1. Pull an image

docker pull <imagename:tag>

The default value for 'tag' is 'latest'

2. Run a container for an image

docker run <imagename:tag>

a) Run in detached mode. This means container will be started and keep running in background

docker run -d <imagename:tag>

b) Run with host port mapping (-p or --publish option).

docker run -d -p80:8080 <imagename:tag>

Note: read about expose option

This mapps container port 8080 with host port 80

c) Run with all container ports mapped to random port (dynamic port mapping, -P or publish-all option)

docker run -d -P <imagename:tag>

d) Run with a user defined name for container

docker run -d --name mycontainer <imagename:tag>

e) Run with a custom hostname for the container (-h or --hostname option)

docker -d -h app.example.com <imagename:tag>

f) Run with a host entry added to container host file (notice the syntax)

docker run -d --add-host=web.example.com:192.168.0.1 <imagename:tag>

g) Run with a environment variable (--env or -e or --env-file options

docker run -d --env JAVA_HOME=/opt/Java <imagename:tag>

docker run -d --env-file /opt/env/env.txt <imagename:tag>

h) Run with a bind mount (2 options -v or --volume and --mount). --mount is more verbose

docker run -d -v/app:/opt/app <imagename:tag> (mount host /app to container /opt/app)

docker run -d --mount source=/app,destination=/opt/app <imagename:tag>

Note: read about docker volumes: https://docs.docker.com/engine/admin/volumes/volumes/

i) Run with option to remove container when at exit

docker -d --rm <imagename:tag>

Without this option docker container is still there and needs to be removed using "docker rm" command

j) Run with interactive session

docker run -it <imagename:tag>

k) Run with a command (in addition to ENTRYPOINT (entrypoint can also be replaced with --entrypoint) or instead or CMD)

docker run <imagename:tag> hostname

3. Stop a running container

docker stop <container ID or name>

4. Remove a stopped container (if --rm option is not given at the time of run)

docker rm <container ID or name>

5. Removing all the stopped containers

docker rm $(docker ps -a -f status=exited -q)

6. Start a stopped container

docker start <container ID or name>

7. Attach local standard input, output, and error streams to a running container

docker attach <container ID or name>

8. Build a container image using dockerfile

docker build <path, it is the docker context e.g. current directory (.). If -f option is not used then it expects dockerfile in the path >

docker build .

docker build . -t myimage:dev (-t is for tagging the image)

docker build . -f /home/ubuntu/Dockerfile -t myimg:latest

9. Commit the changes to writable layer of running container in an image

docker commit <container name or id> <imagename:tag>

10. Run a command in running container

In detached mode - docker exec -d <container name or id> mkdir -p /opt/mydir

In interactive mode (e.g. get a bash session) - docker exec -it <container name or id> bash

11. List containers

docker ps (list running container)

docker ps -a (list all containers, stopped as well)

12. List images in local

docker images

13. Remove image from local

docker rmi <image name or id>

14. Create a tag for image

docker tag <source image> <target image>:<tag>

15. Copy files between host and container

docker cp <host path> <container id or name>:<container path>

docker cp <container id or name>:<container path> <host path>

16. Get information about docker objects (e.g. images or containers etc)

docker inspect <image or container id>

17. Get logs of docker container

docker logs <container id or name>

docker logs -f <container id or name> (similar to tail -f)

18. Get live container statistics (e.g. cpu and memory utilization)

docker stats <container id or name>

19. Log into docker registry

docker login -u <username> -p <password> <registry url>

20. Push image to docker registry

docker push <image name>:<tag>

21. Find changes made to a container (in writable layer)

docker diff <container name or id>