Contents

[Docker 2](#_Toc114216649)

[WHAT IS DOCKER? 2](#_Toc114216650)

[WHY DOCKER? 2](#_Toc114216651)

[USING DOCKER CLIENT? 2](#_Toc114216652)

[WHAT IS CONTAINER? 3](#_Toc114216653)

[RELATION BETWEEN CONTAINER AND IMAGE 4](#_Toc114216654)

[LINUX KERNEL 5](#_Toc114216655)

[DOCKER COMMANDS 5](#_Toc114216656)

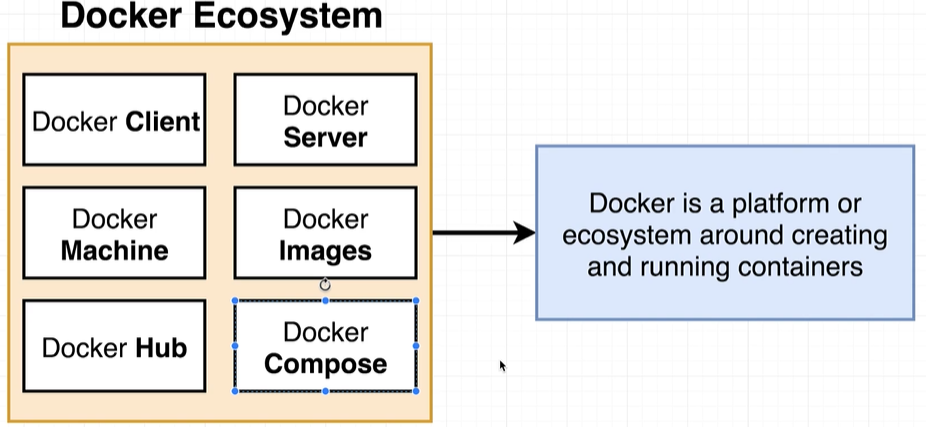
[DOCKER RUN 5](#_Toc114216657)

[LIST ALL RUNNING CONTAINER 6](#_Toc114216658)

[CONTAINER LIFECYCLE 6](#_Toc114216659)

# Docker

## WHAT IS DOCKER?

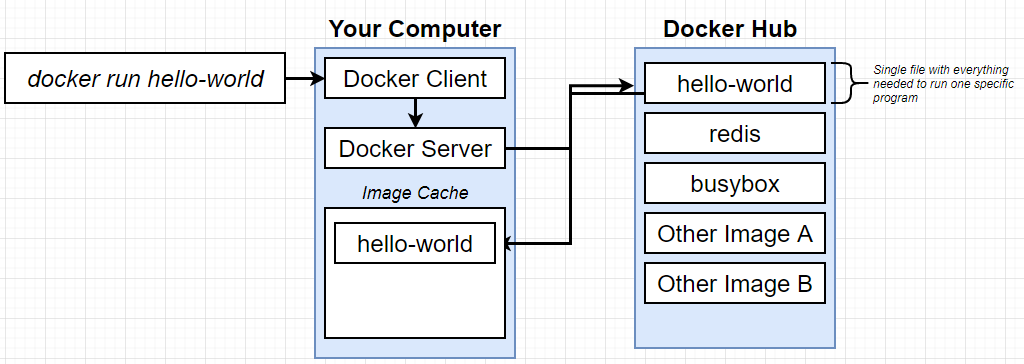


## WHY DOCKER?

* Docker makes it easy to install a software without worrying about set up or dependencies.

## USING DOCKER CLIENT?

|  |  |
| --- | --- |
| **DOCKER VERSION** | docker version |
| **DOWNLOADING AND RUNNING AN IMAGE IN A CONTAINER** | docker run *<image\_name>*  ***docker run hello-world*** |



* The purpose “docker run” command is to run the image in container.
* After we execute a “docker run” command in docker client or docker CLI, it pass the instruction to the Docker Server
* The docker first checks for the image in the local machine and if not found – it tries to download the image from a docker repo called “**Docker Hub”**
* Once it is downloaded from the docker hub – it stores it in image cache (in the local machine)- it avoids the re-downloading of the same image

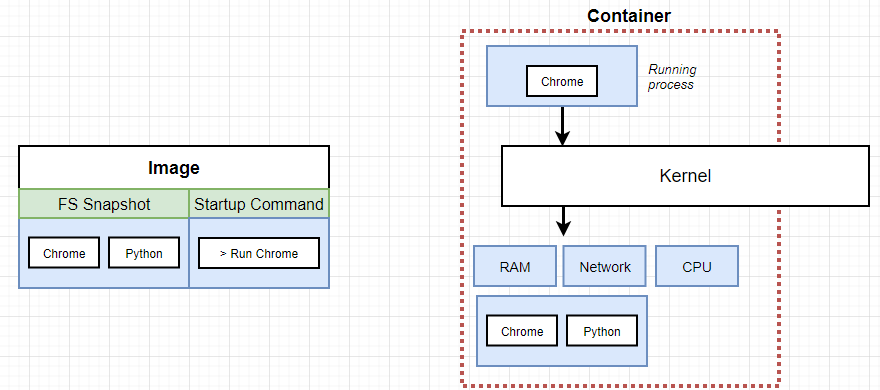
|  |  |
| --- | --- |
| * Once the image is downloaded – It starts a dedicated process called container and start executing the program.   Note: Container is an instance of the image |  |

## WHAT IS CONTAINER?

|  |  |
| --- | --- |
| * In operating system – if an application needs to talk to the physical hardware devices via Kernel. * These apps make system calls to kernel which in turn interact with hardware.   ***Kernel is a running program which governs access to the running application and the physical hardware. Means it decides/ governs – which application can access which hardware device.*** |  |
| * To understand container – Lets consider a hypothetical analogy. * Let’s say, we have Python v2 installed in our machine and we have two apps – Chrome & NodeJS – which needs Python V2 and Node V3. * Due to incompatible- Python version Node JS will not work as it needs Python v3 |  |
|  |  |
| * The work around for this problem – is to create segments in the hard-drive and install the respective python version is the segments * Now based on the system call kernel will redirect it to respective segment. * This process of segregating the hardware and software resources is called “NameSpacing” | |
| * With name-spacing we can be able to isolate a resource per process or group of process. * On the other hand – Control Group is the amount of resource a particular process can use. * The entire vertical in the diagram – of the running process and the segment in the Hard disk is called container. * Hence container is a process which has a set of resources assigned to it.   **CONTAINER ON HIGH LEVEL** |  |
|  |

### RELATION BETWEEN CONTAINER AND IMAGE

* Image is basically – is a snapshot of the file system(like folder)
* The image has a start up command.
* When we run the container – the image gest copied to the container and the start-up command starts an instance of an image. It will have very specific group of resources and hardware.



## LINUX KERNEL

* As the name-spacing and control group feature are only available in Linus OS
* In the window/Mac OS – When we install docker - which in turn install Linux Virtual Machine, which has a Linux Kernel – which facilitate name-spacing and control group. The Linux Kernel host the running process i.e. containers.

|  |  |
| --- | --- |
|  | ***The docker version commnd gives the detail of Linux VM*** |

## DOCKER COMMANDS

### DOCKER RUN

|  |  |
| --- | --- |
| * Docker run command creates and run a container from an image. |  |

#### OVERRIDING DEFAULT COMMANDS

* “docker run” command by default create+ start the container. We can to override the default behavior of these commands. e.g.

|  |  |
| --- | --- |
| ***docker run busybox echo hi there*** | This will print “hi there” message in the command |
| ***docker run busybox echo ls*** | This will list the file system inside the dedicated container for this image |

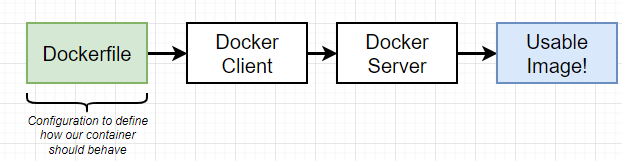
### LIST ALL RUNNING CONTAINER

|  |  |
| --- | --- |
| **docker ps** | This will list all the running containers. |
| **docker ps --all** | To see all the containers – both running and exited containers |
| **docker start <CONTAINER\_ID>** | Starts a specific container. |

## CONTAINER LIFECYCLE

|  |  |
| --- | --- |
|  | * docker run – command is internally create and start the container.   **WHAT IS DOCKER CREATE ?**  **WHAT IS DOCKER START ?** |
|  |  |

## CREATING A DOCKER IMAGE



To create a docker image:

1. We create a Docker file (plain text file) – which will have command / configuration. The configuration defined what different program it contains and what it does when it starts as a container.
2. Once the docker file is created. It is supplied to docker client (docker cli) which in-turn provide the file to docker server. The docker server finally does the heavy lifting to create Docker image.

### CREATING A DOCKER FILE

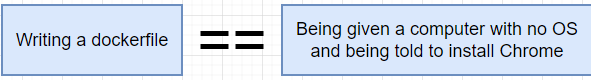
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **EXAMPLE: CREATE AN IMAGE THAT RUNS REDIS SERVER**    After creating a docker file .   |  |  | | --- | --- | | TO BUILD THE DOCKER IMAGE | ***docker build .*** | | TO RUN THE DOCKER IMAGE | docker run <image\_name> | |
| **DOCKER FILE** | |
| RUNNING AND BUILDING A DOCKER IMAGE | |

#### DOCKERFILE IN DETAILS

|  |  |
| --- | --- |
|  | * Every line in a docker file is called instruction – which instruct to some basic preparation step on the custom  1. **FROM:** This is used to specify the docker image we want to use as a base. 2. **RUN:** Command while preparing the custom image 3. **CMD:** What should be executed when the image starts as a container. |

### BASE IMAGE

* To understand the concept of base image – lets understand this with an analogy.
* The analogy can be writing a docker file is equivalent to installing the Chrome in a computer with no OS.



|  |  |
| --- | --- |
|  | * Specifying the base image is equivalent to installing an OS in a computer.   ***apk add --update redis***  This is not a docker command- “**apk**” is a package manager which is built in the alpine image – which can download and install redis.  “This is how base image helps in preparing the custom images” |

## DOCKER BUILD PROCESS

