Basic Docker commands

Course Code: INT332

1 Finding the version

One of the first things you want to know is how to find the installed docker version.

- $ docker --version

- Docker version 18.09.6, build 481bc77

2. Docker info

- Get detailed information about docker installed on the system including the kernel version, number of containers and images, etc.
- **$ docker info**

- Containers: 3
- Running: 1
- Paused: 0
- Stopped: 2
- Images: 3
- Server Version: 18.09.6
- Storage Driver: overlay2
- Backing Filesystem: extfs
- Supports d_type: true Native Overlay Diff: true Logging Driver:

3. Checking History

- Shows the history of a docker image with the image name mentioned in the command

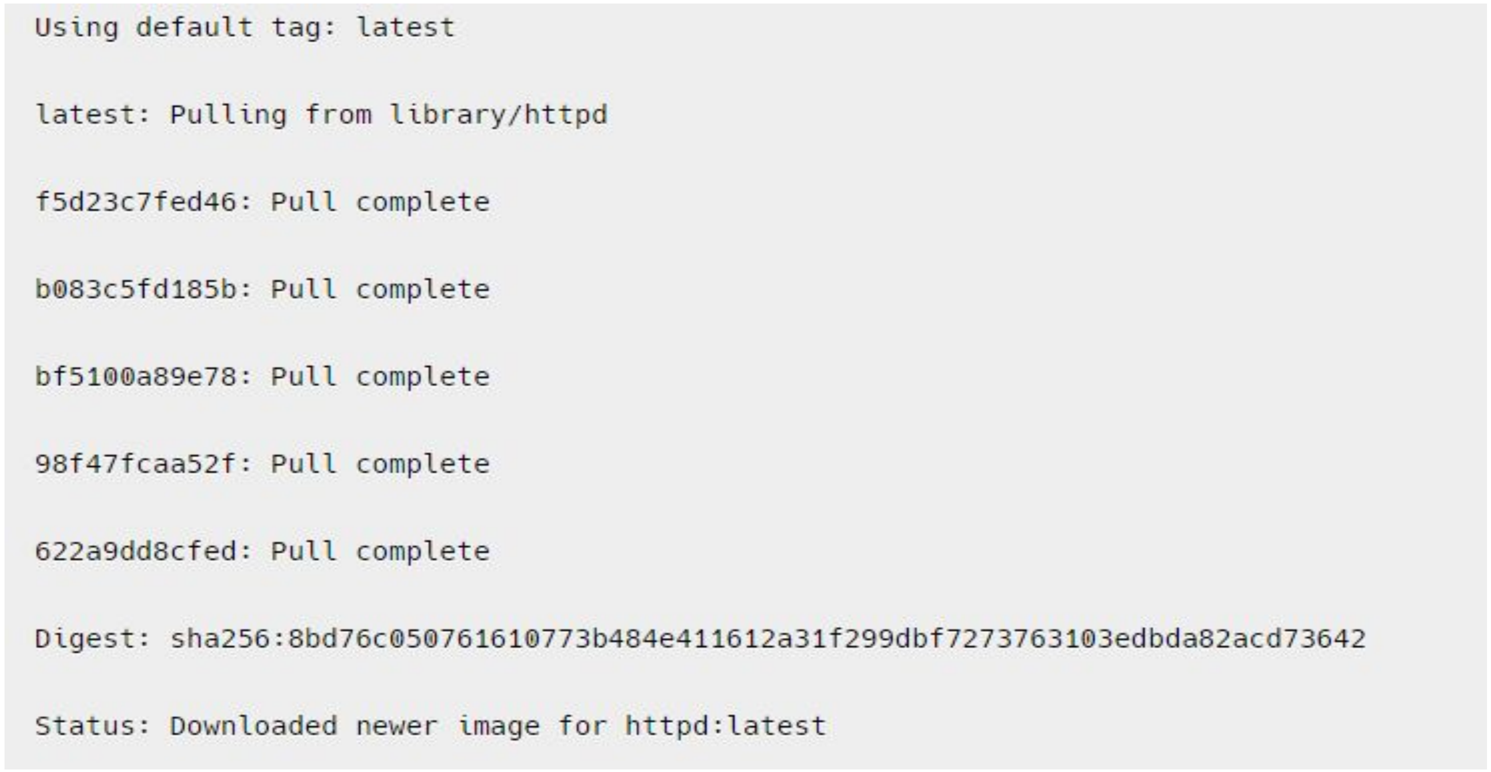
```
$ docker history httpd
```

```
IMAGE      CREATED    CREATED BY    SIZE     COMMENT
```

4 . Downloading Image

When we need to pull the docker image from dockerhub (docker repository) then ,the following example of pulling the the image and here we are taking Apache HTTP server image.

$ docker pull httpd

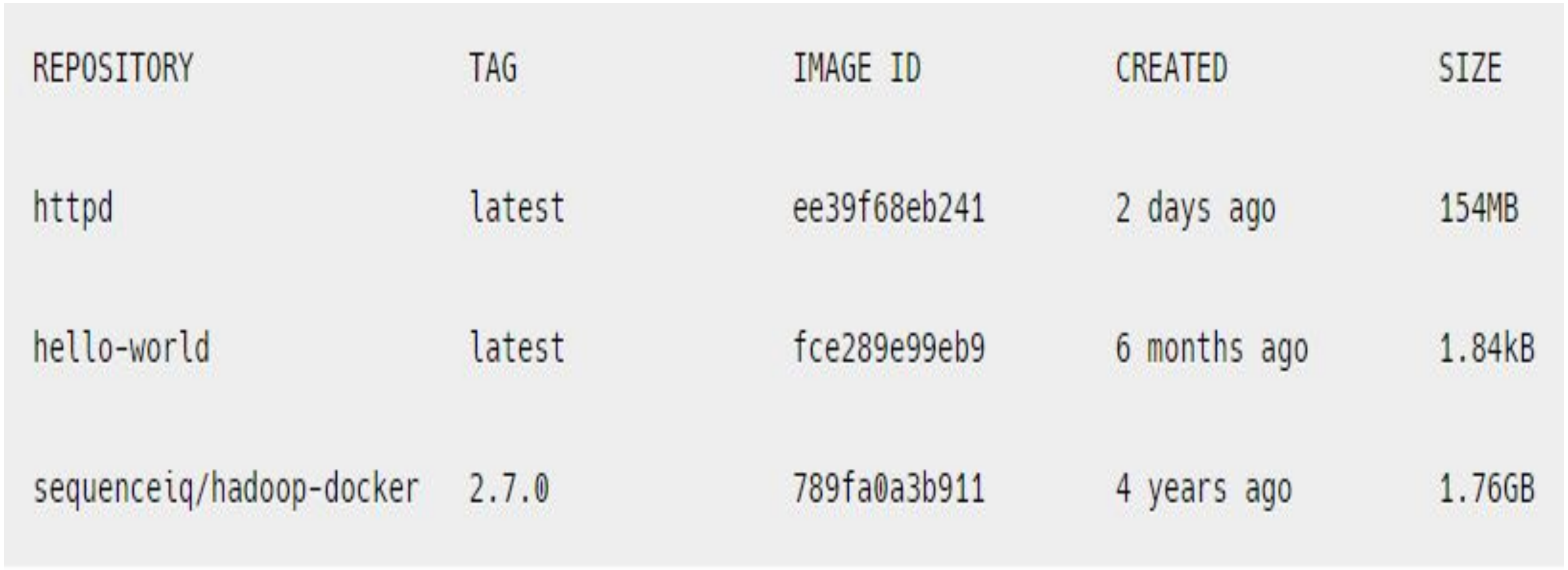


```
Using default tag: latest

latest: Pulling from library/httpd

f5d23c7fed46: Pull complete

b083c5fd185b: Pull complete

bf5100a89e78: Pull complete

98f47fcaa52f: Pull complete

622a9dd8cfed: Pull complete

Digest: sha256:8bd76c050761610773b484e411612a31f299dbf7273763103edbda82acd73642

Status: Downloaded newer image for httpd:latest
```

5. Images

To list all the docker images pulled on the system with image details such as TAG/IMAGE ID/SIZE etc.

$ docker images



```
REPOSITORY                  TAG       IMAGE ID       CREATED         SIZE
httpd                       latest    ee39f68eb241   2 days ago      154MB
hello-world                 latest    fce289e99eb9   6 months ago    1.84kB
sequenceiq/hadoop-docker    2.7.0     789fa0a3b911   4 years ago     1.76GB
```

6.RUN

- The **'docker run'** command is used to create and start a new container from a **Docker** image. The basic syntax of the **'docker run'** command is as follows:
- **docker run [OPTIONS] IMAGE [COMMAND] [ARG…]**
- *where,*
- ***OPTIONS:*** *optional flags that configure the behaviour of the container, such as setting environment variables, mounting volumes, etc.*
- ***IMAGE:*** *the name of the Docker image to be run.*
- ***COMMAND:*** *the command to be executed when the container is started. If not provided, the default command specified in the Docker image will be used.*
- ***ARG:*** *optional arguments passed to the command.*

-d means

Options Available for executing Docker Run Command

- Some commonly used options for executing the docker run command inside a container:
- *-it: Runs the container in interactive mode, allowing you to interact with the container through the command line.*
- *-d: Runs the container in the background.*
- *-- name: Specifies a name for the container.*

  -- rm: Automatically removes the container when it exits.
- *-p: Maps a host port to a container port.*
- -e / --env*: Sets an environment variable inside the container.*
- *-v: Mounts a host volume inside the container.*

1st Example : **Running a Simple Image:**

docker run httpd echo "Hello, World!"

- This command runs a new container from the '***httpd*** ' image and executes the '**echo**' command with the argument "Hello, World!".

2nd example

Running a Container with a Specific Name:

- ***$ docker run --name my-container httpd echo "Hello, World!"***

- ***--name : this option is to give the name of container***
- ***my-container: it is the name of container***
- ***Echo: it is the command***
- ***"hello,world!" – it is argument used***
- This command runs a new container with the specified name "my-container" from the **httpd** image and executes the echo command with the argument "Hello, World!".

What are Environment Variables

Environment variables are **key–value pairs** used by applications to:

- Read configuration
- Store secrets (usernames, passwords)
- Control application behavior **without changing the image**
- *Environment variables allow us to configure containers dynamically.*

Need of environment variables in Docker

Without -e:

•Configuration must be **hard-coded** in the image

•Need to rebuild image for every change

With -e:

•Same image → different behavior

•Secure & flexible configuration

•Widely used in **microservices & cloud deployments**

3rd Example : **Setting Environment:**

- **docker run -e MY_VAR=value httpd env**

- This command sets an environment variable ‘**MY_VAR**’ to the value “**value**” and runs a new container from the ‘**httpd**’ image.

The ‘**env**’ command is executed to display the list of environment variables in the container.

2nd Example: Run a container from Image and display the variable name inside it.

```
docker run -it -e MY_NAME=Harpreet ubuntu bash
```

It will open the container and you can check the env variable inside it as:

```
Echo $ MY_NAME
```

Kindly note :

MY_NAME is created inside the container

Host system is not affected

Multiple Environment Variables

- docker run -e APP_ENV=production -e APP_VERSION=1.0 nginx

Same for MySQL Container

```
docker run -d \
  -e MYSQL_ROOT_PASSWORD=root123 \
  -e MYSQL_DATABASE=college \
  -e MYSQL_USER=admin \
  -e MYSQL_PASSWORD=admin123 \
  mysql:8
```

Without these variables, MySQL container will fail to start

Next is : Passing Environment Variable from Host System

Ex#1:

Step 1: Set variable on host

```
export APP_PORT=8080
```

Step 2:

```
docker run -e APP_PORT nginx env
```

Another way : Using --env-file (Clean & Professional Way)

Create .env file

```
DB_HOST=localhost
DB_USER=root
DB_PASS=secret
```

Then Run container

```
docker run --env-file .env myapp
```

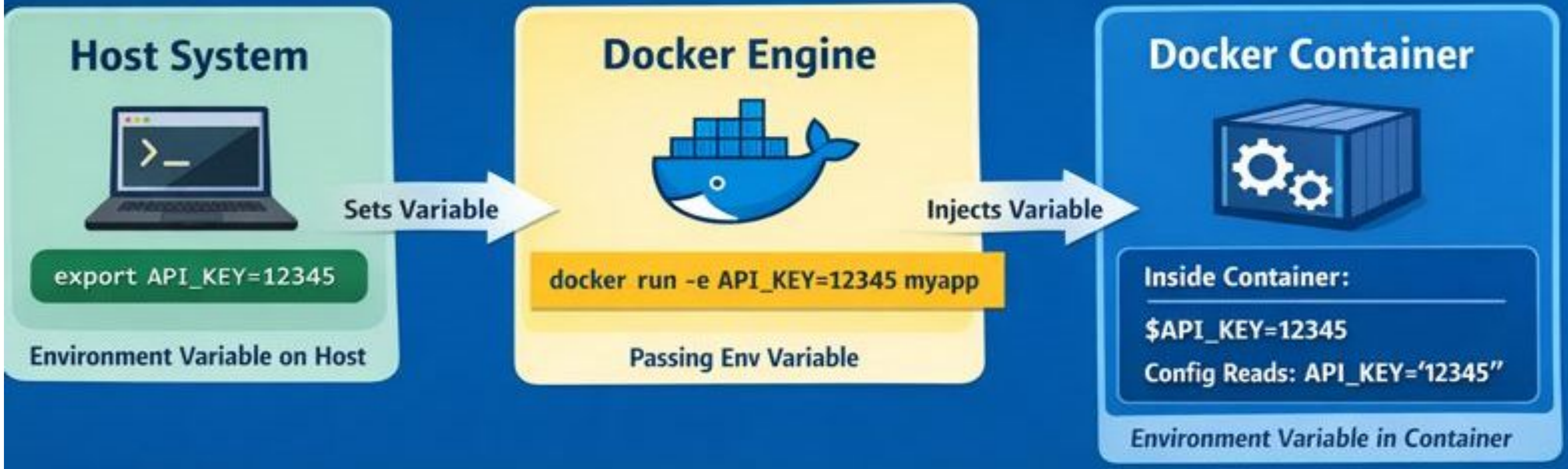
4th Example

$ docker run -it -d httpd

Here :

- **-it** flag enables interactive mode, so you can interact with the container’s command line
- The -d flag runs the container in the background and the **httpd** argument specifies the image name to use for the container. This command will pull the **httpd** image from Docker Hub if it is not already present on your local machine, create a container based on that image, and start it in the background.

Passing Environment Variables to Containers



```
export API_KEY=12345
```

```
docker run -e API_KEY=12345 myapp
```

Inside Container:

$API_KEY=12345

Config Reads: API_KEY='12345"

Dynamic Configuration at Runtime

-p optional flag in run command

- **-p stands for *publish port*.**
  It maps a **port on your host machine** to a **port inside the container**.

In simple words:

- **-p allows users outside the container to access an application running inside the container.**

By default:

- Containers run in an **isolated network**
- Applications inside containers are **NOT accessible from the host/browser**
- So if a web server is running **inside a container**, you **must expose it** using -p.

Syntax of –p

```
docker run -p <HOST_PORT>:<CONTAINER_PORT> IMAGE
```

Part	Description
`HOST_PORT`	Port on your system (Laptop/Server)
`CONTAINER_PORT`	Port where app runs inside container

EX#1:

- Without –p

- docker run nginx

•Try opening browser: But it is Not accessible
As no port mapping done here

Now try to Run Nginx with -p

- docker run -p 8080:80 nginx

What happens internally?

Browser → localhost:8080 → Docker Host → Container:80 → Nginx

- docker run -d -p 3307:3306 --name mysql-test –e MYSQL_ROOT_PASSWORD=root123 mysql

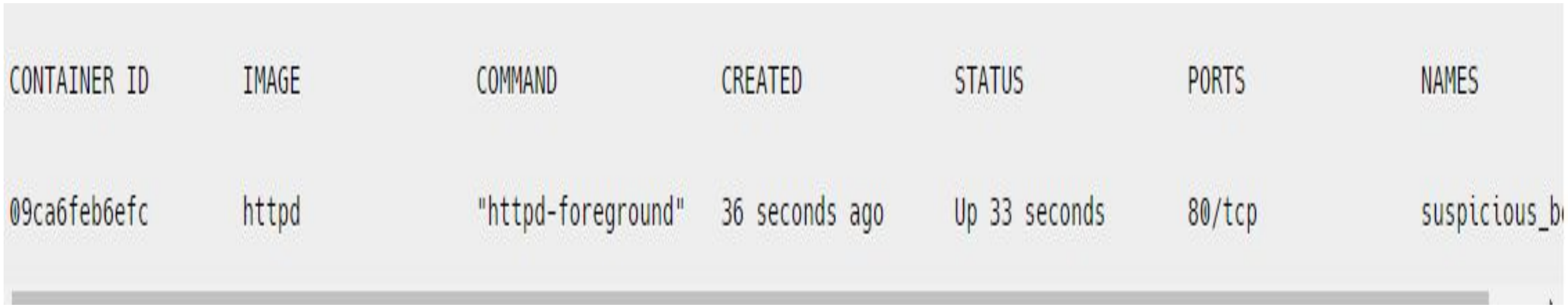
docker exec -it mysql-test bash

mysql -h 127.0.0.1 -P 3306 -u root –p

7 What's running?

- To lists all the docker containers are running with container details.
.

- **$ docker ps**



```
CONTAINER ID    IMAGE          COMMAND             CREATED          STATUS          PORTS          NAMES

09ca6feb6efc    httpd          "httpd-foreground"  36 seconds ago   Up 33 seconds   80/tcp         suspicious_b
```

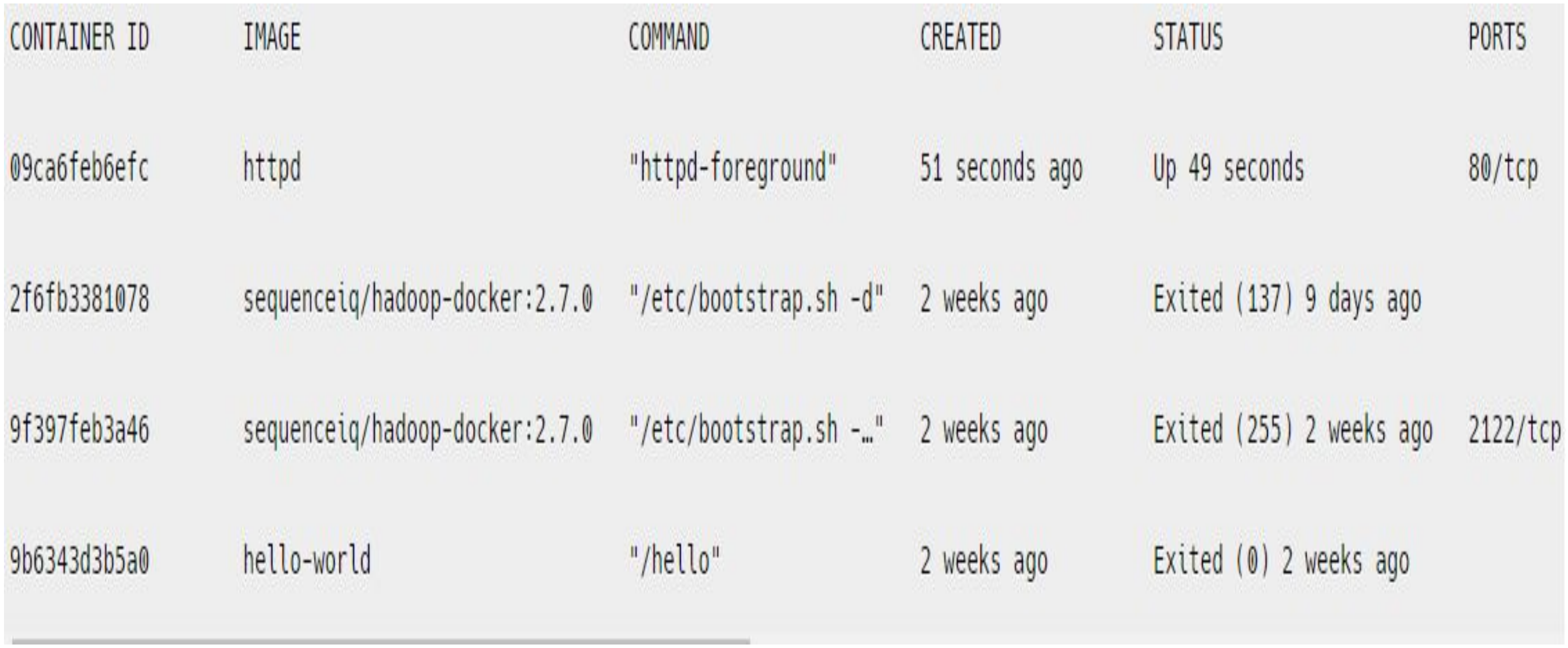
TASK #1:

- Run Ubuntu container
- Pass -e COLLEGE=CSE
- Show echo $COLLEGE
- Stop container then check what happened

8. ps -a

To list all the docker containers running/exited/stopped with container details

- **$ docker ps -a**



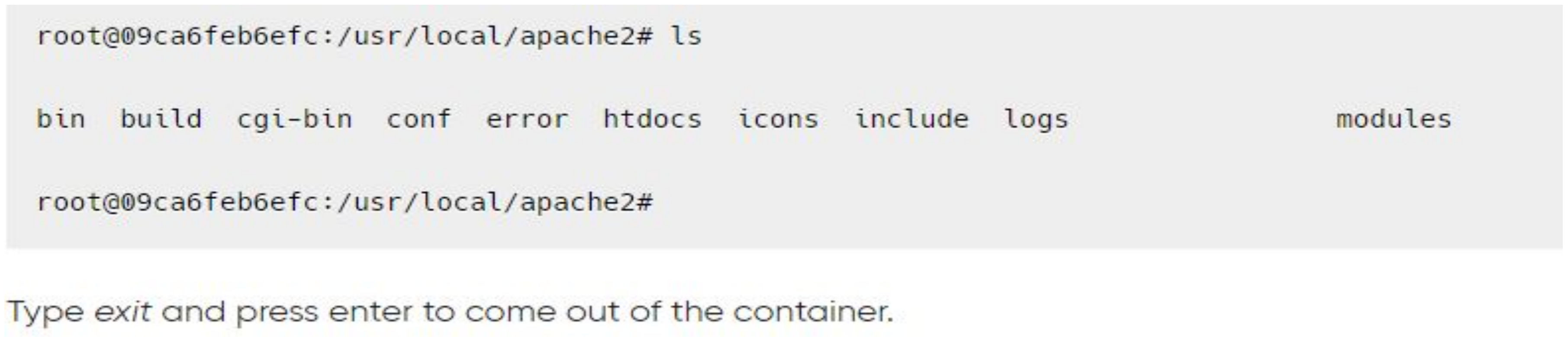
```
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS                    PORTS
09ca6feb6efc   httpd                          "httpd-foreground"       51 seconds ago   Up 49 seconds             80/tcp
2f6fb3381078   sequenceiq/hadoop-docker:2.7.0 "/etc/bootstrap.sh -d"   2 weeks ago      Exited (137) 9 days ago
9f397feb3a46   sequenceiq/hadoop-docker:2.7.0 "/etc/bootstrap.sh -…"   2 weeks ago      Exited (255) 2 weeks ago  2122/tcp
9b6343d3b5a0   hello-world                    "/hello"                 2 weeks ago      Exited (0) 2 weeks ago
```

9. exec

- Access the docker container and run commands inside the container in bash

$ docker exec -it 09ca6feb6efc bash/env

09ca6feb6efc --- it is Id of the container



```
root@09ca6feb6efc:/usr/local/apache2# ls

bin  build  cgi-bin  conf  error  htdocs  icons  include  logs           modules

root@09ca6feb6efc:/usr/local/apache2#
```

Type *exit* and press enter to come out of the container.

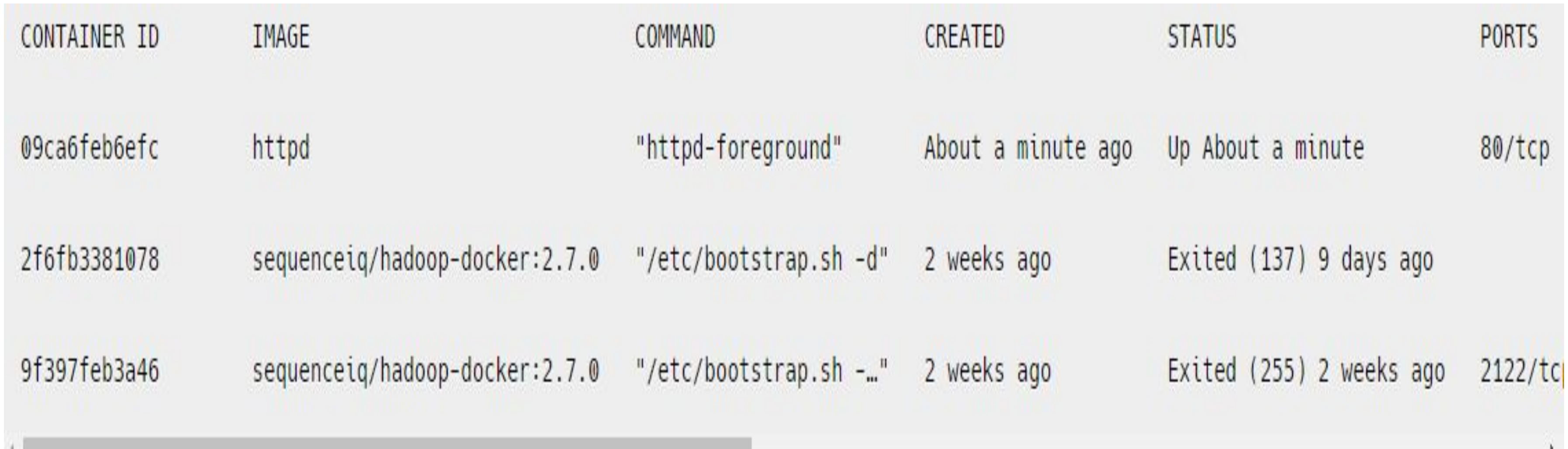
10. Removing container

- Remove the docker container with container id mentioned in the command.

```
$ docker  rm  9b6343d3b5a0
```

Run the below command to check if the container got removed or not.

```
$ docker ps -a
```

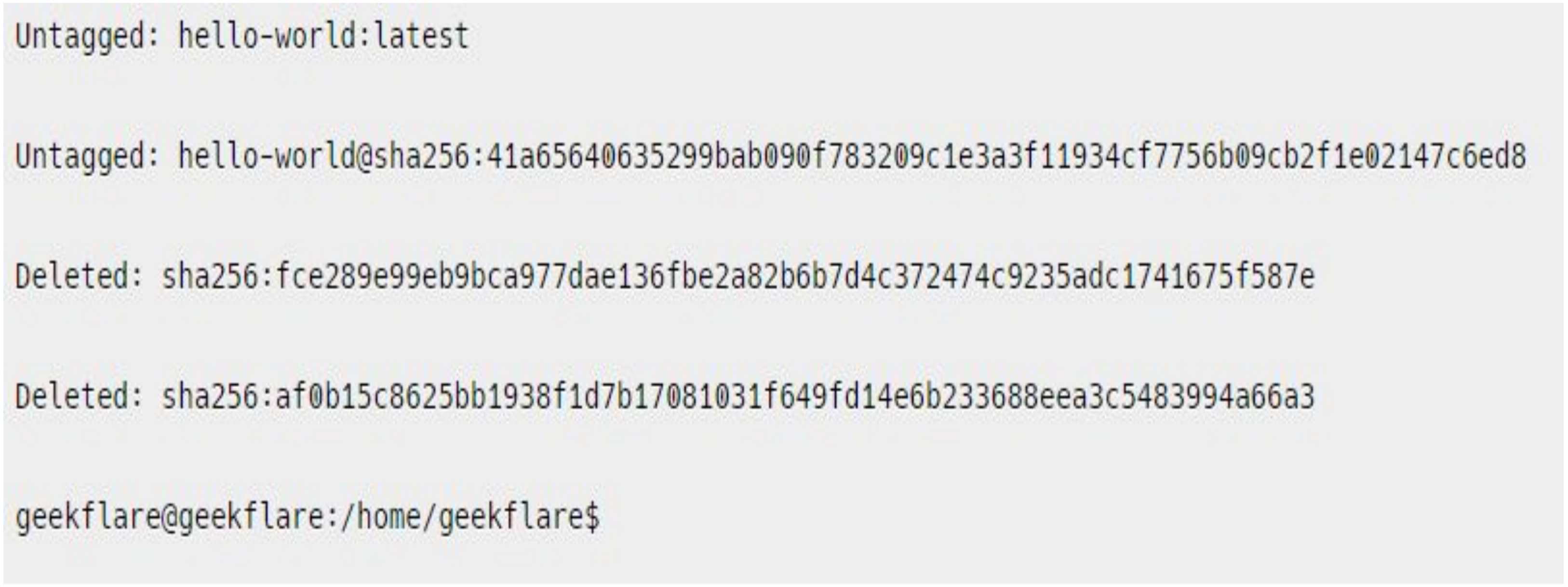


```
CONTAINER ID   IMAGE                           COMMAND                  CREATED              STATUS                    PORTS
09ca6feb6efc   httpd                           "httpd-foreground"       About a minute ago   Up About a minute         80/tcp
2f6fb3381078   sequenceiq/hadoop-docker:2.7.0  "/etc/bootstrap.sh -d"   2 weeks ago          Exited (137) 9 days ago
9f397feb3a46   sequenceiq/hadoop-docker:2.7.0  "/etc/bootstrap.sh -…"   2 weeks ago          Exited (255) 2 weeks ago  2122/tc
```

11. Removing image

- Remove the docker image with the docker image id mentioned in the command

  $ docker rmi fce289e99eb9



```
Untagged: hello-world:latest

Untagged: hello-world@sha256:41a65640635299bab090f783209c1e3a3f11934cf7756b09cb2f1e02147c6ed8

Deleted: sha256:fce289e99eb9bca977dae136fbe2a82b6b7d4c372474c9235adc1741675f587e

Deleted: sha256:af0b15c8625bb1938f1d7b17081031f649fd14e6b233688eea3c5483994a66a3

geekflare@geekflare:/home/geekflare$
```

..contd..

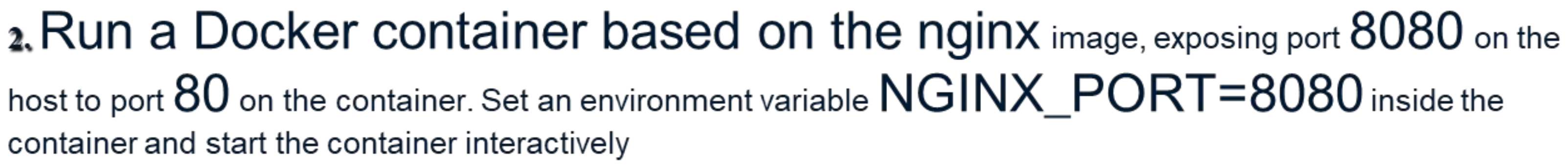
- To remove the image forcefully

```
$ docker rmi –f  <image id>
```

```
docker rmi -f 92fa43a2ff60503fdf250
Untagged: httpd:latest
Untagged:
    httpd@sha256:7765977cf2063fec486b63ddea574faf8fbed285f
    2b17020fa7ef70a4926cdec
Deleted:
    sha256:92fa43a2ff60503fdf250ef0a16d4170aa2a47c4b5a60d4
    27724283543a8792a
```

Practice Questions on Basic Docker Commands

- Run a Docker container named “DB-app" based on the “mongodb" image, and expose port 80 on the host to port 8082 on the container?



2. Run a Docker container based on the nginx image, exposing port 8080 on the host to port 80 on the container. Set an environment variable NGINX_PORT=8080 inside the container and start the container interactively

3. How would you use the docker run command with -it, -e, -v, and --name to: • Set an environment variable APP_ENV=production. • Bind a local directory /app/data to /data inside the container. • Name the container my_app. • Start an interactive terminal in an image called my_image? • Write the full command for the scenario above.

Docker container commands:

1. Container Lifecycle Management

- **Run a container** (creates and starts it)
- docker run -d -p 80:80 --name my_container nginx
- **Start a stopped container**
- docker start <container_id/name>
- **Stop a running container**
- docker stop <container_id/name>
- **Restart a container**
- docker restart <container_id/name>
- **Pause a container**
- docker pause <container_id/name>

…contd….

- **Unpause a paused container**
- docker unpause <container_id/name>
- **Kill a container (force stop)**
- docker kill <container_id/name>
- **Remove a container**
- docker rm <container_id/name>
- **Remove all stopped containers**
- docker container prune

2. Listing and Inspecting Containers

- **List all running containers**
- docker ps
- **List all containers (including stopped ones)**
- docker ps -a
- **Show detailed info about a container**
- docker inspect <container_id/name>
- **Display resource usage stats of a container**
- docker stats <container_id/name>
- **View container logs**
- docker logs <container_id/name>
- **Follow logs in real-time**
- docker logs -f <container_id/name>

3. Container Interaction

- **Execute a command inside a running container**
- docker exec -it <container_id/name> <command>
- **Attach to a running container's interactive shell**
- docker attach <container_id/name>
- **Copy files between container and host**
  - From container to host:
  - docker cp <container_id>:<container_path> <host_path>
  - From host to container:
  - docker cp <host_path> <container_id>:<container_path>
- E.g:

Practice questions:

- 1. You need to start a new container using the nginx image while setting an environment variable ENV_MODE=production. Write the docker run command to achieve this.
- 2. You have a running container named my_app, but it's not behaving as expected. You want to check its logs to debug any errors. Write the command to view its logs and follow new log entries in real-time.
- 3. A container named web_server was stopped. Write the command to:a) Start the container again.b) Stop the container when needed.

Work on Volumes

Basic commands used to create,inpect,verify , delete/remove the volumes

1. docker volume create <volume name>
2. docker volume ls
3. docker volume inspect mydata
4. docker run -d -v <volumename>:<path where to store in image> --name <containername> <imagename>

6. To Store Data in a Volume

```
docker run -it -v <source>:<path of destination>  <image name>
sh
```

Then inside the container

```
 echo “msg or data" > /app/data/filename
```

```
exit
```

To Verify Data Persistence: Run another container using the same volume

```
docker run -it -v <source>:<path of destination>  <image name> sh
```

```
cat <path of file>
```

To Remove a Volume

Docker volume rm <volume name>

To Delete All Unused Volumes

docker volume prune

LOVELY PROFESSIONAL UNIVERSITY
PUNJAB (INDIA)

L
P
U