docker

docker generic commands

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

# get information about selected object
# > docker <object> inspect <object name or id>

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

# remove unused objects
# > docker <object> prune
```

docker images

- image registry
    - collection of docker images
    - types
        - custom or private
            - can be hosted by organizations or companies
        - public
            - hosted by docker at hub.docker.com

```
# list available images
> docker image ls

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

# get the information about the selected image
# > docker image inspect <image name>
> docker image inspect hello-world

# delete selected image
# > docker image rm <image name or id>
> docker image rm hello-world

# delete unused images
> docker image prune
```

custom images

- to create a custom image, create a file named Dockerfile
- the custom image must be created using an existing image (it must have one base image)
- commands
  - FROM: used to specify the base image
  - COPY: used to copy resource(s) from local machine to image
    - syntax: COPY
  - EXPOSE: used to expose a port for container (used later in port forwarding)
  - RUN: used to execute command(s) while building the image
  - CMD: used to execute command when a new container starts
  - WORKDIR: used to set (or create if it does not exist) working directory
  - ENV: used to set an environment variable
- to push your image to docker hub
  - signup with docker hub (free)
  - rename the image with the following pattern
    - name: /
  - login with your name and password
  - push the image to the docker hub

```
# build the image
# > docker image build -t <image name> <context>
> docker image build -t mywebsite .

# create a reference to an image with different name
# > docker image tag <old image name> <new image name>
> docker image tag mywebsite amitksunbeam/mywebsite

# login with docker hub user name and password
> docker login

# push image to docker hub
# note: make sure the image name is in the pattern: <docker user
name>/<image name>
> docker image push <image name>
```

docker containers

- a container can run only one application at a time
- when the application stops (either because the application is finished with its execution or got terminated because of any error), the container gets exited

- containers are not meant to persist the data (as the data will be stored inside the container till it is alive)
- if the container gets removed, all the data stored inside the container will be removed

```
# get the list of running containers
> docker container ls

# get the list of all containers
# statuses: Created, Running (Up), Exited
> docker container ls -a

# create a new container
# every new container gets
# - a unique id assigned
# - a random but unique name assigned
# > docker container create <image name or id>
> docker container create hello-world

# start a container
# > docker container start <container name or id>

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

# get the logs from a container
# > docker container logs <container name or id>

# get the information about the container
# > docker container inspect <container name of id>

# delete exited container
# > docker container rm <container name or id>

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

# run a container in attached mode (in foreground)
# - run = create + start
# > docker container run <image name or id>
> docker container run httpd

# run a container
# params
# - -d: run the container in detached mode (in background)
# - -i: run the container in interacting mode
# - -t: enable the container to get the terminal from it
# - --name: name of the container
# - -p: used to publish a port on the container
# - -e: used to set an environment variable
# - -v: used to attach a volume on the container
# > docker container run -d -i -t --name <container name> -p <source
port>:<destination port> -v <volume name>:<container mount path> <image
name or id>
# > docker container run -d -i -t --name myhttpd -p 8080:80 httpd
> docker container d-itd --name myhttpd -p 8080:80 httpd

# create a mysql container
> docker container run -itd --name mysql -p 3306:3306 -e
MYSQL_ROOT_PASSWORD=root -v mysql-volume:/var/lib/mysql mysql

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

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

# delete unused containers
> docker container prune
```

docker volumes

```
# get the list of volumes
> docker volume ls

# create a new volume
# > docker volume create <volume name>

# delete a volume
# > docker volume rm <volume name>

# delete all unused volumes
> docker volume prune
```