DOCKER CHEATSHEET

sudo usermod -aG docker $USER removes the sudo constraint from docker commands

docker images List locally available images

docker run <image> start a new container from the image
options :-
docker run --name <container> <image> assign a name to the container and start it
-d starts a container in backend
-p <host-port:container-port> map all ports to container
-it interact with the image through command line
-e, --env NAME=hello environment variables

docker rmi <Image> or docker image rm <image> Delete image

docker save -o <archive-file> <image> Saving images to a tar archive

docker pull <image> pull a docker image from a registry or a repository

docker build -t <image> Builds an image from a "dockerfile"

docker image prune Removes all unnecessary docking images

docker create <options> image
options :-
-a, --attach attach stdout/err
-i, --interactive attach stdin (interactive)
--name NAME name your image
-p, --publish 5000:5000 port map
--expose 5432 expose a port to linked containers
-P, --publish-all publish all ports
-v, --volume `pwd`:/app mount (absolute paths needed)
-e, --env NAME=hello environment variables

docker ps List of containers started

docker ps -a List of all containers

docker build -t <Image>: <Tag> Builds a Docker image from a "Dockerfile" or a container.

docker login -u <username> Connects to a remote repository(dockerhub)

docker search <image> To search image in remote repository (dockerhub)

docker push < Image>: <Tag > Push image to remote repository

docker pull < Image> Extracts the image from the remote repository

docker logs <container-name or ID> Fetches the logs of a container

docker start <container> start a container

docker stop <container name> stop a container

docker-compose up Start all services defined in the docker-compose.yml file

docker-compose up -d Starts the containers and leaves them running

docker-compose down Stop and remove all containers, networks, and volumes associated with the services defined in the docker-compose.yml file

docker-compose ps List all containers started by the docker-compose up command, along with their status

docker-compose logs <service> View the logs of a specific service

docker-compose build Build or rebuild all Docker images defined in the docker-compose.yml file

docker-compose stop Stop all containers started by docker-compose up command without removing them

docker kill <container> forced shutdown of running container

docker commit <image ID> <image-name> save a running docker container as an image

docker rm <container> remove a stopped container

docker-compose restart Restart all containers or a specific service

docker-compose exec -it <container-name or ID> bash Open a shell inside a running container

docker-compose pull Pull updated images for services from their respective repositories

docker-compose up --scale <service>=<no. of replicas> Scale a number of containers for a specific service as prescribed in yaml file

docker run -d -p 8000:800 --mount source=<created volume name>, target=/app <image_name> mount volume to a container and run it

docker volume create <Volume> Create a volume

docker volume create --name <volume-name> --opt type=none --opt device=<path of volume directory> --opt o=bind creating volume with options

docker volume ls Image controls

docker volume inspect <volume> Display detailed information on one or more volumes

docker volume rm <Volume> Remove volume

docker volume prune Removes unused volumes

docker network ls List of networks

docker network inspector < Network > Control network information

docker network create < Network > Create a network

docker network rm < Network > Removes a network

docker network connect <Network> <Container> Connect a container to the network

docker network connect --ip < IP > <Network> <Container> Specifies the IP address of the container interface

docker network disconnect <Network> <Container> Disconnect the network container

docker swarm init enable first node of docker system

docker swarm join --token <token> --listen -addr <ip:port> add a node to a swarm cluster

docker swarm join-token retrieve the join token

docker node ls list nodes in a cluster

docker node rm <node-name> remove a node from swarm cluster

docker service <service> --replicas <no. of times> <image> autoscaling services with set number of times

docker service ls list services in docker swarm

docker service ps <service> list containers in a service

docker service rm <service> remove a service

Dockerfile

Dockerfile is a text file it contains some set of instructions to automate docker image creation.

FROM This will set a base image for your Dockerfile, which means that subsequent instructions will be applied to this base image. For the base image this command must be on top of the dockerfile.

RUN It is used to execute commands, it will create a layer in the docker file.

MAINTAINER Author/Owner/Description

COPY Copy files from local system, we need to provide a source and destination.(we cant download from internet using COPY)

ADD Similar to COPY, it provides a feature to download files from the internet,also we can extract the file from the docker image side.

EXPOSE To expose ports (eg- 3306 for mysql)

WORKDIR To set a working directory for a container

CMD To execute commands but during container creation.

ENTRYPOINT Similar to CMD, but has higher priority over CMD, the first commands will be executed by ENTRYPOINT.

ENV Environment variables

ARG To define the name of a parameter and its default value, the difference between ENV and ARG is that after you set ENV using ARG you will not be able to access that later on when you try to run the docker container.

docker-compose.yaml

Basic docker-compose.yaml file example
(yaml Works in key-value format, for eg- key is version value is 3.9)

```
version: "3.9"
services:
  my_app:
      container_name: "django-todo-app"
      build: .
      ports:
      - 8000:8000
      volumes:
      - django-todo-volume:/app
  my_db:
      container_name: "django-mysql-db"
      image: mysql:5.7
      ports:
      - 3306:3306
      environment:
      MYSQL_ROOT_PASSWORD: "test@123"
volumes:
  django-todo-volume:
```

Checkout my GitHub for more cheatsheets

- Nishant Sharma