Docker está formado fundamentalmente por tres componentes:

- Docker Engine
- Docker Client
- Docker Registry

Docker Engine o Demonio Docker:

Es un demonio que corre sobre cualquier distribución de Linux y que expone una API externa para la gestión de imágenes y contenedores (y otras entidades que se van añadiendo en sucesivas distribuciones de docker como volúmenes o redes virtuales). Podemos destacar entre sus funciones principales:

- Creación de imágenes docker.
- Publicación de imágenes en un Docker Registry o Registro de Docker (otro componente Docker que se explicará a continuación).
- Descarga de imágenes desde un Registro de Docker
- Ejecución de contenedores usando imágenes locales.

Otra función fundamental del Docker Engine es la gestión de los contenedores en ejecución, permitiendo parar su ejecución, rearrancarla, ver sus logs o sus estadísticas de uso de recursos.

Docker Client o Cliente Docker

Es cualquier herramienta que hace uso de la api remota del Docker Engine, pero suele hacer referencia al comando *docker* que hace las veces de herramienta de línea de comandos (cli) para gestionar un Docker Engine. La *cli* de docker se puede configurar para hablar con un Docker Engine local o remoto, permitiendo gestionar tanto nuestro entorno de desarrollo local, como nuestros servidores de producción. Los comandos de docker más comunes son:

- *docker info*: da información acerca de la cantidad de contenedores e imágenes que está gestionando la máquina actual, así como los plugins actualmente instalados.

- *docker images*: lista información de las imágenes que se encuentran disponibles en la máquina (nombre, id, espacio que ocupa, el tiempo que transcurrió desde que fue creada).

- *docker build*: crea una imagen desde el fichero `Dockerfile` del directorio actual.

- *docker pull <imagen>:<version>*: descarga en la máquina actual la versión de la imagen indicada. En caso de no indicar la versión descarga todas las que estén disponibles.

- *docker push <imagen>:<version>*: sube la versión de la imagen indicada a un Registro de Docker, permitiendo su distribución a otras máquinas.

- *docker rmi <imagen>:<version>*: elimina una imagen de la máquina actual.

- *docker run <imagen>:<version>*: crea un contenedor a partir de una imagen. Este comando permite multitud de parámetros, que son actualizados para cada versión del Docker Engine, por lo que para su documentación lo mejor es hacer referencia a la página oficial.

- *docker ps*: muestra los contenedores que están corriendo en la máquina. Con el flag `-a` muestra también los contenedores que están parados.

- *docker inspect contenedor*: muestra información detallada de un contenedor en formato json. Se puede acceder a un campo particular con el comando `docker inspect -f '{{.Name}}' contenedor` .

- *docker stop contenedor*: para la ejecución de un contenedor.

- *docker start contenedor*: reanuda la ejecución de un contenedor.

- *docker rm contenedor*: elimina un contenedor. Para borrar todos los contenedores de una máquina se puede ejecutar el comando `docker rm -fv $(docker ps -aq)` .

- *docker logs contenedor*: muestra los logs de un contenedor.

- *docker stats contenedor*: muestra las estadísticas de ejecución de un contenedor, como son la memoria utilizada, la CPU, el disco…

- *docker exec contenedor comando*: ejecuta un comando en un contenedor. Útil para depurar contenedores en ejecución con las opciones `docker exec -it contenedor bash` .

- *docker volume ls*: lista los volúmenes existentes en la máquina. Para un listado completo de los comandos relacionados con volúmenes

ejecuta `docker volume --help`.

- *docker network ls*: lista las redes existentes en la máquina. Para un listado completo de los comandos relacionados con redes ejecuta `docker network --help`.

Docker Registry o Registro Docker

El Registro es otro componente de Docker que suele correr en un servidor independiente y donde se publican las imágenes que generan los Docker Engine de tal manera que estén disponibles para su utilización por cualquier otra máquina. Es un componente fundamental dentro de la arquitectura de Docker ya que permite distribuir nuestras aplicaciones. El Registro de Docker es un proyecto *open source*que puede ser instalado gratuitamente en cualquier servidor, pero Docker ofrece *Dockerhub*, un sistema *SaaS* de pago donde puedes subir tus propias imágenes, acceder a imágenes públicas de otros usuarios, e incluso a imágenes oficiales de las principales aplicaciones como son: MySQL, MongoDB, RabbitMQ, Redis, etc.

El registro de Docker funciona de una manera muy parecida a *git* (de la misma manera que Dockerhub y us métodos de pago funcionan de una manera muy parecida a *Github*). Cada imagen, también conocida como repositorio, es una sucesión de capas. Es decir, cada vez que hacemos un build en local de nuestra imagen, el Registro de Docker sólo almacena el **diff** respecto de la versión anterior, haciendo mucho más eficiente el proceso de creación y distribución de imágenes.