Docker

**Whether it be from your development machine to a remote server for production, or packaging everything for use elsewhere, it is always a challenge when it comes to porting your application stack together with its dependencies and getting it to run without hiccups. In fact, the challenge is immense and solutions so far have not really proved successful for the masses.**

**In a nutshell, docker as a *project* offers you the complete set of higher-level tools to carry everything that forms an application across systems and machines - virtual or physical - and brings along loads more of great benefits with it.**

**Docker achieves its robust application (and therefore, process and resource) containment via *Linux Containers* (e.g. namespaces and other kernel features). Its further capabilities come from a project's own parts and components, which extract all the complexity of working with *lower-level* linux tools/APIs used for system and application management with regards to securely containing processes.**

### Main Docker Parts

1. **docker daemon: used to manage docker (LXC) containers on the host it runs**
2. **docker CLI: used to command and communicate with the docker daemon**
3. **docker image index: a repository (public or private) for docker images**

### Main Docker Elements

1. **docker containers: directories containing everything-your-application**
2. **docker images: snapshots of containers or base OS (e.g. Ubuntu) images**
3. **Dockerfiles: scripts automating the building process of images**

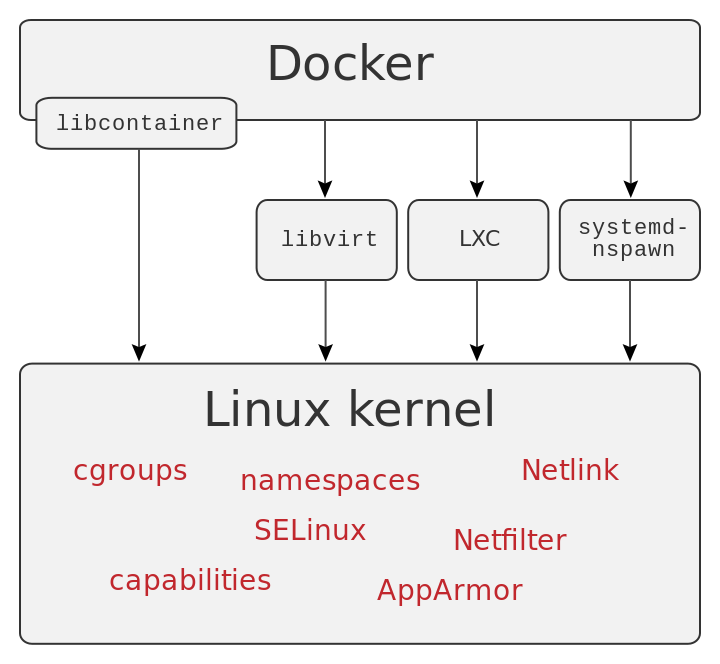
**Docker is an**[**open-source**](https://en.wikipedia.org/wiki/Open-source)**project that automates the deployment of**[**applications**](https://en.wikipedia.org/wiki/Application_software)**inside**[**software containers**](https://en.wikipedia.org/wiki/Software_container).

**Definition:**

**Docker is a tool that can package an application and its dependencies in a virtual container that can run on any Linux server. This helps enable flexibility and portability on where the application can run, whether**[**on premises**](https://en.wikipedia.org/wiki/On-premises_software)**,**[**public cloud**](https://en.wikipedia.org/wiki/Public_cloud)**,**[**private cloud**](https://en.wikipedia.org/wiki/Private_cloud)**,**[**bare metal**](https://en.wikipedia.org/wiki/Bare-metal_server)**,**

**Building on top of facilities provided by the**[**Linux kernel**](https://en.wikipedia.org/wiki/Linux_kernel)**(primarily cgroups and namespaces), a Docker container, unlike a virtual machine, does not require or include a separate operating system.**[**[13]**](https://en.wikipedia.org/wiki/Docker_(software)#cite_note-Linux-14)**Instead, it relies on the kernel's functionality and uses resource isolation (CPU, memory, block I/O, network, etc.) and**[**separate namespaces**](https://en.wikipedia.org/wiki/Namespace_isolation)**to isolate the application's view of the operating system.**

**Docker accesses the Linux kernel's virtualization features either directly using the libcontainer library, which is available as of Docker 0.9, or indirectly via [libvirt](https://en.wikipedia.org/wiki/Libvirt" \o "Libvirt),**[**LXC**](https://en.wikipedia.org/wiki/LXC)**(Linux Containers) or [systemd-nspawn](https://en.wikipedia.org/wiki/Systemd-nspawn" \o "Systemd-nspawn).**



**Because Docker containers are so lightweight, a single server or virtual machine can run several containers simultaneously.**

This type of Linux containers are proven to be fast, portable and secure. The processes that run in a Docker container are always isolated from the main host, preventing outside tampering.

**Lets Start With Docker :**

* **Systemctl start docker.service :Start Docker Service**
* **Systemctl status docker.service : Status(Active/Disabled)**
* **Systemctl enable docker.service : Cretaes Symlink**

**Created symlink from /etc/systemd/system/multiuser.target.wants/docker.service to /usr/lib/systemd/system/docker.service.**

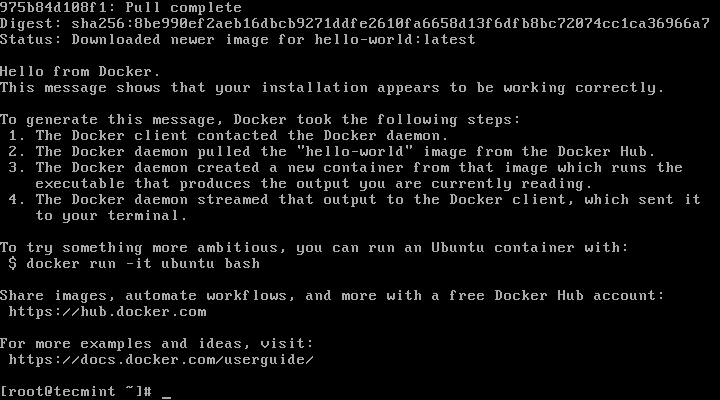
**Docker pulls images from : docker.io/library**

**Trying to pull repository docker.io/library/hello-world ...**

**latest: Pulling from docker.io/library/hello-world**

Finally, run a container test image to verify if Docker works properly, by issuing the following command:

# docker run hello-world



**Pull image from Docker Library**

docker pull centos

**To list all the available Docker images on your host issue the following command:**

# docker images

If you don’t need a Docker image anymore and you want to remove it from the host issue the following command:

# docker rmi centos

**To create a container in Docker**

docker run -it -d <Image Id>

**To stop the container**

Docker stop <ImageID>

**To start the container**

Docker start <ImageID>

**To run a command in Container**

# docker run [local image] [command to run into container]

**To enter into Container**

**docker exec -it <ContainerID> bash**

**To remove Docker Container**

**docker rm 740ad004d045**