```
# ---------- DOCKER CLI -----------

# GENERAL
    # docker help => shows a helplist of all useful docker commands

# IMAGES
    # docker build {PATH TO DOCKERFILE} => builds a Docker image from the specified Dockerfile, where the path can be a local file extension or a URL, images are rebuilt whenever a change is made in the Dockerfile
    # docker pull {IMAGE NAME} => similar to git pull, downloads the specified image from Docker Hub
    # docker create {IMAGE NAME} => creates a new container with the specified image but does not run the container automatically like docker run, docker start must be used to then run this container
    # docker images => displays all docker images and their metadata
    # docker rmi => removes one or more docker images if they do not have a container instance currently running

# CONTAINERS
    # docker run {CONTAINER NAME} => runs the specified docker container and can be augmented with flags, docker will pull the required image from Docker Hub and build the container if the container is not found, then run it
        # -d => detach flag runs a container in the background and returns to the main terminal
        # -p {INNER PORT NUMBER:OUTER PORT NUMBER} => publish flag exposes the specified inner port to the specified port outside the container since the app's runtime is isolated within a given docker container
        # -i => input flag runs the docker container in an interactive mode, which accepts inputs and responds dynamically to said inputs while running the app, by default docker is in non-interactive mode and does not accept user input
        # -t => terminal flag often used in conjunction with the -i input flag (-it), creates a pseudo-terminal that the shell can latch onto
    # docker ps => only shows running containers
        # -a => all flag specifies to show all containers
    # docker start {CONTAINER NAME} => starts the specified container
    # docker stop {CONTAINER NAME} => stops the specified container
    # docker rm {CONTAINER ID} => removes a docker container based on the specified container ID
    # docker exec {CONTAINER ID} {COMMAND} => runs a command in the specified container based on container ID
    # docker logs {CONTAINER ID} => displays console information logged by the specified container based on container ID

$ docker run parklane-cai-fan # assuming the container is not found on my machine, docker pulls the parklane-cai-fan image from Docker Hub, builds the container, and runs it
$ docker run -d parklane-cai-fan # detaches the container from the terminal
$ docker run 10f10 -p 3000:8000 # exposes port 8000 within the parklane-cai-fan container (of container ID 10f10) to the exterior port 3000 outside the container
$ docker stop parklane-cai-fan # stops the docker container parklane-cai-fan from running
$ docker start parklane-cai-fan # starts the docker container parklane-cai-fan and runs it

$ docker ps -a # displays all containers, including those not currently running

$ docker create isle-eating-house # creates a new container from the image isle-eating-house, but does not run it
$ docker rm 82f82 # removes the container isle-eating-house based on its container ID 82f82

$ docker images # displays all existing docker images on your machine
$ docker rmi # removes all docker images that do not currently have a docker container running
$ docker pull a1-dumplings # pulls the a1-dumplings image from Docker Hub and downloads it to our machine

$ docker exec 77b77 ls # runs the command 'ls' in the a1-dumplings container (of container ID 77b77)
$ docker logs 77b77 # displays the console output displayed to the stdout from the executed 'ls' command issued to the a1-dumplings container

$ docker build Desktop/coding/proj/Dockerfile # builds an image from the Dockerfile at the specified local file extension
```

Dockerfile

- file name must be `Dockerfile` with no extension (similar to a Makefile)
- blueprint of a Docker image
- enables further configuration of existing Docker base images