



# DOCKER CHEAT SHEET

## FOR JAVA DEVELOPERS

### DOCKER

#### List all Docker Images

```
docker images -a
```

#### List All Running Docker Containers

```
docker ps
```

#### List All Docker Containers

```
docker ps -a
```

#### Start a Docker Container

```
docker start <container name>
```

#### Stop a Docker Container

```
docker stop <container name>
```

#### Kill All Running Containers

```
docker kill $(docker ps -q)
```

#### View the logs of a Running Docker Container

```
docker logs <container name>
```

PAGE 1



SPRINGFRAMEWORK.GURU

## DOCKER CHEAT SHEET

### Delete All Stopped Docker Containers

Use -f option to nuke the running containers too.

```
docker rm $(docker ps -a -q)
```

### Remove a Docker Image

```
docker rmi <image name>
```

### Delete All Docker Images

```
docker rmi $(docker images -q)
```

### Delete All Untagged (dangling) Docker Images

```
docker rmi $(docker images -q -f dangling=true)
```

### Delete All Images

```
docker rmi $(docker images -q)
```

### Remove Dangling Volumes

```
docker volume rm -f $(docker volume ls -f dangling=true -q)
```

### SSH Into a Running Docker Container

Okay not technically SSH, but this will give you a bash shell in the container.

```
sudo docker exec -it <container name> bash
```

### Use Docker Compose to Build Containers

Run from directory of your docker-compose.yml file.

```
docker-compose build
```

## DOCKER CHEAT SHEET

### Use Docker Compose to Start a Group of Containers

Use this command from directory of your docker-compose.yml file.

```
docker-compose up -d
```

This will tell Docker to fetch the latest version of the container from the repo, and not use the local cache.

```
docker-compose up -d --force-recreate
```

This can be problematic if you're doing CI builds with Jenkins and pushing Docker images to another host, or using for CI testing. I was deploying a Spring Boot Web Application from Jenkins, and found the docker container was not getting refreshed with the latest Spring Boot artifact.

#stop docker containers, and rebuild

```
docker-compose stop -t 1
```

```
docker-compose rm -f
```

```
docker-compose pull
```

```
docker-compose build
```

```
docker-compose up -d
```

### Follow the Logs of Running Docker Containers With Docker Compose

```
docker-compose logs -f
```

Save a Running Docker Container as an Image

```
docker commit <image name> <name for image>
```

## DOCKER CHEAT SHEET

Follow the logs of one container running under Docker Compose

```
docker-compose logs pump <name>
```

### Add Oracle Java to an Image

For CentOS/ RHEL

```
ENV JAVA_VERSION 8u31
```

```
ENV BUILD_VERSION b13
```

```
# Upgrading system
```

```
RUN yum -y upgrade
```

```
RUN yum -y install wget
```

```
# Downloading & Config Java 8
```

```
RUN wget --no-cookies --no-check-certificate --header "Cookie:
oraclelicense=accept-securebackup-cookie"
```

```
"http://download.oracle.com/otn-pub/java/jdk
/$JAVA_VERSION-$BUILD_VERSION/jdk-$JAVA_VERSION-linux-
x64.rpm" -O /tmp/jdk-8-linux-x64.rpm
```

```
RUN yum -y install /tmp/jdk-8-linux-x64.rpm
```

```
RUN alternatives --install /usr/bin/java jar /usr/java/latest/bin/java
200000
```

```
RUN alternatives --install /usr/bin/javaws javaws /usr/java/latest
/bin/javaws 200000
```

```
RUN alternatives --install /usr/bin/javac javac /usr/java/latest/bin/javac
200000
```

## DOCKER CHEAT SHEET

### Add / Run a Spring Boot Executable Jar to a Docker Image

ADD /maven/myapp-0.0.1-SNAPSHOT.jar myapp.jar

RUN sh -c 'touch /myapp.jar'

ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/myapp.jar"]

### Show Running Containers

docker ps

### Show All Containers - Running and stopped

docker ps -a

### Default Tag

'latest' is selected if no other value is specified

### Run A Docker Image

docker run <image name>

### See the Console Output of a Docker Container

docker logs <container name>

### Build a docker image

From the directory of the Dockerfile run:

docker build -t <tag name>

### Stop a docker container

docker kill <container name>

or

docker stop <container name>

## DOCKER CHEAT SHEET

**Parameter that tells docker to run the container as a background process**

**-d**

Example: `docker run -d <image name>`

**List all docker images on your system**

`docker images`

**Map a Host Port to a Container Port**

`-p <host port>: <container port>`

Example:

`docker run -p 8080:8080 <image name>`

**Tail the Console Output of a Running Docker Container**

`docker logs -f <container name>`

**A .java file to a docker image - i.e. the source code**

The Dockerfile

**Remove a Stopped Docker Container**

`docker rm <container name>`

**Specify an Environment Variable for a Docker Container**

`docker run -e MY_VAR=my_prop <image name>`

**Remove a Docker Image from your System**

`docker rmi <image name>`

**Shell into a Running Docker Container**

`docker exec -it <container name> bash`

PAGE 6



## DOCKER CHEAT SHEET

### Share Storage on a Host System with a Docker container

-v <host path>: <container path>

Example:

docker run -v <host path>: <the container path> <image name>

### Name of the Maven plugin we are using for the Course

Fabric8

### Map a Host Port to a Container Port

-p <host port>:<container port>

Example:

docker run -p 8080:8080 <image name>

### Maven Command to Stop Running Image(s)

mvn docker:stop

### Maven Command to Build a Docker Image

mvn clean package

docker:build

### Remove a Stopped Docker Container

docker rm <container name>

### Maven Command Used to Publish a Docker Image to its Repository

mvn docker:push

### Maven Command Used To Start a Docker Image

mvn docker:start

### Run Containers in the Background from Maven

mvn docker:start

PAGE 7



## DOCKER CHEAT SHEET

### XML Tag that has the Runtime Parameters for the Fabric8 Plugin

```
<image>  
<run>  
**params here**  
</run>  
</image>
```

### Map a Host Port to a Container Port in Maven Configuration

```
<ports>  
<port>8080:8080</port>  
</ports>
```

### Parameter that Creates a Network Host Name Reference for a Docker Container to Another Container

```
"--link" {dash dash}  
--link <container name>:<hostname>
```

### Specify Environment Variable for a Docker Container in Maven Configuration

```
<env>  
<parameter_name>{value}</parameter_name>  
</env>
```

### Maven Command Used To Start a Docker Image Interactively

```
mvn docker:run
```

### Where to Store Credentials for Docker Hub

```
~/.m2/settings.xml
```



## DOCKER CHEAT SHEET

Example:

```
<servers>
  <server>
    <id>docker.io</id>
    <username>springframeworkguru</username>
    <password>YourPasswordHere</password>
  </server>
</servers>
```

## DOCKER SWARM

**Is Docker Swarm automatically enabled?**

No, by default, Docker Swarm is not available

**Types of Nodes in a Docker Swarm**

Manager and worker

**Enable the First Node of a Docker Swarm**

docker swarm init

**List Running Services**

docker service ls

**Add a Node to a Swarm Cluster**

docker swarm join --token <token> --listen-addr <ip:port>

**Can manager nodes run containers?**

Yes, manager nodes normally run containers

## DOCKER CHEAT SHEET

### Retrieve the Join Token

`docker swarm join-token`

### List Nodes in a Cluster

`docker node ls`

### Can you run a 'docker node ls' from a worker node?

No. Docker Swarm commands can only be from manager nodes

### List Services in a Docker Swarm

`docker service ls`

### List Containers in a Service

`docker service ps <service name>`

### Remove a Service

`docker service rm <service name>`

### Remove a Node from a Swarm Cluster

`docker node rm <node name>`

### Promote a Node from Worker to Manager

`docker node promote <node name>`

### Change a Node from a Manager to a Worker

`docker node demote <node name>`

### Map a Host Port to a Container Port

`-p <host port>: <container port>`

Example:

`docker run -p 8080:8080 <image name>`