LINUX PERMISSION

**d**rwxrwxrwx

^ first ignored – tells you if drive

|  |  |  |
| --- | --- | --- |
| Sets | Permissions | Decimals |
| Owner | xxx  Group | xxx  Public | xxx | Read | 4  Write | 2  Execute | 1 | O G P  rwx rw- r--  7 6 4 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Permissions | | | Decimals | | |
| Owner | Group | Public | Owner | Group | Public |
| r-- | rw- | rwx | 4 | 6 | 7 |
| rwx | r-- | rw- | 7 | 4 | 6 |
| rw- | rwx | r-- | 6 | 7 | 4 |

$ chmod 0664 <thing>

DOCKER RUN PARAMS

-d: Run as a daemonized process  
--rm: When a container stops, it's deleted  
-i: Use STDIN  
-t: Allocates sudo terminal to interact with the terminal  
--add-host 'hostname:ip address': Add hostname and ip address to /etc/hosts in running container

GET SHELL ON RUNNING DOCKER CONTAINER:

docker exec -it <container name or id> /bin/bash

DEBUGGING

If you want to start debugging a Dockerfile you're working on, do the following:

docker run -d --name=<container name> <your name>/<image name> watch "echo 'test' >> /var/log/test.log"

Next, attach to it:

docker exec -it <name of running container> bash #/bin/sh for alpine

STOP & REMOVE DOCKER CONTAINERS:

docker stop $(docker ps -a -q)

docker rm $(docker ps -a -q)

COPY FILE FROM RUNNING CONTAINER TO HOST

docker cp <container name>:<path to file> <location on host machine to copy file to>

COPY FILE FORM HOST TO RUNNING CONTAINER

docker cp <file> <container name>:<where you want file to go on container>

# USE ENTRYPOINT SCRIPT

Create a file, docker-entrypoint.sh. Make sure it has the following in it:

#!/bin/bash

set -e

commands to run

exec "$@"

Be sure to run chmod +x docker-entrypoint.sh on the host system.

At the bottom of the Dockerfile:

COPY docker-entrypoint.sh /

ENTRYPOINT ["/docker-entrypoint.sh"]

**Resource:** <https://success.docker.com/article/use-a-script-to-initialize-stateful-container-data>

# ATTACH RUNNING CONTAINER TO EXISTING DOCKER NETWORK

docker network connect your-network-name container-name