DOCKER CHEAT SHEET – DOCKER CLI COMMANDS

Images

You can refer to images as the templates for the Docker containers. You can run the following Commands to work with the images:

- ***docker images***: *It will display all images.*
- ***docker import***: *You can create an image from a tarball.*
- ***docker build***: *You can create an image from Dockerfile.*
- ***docker commit***: *You can create an image from a container and temporarily pause it if it is running.*
- ***docker rmi***: *It will remove an image.*
- ***docker load***: *It will load an image from a tar archive as STDIN, including images and tags.*
- ***docker:*** *It will save an image to a tar archive stream to STDOUT with all parent layers, tags, & versions.*
- ***docker history***: *To display the history of an image*
- ***docker tag***: *It will tag the image to a name.*
- ***docker load < my_image.tar.gz***: *It will load an image from the mentioned file along with its history.*
- ***docker push repo[:tag]***: *It will push an image or repo from the registry.*
- ***docker pull repo[:tag]:*** *It will pull an image or repo from the registry*.

Container

Containers are the isolated Docker process that contains the code to be executed.

docker create: *It will create a container without starting it.*

docker rename: *To remane the container.*

docker run: *It will create and start the container in one task.*

docker rm: *It will delete a container.*

docker update: *It will update a container's resource limits.*

docker rm -v: *It will remove the volumes associated with the container.*

docker start: *It will start a container, so it is running.*

docker stop: *It will stop a running container.*

docker restart: *It stops and starts a container.*

docker pause: *It will pause a running container, "freezing" it in place.*

docker unpause: *It will unpause a running container.*

docker wait: *It will block until the running container stops.*

docker kill: *It sends a SIGKILL to a running container.*

docker ps: *It will display the running containers.*

docker logs: *Provide the logs from the container. (You can use a custom log driver, but logs are only available for json-file and journald in 1.10).*

docker inspect: *It checks all the information on a container (including IP address).*

docker port: *It will display the public-facing port of the container.*

docker top: *It will display the running processes in the container.*

docker stats: *It will display the containers' resource usage statistics.*

docker diff: *It will display the changed files in the container's FS.*

docker ps -a: *It will display the running and stopped containers.*

Dockerfile Cheat Sheet

It is a config file that will set up a Docker container whenever you run a docker build on it. To Create docker files, you can use any of the following text editors and their syntax highlighting Module.

The following are some instructions that you can use while working with Dockerfile:

- FROM: *It will set the Base Image for subsequent instructions.*

- RUN: *It will execute any commands in a new layer on top of the current image and then commit the results.*

- CMD: *It will offer the defaults for an executing container.*

- EXPOSE: *It will tell the Docker that the container listens on the specified network ports at runtime.*

- ADD: *It will copy the new files, directories, or remote files to the container.*

- COPY: *It will copy the new files or directories to a container. It copies as root regardless of the USER/WORKDIR settings by default. Use --chown=<user>:<group> to provide the ownership to another user/group.*

- ENTRYPOINT: *It will configure a container that will run as an executable.*

- VOLUME: *It will create a mount point for externally mounted volumes or other containers.*

- USER: *It will set the user name for the following RUN / CMD / ENTRYPOINT Commands.*

- WORKDIR: *It will set the working directory.*

Networks

Docker has a featured network, allowing the containers to connect. You can create three network interfaces with Docker, namely bridge, host, and none.

- **docker network create NAME:** *It will create a new network of bridge type by default.*

● **docker network rm NAME:** *It will remove one or more networks specified by name and make sure that no containers are connected to the deleted network.*

● **docker network ls:** *It will list all the networks.*

● **docker network inspect NAME:** *It will show the detailed information on one or more networks.*
● **docker network connect NETWORK CONTAINER:** *It will connect a container to a Network.*

● **docker network disconnect NETWORK CONTAINER**: *It will disconnect a container from a network.*

Volumes

Docker has volumes that are free-floating filesystems. So there is no need to be connected to a particular container. You can use volumes mounted from data-only containers for portability. As per Docker 1.9.0, it comes with the named volumes that replace data-only containers.

● **docker volume create:** *to create volumes.*

● **docker volume rm:** *To remove volumes.*

● **docker volume ls:** *To list the volumes.*

● **docker volume inspect:** *To inspect the volumes.*

Interaction with container

You can use the following commands to interact with the container.

● **Docker exe -ti container_name command.sh:** *It will run a command in the container.*

● **Docker logs -ft container name:** *It will follow the container log.*

● **Docker commit -m "commit message" -a "author" container_name username/image_name: tag:** *It will save the running container as an image.*

Build

You can use the following commands to build the images from a Docker file.

- **Docker build -t myapp :1.0-** *will build an image from the Docker file and tag it.*
- **Docker images-** *it will list all the images that are locally stored.*
- **Docker rmi alpine:** *3.4 will delete an image from the Docker Store.*

Cleanup

To optimize the usage of the resources, you need to clean up the resources frequently to maintain the performance. You can run the following commands to clean up resources.

- **Docker image prune:** *It will clean an unused/dangling image*
- **Docker image prune -a:** *It will remove an image not used in a container.*
- **Docker system prune:** *It will prune the entire system.*
- **Docker kills $ (docker ps -q):** *It will v.*
- **docker rm $(docker ps -a -q):** *It will delete all stopped containers*
- ***docker rmi $(docker images -q):*** *It will delete all images.*