 Docker Run

|  |
| --- |
| docker run -it --name=cont1 ubuntu /bin/bash |

* Crea un contenedor con la imagen “ubuntu” (al no especificar, toma versión “latest”), le establece un nombre “cont1” y lanza en modo interactivo una shell “bash”.

|  |
| --- |
| docker run -d -p 1200:80 nginx |

* Crea un contenedor con la versión “latest” de la imagen “nginx” y lo lanza en “background”, exponiendo el puerto 80 del contenedor en el puerto 1200 de la máquina anfitrión.

|  |
| --- |
| docker run -it -e MENSAJE=HOLA ubuntu:14.04 bash |

* Crea un contenedor con la imagen “ubuntu”, versión “14.04” y establece la variable de entorno “MENSAJE”.

 Docker ps

|  |
| --- |
| docker ps |

* Muestra información de los contenedores en ejecución.

|  |
| --- |
| docker ps -a |

* Muestra información de todos los contenedores, tanto parados como en ejecución.

 Docker Start/Stop/Restart

|  |
| --- |
| docker start micontenedor |

* Arranca el contenedor con nombre “mi contenedor”.

|  |
| --- |
| docker start -ai micontenedor |

* Arranca el contenedor con nombre “mi contenedor”, enlazando el comando ejecutado al arranque a la entrada y salida estándar de la terminal del anfitrión.

 Docker Exec

|  |
| --- |
| docker exec -it -e FICHERO=prueba cont bash |

* Lanza en el contenedor “cont” (que debe estar arrancado) el comando “bash”, estableciendo la variable de entorno “FICHERO” y enlazando la ejecución de forma interactiva a la entrada y salida estándar del anfitrión.

|  |
| --- |
| docker exec -d cont touch /tmp/prueba |

* Lanza en el contenedor “cont” (que debe estar arrancado) el comando “touch /tmp/prueba”. Este comando se ejecuta en segundo plano, generando el fichero “/tmp/prueba”.

 Docker attach

|  |
| --- |
| docker attach idcontainer |

* Enlaza nuestra terminal la entrada/salida de nuestra al proceso en segundo plano del contenedor “idcontainer”.

 Docker logs

|  |
| --- |
| docker logs -n 10 idcontainer |

* Muestra las 10 últimas líneas de la salida estandar producida por el proceso en ejecución en el contendor.

 Docker cp

|  |
| --- |
| docker cp idcontainer:/tmp/prueba ./ |

* Copia el fichero “/tmp/prueba” del contenedor “idcontainer” al directorio actual del anfitrión.

|  |
| --- |
| docker cp ./miFichero idcontainer:/tmp |

* Copia el fichero “miFichero” del directorio actual del anfitrión a la carpeta “/tmp” del contenedor.

 Gestión de imágenes

|  |
| --- |
| docker images |

* Información de imágenes locales disponibles.

|  |
| --- |
| docker search ubuntu |

* Busca la imagen “ubuntu” en el repositorio remoto (por defecto Docker Hub).

|  |
| --- |
| docker pull alpine |

* Descarga localmente imagen “alpine”.

|  |
| --- |
| docker history alpine |

* Muestra la historia de creación de la imagen “alpine”.

|  |
| --- |
| docker rmi ubuntu:14.04 |

* Elimina localmente la imagen “ubuntu” con tag “14.04”.

|  |
| --- |
| docker rmi $(docker images -q) |

* Borra toda imagen local que no esté siendo usada por un contenedor.

|  |
| --- |
| docker rm IDCONTENEDOR |

* Borra un contenedor con IDCONTENEDOR.

|  |
| --- |
| docker stop $(docker ps -a -q) |

* Para todos los contenedores del sistema.

|  |
| --- |
| docker rm $docker ps -a -q) |

* Borra todos los contenedores parados del sistema.

|  |
| --- |
| docker system prune -a |

* Borra todas las imágenes y contenedores parados del sistema.

 Creación de imágenes a partir de contenedores

|  |
| --- |
| docker commit -m “comentario” IDCONTENEDOR usuario/imagen:version |

* Hace commit de un contenedor existente a una imagen local.

|  |
| --- |
| docker save -o copiaSeguridad.tar imagenA |

* Guarda una copia de seguridad de una imagen en fichero “.tar”.

|  |
| --- |
| docker load -i copiaSeguridad.tar |

* Restaura una copia de seguridad de una imagen en fichero “.tar”.

 Docker Hub

|  |
| --- |
| docker login |

* Permite introducir credenciales del registro (por defecto “Docker Hub”).

|  |
| --- |
| docker push usuario/imagen:version |

* Permite subir al repositorio una imagen mediante “push”.

 Ejemplo de Dockerfile

|  |
| --- |
| **FROM** alpine **MAINTAINER** Sergi <sergi.profesor@gmail.com> *#Actualizamos e instalamos paquetes con APK para Alpine* **RUN** apk update && apk add apache2 php php-apache2 openrc tar *#Copiamos script para lanzar Apache 2* **ADD** ./start.sh /start.sh *#Descargamos un ejemplo de <?php phpinfo(); ?> por enseñar como bajar algo de Internet* *#Podría haber sido simplemente* *#RUN echo "<?php phpinfo(); ?>" > /var/www/localhost/htdocs/index.php* **ADD** https://gist.githubusercontent.com/SyntaxC4/5648247/raw/94277156638f9c309f2e36e19bff378ba7364907/info.php /var/www/localhost/htdocs/index.php *# Si quisiéramos algo como Wordpress haríamos* *#ADD http://wordpress.org/latest.tar.gz /var/www/localhost/htdocs/wordpress.tar.gz* *#RUN tar xvzf /var/www/localhost/htdocs/wordpress.tar.gz && rm -rf /var/www/localhost/htdocs/wordpress.tar.gz*  *# Usamos usuario y grupo www-data. El grupo lo crea Apache, pero si quisiéramos crear grupo* *# Grupo www-data RUN set -x && addgroup -g 82 -S www-data* *# Creamos usuario www-data y lo añadimos a ese grupo* **RUN** adduser -u 82 -D -S -G www-data www-data *# Hacemos todos los ficheros de /var/www propiedad de www-data* *# Y damos permisos s esos ficheros y a start.sh* **RUN** chown -R www-data:www-data /var/www/ && chmod -R 775 /var/www/ && chmod 755 /start.sh *#Indicamos puerto a exponer (para otros contenedores) 80* **EXPOSE** 80 *#Comando lanzado por defecto al instalar el contendor* **CMD** /start.sh |

* Ejemplo de fichero “Dockerfile”.