Input :- $ docker
Output:-
A self-sufficient runtime for containers
Options:
```
      --config string      Location of client config files (default "/home/abhishek/.docker")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default "/home/abhishek/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default "/home/abhishek/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default "/home/abhishek/.docker/key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit

Management Commands:
  builder     Manage builds
  config      Manage Docker configs
  container   Manage containers
  engine      Manage the docker engine
  image       Manage images
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  secret      Manage Docker secrets
  service     Manage services
  stack       Manage Docker stacks
  swarm       Manage Swarm
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  deploy      Deploy a new stack or update an existing stack
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
```

```
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  ps          List containers
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  rmi         Remove one or more images
  run         Run a command in a new container
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  search      Search the Docker Hub for images
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  version     Show the Docker version information
  wait        Block until one or more containers stop, then print their exit codes

Run 'docker COMMAND --help' for more information on a command
```

Input:- $ docker -v
Output:- Docker version 18.09.7, build 2d0083d

To start docker :- sudo service docker start

To give permission :- sudo chmod 666 /var/run/docker.sock

Input:- docker info
Output:-

```
 Containers: 2
 Running: 0
 Paused: 0
 Stopped: 2
Images: 1
Server Version: 18.09.7
Storage Driver: overlay2
 Backing Filesystem: extfs
 Supports d_type: true
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
Swarm: inactive
Runtimes: runc
```

Default Runtime: runc
Init Binary: docker-init
containerd version: 894b81a4b802e4eb2a91d1ce216b8817763c29fb
runc version: 425e105d5a03fabd737a126ad93d62a9eeede87f
init version: fec3683
Security Options:
 apparmor
 seccomp
  Profile: default
Kernel Version: 4.15.0-54-generic
Operating System: Ubuntu 18.04.2 LTS
OSType: linux
Architecture: x86_64
CPUs: 8
Total Memory: 7.652GiB
Name: abhishek-Latitude-5490
ID: H7SM:M3XB:UBND:XVBO:6XNC:IQOI:332U:XKPP:UE4P:JVOK:73UE:BIQE
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false
Product License: Community Engine

input : - $ docker images
output :-
REPOSITORY TAG IMAGE ID CREATED SIZE
hello-world latest fce289e99eb9 6 months ago 1.84kB

input :- $ docker ps -a
output : -
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
3df4ac99e241 hello-world "/hello" 16 hours ago Exited (0) 16 hours ago amazing_napier
0d1376e9e6de hello-world "/hello" 10 days ago Exited (0) 10 days ago mystifying_hellman

input:- $ docker run hello-world
output :-
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/

To stop docker :- $ sudo service docker stop

input:- $ docker version
output :-
Client:
Version: 18.09.7
API version: 1.39
Go version: go1.10.8
Git commit: 2d0083d
Built: Thu Jun 27 17:56:23 2019
OS/Arch: linux/amd64
Experimental: false

Server: Docker Engine - Community
Engine:
Version: 18.09.7
API version: 1.39 (minimum version 1.12)
Go version: go1.10.8
Git commit: 2d0083d
Built: Thu Jun 27 17:23:02 2019
OS/Arch: linux/amd64
Experimental: false

input :- $ docker images --help
output :-
List images
Options:
-a, --all Show all images (default hides intermediate images)
--digests Show digests
-f, --filter filter Filter output based on conditions provided
--format string Pretty-print images using a Go template
--no-trunc Don't truncate output
-q, --quiet Only show numeric IDs

To login into docker hub :- $ docker login

To pull an image(e.g. ubuntu) from dockerhub :- $ docker pull ubuntu

To list images id :- $ docker images -q

output :- 4c108a37151f
fce289e99eb9

input :- $ docker images -a
output :-
REPOSITORY TAG IMAGE ID CREATED SIZE
ubuntu latest 4c108a37151f 2 weeks ago 64.2MB
hello-world latest fce289e99eb9 6 months ago 1.84kB

To remove an image :- $ docker rmi fce289e99eb9

To go inside a container :- $ docker run -it ubuntu
output :-
root@daa6448db610:/# ls
bin dev home lib64 mnt proc run srv tmp var
boot etc lib media opt root sbin sys usr

To see container running :- $ docker ps
output :-
CONTAINER ID IMAGE COMMAND CREATED STATUS
PORTS NAMES
daa6448db610 ubuntu "/bin/bash" About a minute ago Up About a minute
musing_sinoussi

To start a container :- $ docker start daa6448db610
output :- daa6448db610

To stop a container :- $ docker stop daa6448db610

To check status of system :- $ docker stats

To check disk usage :- $ docker system df
output :-
TYPE TOTAL ACTIVE SIZE RECLAIMABLE
Images 2 2 64.19MB 0B (0%)
Containers 5 0 3B 3B (100%)
Local Volumes 0 0 0B 0B
Build Cache 0 0 0B 0B

To remove unused data (removes containers):- $ docker system prune
(removes images too) :- $ docker system prune -a

To list images with filter :- $ docker images -f 'dangling=false'
output :-
REPOSITORY TAG IMAGE ID CREATED SIZE
ubuntu latest 4c108a37151f 2 weeks ago 64.2MB

To get only image id :- $ docker images -f 'dangling=false' -q
output :- 4c108a37151f

To run an image which is actually creating a container out of an image :-
$ docker run --name MyUbuntu -it ubuntu bash

To inspect an image :- $ docker inspect ubuntu

To pause a container :- $ docker pause MyUbuntu

To unpause a container :- $ docker unpause MyUbuntu

Input :- $ docker top MyUbuntu
Output :-

```
UID                 PID                 PPID                C                   STIME               TTY                 TIME                CMD
root                9432                9405                0                   12:02               pts/0               00:00:00            /bin/bash
```

Input :- $ docker stats MyUbuntu
Output :-

```
CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O             BLOCK I/O           PIDS
e8fedbf69559        MyUbuntu            0.00%               1.93MiB / 7.652GiB    0.02%               3.68kB / 0B         0B / 0B             1
```

To attach to a container :- $ docker attach MyUbuntu
To kill a container :- $ docker kill MyUbuntu

To remove a container : - $ docker rm MyUbuntu

To get history of an image :- $ docker history ubuntu
output :-

```
IMAGE               CREATED             CREATED BY                                      SIZE                COMMENT
4c108a37151f        2 weeks ago         /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>           2 weeks ago         /bin/sh -c mkdir -p /run/systemd && echo 'do…   7B
<missing>           2 weeks ago         /bin/sh -c set -xe   && echo '#!/bin/sh' > /…   745B
<missing>           2 weeks ago         /bin/sh -c [ -z "$(apt-get indextargets)" ]     987kB
<missing>           2 weeks ago         /bin/sh -c #(nop) ADD file:4e6b5d9ca371eb881…   63.2MB
```

Jenkins on docker
To pull jenkins :- $ docker pull jenkins
To run jenkins on port 9090 :-

```
 docker run --name myjenkins2 -p 9090:8080 -p 50000:50000 -v
~/Desktop/Jenkins_Home:/var/jenkins_home jenkins
```

To create a volume :- $ docker volume create jenkinsVolume
To list volumes :- $ docker volume ls
output :-

```
DRIVER              VOLUME NAME
local               jenkinsVolume
```

To inspect a volume :- docker volume inspect jenkinsVolume
output :-

```
[
    {
        "CreatedAt": "2019-07-09T18:04:44+05:30",
```

```
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/jenkinsVolume/_data",
        "Name": "jenkinsVolume",
        "Options": {},
        "Scope": "local"
    }
]
```

To inspect a container :-$ docker inspect myjenkins3

To build a docker file :

Step 1 : Create a file named Dockerfile
Step 2 : Add instructions in Dockerfile
Step 3 : Build dockerfile to create image
Step 4 : Run image to create container

Creating a basic docker file:-

```
#getting base image ubuntu
FROM ubuntu
MAINTAINER abhishek kumar <abhishek.6059@gmail.com>
RUN apt-get update                                  #(gets executed during building of image)
CMD ["echo", "Hello World...! from my first docker image"]
#(gets executed when container is created)
```

To build Docker file :- $ docker build -t myimage1:1.0 . or (location of dockerfile)

To run image to create container :- $ docker run c362d40ee5e4

Docker Compose (create a docker compose file):
tool for defining and running multi container applications

Step 1 : install docker compose
Step 2 : Create docker compose file at any location on your system
docker-compose.yml
Step 3 : Check the validity of file by command
docker-compose config
Step 4 : Run docker-compose.yml file by command
docker-compose up -d
Steps 5 : Bring down application by command
docker-compose down

To scale up a container :- $ docker-compose up -d --scale database=4
Docker Volumes :
To create a volume :- $ docker volume create myvol1
To list volumes :- $ docker volume ls
To inspect a volume :- $ docker inspect myvol1
To remove a volume :- $ docker volume rm jenkinsVolume
To remove unused volumes :- $ docker volume prune