Dockers 101 – Series 1 of N – Introduction

PUBLISHED ON *April 1, 2018April 3, 2018 by Mohd Naeem*

What is Docker?

- As per the docker documentation –
  - **Docker:**
    - is a platform to **develop**, **deploy** and **run** application using a concept called as containerization.
  - **Containerization:**
    - uses images and containers which Scalable, Portable, Flexible, Lightweight, Stackable, Interchangeable making software development, deployment(CI-CD i.e. Continuous Integration and Continuous Delivery) and execution of application seamless
  - **Images:**
    - are executable packages which contains the everything to run an application – the code, run-time, libraries or dependencies, configuration files, environmental variables.
  - **Containers:**
    - are the run-time instances of the image which runs in a docker environment.
    - It means that containers have a state and they run in memory while images are the blueprints. The analogy can be similar to a class and its objects.
    - If use the **ps** command – docker ps – you can see a container running as a process on the docker host.

Docker Vs Virtual Machines(VMs)

- Please see this image snapshot to understand difference between docker and VM:
- **Virtual Machines(VMs):**
  - They use a full blown VM OS which uses resources more than it might need as each VM will have a definite defined memory, hard disk, network etc
  - It uses a technology Hypervisor to abstract or simulate the host machines resources as memory, storage, CPU etc, that is why they VMs are slower to setup.
- **Dockers:**
  - The container share the resources assigned to docker on the OS, so no container will use more resources than needed.
  - Its lightweight as it does not use a fully blown OS.
  - Faster to setup
- Below is a snapshot clarify

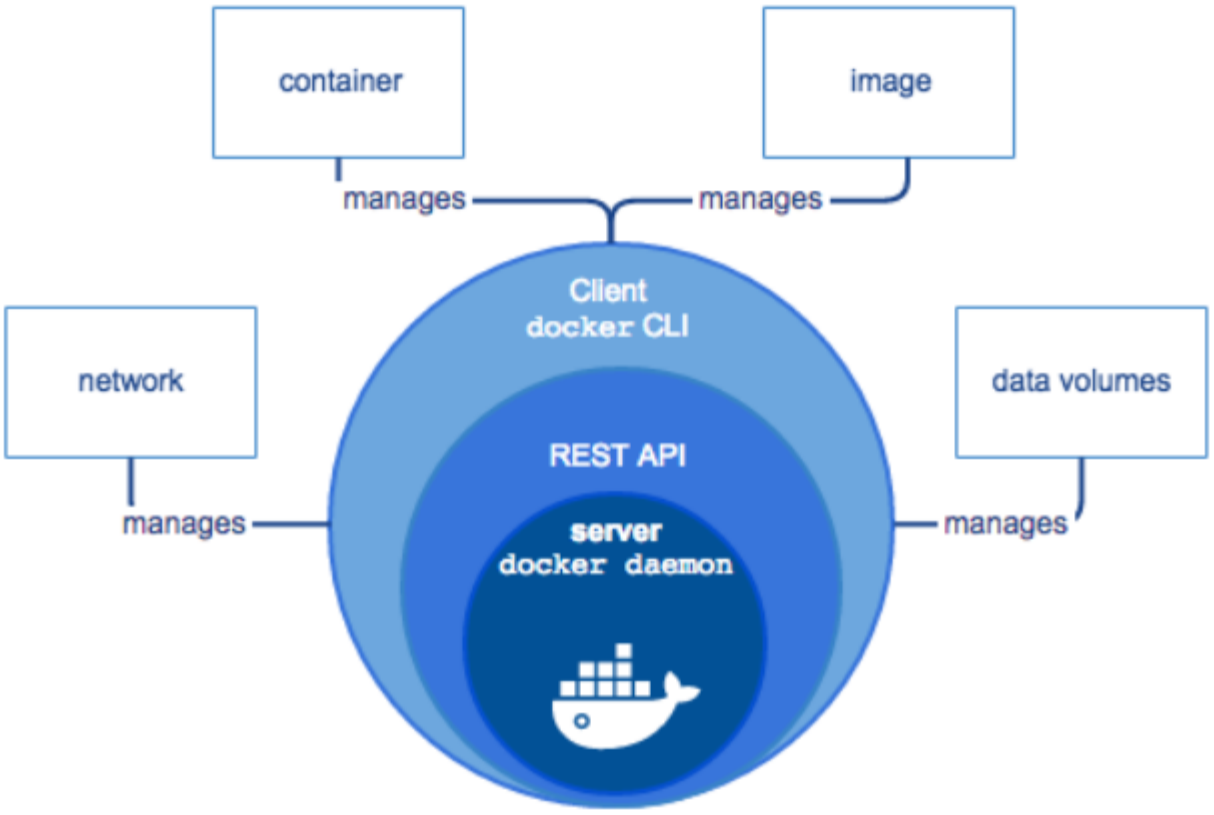
What dockers can do for us?

- **Scanario 1** – Imagine you have to deploy a **highly scalable** and **highly available** website behind a **load balancer** with a fleet of multiple web servers , applications servers and database server. Imagine you have do the following
  - **Manually Install** all that is required:

        - install the guest OS first e.g Linux, then Apache the web server, MySQL the database server, PHP or Python etc to process the application code.
        - Imagine if you have to manually to do this on 100s or 1000s of server????????
    - **Automate** using **script**
        - write shell scripts to install Linux, Apache, MySQL, PHP or Python etc
        - copy these scripts to the specific servers to execute on boot up.
        - Imagine if you have to use scripts to do this on 100s or 1000s of server????????
    - Use an **Orchestrator** tool like D**ocker**, K**ubernetes** etc to do this for using a orchestration file. So now installing software, dependencies to run an application is not a tedious error prone activity but it has been now limited to the orchestration file. In Docker terms its is called a Dockerfile.
- **Scanario 2** – Now imagine you are constantly doing lot of changes and you have constantly deploy it to servers from scratch, you will have to download, install and configure everything from Linux, Apache, MySQL, PHP/Python. How will you manage if your company can't afford large down times also it will take lot of time
    - In Docker world, docker uses images and containers and also uses caching and layered approach.
    - It first searches for an image locally or from cache, if it does not find it then only it downloads it from repository(called a dockerhub)
    - You can first pull the base image first (say for Linux), then build over it as containers where 1 container will hold 1 responsibility e.g a container will have database, another one the Apache, the third one your application code if we think of a 3 tier application.
    - An image downloaded once is cached and can be used by multiple containers
    - You can use orchestration tools like docker-compose, kubernetes to orchesrtate development, deployment and running of your application.

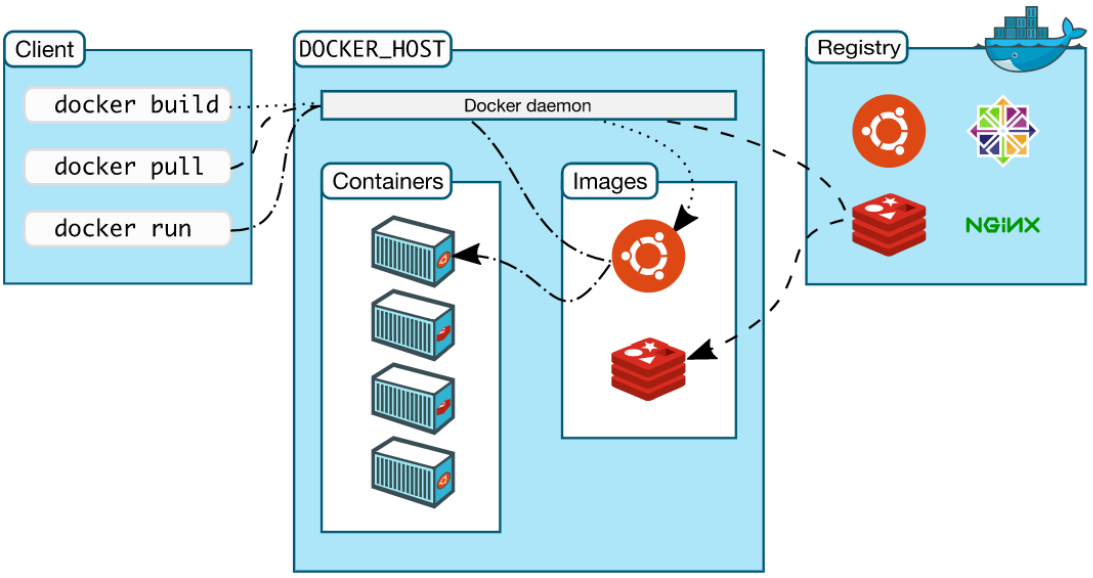
Features and Architecture of Docker:

- **Features**:
    - Immutability – the images are immutable, they don't change unless you make changes and build and deploy another image
    - Disposibility – images are disposable so that you can create another one after disposing curret one
    - SRP(Single Responsibility Principle) – each image holds a single responsibility e.g. in a three tier application, it would be ideal to have 3 images to represent the front end , business layer and database layer
- **Architecture**:
    - docker is essentially a client server architecture called as **Docker Engine**, where client and server can run on same machine or different machines
    - Below is the snapshot of a **Docker Engine**:



- *Source: https://docs.docker.com/engine/docker-overview/#docker-engine (https://docs.docker.com/engine/docker-overview/#docker-engine)*

- **Docker client** –
  - the client interacts with server to help execute the docker commands exposed by docker APIs
  - E.g docker pull, docker pull, docker run etc
- **Docker Daemon**(server) –
  - manages the docker objects – images and containers
- **Docker registry**(placeholder for repository)
  - holds the images.
  - Docker Hub and Docker Cloud are public docker registries
- As per docker documentation website – the below is the architecture of docker



Source: https://docs.docker.com/engine/docker-overview/#docker-architecture (https://docs.docker.com/engine/docker-overview/#docker-architecture)

Docker Installation:

- Docker supports a community edition(**Docker CE**) and an enterprise edition(**Docker EE**) and supports platforms like Linux, Mac, Windows
- **Steps** for **Docker Installation**:
  - Login to your host(I am using CentOS Linux server as my host)
  - Step 1 : **Installation** of **pre-requisite** packages:
    - Docker needs yum-utils, device-mapper-persistent-data and lvm2 as prerequisites.
    - **sudo yum install -y yum-utils device-mapper-persistent-data lvm2**

```
[root@mnaeemsiddiqui3 user]# whoami
root
[root@mnaeemsiddiqui3 user]# pwd
/home/user
[root@mnaeemsiddiqui3 user]# sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

  - Step 2 : **Setting up** Docker yum **repository and install docker**:
    - **sudo yum-config-manager –add-repo https://download.docker.com/linux/centos/docker-ce.repo (https://download.docker.com/linux/centos/docker-ce.repo)**
    - **sudo yum -y install docker-ce**

```
[root@mnaeemsiddiqui3 user]# sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
Loaded plugins: fastestmirror
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to /etc/yum.repos.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
[root@mnaeemsiddiqui3 user]#
[root@mnaeemsiddiqui3 user]# sudo yum install docker-ce
```

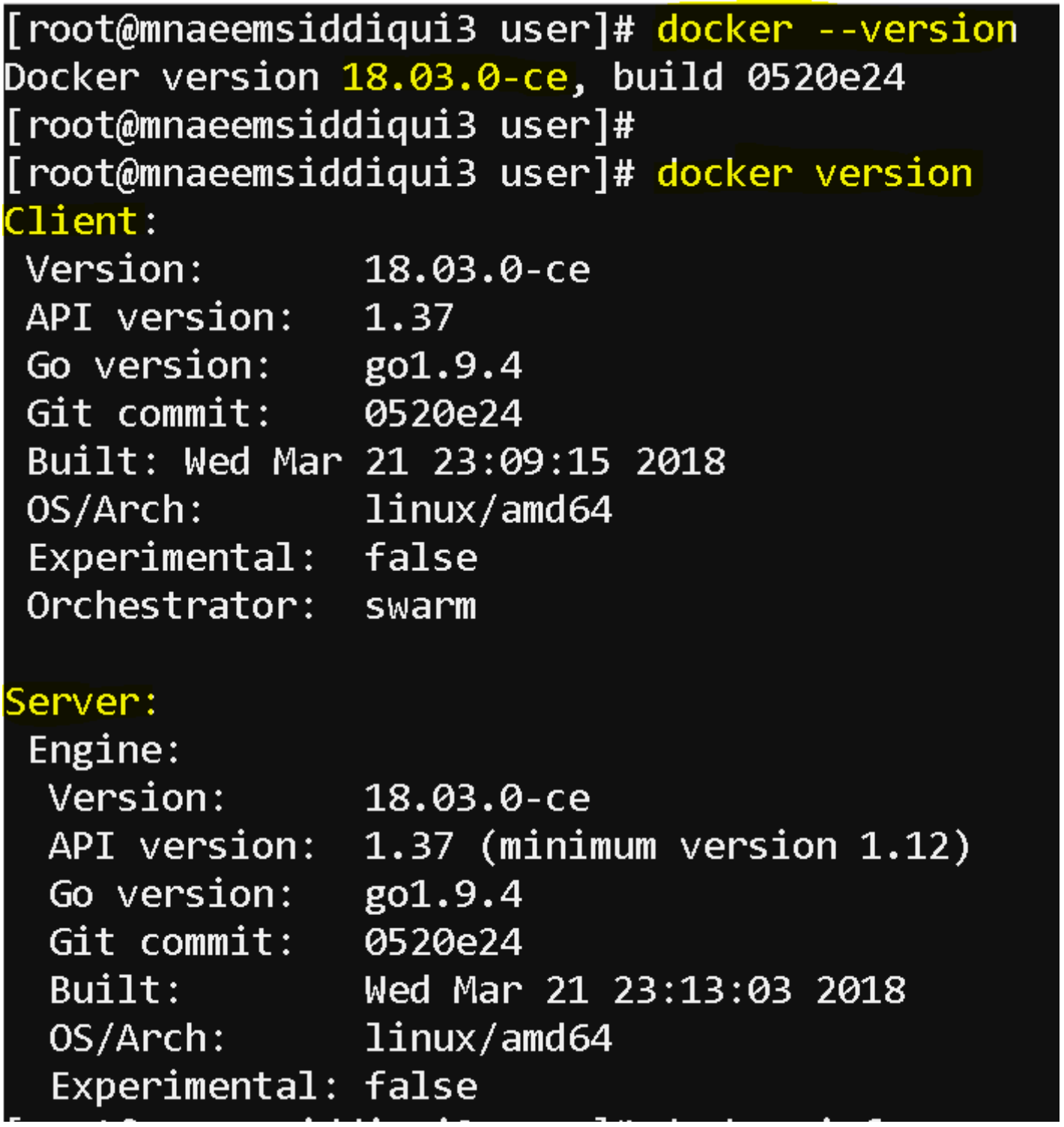
  - Step 3 : **Start** Docker :
    - **sudo systemctl start docker**
    - **sudo systemctl enable docker**
  - Step 4 : **Run** Hello world to test docker installation
    - **sudo docker run hello-world**

```
[root@mnaeemsiddiqui3 user]# sudo systemctl start docker && sudo systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.
[root@mnaeemsiddiqui3 user]# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
ca4f61b1923c: Pull complete
Digest: sha256:97ce6fa4b6cdc0790cda65fe7290b74cfebd9fa0c9b8c38e979330d547d22ce1
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/
```

- **Steps** for **Docker Uninstall:**
  - **sudo yum remove docker-ce** # uninstalls docker
  - **sudo rm -rf /var/lib/docker** # removes dicker folder

Docker Commands:

- To check docker version(detailed client server info) : **docker version**
- To check docker info(short info) : **docker –version**



- To check docker info(detailed docker metadata info) : **docker info**

```
[root@mnaeemsiddiqui3 user]# docker info
Containers: 1
 Running: 0
 Paused: 0
 Stopped: 1
Images: 1
Server Version: 18.03.0-ce
Storage Driver: devicemapper
 Pool Name: docker-202:1-75533126-pool
 Pool Blocksize: 65.54kB
 Base Device Size: 10.74GB
 Backing Filesystem: xfs
 Udev Sync Supported: true
 Data file: /dev/loop0
 Metadata file: /dev/loop1
 Data loop file: /var/lib/docker/devicemapper/devicemapper/data
 Metadata loop file: /var/lib/docker/devicemapper/devicemapper/metadata
 Data Space Used: 19.4MB
 Data Space Total: 107.4GB
 Data Space Available: 15.5GB
 Metadata Space Used: 593.9kB
 Metadata Space Total: 2.147GB
 Metadata Space Available: 2.147GB
 Thin Pool Minimum Free Space: 10.74GB
 Deferred Removal Enabled: true
 Deferred Deletion Enabled: true
 Deferred Deleted Device Count: 0
 Library Version: 1.02.140-RHEL7 (2017-05-03)
```

- To **pull an image from Docker Hub**:
  - pull the **latest image** :
    - to pull latest Ubuntu – **docker pull ubuntu**
    - to pull latest CentOS – **docker pull centos**
  - pull the **tagged image**:
    - to pull tagged Ubuntu – **docker pull ubuntu:trusty**
    - to pull tagged CentOS – **docker pull centos:7**
- To **create and run a container**:
  - To **run** the container in **interactive** mode:
    - in interactive mode you are running the container in foreground
    - its kind of sshing(not actually though)
    - use **exit** to exit out of the container
    - **docker run -it ubuntu bash**
    - **docker run -it ubuntu ls**
    - bash , ls are commands for the container

```
[root@mnaeemsiddiqui3 user]# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
22dc81ace0ea: Pull complete
1a8b3c87dba3: Pull complete
91390a1c435a: Pull complete
07844b14977e: Pull complete
b78396653dae: Pull complete
Digest: sha256:e348fbbea0e0a0e73ab0370de151e7800684445c509d46195aef73e090a49bd6
Status: Downloaded newer image for ubuntu:latest
[root@mnaeemsiddiqui3 user]#
[root@mnaeemsiddiqui3 user]# docker pull ubuntu:trusty
trusty: Pulling from library/ubuntu
99ad4e3ced4d: Pull complete
ec5a723f4e2a: Pull complete
2a175e11567c: Pull complete
8d26426e95e0: Pull complete
46e451596b7c: Pull complete
Digest: sha256:ed49036f63459d6e5ed6c0f238f5e94c3a0c70d24727c793c48fded60f70aa96
Status: Downloaded newer image for ubuntu:trusty
[root@mnaeemsiddiqui3 user]#
[root@mnaeemsiddiqui3 user]# docker run -it ubuntu bash
root@4f903c6c83fa:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@4f903c6c83fa:/# exit
exit
[root@mnaeemsiddiqui3 user]# docker run -it ubuntu ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
```

- To **list** all **images – docker images**
- To **list** all **containers (active)– docker container ls**
  - In snapshot below it returns no results as there is no active container running.
  - When we will run ubuntu in detached mode, we will try this command again to see if it shows an active container
- To **list** all containers **(container ID not truncated)– docker container ls –no-trunc**
- To **list** all containers **(container ID only)– docker container ls -q**
- Filter containers – **docker container ls -a filter <filtercondition>**
  - docker container ls -a filter “excited=0”
  - docker container ls -a filter “excited=1”
- To attach a container – **docker container attach <container-ID>**
  - it similar to running a container in interactive mode.
- To **list** all **containers (active and stopped)– docker container ls -a**
  - it lists all containers including active and stopped once
- to check the differential(delta of changes) between the original and modified container – **docker container diff <container-id>**
- to check image history – **docker image history <image>**

```
[root@mnaeemsiddiqui3 user]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest              f975c5035748        3 weeks ago         112MB
ubuntu              trusty              a35e70164dfb        3 weeks ago         222MB
hello-world         latest              f2a91732366c        4 months ago        1.85kB
[root@mnaeemsiddiqui3 user]#
[root@mnaeemsiddiqui3 user]# docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@mnaeemsiddiqui3 user]# docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
621c74904b90        ubuntu              "ls"                5 minutes ago       Exited (0) 5 minutes ago                        epic_proskuriakova
4f903c6c83fa        ubuntu              "bash"              6 minutes ago       Exited (0) 5 minutes ago                        sharp_hugle
0fbab3f2b5a8        hello-world         "/hello"            37 minutes ago      Exited (0) 37 minutes ago                       determined_davinci
[root@mnaeemsiddiqui3 user]#
```

- To **run** the container in **detached**(background) mode:
  - in detached mode you are running the container in background
  - extra switch -d for detached mode
  - **docker run -it -d ubuntu bash**
  - **docker run -it -d ubuntu ls**

- bash , ls are commands for the container
- you will then use docker exec command to run the command on the active container
  - first lets see if there is an active container running – **docker container ls**
  - now run the container and command
    - **docker exec -it <container-id-or-name> <command>**

```
[root@mnaeemsiddiqui3 user]# docker run -it -d ubuntu bash
56891b54def64907dcc6c2540baf831ce97f5d7101b01b880e679e2850272b9c
[root@mnaeemsiddiqui3 user]# docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
56891b54def6        ubuntu              "bash"              11 seconds ago      Up 9 seconds
s_cori
[root@mnaeemsiddiqui3 user]# docker exec -d 56891b54def6 ls
[root@mnaeemsiddiqui3 user]# docker exec -d 56891b54def6 bash
[root@mnaeemsiddiqui3 user]# ls
Desktop  VNCHOWTO  xrdp-chansrv.log
[root@mnaeemsiddiqui3 user]# docker exec -it 56891b54def6 bash
root@56891b54def6:/# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
root@56891b54def6:/# exit
exit
[root@mnaeemsiddiqui3 user]# docker exec -it 56891b54def6 ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
[root@mnaeemsiddiqui3 user]#
```

- This script from here – **https://testbucket786786.s3.amazonaws.com/docker/docker-installer.sh (https://testbucket786786.s3.amazonaws.com/docker/docker-installer.sh)** will also install docker and docker-compose(to be used later) on a host
  - wget https://testbucket786786.s3.amazonaws.com/docker/docker-installer.sh
  - chmod +x docker-installer.sh
  - ./docker-installer.sh

CATEGORIES DOCKERS •