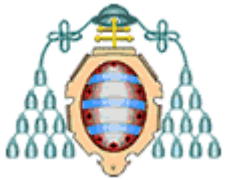
Docker

Jesús Morán y Cristian Augusto

Grupo de Investigación en Ingeniería del Software
http://giis.uniovi.es
Universidad de Oviedo



Problemas de “desarrollo”

No sé qué pudo ocurrir… en mi PC funcionaba

Instala la nueva versión de java

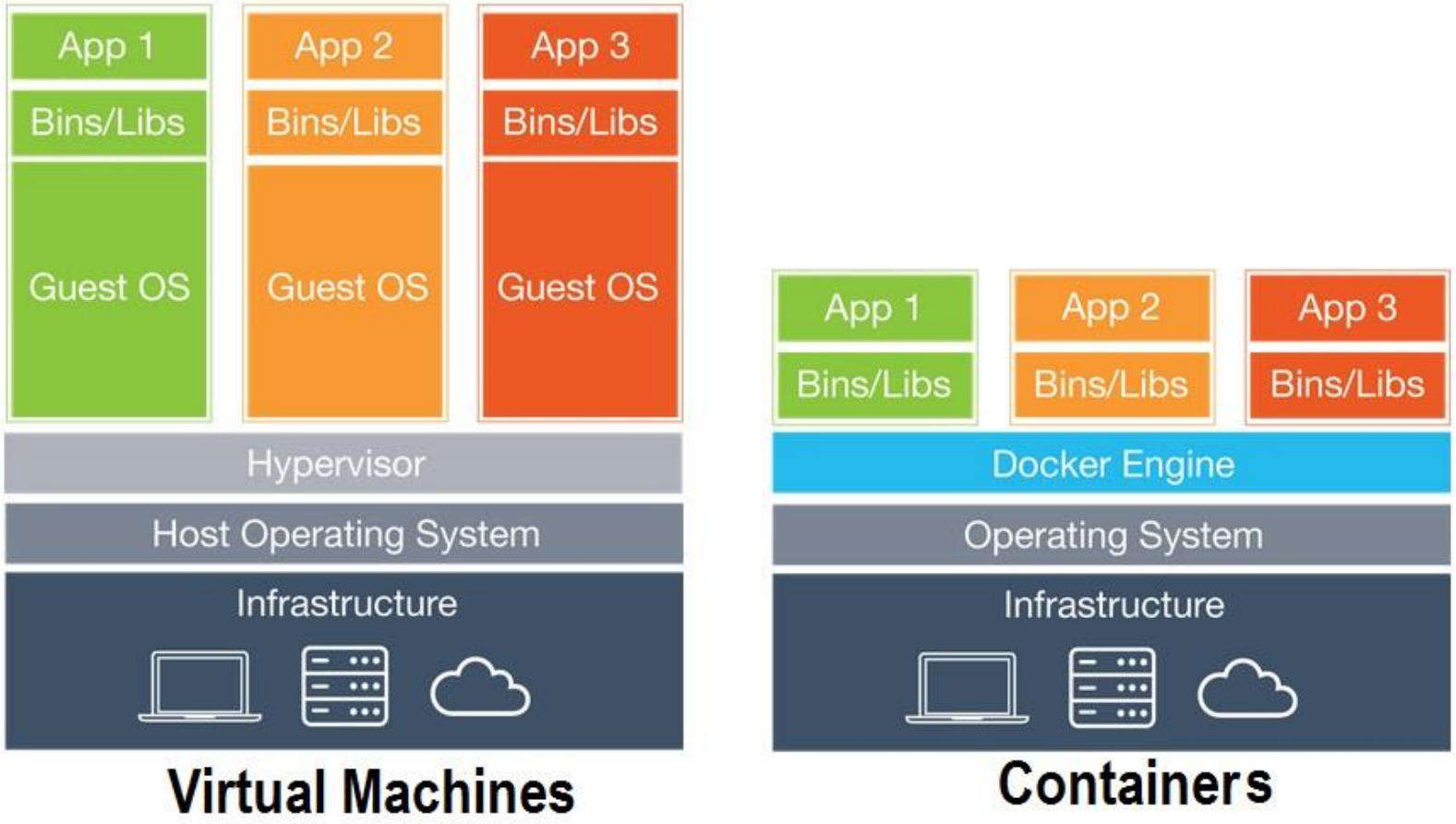
Falta una librería

Tienes que abrir el puerto 8000

Prueba a instalar tomcat

…

Máquina virtual vs Docker



Crédito de la imagen: docker.com

Solución

- Aplicaciones/servicios autocontenidos
- Fáciles de desplegar
- Fáciles de gestionar
- Fáciles de portar
- Seguros
- Aislados
- “Estándar”

*

Terminología docker

- **Imagen**: "*An executable package that includes everything needed to run an application -the code, a runtime, libraries, environment variables, and configuration files.*" Docker.com

- **Contenedor**: "A runtime instance of an image--what the image becomes in memory when executed (that is, an image with state, or a user process" Docker.com

Instalación en RHEL-based

- **Instalación** Docker:
  - `sudo yum install -y yum-utils device-mapper-persistent-data lvm2`
  - `sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo`
  - `sudo yum insall docker-ce docker-ce-cli containerd.io`
- En algún RHEL-based hay conflictos: `sudo yum remove podman buildah`
- **Inicializar** Docker: `sudo systemctl start docker`

Hello World

- Descargamos la imagen de hello-world

```
docker pull hello-world
```

- Creamos un contenedor

```
docker create --name miHelloWorld hello-world
```

- Arrancamos el contenedor

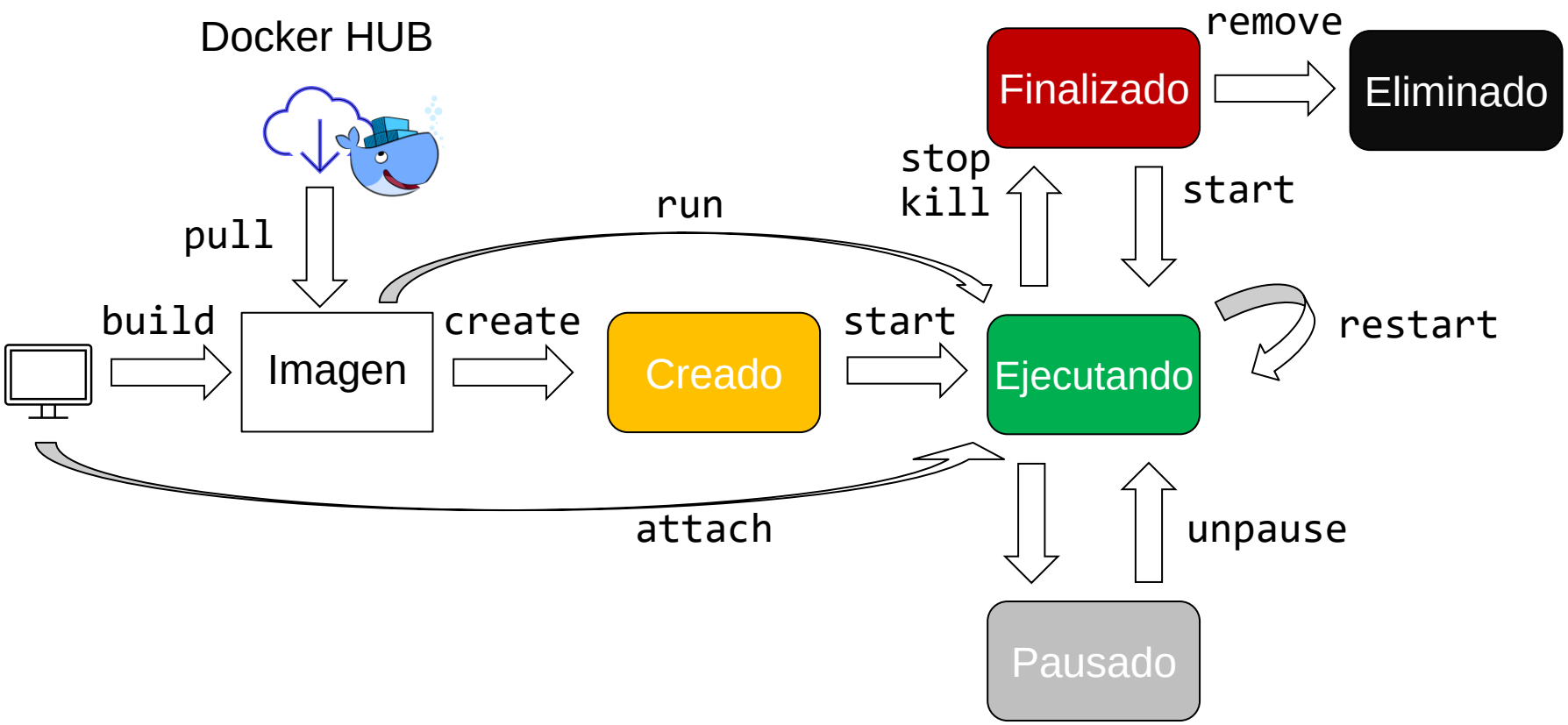
```
docker start --attach --interactive miHelloWorld
```

Hello World

- Salida:

```
[moranjesus@localhost ~]$ sudo docker start --attach --interactive miHelloWorld

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/

[moranjesus@localhost ~]$
```

Ciclo de vida de un contenedor



Contenedor ubuntu

- Descargamos la imagen de ubuntu

```
docker pull ubuntu
```

- Creamos un contenedor

```
docker create --tty --name miUbuntu ubuntu
```

- Arrancamos el contenedor

```
[moranjesus@localhost ~]$ sudo docker start --attach --interactive miUbuntu
root@acfc24995cd2:/#
```

Ciclo de vida de un contenedor

- Arrancar

```
docker start --attach --interactive MiUbuntu
```

- Reiniciar

```
docker restart miUbuntu
```

- Pausar

```
docker pause miUbuntu
```

- Des-pausar

```
docker unpause miUbuntu
```

Ciclo de vida de un contenedor

- Apagar

```
docker stop MiUbuntu
```

- Matar

```
docker kill miUbuntu
```

- Conectarse

```
[moranjesus@localhost ~]$ sudo docker attach miUbuntu
root@acfc24995cd2:/#
```

- Ejecutar un comando

```
[moranjesus@localhost ~]$ sudo docker exec miUbuntu cat /etc/passwd | grep mor*
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
[moranjesus@localhost ~]$
```

Compartir datos

- Opción mount

```
docker create --name miUbuntu --tty --interactive
--mount type=bind,
         source=/home/moranjesus/misDatosEnHost,
        target=/misDatosEnContenedor,
         bind-propagation=shared ubuntu
```

```
[moranjesus@localhost ~]$ ls -l /home/moranjesus/misDatosEnHost/
total 8
-rw-rw-r--. 1 moranjesus moranjesus 13 Apr 12 10:40 archivo1
-rw-r--r--. 1 root       root       13 Apr 12 10:51 archivo2
[moranjesus@localhost ~]$
```

Host

Contenedor miUbuntu

```
root@4c3238aa97bb:/# ls -l /misDatosEnContenedor/
total 8
-rw-rw-r--. 1 1000 1000 13 Apr 12 08:40 archivo1
-rw-r--r--. 1 root root 13 Apr 12 08:51 archivo2
root@4c3238aa97bb:/#
```

- Opción volumen: menos verbosa

Contenedor servidor apache

- Descargamos la imagen del servidor apache

```
docker pull httpd
```

- Creamos un contenedor exponiendo puerto 80

```
docker create --name miServidorWeb --expose 80 httpd
```

- Arrancamos el contenedor

```
docker start miServidorWeb
```

Sólo se puede acceder a través de la IP del contenedor

Contenedor servidor apache

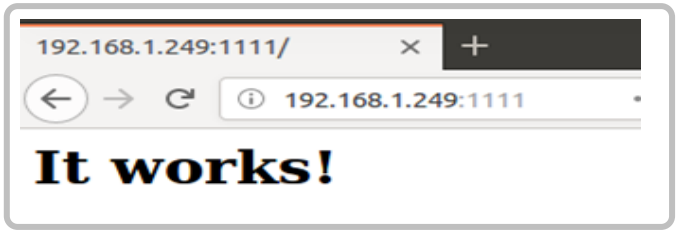
- Creamos un contenedor publicando puerto

```
docker create --name miServidorWeb --publish 1111:80 httpd
```

- Arrancamos el contenedor

```
docker start miServidorWeb
```

Se puede acceder a través del puerto 80 del contenedor [está expuesto]



Se puede acceder a través del puerto 1111 del host [está publicado]

Crear imagen desde contenedor

1. Arrancamos un contenedor

```
docker start --attatch --interactive miUbuntu
```

2. Lo modificamos (instalamos Maven)

```
root@50ab61106e5a:/# mvn -version
Warning: JAVA_HOME environment variable is not set.
Apache Maven 3.3.9
Maven home: /usr/share/maven
```

3. Creamos una imagen a partir del contenedor

```
docker commit miUbuntu ubuntuconmaven
```

Crear imagen con Docker File

1. Creamos un Docker File llamado miImagen

```
FROM ubuntu
MAINTAINER Jesus Moran
RUN apt-get update
RUN apt-get install -y maven
```

2. Creamos la imagen a partir del Docker File

```
docker build --file miImagen -t ubuntumavendesdedockerfile
/home/moranjesus
```

Opciones: FROM, RUN, ENTRYPOINT, CMD, LABEL, EXPOSE, COPY, …

Crear imagen con Docker File

- Primera instrucción que se ejecutará al arrancar el contenedor
  - El proceso tendrá el PID1
- En Docker el PID 1 no suele ser el proceso init
  - Docker está pensado para ejecutar un único servicio
- Si queremos instalar systemd en un contenedor:
  - Ejecutar contenedor con las capabilities de adminsistracción: --cap-add SYS_ADMIN
    - (Alternativa) Ejecutar docker en modo privilegiado: --privileged=true
  - Crear un volumen con el anfitrión: –v /sys/fs/cgroup:/sys/fs/cgroup:ro
  - (Opcional) Evitar servicios innecesarios: eliminar archivos /etc/systemd/system/*wants/ y /lib/systemd/system/*wants/
  - Instalar systemd y ejecutar como entrypoint /sbin/init

Opciones: FROM, RUN, ENTRYPOINT, CMD, LABEL, EXPOSE, COPY, …

Crear imagen de un programa

1. Creamos un programa holaMundo y lo compilamos
2. Creamos un Docker File llamado miImagenJava

```
FROM alpine
ADD holaMundo.class holaMundo.class
RUN apk --update add openjdk8-jre
ENTRYPOINT ["java", "holaMundo"]
```

3. Creamos la imagen a partir del Docker File

```
docker build --file miImagenJava -t holamundojava
/home/moranjesus
```

Gestión de contenedores e imágenes

- Buscar imágenes en repositorio remoto

```
C:\Users\crist>docker search ubuntu
NAME                                DESCRIPTION                                     STARS     OFFICIAL
ubuntu                              Ubuntu is a Debian-based Linux operating sys…   10416     [OK]
dorowu/ubuntu-desktop-lxde-vnc      Docker image to provide HTML5 VNC interface …   385
rastasheep/ubuntu-sshd              Dockerized SSH service, built on top of offi…   240
```

- Ver imágenes del repositorio local:

```
C:\Users\crist>docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
selenoid/vnc                chrome_78.0         67312aa40b2f        2 months ago        897MB
eexit/mirror-http-server    latest              9290012d0cef        4 months ago        285MB
aerokube/selenoid           1.8.4               ec8a96615a09        13 months ago       13.1MB
mysql                       5.7.22              6bb891430fb6        18 months ago       372MB
```

- Ver contenedores en ejecución:

```
C:\Users\crist>docker ps
CONTAINER ID        IMAGE                           COMMAND                  CREATED             STATUS              PORTS
d229c3effb1b        codeurjc/full-teaching:2.3.0    "/bin/sh -c '/wait.s…"   2 months ago        Up 39 seconds       5000/tcp
```

Gestión de contenedores e imágenes

- Ver todos los contenedores

```
C:\Users\crist>docker ps --all
CONTAINER ID        IMAGE                                COMMAND                  CREATED             STATUS
d229c3effb1b        codeurjc/full-teaching:2.3.0         "/bin/sh -c '/wait.s…"   2 months ago        Exited (1) 4 minutes ago
197c9ff3a74b        openvidu/openvidu-server-kms:2.3.0   "/usr/bin/supervisord"   2 months ago        Exited (0) 2 months ago
5c339a9520aa        mysql:5.7.22                         "docker-entrypoint.s…"   2 months ago        Exited (0) 2 months ago
ceb57b9083f4        codeurjc/full-teaching:2.3.0         "/bin/sh -c '/wait.s…"   3 months ago        Exited (137) 3 months ago
```

- Eliminar contenedor

```
docker rm miUbuntu
```

- Ver información estática de contenedores:

```
docker inspect miUbuntu
```

- Ver información dinámica de contenedores

```
docker stats miUbuntu
```

Docker: limitar recursos

- Opciones:
  - --memory: memoria máxima que puede utilizar
  - --memory-swap: memoria + swap máxima
  - …

```
jesus@injtest:~$ sudo docker run -it --rm --memory 512m --memory-swap 512m alpin
e:latest /bin/sh
WARNING: Your kernel does not support swap limit capabilities or the cgroup is n
ot mounted. Memory limited without swap.
/ #
jesus@injtest:~$
```

Problema: No funciona porque se tienen que activar los cgroups del kernel

Docker: limitar recursos

- Activar cgroups v1:
  - Activarlo como opción de arranque:

Archivo /etc/default/grub

```
GRUB_CMDLINE_LINUX_DEFAULT="cgroup_enable=memory swapaccount=1"
```

  - Actualizar el grub:

```
jesus@injtest:~$ sudo update-grub
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.4.0-91-generic
Found initrd image: /boot/initrd.img-5.4.0-91-generic
Found linux image: /boot/vmlinuz-5.4.0-90-generic
Found initrd image: /boot/initrd.img-5.4.0-90-generic
Found linux image: /boot/vmlinuz-5.4.0-86-generic
Found initrd image: /boot/initrd.img-5.4.0-86-generic
Adding boot menu entry for UEFI Firmware Settings
done
jesus@injtest:~$
```

  - Reiniciar la máquina

Docker: limitar recursos

- Activar cgroups ~~v1~~ v2:
  - Sólo para SO con kernel >= 4.15
  - Sólo para Docker >= 20.10
  - Activar cgroups v2 como opción de arranque:

Archivo /etc/default/grub

```
GRUB_CMDLINE_LINUX="systemd.unified_cgroup_hierarchy=1"
```

  - Actualizar el grub
  - Reiniciar la máquina
  - Nota: si queremos utilizar cgroups v1:

GRUB_CMDLINE_LINUX="systemd.unified_cgroup_hierarchy=**0**"

Docker: limitar recursos

- Tras activar cgroup:

```
jesus@injtest:~$ sudo docker run -it --rm --memory 512m --memory-swap 512m --name
"alpine_limitado" alpine:latest /bin/sh
/ #
```

`docker stats`

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
d1ad67693f04	8_65536-131072_0_9	100.60%	3.764GiB / 11GiB	34.22%	1.01kB / 0B	18.4MB / 0B	34
9396294b26d9	7_65536-131072_3_10	100.75%	4.204GiB / 11GiB	38.22%	1.15kB / 0B	0B / 0B	34
924a4266b58a	7_65536-131072_3_9	101.38%	4.068GiB / 11GiB	36.98%	1.15kB / 0B	11.5MB / 0B	34
3a78d55a8951	6_65536-131072_3_11	100.03%	8.983GiB / 11GiB	81.67%	1.57kB / 0B	0B / 0B	34
a93e5955f517	prometheus	0.00%	241.9MiB / 47.41GiB	0.50%	4.94GB / 86.5MB	164MB / 2.7GB	23
5804c90ed26f	caddy	0.00%	33.06MiB / 47.41GiB	0.07%	481MB / 470MB	34.8MB / 0B	22
35757cb3d043	cadvisor	7.98%	59.8MiB / 47.41GiB	0.12%	87.9MB / 5.4GB	95.2MB / 0B	42
82c67db4f94e	alpine_limitado	0.00%	1.684MiB / 512MiB	0.33%	726B / 0B	0B / 0B	1

Docker

- Docker engine: daemon y cliente de contenedores
- Docker machine: gestionar máquinas virtuales con Docker engine
- Docker hub: repositorio de imágenes
- Docker compose: gestión de múltiples contenedores

Docker compose

- Instalación:
  - Descargar docker-compose

```
jesus@injtest:/disk0$  sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname
 -m)" -o /usr/local/bin/docker-compose
[sudo] password for jesus:
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   664  100   664    0     0   5487      0 --:--:-- --:--:-- --:--:--  5487
100 12.1M  100 12.1M    0     0  15.9M      0 --:--:-- --:--:-- --:--:-- 68.2M
```

  - Dar permisos de ejecución

```
jesus@injtest:/disk0$ sudo chmod +x /usr/local/bin/docker-compose
```

  - Instalar command line completion

```
jesus@injtest:/disk0$ sudo curl    -L https://raw.githubusercontent.com/docker/compose/1.29.2/contrib/com
pletion/bash/docker-compose  -o /etc/bash_completion.d/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 13500  100 13500    0     0   117k      0 --:--:-- --:--:-- --:--:--  117k
```

  - Actualizar la configuración de la terminal

```
jesus@injtest:/disk0$ source ~/.bashrc
```

Docker Compose

```
docker-compose --version
```

- Desde Docker Desktop 3.4.0 hay dos versiones
  - La versión docker-compose: implementada en python
  - La versión docker compose (dentro de docker-cli):
    - Se conoce como Compose V2
    - Implementada en Go
    - Todavía no tiene toda la funcionalidad de docker-compose
    - Más amigable con la nube:
      - Backends para guardar imágenes: ECS de AWS, ACI de Azure,…

```
docker compose --version
```

Docker compose

```
version: '3.3’

services:
    db:
        image: mysql:5.7
        volumes:
            - dbdata:/var/lib/mysql
        restart: always
        environment:
            MYSQL_ROOT_PASSWORD: somewordpress
            MYSQL_DATABASE: wordpress
            MYSQL_USER: wordpress
            MYSQL_PASSWORD: wordpress
    wordpress:
        depends_on:
            - db
        image: wordpress:latest
        ports:
            - "8000:80"
        restart: always
        environment:
            WORDPRESS_DB_HOST: db:3306
            WORDPRESS_DB_USER: wordpress
             WORDPRESS_DB_PASSWORD: wordpress
volumes:
         dbdata:
```

Docker Compose

- Desplegar/arrancar los contenedores

```
docker-compose up
```

- Parar los contenedores

```
docker-compose stop
```

- Eliminar contenedores + red

```
docker-compose down
```

- Eliminar contenedores + red + volúmenes

```
docker-compose down --volumes
```

Docker Compose

- “Construir” o “Reconstruir” las imágenes

```
docker-compose build
```

- Reiniciar los contenedores

```
docker-compose restart
```

- Descargar las imágenes necesarias

```
docker-compose pull
```

- Modo “detached” (no se muestran los logs)

```
docker-compose --detach
```

Monitorizar Docker

- Crear docker-compose con servicios:
  - cAdvisor: obtiene métrica del uso de contenedores
  - Prometheus: base de datos que almacena las métricas
  - caddy: servidor web y proxy reverso

Monitorizar Docker

docker-compose.yml

```yaml
version: '3.9'

volumes:
    prometheus_data: {}

services:
  prometheus:
    image: prom/prometheus:v2.33.3
    container_name: prometheus
    expose:
      - 9090
    command:
      - --config.file=/etc/prometheus/prometheus.yml
      - --storage.tsdb.retention.time=1y
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml:ro
      - prometheus_data:/prometheus:rw
    depends_on:
      - cadvisor

  cadvisor:
    image: gcr.io/cadvisor/cadvisor:v0.43.0
    container_name: cadvisor
    expose:
      - 8080
    volumes:
      - /:/rootfs:ro
      - /var/run:/var/run:rw
      - /sys:/sys:ro
      - /var/lib/docker/:/var/lib/docker:ro
    command:
      - "-enable_load_reader=true"

  caddy:
    image: caddy:2.4.6
    container_name: caddy
    ports:
      - "9090:9090"
      - "8080:8080"
    volumes:
      - ./Caddyfile:/etc/caddy/Caddyfile
    environment:
      - ADMIN_USER=${ADMIN_USER:-admin}
      - ADMIN_PASSWORD=${ADMIN_PASSWORD:-admin}
      - ADMIN_PASSWORD_HASH=${ADMIN_PASSWORD_HASH:-[redacted]
```

Monitorizar Docker

docker-compose.yml

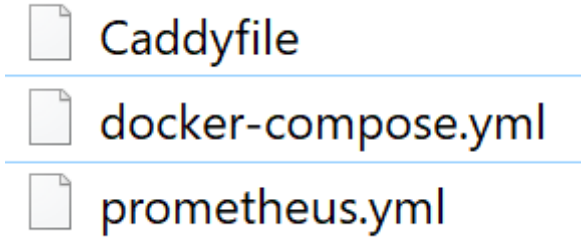
```yaml
version: '3.9'

volumes:
    prometheus_data: {}

services:
  prometheus:
    image: prom/prometheus:v2.33.3
    container_name: prometheus
    expose:
      - 9090
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml:ro
      - prometheus_data:/prometheus:rw
    depends_on:
      - cadvisor

  cadvisor:
    image: gcr.io/cadvisor/cadvisor:v0.43.0
    container_name: cadvisor
    expose:
      - 8080
    volumes:
      - /:/rootfs:ro
      - /var/run:/var/run:rw
      - /sys:/sys:ro
      - /var/lib/docker/:/var/lib/docker:ro
    command:
      - "-enable_load_reader=true"

  caddy:
    image: caddy:2.4.6
    container_name: caddy
    ports:
      - "9090:9090"
      - "8080:8080"
    volumes:
      - ./Caddyfile:/etc/caddy/Caddyfile
    environment:
      - ADMIN_USER=${ADMIN_USER:-admin}
      - ADMIN_PASSWORD=${ADMIN_PASSWORD:-admin}
      - ADMIN_PASSWORD_HASH=${ADMIN_PASSWORD_HASH:-
```

Para saber cuál es el hash de la contraseña que queremos asignar: escribimos la contraseña

```
jesus@injtest:/disk0/monitorizar$ sudo docker run --rm caddy:2.4.6 caddy hash-password --plaintext '
jesus@injtest:/disk0/monitorizar$
```

Copiamos el hash a la variable de entorno

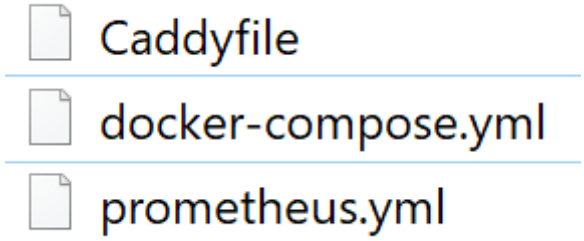
Monitorizar Docker



- Archivos:
  - docker-compose-yml: contiene los contenedores
  - prometheus.yml:
    - Cada 10s guarda las métricas de cadvisor

```
scrape_configs:
  - job_name: cadvisor
    scrape_interval: 10s
    static_configs:
      - targets:
          - cadvisor:8080
```

Monitorizar Docker



- Archivos:
  - docker-compose-yml: contiene los contenedores
  - prometheus.yml:
  - Caddyfile: autenticación básica (usuario y contraseña) y proxy reverso a cadvisor y prometheus

```
:9090 {
    basicauth /* {
        {$ADMIN_USER} {$ADMIN_PASSWORD_HASH}
    }
    reverse_proxy prometheus:9090
}

:8080 {
    basicauth /* {
        {$ADMIN_USER} {$ADMIN_PASSWORD_HASH}
    }

    reverse_proxy cadvisor:8080
}
```

Monitorizar Docker

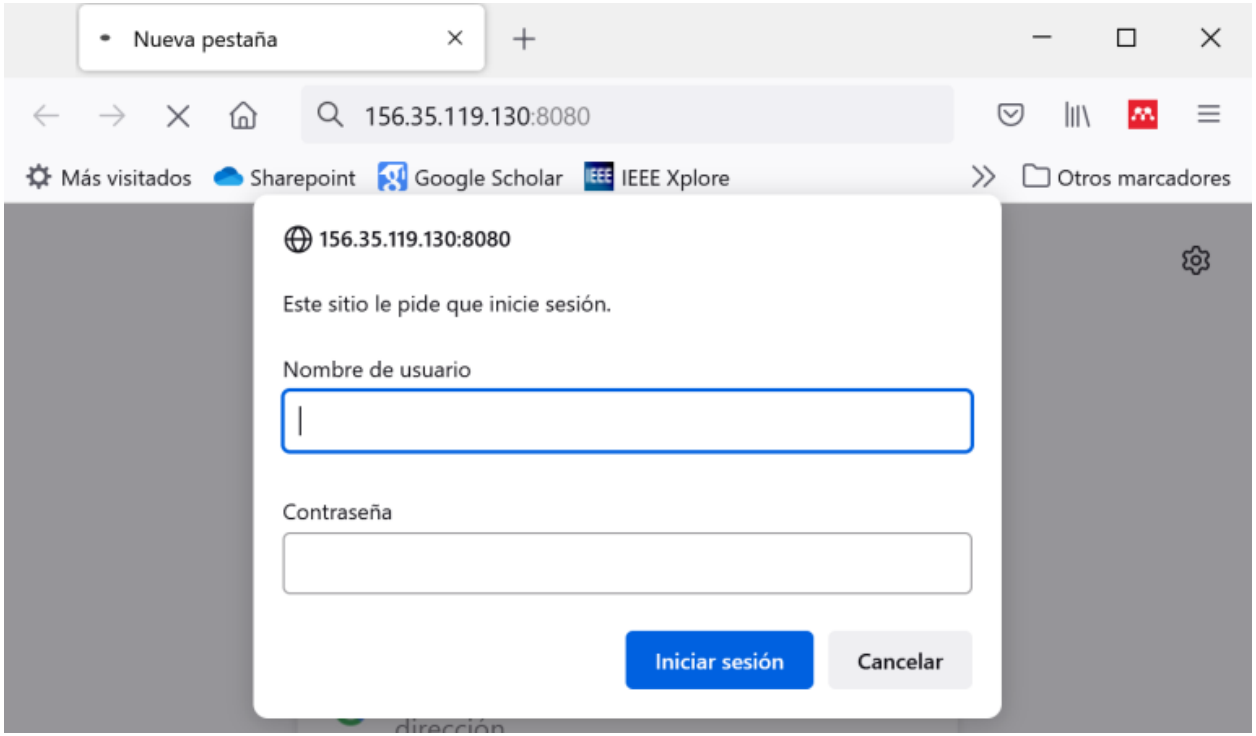
- Desplegar servicio de monitorización:

```
jesus@injtest:/disk0/monitorizar$ sudo docker-compose up --detach
Creating network "monitorizar_default" with the default driver
Creating volume "monitorizar_prometheus_data" with default driver
Creating cadvisor ... done
Creating caddy    ... done
Creating prometheus ... done
jesus@injtest:/disk0/monitorizar$
```

  - Crea red
  - Crea volumen para guardar los datos de la bd
  - Crea contenedores

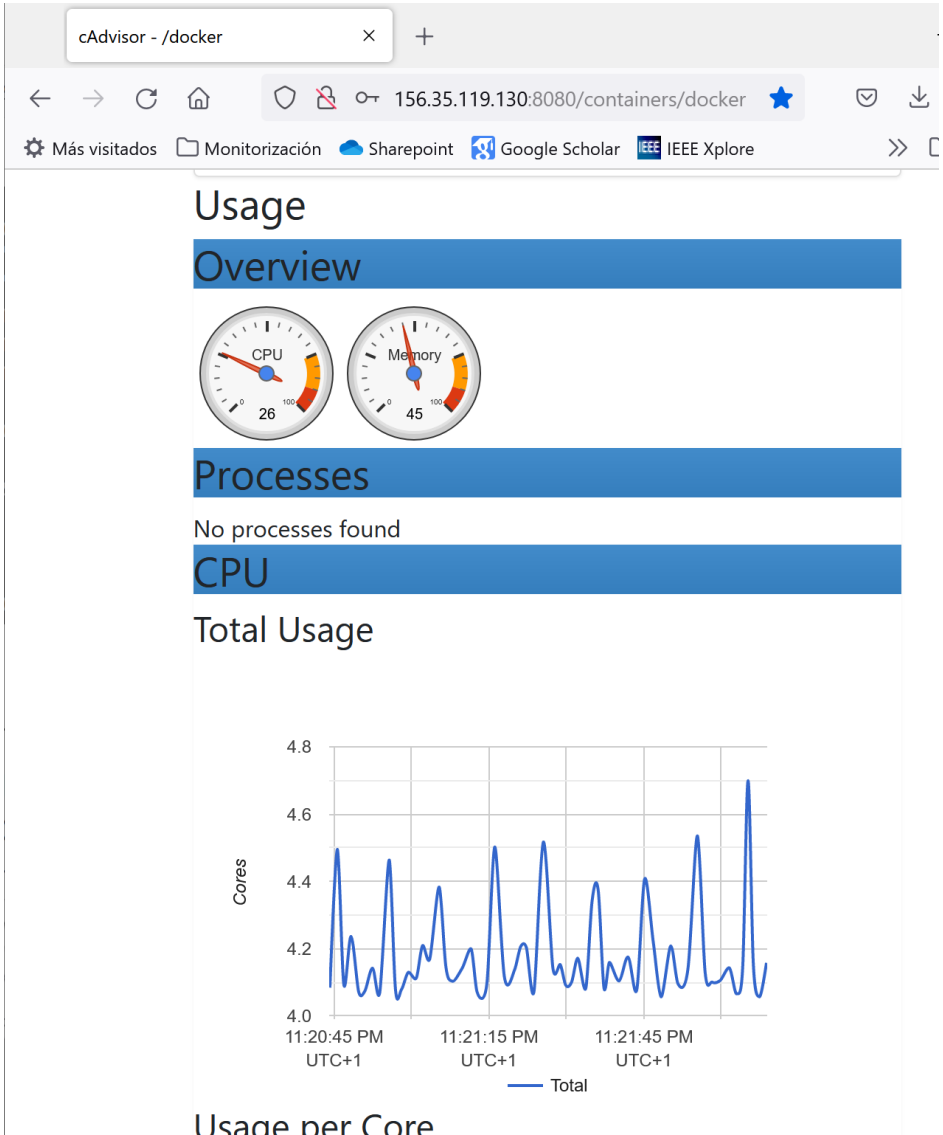
Monitorizar Docker

- cAdvisor



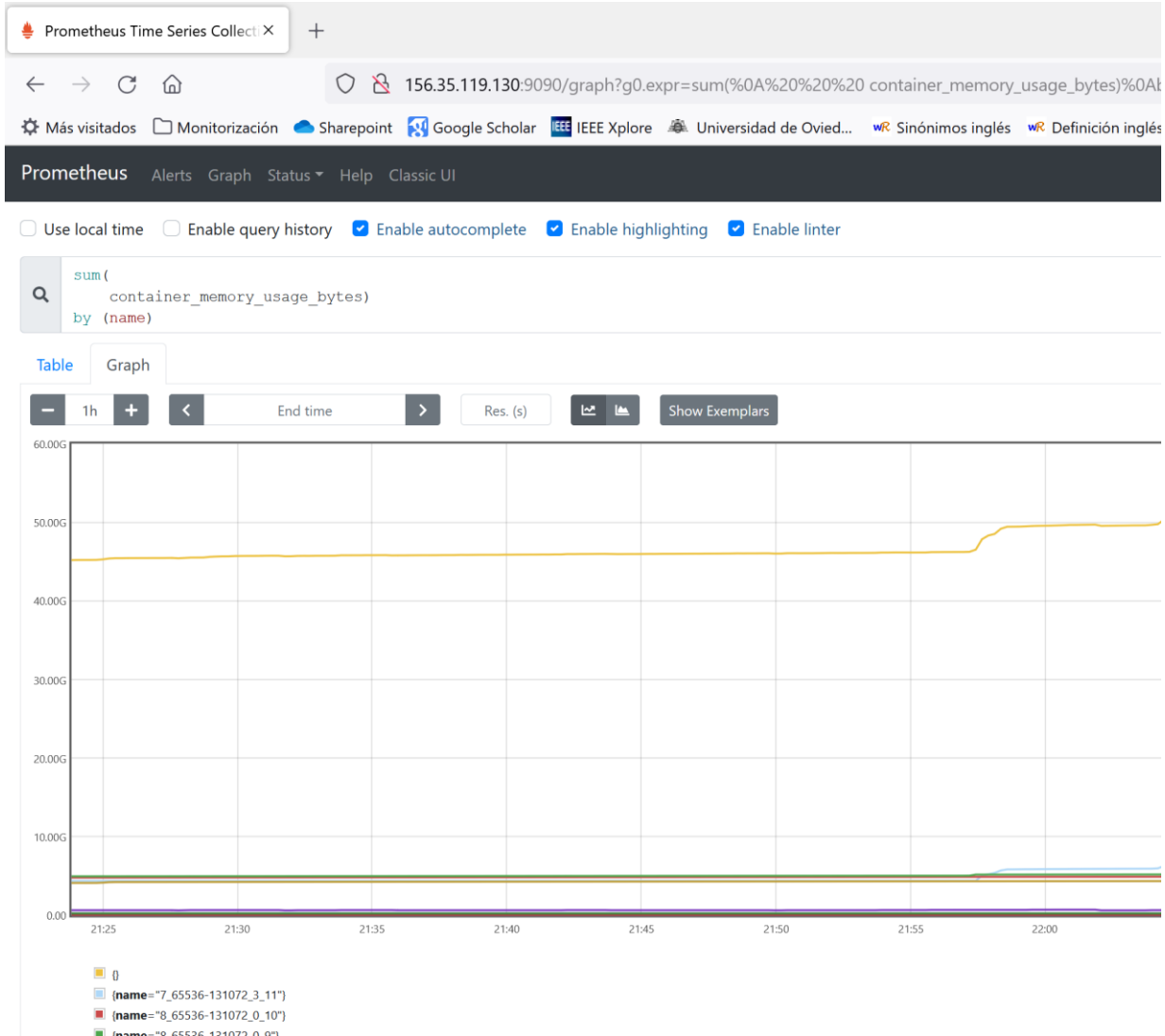
Monitorizar Docker

- cAdvisor



Monitorizar Docker

- Prometheus:



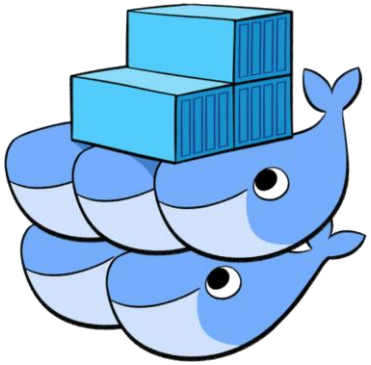
Data + Dev + Op

- Devops:
  - Desarrollo + Operaciones
  - Integración continua
  - Despliegue continuo
  - Entrega continua

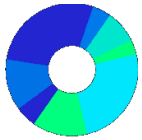
- DataOps
  - Desarrollo + Operaciones + científicos/analistas de datos
  - Analítica continua

Orquestación de contenedores

- Organizar y dirigir cómo se despliegan los contenedores. Alternativas:



Apache MESOS™



Jesús Morán y Cristian Augusto

Grupo de Investigación en Ingeniería del Software

http://giis.uniovi.es

Universidad de Oviedo