docker

# DOCKER Cheat Sheet

Docker provides the ability to package and run an application in a loosely isolated environment called a Container. The isolation and security allows you to run many Containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host. You can easily share Containers while you work, and be sure that everyone you share with gets the same Container that works in the same way.

## INSTALLATION

**Docker Desktop is available for Mac, Linux and Windows**
https://docs.docker.com/desktop

**Docker Engine is available for Mac, Linux and Windows**
https://docs.docker.com/engine

**Check out docs for information on using Docker**
https://docs.docker.com

## IMAGES

Docker images are a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings. In simple words a Docker image is an executable file, that creates a Docker container

**List local Images**
```
docker image ls
```

**Delete an Image**
```
docker image rm <image> (or <image_id>)
```

**List Dangling Images**
```
docker images -f dangling=true
```

**Remove all unused Images**
```
docker image prune -a
```

**Remove all Dangling Images**
```
docker image prune
```

**Build an Image from a Dockerfile**
```
docker build -t <image>:<tag> .
```

**Build an Image from a Dockerfile without the cache**
```
docker build -t <image>:<tag> . –no-cache
```

**Retag a local Image**
```
docker tag <old_name>:<tag> <new_name>:<tag>
```

## DOCKER HUB

Docker Hub is a service provided by Docker for finding and sharing Docker images. We can create repositories from which we can push and pull the docker images, allowing us to share container images within our team, organization, customers. Learn more and find images at https://hub.docker.com

**Login into Docker Hub**
```
docker login
```

**Search Docker Hub for an Image**
```
docker search <image>
```

**Pull an Image from Docker Hub**
```
docker pull <image>
```

**Push an Image to Docker Hub**
```
docker push <username>/<image>:<tag>
```

## GENERAL COMMANDS

**Check Docker version**
```
docker version
```

**Get help with Docker. Can also use --help on all subcommands**
```
docker --help
```

**Inspect a Docker Object**
```
docker inspect <object_name> (or <object_id>)
```

## CONTAINERS

A Container is a runtime instance of a docker image. A Container will always run the same, regardless of the infrastructure.

**Create and run a Container from an image, with a custom name**
```
docker run --name <container_name> <image>
```

**Create and run a Container with Pseudo Terminal (Usually for OS)**
```
docker run --it <image>
```

**Run a Container and Publish a Container's Port to the Host**
```
docker run -p <host_port>:<container_port> <image>
```

**Run a Container in the Background**
```
docker run -d <image> (or <image_id>)
```

**List running Containers**
```
docker ps
```

**List all Docker Containers (running and stopped)**
```
docker ps -a
```

**Start a Stopped Container:**
```
docker start <container_name> (or <container_id>)
```

**Stop a Running Container:**
```
docker stop <container_name> (or <container_id>)
```

**Remove a Stopped Container**
```
docker rm <container_name> (or <container_id>)
```

**Remove all Stopped Containers**
```
docker container prune
```

**Remove all Running and Stopped Containers**
```
docker container rm -f $(docker ps -aq)
```

**Attach to an Running Container Process:**
```
docker attach <container_name> (or <container_id>)
```

**Open a Shell inside a Running Container**
```
docker exec --it <container_name> sh
```

**Fetch and follow the Logs of a Container**
```
docker logs -f <container_name> (or <container_id>)
```

**Set a Memory Limit for a New Container**
```
docker run -m <value> <image>
```

**Set Memory Reservation along with Memory Limit**
```
docker run -m <value> —memory-reservation <value> <image>
```

**Set CPU Limit for a New Container**
```
docker run --cpus <value> <image>
```

**View Container Resource usage Stats**
```
docker container stats
```

docker

# DOCKER Cheat Sheet

## VOLUMES

Volumes are used for persisting data generated by and used by Docker Containers. Bind Volume mounts are dependent on the directory structure and OS of the host machine, Docker Volumes are completely managed by Docker.

**Create a Docker Volume**
```
docker volume create <volume_name>
```

**List Docker Volumes**
```
docker volume ls
```

**Mount a Docker Volume to a new Container**
```
docker run -v <volume_name>:<container_path> <image>
```

**Create a Read-Only Volume Mount**
```
docker run -v <volume_name>:<container_path>:ro <image>
```

**Delete a Docker Volume**
```
docker volume rm <volume_name>
```

**Delete all unused Docker Volumes**
```
docker volume prune
```

**Mount a Bind Mount to a new Container**
```
docker run -v <host_path>:<container_path> <image>
```

## NETWORKS

Docker Networks enable containers communication with other containers and also with the Docker Host. By default, Docker creates three network drivers called bridge, host and none.

**Create a Docker Network**
```
docker network create —driver bridge <network_name>
```

**List Docker Networks**
```
docker network ls
```

**Attach a Docker Network to a new Container**
```
docker run --network <network_name> <image>
```

**Create a connection between Container and another Network**
```
Docker network connect <network_name> <container_name>
```

**Remove the connection between Container and another Network**
```
Docker network disconnect <network_name> <container_name>
```

**Delete a Docker Network**
```
docker network rm <network_name>
```

**Remove all unused Docker Networks**
```
docker network prune
```

## DOCKERFILE INSTRUCTIONS

A Dockerfile is a text document that contains all the commands used to update the base image.
Default Name: Dockerfile

**FROM: Used to set a base Image**
```
FROM <image>:<tag>
```

**RUN: Used to run specified Commands**
```
RUN <command>
```

**CMD: Used to set the default Execution Point**
```
CMD ["<command>"]
```

**ENTRYPOINT: Used to set the default Execution Point, Subsequent CMD's passed are taken as arguments**
```
ENTRYPOINT ["<command>"]
```

**COPY: Used to Copy Files from Host Machine to Containers**
```
COPY <host_path> <container_path>
```

**ADD: Copy files, Extracted archives and Files from URL's to containers**
```
ADD <host_path/archive_file/url> <container_path>
```

**ENV: Used to set Environment Variables**
```
ENV <key>=<value>
```

**ARG: Used to define Variables, whose value passed with Docker Build**
```
ARG <variable>
```

**EXPOSE: Used to Expose a container**
```
EXPOSE <port>
```

**WORKDIR: Used to set a working directory in Container**
```
WORKDIR <path>
```

**USER: Used to set an User for Dockerfile Instructions**
```
USER <username>
```

**SHELL: Used to set an environment for executing Run Instructions**
```
SHELL ["<executable>", “<parameters>"]
```