Docker

installation

```
# update the apt cache
> sudo apt-get update

# install the certificate tool to communicate with apt server
# CA: certificate authority
# curl: console url
> sudo apt-get install ca-certificates curl

# install the apt-keys
> sudo install -m 0755 -d /etc/apt/keyrings
> sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
> sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
> echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu
\
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}")
stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# update the apt cache again
> sudo apt-get update

# install docker and its dependencies
# docker-ce: community edition
# container.id: used to run the containers
# docker-buildx-plugin: used to create images for different CPU
architecture
# docker-compose-plugin: used to run micro-services application
> sudo apt-get install docker-ce docker-ce-cli containerd.io docker-
buildx-plugin docker-compose-plugin

# check if the docker is running
> sudo systemctl status docker

# start the docker service
> sudo systemctl start docker

# start the docker on startup
> sudo systemctl enable docker
```

generic commands

```
# get a list of objects
# > docker <object> ls

# get details of selected object
# > docker <object> inspect <object id or name>

# delete selected object
# > docker <object> rm <object id or name>

# delete dangling objects
# > docker <object> prune
```

images

```
# get the list of images
> docker image ls

# download an image from docker registry
# > docker image pull <image name>
> docker image pull hello-world

# get the details of selected image
# > docker image inspect <image name or id>
> docker image inspect hello-world

# remove a selected image
# > docker image rm <image name or id>
# note:
# - image CAN NOT be deleted if at one container of the image is in use
# - --force can be applied to delete an image for which a container is
running, which will create a dangling image (the image will be unlisted)
> docker image rm hello-world

# remove dangling and unused images
> docker image prune

# build a new image
# > docker image build -t <image name>:<image tag> <context>
> docker image build -t myimage .

# create an alias to existing image
# > docker image tag <old name> <name name>
> docker image tag mywebsite amitksunbeam/mywebsite

# login with docker hub
# > docker login -u <docker user name>

# push the image to docker hub
# > docker image push <docker hub username>/<image name>
```

dockerfile

- text file with instruction and it argument(s) pairs
- instructions
    - FROM
        - used to decide the base image for the custom image
        - every custom image must be created using one base image
    - COPY
        - used to copy file(s) to the image
        - syntax
            - COPY
            - e.g. COPY index.html /usr/local/apache2/htdocs
                - the above command will copy index.html from local directory to /usr/local/apache2/htdocs path in image
    - EXPOSE
        - used to expose a port for the consumer
    - WORKDIR
        - used to set the working directory
    - RUN
        - used to execute a command at the time of building the image
        - the command will get executed only once
        - you may have multiple RUN instructions
    - CMD
        - used to execute a command at the time of creating container
        - one image can have one and only one CMD instruction
        - dockerfile must contain the CMD instruction as the last instruction
    - ENV
        - used to set an environment variable
        - syntax: ENV =

container

```


# get the list of running containers
> docker container ls

# get the list of all containers
# states: created, up (running), exited (stopped)
> docker container ls -a

# delete a stopped container
# > docker container rm <container id or name>

# remove a running container
# > docker container rm --force <container id or name>

# create a container
# > docker container create <image name or id>
> docker container create hello-world

# get details of selected container
# > docker container inspect <container name or id>

# start a (created/stopped) container
# > docker container start <container id or name>

# stop a running container
# > docker container stop <container id or name>

# get the logs generated by the container
# > docker container logs <container id or name>


# run a container in attached mode
# - create a new container and start the application inside the container
# - run = create + start
# > docker container run <image name or id>

# attached mode
# - the container by default gets attached with the terminal
# - the logs will be shown directly on the terminal
# - the container gets the commands directly from the terminal
# - if terminal stops, the container will also stop

# detached mode
# - the container will run in the background

# execute a command inside a container
# > docker container exec <container name or id> <command>
> docker container exec myhttpd date

# get a terminal out from a container
# > docker container exec -it <container name> <bash or sh>
> docker container exec myhttpd -it bash

# run a container
# - -d: start the container in detached mode
# - -i: let the user interact with the container
# - -t: let the user get the terminal from the container
# - --name: sets the container name
# - -p: used to publish a port
#   - -p <OS/HOST port>:<container port>
# - -e: used to set an environment variable
#   - -e <env var name>=<env var value>
# > docker container run -d -i -t --name <container name> -p <os port>:
```

```
<container port> <image name or id>
# > docker container run -itd --name <container name> -p <os port>:
<container port> <image name or id>
> docker container run -itd --name myhttpd -p 9090:80 httpd

# create a container for mysql
> docker container run -itd --name mysql -p 3306:3306 -e
MYSQL_ROOT_PASSWORD=root mysql
```