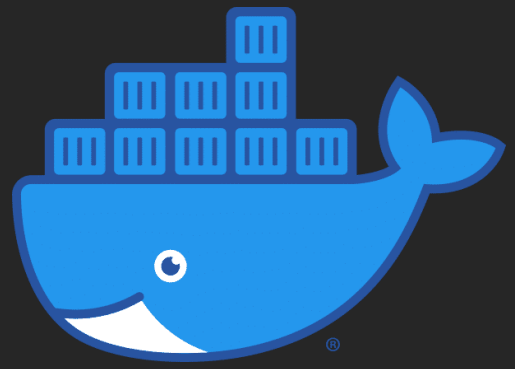


Learn Docker

https://www.linkedin.com/in/themr255/

- Agenda -

- Docker Introduction
- Docker Architecture
- Docker Vs Virtual Machine
- Docker Installation - RHEL -
- Docker Installation - MAC -
- Docker Installation - Windows -
- Docker Hello World
- Docker Lifecycle
- Docker Commands
- Docker Commands Cont.

Docker Introduction

- **What is Docker**
  - **Docker is an open platform** for developing, shipping, and running applications.
  - Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.
  - With Docker, you can manage your infrastructure in the same ways you manage your applications.
  - Docker is written in **Go programming** language.
  - Docker uses a technology called **namespaces** to provide the **isolated workspace** called the container. When you run a container, Docker creates a set of namespaces for that container.

- **Creator of Docker**
  - Docker Inc. was founded by Kamel Founadi, Solomon Hykes, and Sebastien Pahl during the Y Combinator Summer 2010 startup incubator group and launched in **2011**.

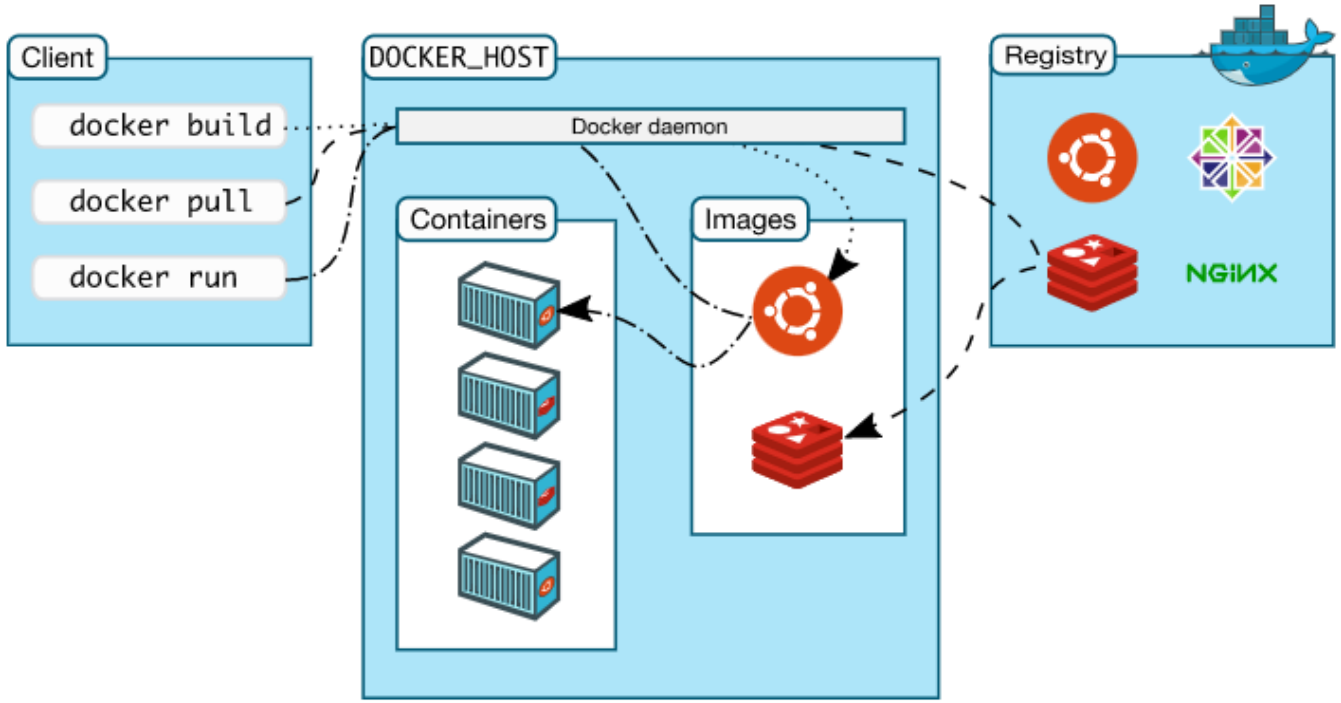
- **Docker Editions**
  - Docker is available in two editions
    1) Community Edition
    2) Enterprise Edition

- **Use cases of Docker**
  - Check use cases of Docker here.

Docker Architecture

- Docker uses a **client-server architecture**.
- The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers.
- The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a **REST API**, over UNIX sockets or a network interface.
- Another Docker client is **Docker Compose**, that lets you work with applications consisting of a set of containers.
- Please check below architecture for Docker,

- **Components of Docker –**
  1) Docker Daemon
  2) Docker Client
  3) Docker Registry
  4) Docker Images
  5) Docker Containers



Docker Architecture Cont.

- **Components of Docker –**

1) **Docker Daemon -**
   - The Docker daemon (dockerd) **listens for Docker API requests** and **manages Docker objects** such as images, containers, networks, and volumes.
   - A daemon can also communicate with other daemons to manage Docker services.
   - The **Docker daemon** called: **dockerd is the Docker engine** which represents the server.

2) **Docker Client -**
   - The Docker client (docker) is the primary way that many **Docker users interact with Docker**. When you use docker commands such as docker run, the **client sends these docker commands to docker daemon (dockerd)**, which carries them out.
   - The docker command uses the **Docker API**. The Docker client can communicate with more than one daemon.

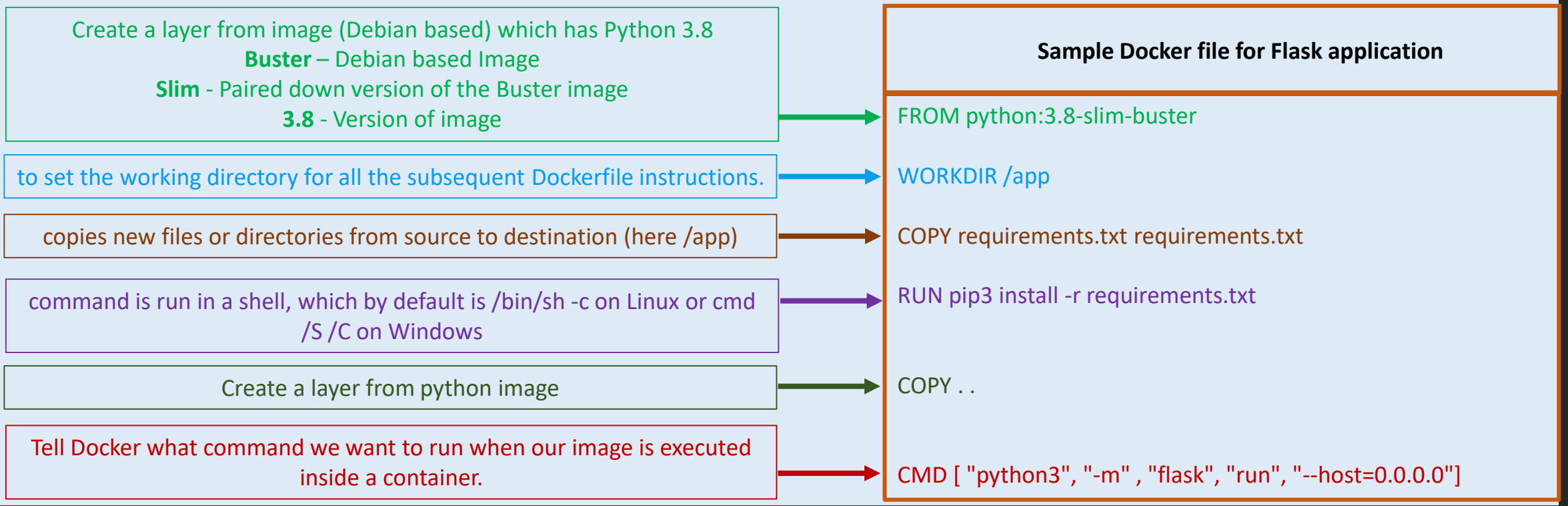
3) **Docker Registry -**
   - A **Docker registry stores Docker images**.
   - Docker Hub is a **public registry** that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own **private registry**.
   - When you use the **docker pull or docker run** commands, the required images are pulled from your configured registry.
   - When you use the **docker push** command, your image is pushed to your configured registry.

Docker Architecture Cont.

- **Components of Docker –**

4) Docker Images -

- An image is a **read-only template with instructions for creating a Docker container**. Often, an image is based on another image, with some additional customization. For example, you may build an image which is based on the Ubuntu image, but installs the Apache web server and your application, as well as the configuration details needed to run application.
- Docker template is **written in a language called YAML**, which stands for Yet Another Markup Language.
- The Docker image is built within the YAML file and then **hosted as a file in the Docker registry**.
- Example of Docker Image



Docker Architecture Cont.

- **Components of Docker –**

  5) Docker Container -

  - A container is a **runnable instance of an image**. You can create, start, stop, move, or delete a container using the Docker API or CLI.
  - You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.
  - By default, a container is relatively well isolated from other containers and its host machine. You can control how isolated a container's network, storage, or other underlying subsystems are from other containers or from the host machine.
  - **Docker Container is a running environment for Image.** Images like Redis, Ubuntu, Nginx, Alphine and etc.

Docker Vs Virtual Machine

Docker	Virtual Machine
OS level process isolation	Hardware-level process isolation
Dockers make use of the execution engine (Docker Engine).	VMs make use of the hypervisor.
Each container can share OS	Each VM has a separate OS
Containers are lightweight (KBs/MBs)	VMs are of few GBs
Less resource usage	More resource usage
Boots in a few seconds.	It takes a few minutes for VMs to boot.
Pre-built VMs are difficult to find	Pre-built docker containers are easily available
Containers are destroyed and re-created rather than moving	VMs can move to new host easily
Containers can be created in seconds	Creating VM takes a relatively longer time

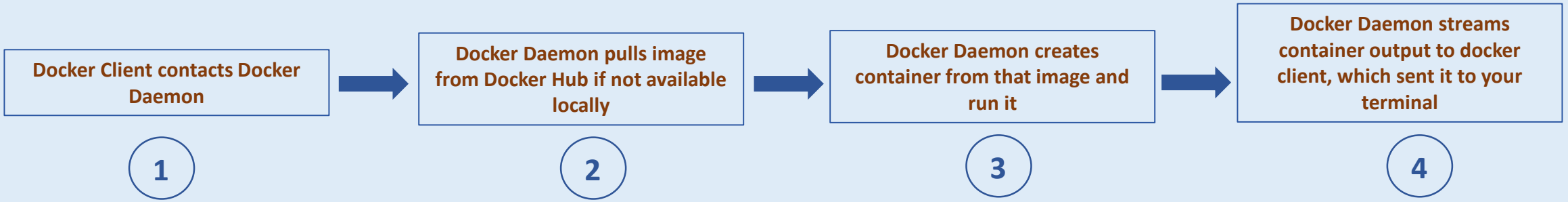
Docker Installation

- **To install Docker, please follow instructions given on below links**
  - **For RHEL**
    https://docs.docker.com/engine/install/rhel/
  - **For CentOS**
    https://docs.docker.com/engine/install/centos/
  - **For Ubuntu**
    https://docs.docker.com/engine/install/ubuntu/
  - **For Mac**
    https://docs.docker.com/desktop/install/mac-install/
  - **For Windows**
    https://docs.docker.com/desktop/install/windows-install/

Docker – Hello World

- **Demonstrating Hello World Example**
  - In our first example, we are going to run a container with hello world image.
  - To perform this practical, first check docker is running or not and if not then first docker daemon.
  - To use this hello world image, to create and run this container with that image use below command
    docker run hello-world

  - So how above command works in background???
  - We are running docker run command which is responsible for launching a container.
  - hello-world is the name of the image created by someone on Dockerhub.
  - So when we run above command then first it will search that image locally and if image is not available locally then it will search in Dockerhub.
  - Once hello-world image has been downloaded, docker creates a container using that image and executes it.
  - Check below flow for docker hello-world example



Flow in background

Docker – Hello World Cont.

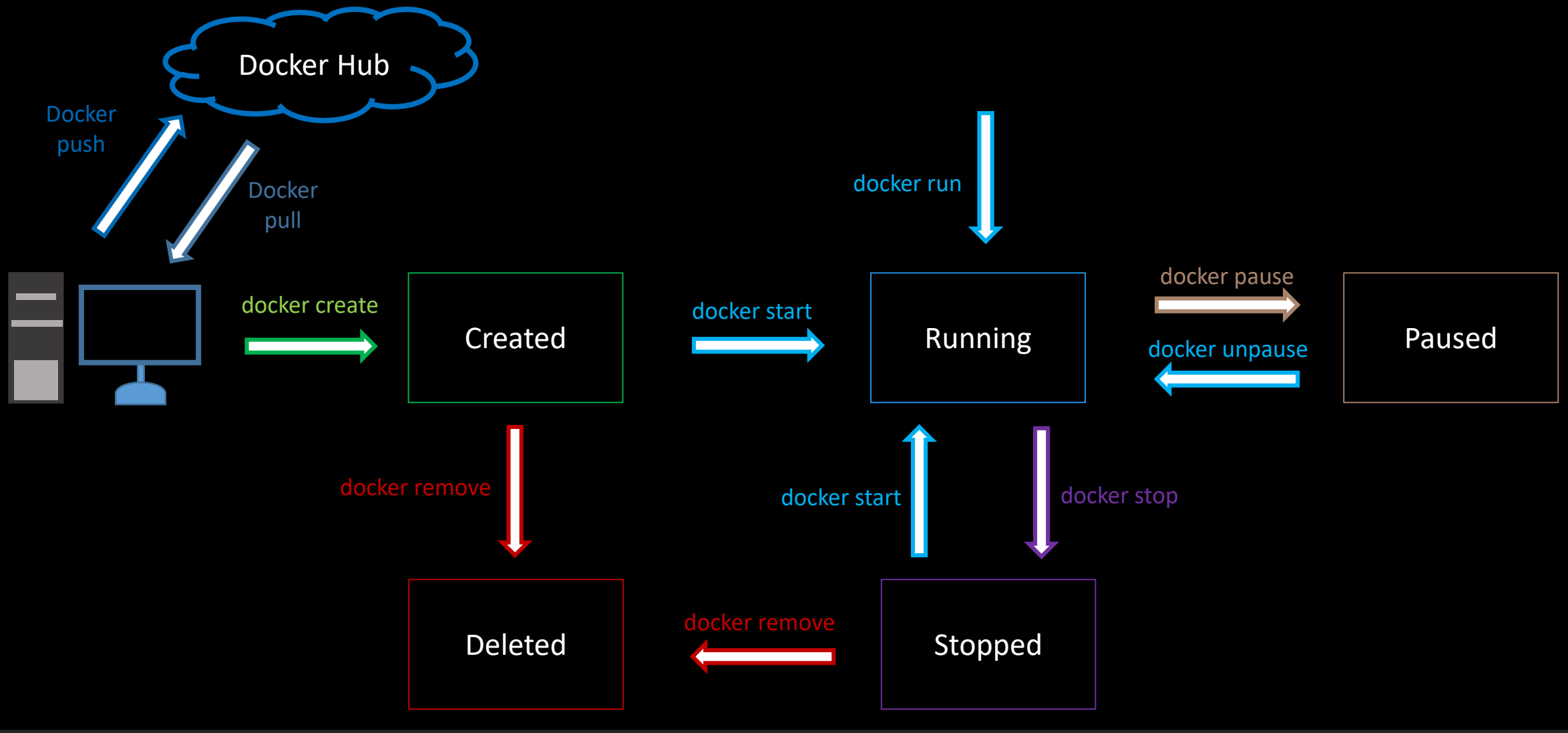
- **Demonstrating Hello World Example**

  Console output –

```
C:\Users\Mahesh> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:94ebc7edf3401f299cd3376a1669bc0a49aef92d6d2669005f9bc5ef028dc333
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://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

C:\Users\Mahesh>
```

Docker Lifecycle

- **Please see below flow for docker lifecycle.**



Docker Commands

1) **docker pull -** Pull an image or a repository from a registry.
   - In below example we are pulling **Nginx** image.
   - You can also pull image with specific tag as you see in below example, we are pulling **nginx:alpine-slim** image (Paired down version of main Nginx image)

```
C:\Users\Mahesh> docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
3f4ca61aafcd: Pull complete
50c68654b16f: Pull complete
3ed295c083ec: Pull complete
40b838968eea: Pull complete
88d3ab68332d: Pull complete
5f63362a3fa3: Pull complete
Digest: sha256:0047b729188a15da49380d9506d65959cce6d40291ccfb4e039f5dc7efd33286
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest

C:\Users\Mahesh> docker pull nginx:alpine-slim
alpine-slim: Pulling from library/nginx
c158987b0551: Pull complete
1e35f6679fab: Pull complete
cb9626c74200: Pull complete
b6334b6ace34: Pull complete
f1d1c9928c82: Pull complete
9b6f639ec6ea: Pull complete
Digest: sha256:6dbf9cb07de10c545d67e346042d628be0dd098f3207c30e23b84af9f146eced
Status: Downloaded newer image for nginx:alpine-slim
docker.io/library/nginx:alpine-slim

C:\Users\Mahesh>
```

Docker Commands Cont.

2) docker images - List and show images, their repository and tags, and their size.

```
C:\Users\Mahesh> docker images
REPOSITORY      TAG         IMAGE ID      CREATED       SIZE
nginx           latest      1403e55ab369  13 days ago   142MB
nginx           alpine-slim fe464c9f63f9  2 weeks ago   11.5MB
hello-world     latest      feb5d9fea6a5  15 months ago 13.3kB

C:\Users\Mahesh>
```

3) docker image ls - List and show images, their repository and tags, and their size.

```
C:\Users\Mahesh> docker image ls
REPOSITORY      TAG         IMAGE ID      CREATED       SIZE
nginx           latest      1403e55ab369  13 days ago   142MB
grafana/grafana latest      83f377cc32a0  2 weeks ago   317MB
prom/prometheus latest      b816c2dd5971  2 weeks ago   223MB
nginx           alpine-slim fe464c9f63f9  2 weeks ago   11.5MB
mysql           latest      7484689f290f  3 weeks ago   538MB
jenkins/jenkins lts         25fa92c47840  7 weeks ago   463MB
hello-world     latest      feb5d9fea6a5  15 months ago 13.3kB

C:\Users\Mahesh>
```

Docker Commands Cont.

4) docker run - To create and start a new container from docker image.

- **Note** – If you are running docker run command and if image is not available locally, then it will download image from Docker Hub & will create and run the container.
- And if I run **docker run nginx** command again then it will create a new container and will run it.

```
C:\Users\Mahesh> docker run nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/01/03 13:36:10 [notice] 1#1: using the "epoll" event method
2023/01/03 13:36:10 [notice] 1#1: nginx/1.23.3
2023/01/03 13:36:10 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2023/01/03 13:36:10 [notice] 1#1: OS: Linux 5.10.16.3-microsoft-standard-WSL2
2023/01/03 13:36:10 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/01/03 13:36:10 [notice] 1#1: start worker processes
2023/01/03 13:36:10 [notice] 1#1: start worker process 29
2023/01/03 13:36:10 [notice] 1#1: start worker process 30
2023/01/03 13:36:10 [notice] 1#1: start worker process 31
2023/01/03 13:36:10 [notice] 1#1: start worker process 32
2023/01/03 13:36:10 [notice] 1#1: start worker process 33
2023/01/03 13:36:10 [notice] 1#1: start worker process 34
2023/01/03 13:36:10 [notice] 1#1: start worker process 35
2023/01/03 13:36:10 [notice] 1#1: start worker process 36
```

Docker Commands Cont.

5) docker ps - List only running containers.

```
C:\Users\Mahesh> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS     NAMES
8de1c2fa0fd5   nginx     "/docker-entrypoint.…"   13 minutes ago   Up 2 minutes   80/tcp    sweet_lichterman

C:\Users\Mahesh>
```

6) docker ps -a - List both running and stopped containers.

- You can either use docker ps -a or docker ps --all

```
C:\Users\Mahesh> docker ps -a
CONTAINER ID   IMAGE             COMMAND                  CREATED         STATUS                     PORTS     NAMES
8de1c2fa0fd5   nginx             "/docker-entrypoint.…"   11 minutes ago   Up 2 seconds               80/tcp
sweet_lichterman
9a964a846329   hello-world       "/hello"                 7 hours ago      Exited (0) 7 hours ago             gallant_cerf

C:\Users\Mahesh>
```

Docker Commands Cont.

7) docker stop - Stop a running container

- To stop a container you can use below command

  docker stop container_id

- There is one more command which you can use to stop a container

  docker container stop container_id

```
C:\Users\Mahesh> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS     NAMES
8de1c2fa0fd5   nginx     "/docker-entrypoint.…"   33 minutes ago   Up 22 minutes   80/tcp    sweet_lichterman


C:\Users\Mahesh> docker stop 8de1c2fa0fd5
8de1c2fa0fd5

C:\Users\Mahesh> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

C:\Users\Mahesh>
```

Docker Commands Cont.

8) docker start - To start a container

- To start a container you can use below command

  docker start container_id

- There is one more command which you can use to start a container

  docker container start container_id

- To start a container, first get container ID of that container using command and then start your container as shown below – docker ps -a

```
C:\Users\Mahesh> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS                     PORTS     NAMES
8de1c2fa0fd5   nginx          "/docker-entrypoint.…"   42 minutes ago  Exited (0) 3 seconds ago             sweet_lichterman
9a964a846329   hello-world    "/hello"                 8 hours ago     Exited (0) 8 hours ago               gallant_cerf

C:\Users\Mahesh> docker start 8de1c2fa0fd5
8de1c2fa0fd5

C:\Users\Mahesh> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS     NAMES
8de1c2fa0fd5   nginx     "/docker-entrypoint.…"   42 minutes ago  Up 1 second    80/tcp    sweet_lichterman

C:\Users\Mahesh>
```

Docker Commands Cont.

9) docker log - To check logs of container

- To get logs of a container, first get container ID of that container using command and then start your container as shown below – docker ps –a
- Then you can use below command to fetch logs of container

  docker log container_id
- To view the last few lines of container log you can use command like - **docker logs -f --tail 10 container_id**

```
C:\Users\Mahesh> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS                     PORTS     NAMES
8de1c2fa0fd5   nginx          "/docker-entrypoint.…"   42 minutes ago  Exited (0) 3 seconds ago             sweet_lichterman
9a964a846329   hello-world    "/hello"                 8 hours ago     Exited (0) 8 hours ago               gallant_cerf

C:\Users\Mahesh> docker logs 8de1c2fa0fd5
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/01/03 13:36:10 [notice] 1#1: using the "epoll" event method
2023/01/03 13:36:10 [notice] 1#1: nginx/1.23.3
2023/01/03 13:36:10 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2023/01/03 13:36:10 [notice] 1#1: OS: Linux 5.10.16.3-microsoft-standard-WSL2
2023/01/03 13:36:10 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/01/03 13:36:10 [notice] 1#1: start worker processes
2023/01/03 13:36:10 [notice] 1#1: start worker process 29
2023/01/03 13:36:10 [notice] 1#1: start worker process 30
```

Docker Commands Cont.

10) docker exec -it - To run a command in a **running container.**

- Running an Interactive Shell in a Docker Container
- The **-i flag** keeps input open to the container, and the **-t flag** creates a pseudo-terminal that the shell can attach to.
- To create interactive Shell session in container -> **docker exec –it container_id sh**
- To create interactive Bash session in container -> **docker exec –it container_id bash**
- To **terminate interactive session** use **exit** command

```
C:\Users\Mahesh> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS                     PORTS     NAMES
8de1c2fa0fd5   nginx          "/docker-entrypoint...."   42 minutes ago   Exited (0) 3 seconds ago             sweet_lichterman
9a964a846329   hello-world    "/hello"                 8 hours ago     Exited (0) 8 hours ago               gallant_cerf

C:\Users\Mahesh> docker exec -it 8de1c2fa0fd5 sh
# pwd
/
# ls -lrt | tail
drwxr-xr-x   1 root root 4096 Dec 19 00:00 lib
drwxr-xr-x   2 root root 4096 Dec 19 00:00 bin
-rwxrwxr-x   1 root root 1616 Dec 21 11:28 docker-entrypoint.sh
drwxrwxrwt   1 root root 4096 Dec 21 11:28 tmp
drwxr-xr-x   1 root root 4096 Dec 21 11:28 docker-entrypoint.d
drwxr-xr-x   1 root root 4096 Jan  3 13:36 etc
dr-xr-xr-x  11 root root    0 Jan  3 14:48 sys
dr-xr-xr-x 239 root root    0 Jan  3 14:48 proc
drwxr-xr-x   5 root root  340 Jan  3 14:48 dev
drwxr-xr-x   1 root root 4096 Jan  3 14:48 run
# exit
C:\Users\Mahesh>
```

References

- https://www.docker.com/company/newsroom/media-resources/
- https://docs.docker.com/get-started/overview/
- https://www.docker.com/use-cases/#:~:text=Docker%20allows%20you%20to%20run,the%20Docker%20engine%20as%20containers
- https://hub.docker.com/_/python
- https://dockerlabs.collabnix.com/beginners/helloworld/
- https://docs.docker.com/language/python/build-images/
- https://medium.com/swlh/alpine-slim-stretch-buster-jessie-bullseye-bookworm-what-are-the-differences-in-docker-62171ed4531d
- https://docs.docker.com/engine/reference/builder/
- https://docs.docker.com/get-started/02_our_app/
- https://medium.com/future-vision/docker-lifecycle-tutorial-and-quickstart-guide-c5fd5b987e0d
- https://linuxhandbook.com/container-lifecycle-docker-commands/
- https://geekflare.com/docker-vs-virtual-machine/
- https://cloudacademy.com/blog/docker-vs-virtual-machines-differences-you-should-know/

DevOps Engineer

Mahesh Mahajan

Dev+Ops Engineer & Learner | 5 HackerRank

https://www.linkedin.com/in/themr255/

If you find this post helpful, then please share.
Follow **@TheMr255** for more DevOps content.

Like **Comment** **Save**