Grid y Computación de Altas Prestaciones
GRyCAP

Prácticas.
Contenedores Docker (II)

Máster Universitario en Computación en la Nube y de Altas Prestaciones

Objetivos

Grid y Computación de Altas Prestaciones GRyCAP

- Entender cómo usar de forma apropiada las funcionalidades básicas proporcionadas por la **tecnología docker** para gestionar:
  - **procesos (batch e interactivos)** empaquetados en contenedores**.**
  - **volúmenes de datos** en los contenedores.
  - **creación manual de imágenes** de contenedores**.**
- Probar y testear dichas opciones a través de un host local desplegado en la nube mediante Docker-CE.

Contenidos

Grid y Computación de Altas Prestaciones
GRyCAP

P5. Gestión Interactiva de Contenedores.

P6. Gestión de Volúmenes de Datos.

P7. Creación Manual de Imágenes.

PA3. Solver Python Suma Vectores.

PA4. Solver Octave Ecuaciones Lineales.

P5. Gestión Interactiva de Contenedores
Comandos Básicos Docker

Grid y Computación de Altas Prestaciones
GRyCAP

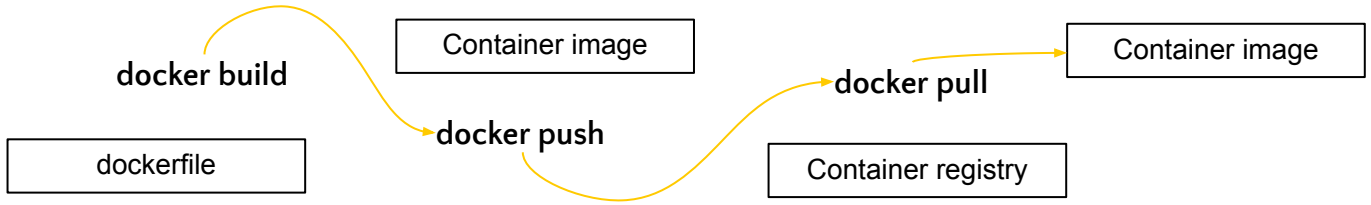
- El objetivo de esta práctica es:
  - entender cómo funcionan los **comandos** fundamentales de **Docker** para interactuar con contenedores a través de terminales interactivos.
  - utilizar los comandos de Docker para la gestión Interactiva de Contenedores.

Uso

- Listado imagenes (docker images)
- Descarga desde Docker Hub (docker pull <imagen>)
- Eliminar imagen (docker rmi)
- Consulta listado (docker ps)
- Consulta Contenedor (docker Inspect)
- Crear Contenedor (docker create)
- Activación (docker start)
- Parada (docker stop)
- Ejecución (docker Run)
- Borrado (docker rm)
- Consulta puertos (docker port)
- ejecución (docker attach)
- Ejecución (docker exec)
- Volúmenes (docker cp)

Creación Imágenes

- Creación del contenedor
  - Vía creación del Dockerfile
    - Creación de la imagen (docker build).
  - Vía actualización
    - Descarga (docker pull) y actualización.
    - docker commit.
- Publicación (docker push)



P5. Gestión Interactiva de Contenedores
Docker create/start/run

Grid y Computación de Altas Prestaciones GRyCAP

- *docker create/start y docker run* crean y arrancan contenedores.
- **IMPORTANTE:**
  Docker start lanza el contenedor en background siempre.
  Docker run lanza contenedores en primer plano. Se puede en segundo plano (opción -d).
- En tareas anteriores relativas a un servidor web de nginx:
  - No podíamos entrar en el contenedor a través de un terminal y cambiar su configuración para que el puerto 443 estuviera activo y sirviera peticiones https, dado que por defecto nginx solo abre http.
- Para solucionar estos problemas, debemos de crear los contenedores mediante **docker create** o **docker run** con opciones específicas que permitan abrir canales interactivos. En ambos casos son las mismas opciones.

docker create <opciones> <[nombre_imagen:etiqueta/s][IDImagen]> <comando>

docker run <opciones> <nombre_imagen:etiqueta/s> <comando>

- Opciones nuevas para la gestión de terminales interactivas

-i → Mantiene la entrada estándar STDIN abierta en el proceso asociado del contenedor a través de un canal de comunicación, para así poder interactuar. El proceso edel contenedor debe de ser interactivo.

-t → Abre un pseudoterminal (dentro del contenedor) para interactuar con el proceso si este es interactivo (Ej. bash, sh etc...). El pseudoterminal interactúa con la entrada estándar desde dentro del contenedor con el proceso si la opción –i está activa.

-it → es el equivalente a activar -i -t por separado.

- Si optamos por **docker start** para arrancar el contenedor, para poder interactuar con el contenedor se debe de arrancar el contenedor con la captura del salida/error estándar en el terminal local (opción -a)**.**

docker start -a <[nombrecontenedor][IDContenedor]>

P5. Gestión Interactiva de Contenedores
Docker create/start

Grid y Computación de Altas Prestaciones
GRyCAP

- **Tarea 5.1:** Crea un contenedor *mi_con1_int* con un canal interactivo (opción -i en create) y un pseudoterminal (opción -t en create). Utiliza la imagen *ubuntu:focal* y utiliza el comando /bin/hostname. Una vez creado, arranca el contenedor en segundo plano activando la salida/error estándar (opción -a en start) para que se muestre en nuestro terminal local la salida del contenedor.

¿Abre un psedoterminal para interactuar con el contenedor?¿porque?

P5. Gestión Interactiva de Contenedores
Docker create/start

Grid y Computación de Altas Prestaciones
GRyCAP

- **Tarea 5.2:** Crea y arranca en segundo plano un contenedor *mi_con2_int* con un pseudoterminal (opción -t de create) para interactuar. **NO utilices la opción de -i.** Utiliza la imagen *ubuntu:focal* y **utiliza el comando /bin/bash** (proceso interactivo).

¿Abre una terminal para interactuar?¿porque?

¿Puedes interactuar con el proceso bash a través del pseudoterminal? ¿Por qué?

P5. Gestión Interactiva de Contenedores
Docker create/start

- **Tarea 5.3:** Crea y arranca en segundo plano un contenedor *mi_con3_int* con un pseudoterminal (opción -t de create) para interactuar y un canal de interacción (opción -i de create). Utiliza la imagen *ubuntu:focal* y utiliza el comando /bin/bash (proceso interactivo).

¿Abre una terminal para interactuar?¿porque?

No! a diferencia de la tarea anterior, ahora el el contenedor si que abre un canal de comunicación de la entrada

P5. Gestión Interactiva de Contenedores
Docker attach

Grid y Computación de Altas Prestaciones GRyCAP

- ***docker attach*** conecta tu terminal local (ejecutado en primer plano) al contenedor que se está ejecutando en un segundo plano. Pone el terminal local y el contenedor en el mismo plano de ejecución.

- Este comando solo funcionará si el contenedor ha sido creado con un canal interactivo (opción -i de create) y un pseudoterminal (opción -t de create) que permita interactuar con el proceso del contenedor.

docker attach <[nombre_contenedor][IDContenedor]>

- **Parámetros:**

<nombre_contenedor> → Etiqueta del contenedor a poner en marcha que se le asignó en el momento de su creación.

<IDContenedor> → Identificador del contenedor (Completo o resumido)

P5. Gestión Interactiva de Contenedores
Docker attach

Grid y Computación de Altas Prestaciones
GRyCAP

- **Tarea 5.4:** Crea y arranca en segundo plano un contenedor *mi_con4_int* con un canal de interacción (opción -i de create) y un pseudoterminal (opcion -t de create). Utiliza la imagen *ubuntu:focal* y utiliza el comando /bin/bash (proceso interactivo). Luego conectate al contenedor con tu terminal local.

¿puedes interactuar con el proceso bash?

P5. Gestión Interactiva de Contenedores
Docker exec

Grid y Computación de Altas Prestaciones GRyCAP

- El comando **docker exec** permite lanzar nuevos comandos sobre un contenedor que **ya esté en marcha**.
- El contenedor sobre el que se aplica exec debe de tener un proceso interactivo en marcha que permita la ejecución de nuevos comandos (Ej. bash, sh).
- Es un mecanismo para poder lanzar diferentes procesos (comandos) en un mismo contenedor de **forma interactiva**.

docker exec <opciones><[nombre_contenedor][IDContenedor]><comando>

- **Parámetros:**
  - <nombre_contenedor> → Etiqueta del contenedor a poner en marcha que se le asignó en el momento de su creación.
  - <IDContenedor> → Identificador del contenedor (Completo o resumido)
  - <comando> → Comando a ejecutar en el contenedor.
- **Opciones:**
  - -i → Misma opción que el comando create, pero asociado al nuevo proceso (comando) lanzado.
  - -t → Misma opción que el comando create, pero asociado al nuevo proceso (comando) lanzado.

P5. Gestión Interactiva de Contenedores
Docker exec

Grid y Computación de Altas Prestaciones GRyCAP

- **Tarea 5.5:** Crea y arranca en segundo plano un contenedor *mi_con5_int* **SIN** canal de interacción y **SIN** un pseudoterminal. Utiliza la imagen *ubuntu:focal* y utiliza el comando /bin/bash (proceso interactivo). Lanza el comando */bin/ls* mediante el comando *docker exec.*

¿qué ocurre?

P5. Gestión Interactiva de Contenedores
Docker exec

Grid y Computación de Altas Prestaciones
GRyCAP

- **Tarea 5.6:** Crea y arranca en segundo plano un contenedor *mi_con6_int* **CON** canal de interacción y **SIN** un pseudoterminal. Utiliza la imagen *ubuntu:focal* y utiliza el comando /bin/bash (proceso interactivo). Lanza estos comandos */bin/ls , /bin/ps y /bin/pwd* mediante el comando *docker exec a través de sucesivas ejecuciones.*

¿qué ocurre?

P5. Gestión Interactiva de Contenedores
Docker exec

- **Tarea 5.7:** Crea y arranca en segundo plano un contenedor *mi_con7_int* **CON** canal de interacción. Utiliza la imagen *ubuntu:focal* y utiliza el **comando /bin/bash** (proceso interactivo). Lanza el proceso interactivo ***/bin/bash*** mediante el comando *docker exec con la opción -i -t activa.*

¿qué crees que va ocurrir?

P5. Gestión Interactiva de Contenedores

Evidencias

Grid y Computación de Altas Prestaciones GRyCAP

- Al final de esta actividad práctica deberás de tener el entorno **Docker-CE** con una serie de contenedores creados.
- Verifica que están todos los contenedores con docker ps -a:

```
josegqui@MVdocker-josegqui:~$ docker ps -a
CONTAINER ID   IMAGE                COMMAND                  CREATED              STATUS                        PORTS                                                                              NAMES
0c5bb57057ce   ubuntu:focal         "/bin/bash"              14 minutes ago       Up 14 minutes                                                                                                    mi_con7_int
aa5eb942c135   ubuntu:focal         "/bin/bash"              19 minutes ago       Up 19 minutes                                                                                                    mi_con6_int
2c00b6784dcb   ubuntu:focal         "/bin/bash"              23 minutes ago       Exited (0) 23 minutes ago                                                                                        mi_con5_int
3440a2a85000   ubuntu:focal         "/bin/bash"              31 minutes ago       Exited (0) 23 minutes ago                                                                                        mi_con4_int
6a7214e74641   ubuntu:focal         "/bin/bash"              About an hour ago    Up About an hour                                                                                                 mi_con3_int
461198b5d33d   ubuntu:focal         "/bin/hostname"          About an hour ago    Exited (0) About an hour ago                                                                                     mi_con1_int
93cb38a15a44   ubuntu:focal         "/bin/bash"              About an hour ago    Up About an hour                                                                                                 mi_con2_int
07d27ff4149f   tomcat               "catalina.sh run"        3 hours ago          Exited (143) 2 hours ago                                                                                         servidor_web
5a297c787418   gnuoctave/octave     "octave --eval 'sin(…"   23 hours ago         Exited (0) 3 hours ago                                                                                           solver_octave
987a610f63f1   nginx:latest         "/docker-entrypoint.…"   24 hours ago         Exited (255) 3 hours ago      0.0.0.0:32768->80/tcp, [::]:32768->80/tcp, 0.0.0.0:32769->443/tcp, [::]:32769->443/tcp   mi_con4_web
b0bb2087b4bb   nginx:latest         "/docker-entrypoint.…"   24 hours ago         Exited (255) 3 hours ago      0.0.0.0:8080->80/tcp, [::]:8080->80/tcp, 0.0.0.0:8443->443/tcp, [::]:8443->443/tcp       mi_con3_web
03997d24aba4   nginx:latest         "/docker-entrypoint.…"   24 hours ago         Exited (0) 24 hours ago                                                                                          mi_con2_web
e958135b28ba   ubuntu:focal         "/bin/hostname"          24 hours ago         Exited (0) 24 hours ago                                                                                          mi_con3_run2
4394256e41f2   ubuntu:focal         "/bin/hostname"          24 hours ago         Exited (0) 24 hours ago                                                                                          mi_con3_run
fb06e155d0ca   ubuntu:focal         "/bin/hostname"          24 hours ago         Created                                                                                                          mi_con3
c9498d3a048c   ubuntu:focal         "/bin/hostname joseg…"   24 hours ago         Created                                                                                                          mi_con2
f078c8dc04db   ubuntu:focal         "/bin/hostname"          24 hours ago         Exited (0) 24 hours ago                                                                                          mi_con1
d2fd989ebc2f   hello-world          "/hello"                 25 hours ago         Exited (0) 25 hours ago                                                                                          competent_dijkstra
```

- Para el portafolio, hay que recoger las siguientes evidencias:
  - Captura de pantalla del terminal con la salida del comando anterior donde aparezca tu usuario.
  - Texto donde indique la tarea y el comando o comandos utilizados.

Contenidos

P5. Gestión Interactiva de Contenedores.

P6. Gestión de Volúmenes de Datos.

P7. Creación Manual de Imágenes.

PA3. Solver Python Suma Vectores.

PA4. Solver Octave Ecuaciones Lineales.

P6. Gestión de Volúmenes de Datos
Comandos Básicos Docker

Grid y Computación de Altas Prestaciones
GRyCAP

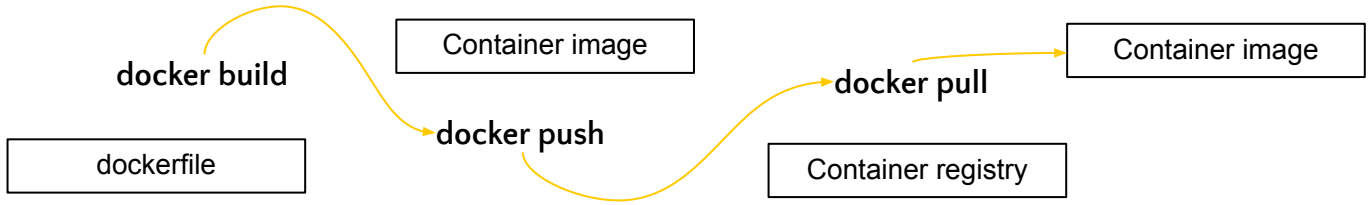
- El objetivo de esta práctica es:
  - entender cómo funcionan los **comandos** fundamentales de **Docker** para el montaje y gestión de volúmenes de datos entre contenedores y el host.
  - utilizar los comandos de Docker para la gestión de volúmenes de datos.

Uso

- Listado imagenes (docker images)
- Descarga desde Docker Hub (docker pull <imagen>)
- Eliminar imagen (docker rmi)
- Consulta listado (docker ps)
- Consulta Contenedor (docker Inspect)
- Crear Contenedor (docker create)
- Activación (docker start)
- Parada (docker stop)
- Ejecución (docker Run)
- Borrado (docker rm)
- Consulta puertos (docker port)
- ejecución (docker attach)
- Ejecución (docker exec)
- Volúmenes (docker cp)

Creación Imágenes

- Creación del contenedor
  - Vía creación del Dockerfile
    - Creación de la imagen (docker build).
  - Vía actualización
    - Descarga (docker pull) y actualización.
    - docker commit.
- Publicación (docker push)



P6. Gestión de Volúmenes de Datos
Introducción

Grid y Computación de Altas Prestaciones
GRyCAP

- **Tarea 6.1:** Crea y arranca un contenedor en segundo plano *mi_con1_vol*. El contenedor debe de lanzar el proceso interactivo */bin/bash,* basado con la imagen ubuntu:focal y la opción -i activada. Una vez arrancado, primero lanza con *docker exec* el siguiente comando */bin/touch /tmp/mifichero.txt y luego docker exec* el siguiente comando /bin/ls /tmp.

  Para el contenedor y vuelvelo arrancar. Después lanza un *docker exec* el siguiente comando /bin/ls/tmp

¿qué ocurre con el fichero mifichero.txt?

P6. Gestión de Volúmenes de Datos
Tareas Introducción

- **Tarea 6.2:** Ahora elimina el contenedor mi_con1_vol y vuelvelo a crear con el mismo nombre y de la misma manera que en la tarea 6.1. Después arranca el contenedor y ejecuta *docker exec* el siguiente comando /bin/ls/tmp

¿qué ocurre con el fichero mifichero.txt?

P6. Gestión de Volúmenes de Datos
Docker volume create / ls

Grid y Computación de Altas Prestaciones GRyCAP

- ***docker volume create*** permite crear volúmenes de datos persistentes en el entorno docker que pueden ser montados en contenedores en el proceso de su creación.
- Los volúmenes persistentes montados en un contenedor, aunque el contenedor sea eliminado, los datos del volumen persisten y pueden ser montados nuevamente en otros contenedores.

docker volume create <nombre_volumen>

- **Parámetros:**

  <nombre_volumen> → Nombre del volumen.

- ***docker volume ls*** permite listar volúmenes de datos persistentes creados en el entorno docker.

docker volume ls

- ***docker volume rm*** permite eliminar volúmenes de datos persistentes creados en el entorno docker.

docker volume rm <opciones> <[nombre_volumen]>

- **Parámetros:**

  <nombre_volumen> → Nombre del volumen.

- **Opciones:**

  -f → fuerza la eliminación de un volumen aunque esté montado en contenedores activos.

P6. Gestión de Volúmenes de Datos
Docker Volumen Create

Grid y Computación de Altas Prestaciones
GRyCAP

- **Tarea 6.3:** Crea un volumen persistente y llámalo mi_vol_persistente. Luego lista los volúmenes persistentes creados en docker.

P6. Gestión de Volúmenes de Datos
Docker create/run

Grid y Computación de Altas Prestaciones GRyCAP

- ***docker create*** y ***docker run*** permiten crear contenedores a partir de una imagen que tengamos en el registro local.
- **Los datos dentro del contenedor son volátiles**. Cuando se elimina el contenedor, el sistema de ficheros desaparece.
- Cuando se crea una imagen, **se pueden montar volúmenes de datos persistentes**, para que cuando se elimine un contenedor, los datos se mantengan y se puedan montar en nuevos contenedores. Para ello se requieren de unas opciones específicas:

docker create <opciones> <[nombre_imagen:etiqueta/s][IDImagen]> <comando>
docker run <opciones> <nombre_imagen:etiqueta/s> <comando>

- Opciones nuevas para la gestión de volúmenes

--volume o -v <directorio_host>:<directorio_contenedor> → Monta como un volumen de datos persistente dentro del contenedor montando el directorio del host indicado. El control y persistencia de este directorio lo gestiona el host.

--volume o -v <volumen>:<directorio_contenedor> → Monta un volumen de datos persistente previamente creado previamente (docke volume create) dentro del contenedor. El control de este directorio lo gestiona docker y sólo es accesible desde dentro del contenedor donde se monta.

--volume o -v <directorio_contenedor> → Crea un nuevo volumen de datos persistente anónimo, identificado por un id, y lo monta dentro del contenedor. El control de este directorio lo gestiona docker y sólo es accesible desde dentro del contenedor donde se monta.

P6. Gestión de Volúmenes de Datos
Docker create

Grid y Computación de Altas Prestaciones
GRyCAP

- **Tarea 6.4:** Crea y arranca un contenedor en segundo plano mi_con2_vol que monte el volumen mi_vol_persistente (creado en la tarea 6.3.) en la carpeta /persiste del contenedor.
  Una vez arrancado, primero lanza con docker exec el siguiente comando /bin/touch /persiste/mifichero.txt y luego docker exec el siguiente comando /bin/ls /tmp.
  Elimina el contenedor y vuelve a crearlo. Después lanza un docker exec el siguiente comando /bin/ls /persiste

¿qué ocurre con el fichero mifichero.txt?

P6. Gestión de Volúmenes de Datos
Docker create

- **Tarea 6.5:** Crea y arranca un contenedor en segundo plano *mi_con3_vol* que monte el directorio *$HOME* de la máquina host dentro del contenedor en el directorio */mnt*. El contenedor debe de lanzar el proceso interactivo */bin/bash* y tener un psedoterminal (opcion -t de docker create) con un canal abierto de interacción (opción -i de docker create). Una vez arrancado, lanza con *docker exec* el siguiente comando *cat /mnt/mi_nombre.txt*

  El contenedor debe de ejecutar el comando “cat /mnt/mi_nombre.txt” cuando arranque. Utiliza la imagen de ubuntu:focal.

¿qué ocurre con el fichero mi_nombre.txt?

P6. Gestión de Volúmenes de Datos
Docker create

Grid y Computación de Altas Prestaciones
GRyCAP

- **Tarea 6.6:** Crea el fichero mi_nombre.txt (con tu nombre real) en el host en la carpeta $HOME y vuelve a lanzar con *docker exec* el siguiente comando *cat /mnt/mi_nombre.txt* utilizando el contenedor mi_con3_vol.

¿qué ocurre con el fichero mi_nombre.txt?

P6. Gestión de Volúmenes de Datos
Docker create

- **Tarea 6.7:** Entra en el pseudoterminal interactivo del contenedor mi_con3_vol (docker attach) y cambia el contenido del fichero /mnt/mi_nombre.txt poniendo tu nombre un par de veces.

¿qué ocurre con el fichero mi_nombre.txt?

P6. Gestión de Volúmenes de Datos
Docker create

- **Tarea 6.8:** Crea y arranca un contenedor en segundo plano *mi_con4_vol* que cree un **volumen persistente anónimo** y lo monte en directorio */persiste* del contenedor. El contenedor debe de lanzar el proceso interactivo */bin/bash* y tener un psedoterminal (opcion -t de docker create) con un canal abierto de interacción (opción -i de docker create).

  Una vez arrancado, lanza con *docker exec* el siguiente comando *touch /persiste/fichero.txt* y después el siguiente comando */bin/ls/ persiste.*

Lista los volúmenes existentes. ¿Qué observas?

P6. Gestión de Volúmenes de Datos
Docker cp

Grid y Computación de Altas Prestaciones GRyCAP

- **docker cp** permite el copiado de datos desde host a un contenedor activo y viceversa.
- Es un copiado recursivo, por lo que copia ficheros y directorios.

docker cp <directorio_host><nombre_contenedor:directorio_contenedor>
docker cp <nombre_contenedor:directorio_contenedor><directorio_host>

- Parámetros:

<directorio_host> → Path del directorio o fichero en el host.
<directorio_contenedor> → Path del directorio o fichero en el contenedor.

P6. Gestión de Volúmenes de Datos
Docker cp

- **Tarea 6.9:** Arranca mi_con3_vol. Crea un nuevo fichero mi_fichero1.txt en tu host y cópialo en tmp dentro del contenedor mi_cont1_vol. Luego lanza con exec un *cat /tmp/mi_fichero1.txt*

¿que ocurre?

¿que ocurrirá si eliminamos el contenedor, lo volvemos a crear, arrancar y lanzamos el mismo exec?

P6. Gestión de Volúmenes de Datos
Docker cp

- **Tarea 6.10:** Crea un nuevo nuevo fichero mi_fichero3.txt en el directorio /tmp del contenedor lanzando el comando *"touch /tmp/mi_fichero2.txt" a través de docker exec* y cópialo al HOME de local desde dentro del contenedor.

P6. Gestión de Volúmenes de Datos
Evidencias

- Al final de esta actividad práctica deberás de tener el entorno **Docker-CE** con una serie de contenedores creados.
- Verifica que están todos los contenedores con docker ps -a:

- Para el portafolio, hay que recoger las siguientes evidencias:
  - Captura de pantalla del terminal con la salida del comando anterior donde aparezca tu usuario.
  - Texto donde indique la tarea y el comando o comandos utilizados.

Contenidos

Grid y Computación de Altas Prestaciones
GRyCAP

P5. Gestión Interactiva de Contenedores.

P6. Gestión de Volúmenes de Datos.

P7. Creación Manual de Imágenes.

PA3. Solver Python Suma Vectores.

PA4. Solver Octave Ecuaciones Lineales.

P7. Creación Manual de Imágenes
Comandos Básicos Docker

Grid y Computación de Altas Prestaciones
GRyCAP

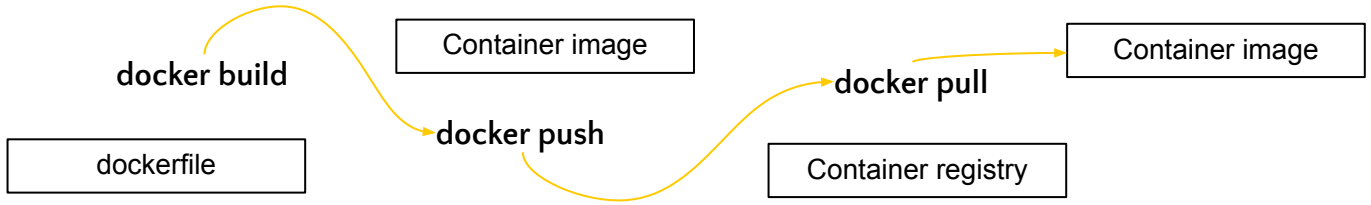
- El objetivo de esta práctica:
  - entender los **comandos** fundamentales de **Docker** para la creación manual de imágenes de contenedores .
  - utilizar los comandos de Docker para la creación de imágenes docker.

Uso

- Listado imagenes (docker images)
- Descarga desde Docker Hub (docker pull <imagen>)
- Eliminar imagen (docker rmi)
- Consulta listado (docker ps)
- Consulta Contenedor (docker Inspect)
- Crear Contenedor (docker create)
- Activación (docker start)
- Parada (docker stop)
- Ejecución (docker Run)
- Borrado (docker rm)
- Consulta puertos (docker port)
- ejecución (docker attach)
- Ejecución (docker exec)
- Volúmenes (docker cp)

Creación Imágenes

- Creación del contenedor
  - Vía creación del Dockerfile
    - Creación de la imagen (docker build).
  - Vía actualización
    - Descarga (docker pull) y actualización.
    - docker commit.
- Publicación (docker push)



P7. Creación Manual de Imágenes
Docker commit

Grid y Computación de Altas Prestaciones
GRyCAP

- **docker commit** permite generar una nueva imagen de un contenedor a partir de un contenedor existente.
- Es útil cuando lanzamos un contenedor interactivo y a través del pseudoterminal podemos instalar paquetes, aplicaciones o servicios específicos dentro del contenedor. Cuando el contenedor está ya actualizado, podemos generar una imagen nueva a partir de dicho contenedor, la cual contendrá ya todos los cambios.

docker commit <etiqueta_contenedor> <etiqueta_nueva_imagen>

- **Parámetros:**

<etiqueta_contenedor> → Etiqueta del contenedor a utilizar para generar la nueva imagen.

<etiqueta_nueva_imagen:tags> → Etiqueta que se va a asignar a la nueva imagen basada en el contenedor.

P7. Creación Manual de Imágenes
Docker commit

- **Tarea 7.1:** Entra en el contenedor mi_con3_int abriendo su pseudoterminal (**docker attach**). Instala python en el contenedor ejecutando desde el pseudoterminal los siguientes comandos:
  - *apt-get update*
  - *apt-get install python*

  A continuación sal del pseudoterminal y crea una imagen que se llame *ubuntu:python.*

P7. Creación Manual de Imágenes
Evidencias

- Al final de esta actividad práctica deberás de tener el entorno **Docker-CE** con una serie de contenedores creados.
- Verifica que están todos los contenedores con docker ps -a:

- Para el portafolio, hay que recoger las siguientes evidencias:
  - Captura de pantalla del terminal con la salida del comando anterior donde aparezca tu usuario.
  - Texto donde indique la tarea y el comando o comandos utilizados.

Contenidos

Grid y Computación de Altas Prestaciones
GRyCAP

P5. Gestión Interactiva de Contenedores.

P6. Gestión de Volúmenes de Datos.

P7. Creación Munual de Imágenes.

PA3. Solver Python Suma Vectores.

PA4. Solver Octave Ecuaciones Lineales.

PA3. Solver Python Suma Vectores

Grid y Computación de Altas Prestaciones GRyCAP

- Tienes que crear un contenedor (llámalo solver_suma_vec) que pueda sumar vectores mediante un programa python que utilicen la librería numpy.
- El programa python es el siguiente ("suma_vectores.py"):

```
import numpy as np
import sys

def sumar_vectores(vector1, vector2):
    v1 = np.array(vector1, dtype=float)
    v2 = np.array(vector2, dtype=float)

    if v1.shape != v2.shape:
        raise ValueError("Los vectores deben tener la misma dimensión")

    return v1 + v2

if __name__ == "__main__":
    # Verificar que se recibieron exactamente 2 argumentos
    if len(sys.argv) != 3:
        print("Uso: python programa.py '1,2,3' '4,5,6'")
        sys.exit(1)

    # Convertir los argumentos en listas de números
    vector1 = [float(x) for x in sys.argv[1].split(',')]
    vector2 = [float(x) for x in sys.argv[2].split(',')]

    try:
        resultado = sumar_vectores(vector1, vector2)
        print("Suma de vectores:", resultado)
    except ValueError as e:
        print("Error:", e)
```

- Se lanzaría desde un terminal de la siguiente manera:

```
python suma_vectores.py "1,2,3" "4,5,6"
```

PA3. Solver Python Suma Vectores

- Para ello sigue los siguientes pasos:
  - Crea un contenedor interactivo con pseudoterminal a partir de una imagen ubuntu:focal (Solo puedes usar esta imagen).
  - Instala python y la librería numpy en el contenedor.
    - apt-get update
    - apt-get install python
    - apt-get install python-numpy
  - Copia el programa que suma vectores en python en $HOME dentro del contenedor.
  - Crea una nueva imagen llamada *mi_img_suma_vectores:latest* a partir del contenedor.
  - Crea un nuevo contenedor con la nueva imagen que permita el lanzamiento de suma de dos vectores.

- **La evidencia va a ser el propio contenedor creado y una captura de pantalla de su ejecución.**

Contenidos

Grid y Computación de Altas Prestaciones
GRyCAP

P5. Gestión Interactiva de Contenedores.

P6. Gestión de Volúmenes de Datos.

P7. Creación Manual de Imágenes.

PA3. Solver Python Suma Vectores.

PA4. Solver Octave Ecuaciones Lineales.

PA4. Solver Octave Ecuaciones Lineales

- Tienes que resolver un sistema de ecuaciones lineales a través de octave.

$$Ax = b \; --> \; x = b/A$$

- La entrada deben ser de 3 ficheros que tendrán que estar en el host y montados dentro del contenedor en el directorio /octave. Estos son:
  - `octave/procesa.m`
  - `octave/A`
  - `octave/b`
- El contenido de procesa.m es:

```
cd /octave
load A;
load b;
x=A\b;
save x;
err=norm(x-ones(size(x)))
```

- Cambiamos al directorio /octave donde está la matriz A y el vector b.
- Cargamos la matriz de coeficientes A
- Cargamos el vector de términos independientes b
- Resolvemos el sistema
- Guardamos la solución del sistema
- Calculamos el error cometido comparando con la solución esperada.

PA4. Solver Octave Ecuaciones Lineales

Grid y Computación de Altas Prestaciones GRyCAP

- Descarga los fichero de test A, B y procesa.m de poliformaT (Carpeta PA4), guárdalos en una carpeta del host y hazlos accesible desde dentro del contenedor para que se pueda ejecutar.

- Prepara en el registro local una imagen de contenedor con octave. NO puedes acceder a docker Hub, si no tienes que preparar una imagen que contenga octave basada en ubuntu:focal a mano a través de un terminal interactivo.

  - **Para** instalar manualmente octave sobre un ubuntu puedes utilizar el comando *`apt-get install -y octave`*. Al crear la nueva imagen debes de llamarla "*mi_img_octave:latest*".

- Una vez creada la imagen correctamente, crea y lánza un contenedor para que ejecute el programa que resuelve el sistema de ecuaciones linea ("*octave /octave/procesa.m*"
- **La evidencia va a ser el propio contenedor creado y una captura de pantalla de su ejecución.**