Yagna®

docker

# Cheat Sheet

## Docker

Docker is a platform or ecosystem around creating and running containers.

Docker is a set of coupled software-as-a-service and platform-as-a-service products that use operating-system-level virtualization to develop and deliver software in packages called containers. The software that hosts the containers is called Docker Engine.It was first started in 2013 and is developed by Docker, Inc.

Docker is an open platform for developers, it’s a mechanism that helps in isolating the dependencies per each application by packing them into containers. Containers are scalable and safer to use and deploy as compared to previous approaches.

## Docker Cheat-Sheet

### Container Commands

```
docker container ls

docker container ls -a

docker container run -d -p 3306:3306 --name=localMysql -e MYSQL_ROOT_PASSWORD=abc123 -v
myvolume:/var/lib/mysql

--network mynetwork mysql:5.7

docker container logs -f <name-of-container or id>

docker container exec -it <name-of-container or id> <cmd>

docker container rm <name-of-container or id>

docker container prune

docker container inspect <name-of-container>
```

### Image Commands

```
docker image pull <imagename>:<tag>

docker image ls

docker image rm

docker image build -t <docker-hub-username>/<image-name>:<tag>

docker image prune
```

### Volume commands

```
docker volume create <name>

docker volume ls

docker volume inspect <name-of-volume>

docker volume rm <name-of-volume>

docker volume prune
```

**Network commands**

docker network create <name>

docker network ls

docker network inspect <name-of-volume>

docker network rm <name-of-volume>

docker network prune

**docker-compose commands**

docker-compose up -d

docker-compose logs

docker-compose down

docker-compose stop

docker-compose restart

**Important Link to refer**

Link 1: To install docker and docker-compose on local env
https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04

Link 2: To play with docker swarm we need a cluster, for that you can follow below link which will give you ready-made cluster
https://labs.play-with-docker.com/

Link 3: Best practises for Dockerfile from docker specification page
https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

Link 4: For all docker commands follow the Docker specification
pagehttps://docs.docker.com/engine/reference/commandline/docker/

Link 5: Please refer the sample microservice i have created over the GitHub Repository which contains Dockerfile, docker-compose.yaml and docker-stack.yaml for docker related operations. Do read the README.md file for running instructions
Github: https://github.com/docker-for-all/microservice-for-docker-demo

**UseCase 1:** Use hello-world(existing image from docker hub) image to run our first containerized application

Step 1: docker image pull hello-world

Step 2: docker container run --name=MyHelloWorldContainer hello-world

Step 3: docker container ls

Step 4: docker container ls -a

Step 5: docker container logs -f MyHelloWorldContainer

Step 6: docker container inspect MyHelloWorldContainer

Step 7: docker container stop MyHelloWorldContainer

Step 8: docker container rm MyHelloWorldContainer

**UseCase 2:** Use tomcat image from docker hub to create container and expose port 8080

Step 1: docker image pull tomcat

Step 2: docker container run -d -name=mytomcatcontainer -p 8080:8080 tomcat

Step 3: docker container ls

Step 4: docker container exec -it mytomcatcontainer bash

Step 5: docker container logs -f mytomcatcontainer

Step 6: docker container stop mytomcatcontainer

Step 7: docker container rm mytomcatcontainer

**UseCase 3:** MySQL container example

Create Network:
docker network create myNetwork
NOTE: by default the network created is of type "bridge".

Create Volume:
docker volume create myData
NOTE: by default the volume created is of type "local”

Step 1: docker image pull mysql:5
Step 2: docker container run

```
        --name=myMySqlDB
        -e MYSQL_ROOT_PASSWORD=Yagna123
        --network myNetwork
        -v mydata:/var/lib/mysql
        -p 3306:3306
        -d mysql:5
```

**UseCase 4:** Create our own image using Dockerfile and push it to docker hub

EXAMPLE 1: Image for JDK 1.8

Step 1: Create a dockerfile with name as Dockerfile.JDK-1.8

Step 2: Please find dockerfile with the above mentioned name and see the code structure

Step 3: docker image build -f Dockerfile.JDK-1.8 -t cloudgeekview/my-jdk:1.8.
OR
docker image build -f Dockerfile.JDK-1.8.
docker tag cloudgeekview/my-jdk:1.8

Step 4: docker login

Step 5: docker push cloudgeekview/my-jdk:1.8

EXAMPLE 2: Create an image for a microservice.

NOTE: Please refer
Github: https://github.com/docker-for-all/microservice-for-docker-demo

**UseCase 5:** Run multiple container at once using docker-compose file

Step 1: Create a new microservice which will call another database service

Step 2: Create database service using mysql database.

Step 3: Create a dockerfile for the new microservice created above.

Step 4: Build microservice image and push it to docker hub.

Step 5: Create docker-compose.yaml file which will contain two services named as microservice and database.

Step 6: docker-compose up -d --> To create containers defined in docker-compose file

Step 7: docker-compose down  --> To stop running containers

## Docker Swarm

All Swarm objects can and should be described in manifests called stack files; these YAML files describe all the components and configurations of your Swarm app, and can be used to easily create and destroy your app in any Swarm environment.

1) docker swarm

2) Set up a swarm with single node

```
                docker swarm init
                         OR
               docker swarm init --advertise-addr 192.168.100.230
```

3) Create our newtork of type overlay
docker network create --driver overlay my-network

4) Create two services

- microservice docker service create -d -p 9999:9999 --network my-network --name=microservice cloudgeekview/microservice
- database    docker service create -d -e MYSQL_ROOT_PASSWORD=Yagna123 -p 3306:3306 --network my-network mysql:5
  docker service ls
  docker container ls

5) docker node ls

6) docker service logs -f microservice

### Docker Stack

Its syntax is almost same as the syntax of docker compose with few additional attributes.

- Create a file with name as docker-stack.yml which will contain 2 services i.e, microservice and database.
- To deploy the Stack
  docker stack deploy -c docker-stack.yaml
- docker stack ls
- docker stack ps <name-of-stack>
- docker service ps <service-name>

NOTE: Main difference btw services and container is that services are resilient to failure.

## Other useful Commands

```
docker version
docker --version
docker info
docker --help
docker events
docker container stats <container-name or container-id>
docker-compose --help
docker-compose up
docker-compose down
docker-compose stop
docker-compose start
docker-compose restart <service-name>
docker-compose logs -t -f --tail <no of lines>
docker-compose logs -t -f --tail <no of lines> <name-of-service1> <name-of-service2> ... <name-of-service N>
```