**Docker Cheatsheet**

**Docker Commands with Images**

* docker pull image-name: Pull the image from Docker Hub.
* docker images: Show available images.
* docker images ls --filter 'reference=image-name:tag': Show images with a specific name and tag.
* docker rmi imgae-name/image-id: Remove an image by name or ID.
* docker image prune: Remove unused images.

**Docker Command to Find Images from Docker Hub**

* docker search image-name --filter 'is-official=true': Search for official images.

**Docker Container**

* docker ps or docker container ls: Show all running containers.
* docker container ls --filter 'status=exited' --filter 'ancestor=ubuntu': Show only containers with the Ubuntu image that have exited.
* docker container inspect container-name: Show detailed information about a container.

**Run Container**

* docker run image-name: Run a container based on an image.
* docker run --rm image-name: Run a container based on a specified image and remove it once it stops.
* docker rm container-name: Remove a container.
* docker rm $(docker container ls --filter 'status=exited' -q): Remove all exited containers.
* docker start image-name/id: Start a container.
* docker stop image-name/id: Stop a container.

**Port Mapping**

Port mapping is used for accessing a container from outside of the local machine. It can only be applied during the creation of a container; after creation, container port mapping is not allowed.

* -p own\_port:image\_port: Manual port mapping.
* -P: Automatic port mapping.

**Additional Options**

* --name give-name: During the creation of an image or container, use this to give it a name.
* -it: Interact with an image's terminal (e.g., Ubuntu for bash, Windows for cmd, and DB for shells).
* -d: Run a container in detached mode (in the background).

**Docker Exec**

This command is used to access the terminal of running containers.

Example: docker exec -it container-id terminal-name

**Show Container Logs**

* docker logs container-name: Show logs for a container.

**Docker Network Commands**

* docker network create give-name: Create a network.
* docker network ls: Show a list of networks.

**Docker Compose**

* apt install docker-compose: Install Docker Compose.
* docker-compose -f filename.yaml up: Run the container but not in detached mode.
* docker-compose -f filename.yaml up -d: Run the container in detached mode.
* docker-compose -f filename.yaml down: Stop and remove the container.

**docker-compose.yml**

version: "3"

services:

mongo:

image: mongo

ports:

- 27017:27017

environment:

- MONGO\_INITDB\_ROOT\_USERNAME=admin

- MONGO\_INITDB\_ROOT\_PASSWORD=pass

mongo-express:

image: mongo-express

ports:

- 8081:8081

environment:

- ME\_CONFIG\_MONGODB\_ADMINUSERNAME=admin

- ME\_CONFIG\_MONGODB\_ADMINPASSWORD=pass

- ME\_CONFIG\_MONGODB\_PORT=27017

- ME\_CONFIG\_MONGODB\_SERVER=mongo

**Dockerfile**

Create your own custom image using a Dockerfile.

* docker build -t name-file .: Create an image using a Dockerfile.

### Example Dockerfile for a Node App

FROM node # this official image name this file for node image

RUN mkdir -p /home/app # using RUN syntax we run any command in sh

COPY . /home/app # copy project from current directory to give directory . means current work directory

CMD['node', '/home/app/index.js'] # CMD MEANS this command run continuously

### Example Dockerfile for a Java Web App

# Use an official Tomcat image as the base image

FROM tomcat:8.5-jdk8-openjdk

# Set the working directory to /usr/local/tomcat/webapps

WORKDIR /usr/local/tomcat/webapps

# Copy the WAR file to the webapps directory

COPY target/myapp.war /usr/local/tomcat/webapps/myapp.war

# Expose the Tomcat server port

EXPOSE 8080

# Set the default command to start the Tomcat server

CMD ["catalina.sh", "run"]