## List Docker CLI commands

docker

docker container --help

## Display Docker version and info

docker --version

docker version

docker info

## Execute Docker image

docker run hello-world

## List Docker images

docker image ls

**# login to terminal of image**

docker run -t -i <<image name>> /bin/bash

**# login to container**

docker exec -it <container name> /bin/bash

# **ip address of container**

docker inspect b63bdaed85cc

## List Docker containers (running, all, all in quiet mode)

docker container ls

docker container ls --all

docker container ls -aq

**Import and export images**

Docker images can be saved using image save command to a .tar file:

docker image save helloworld > helloworld.tar

These tar files can then be imported using load command:

docker image load -i helloworld.tar

**CONTAINER**

docker build -t friendlyhello . # Create image using this directory's Dockerfile

docker run -p 4000:80 friendlyhello # Run "friendlyname" mapping port 4000 to 80

docker run -d -p 4000:80 friendlyhello # Same thing, but in detached mode

docker container ls # List all running containers

docker container ls -a # List all containers, even those not running

docker container stop <hash> # Gracefully stop the specified container

docker container kill <hash> # Force shutdown of the specified container

docker container rm <hash> # Remove specified container from this machine

docker container rm $(docker container ls -a -q) # Remove all containers

docker image ls -a # List all images on this machine

docker image rm <image id> # Remove specified image from this machine

docker image rm $(docker image ls -a -q) # Remove all images from this machine

docker login # Log in this CLI session using your Docker credentials

docker tag <image> username/repository:tag # Tag <image> for upload to registry

docker push username/repository:tag # Upload tagged image to registry

docker run username/repository:tag # Run image from a registry

**COMMANDS ON CONTAINER**

**login to docker container**: docker exec -it <mycontainer> bash

apt-get update

apt-get install yum

**SERVICES**

docker stack ls # List stacks or apps

docker stack deploy -c <composefile> <appname> # Run the specified Compose file

docker service ls # List running services associated with an app

docker service ps <service> # List tasks associated with an app

docker inspect <task or container> # Inspect task or container

docker container ls -q # List container IDs

docker stack rm <appname> # Tear down an application

docker swarm leave --force # Take down a single node swarm from the manager

**SWARMS**

docker-machine create --driver virtualbox myvm1 # Create a VM (Mac, Win7, Linux)

docker-machine create -d hyperv --hyperv-virtual-switch "myswitch" myvm1 # Win10

docker-machine env myvm1 # View basic information about your node

docker-machine ssh myvm1 "docker node ls" # List the nodes in your swarm

docker-machine ssh myvm1 "docker node inspect <node ID>" # Inspect a node

docker-machine ssh myvm1 "docker swarm join-token -q worker" # View join token

docker-machine ssh myvm1 # Open an SSH session with the VM; type "exit" to end

docker node ls # View nodes in swarm (while logged on to manager)

docker-machine ssh myvm2 "docker swarm leave" # Make the worker leave the swarm

docker-machine ssh myvm1 "docker swarm leave -f" # Make master leave, kill swarm

docker-machine ls # list VMs, asterisk shows which VM this shell is talking to

docker-machine start myvm1 # Start a VM that is currently not running

docker-machine env myvm1 # show environment variables and command for myvm1

eval $(docker-machine env myvm1) # Mac command to connect shell to myvm1

& "C:\Program Files\Docker\Docker\Resources\bin\docker-machine.exe" env myvm1 | Invoke-Expression # Windows command to connect shell to myvm1

docker stack deploy -c <file> <app> # Deploy an app; command shell must be set to talk to manager (myvm1), uses local Compose file

docker-machine scp docker-compose.yml myvm1:~ # Copy file to node's home dir (only required if you use ssh to connect to manager and deploy the app)

docker-machine ssh myvm1 "docker stack deploy -c <file> <app>" # Deploy an app using ssh (you must have first copied the Compose file to myvm1)

eval $(docker-machine env -u) # Disconnect shell from VMs, use native docker

docker-machine stop $(docker-machine ls -q) # Stop all running VMs

docker-machine rm $(docker-machine ls -q) # Delete all VMs and their disk images

**STACKS**

**MY DOCKER IMAGES**

1. **ubuntu-18.04-with-java8**

dockerfiile: <https://github.com/lovetaneja/docker/blob/master/ubuntu-18.04-with-java8/dockerfile>

docker build . -t love3001/java-repository:ubuntu-18.04-with-java8

docker run -t -i love3001/java-repository:ubuntu-18.04-with-java8

java -version

docker push love3001/java-repository:ubuntu-18.04-with-java8

1. **hello-world**

dockerfile: <https://github.com/lovetaneja/docker/tree/master/docker-helloworld>

docker build . -t love3001/java-repository:hello-world

docker run -t -i love3001/java-repository:hello-world /bin/bash

docker push love3001/java-repository:hello-world

1. **tomcat-9.0.18**

dockerfile: <https://github.com/lovetaneja/docker/blob/master/tomcat/dockerfile>

docker build . -t love3001/java-repository:tomcat-9.0.18

docker run -d -p 8080:8080 love3001/java-repository:tomcat-9.0.18

docker push love3001/java-repository:tomcat-9.0.18

1. **Run Tomcat with Docker Stack (using docker-compose.yml)**

docker-compose file: <https://github.com/lovetaneja/docker/blob/master/tomcat/docker-compose.yml>

docker stack deploy -c docker-compose.yml tomcatStack

docker stack ls

docker stack ps tomcatStack

docker container ls

docker exec -it <container name> /bin/bash

docker inspect <container id>

docker stack rm tomcatStack

1. **Run MySQL Instances with Adminer**

docker-compose file: <https://github.com/lovetaneja/docker/blob/master/mysql/docker-compose.yml>

docker stack deploy -c docker-compose.yml mysqlStack

<http://localhost:8080>

1. **SWARMS (using AWS)**

docker-machine create --driver amazonec2 --amazonec2-access-key AKIAICAAAOOFRHNFONFQ --amazonec2-secret-key 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 aws01

docker-machine env aws01

eval $(docker-machine env aws01)

docker-machine ls

docker swarm init

docker node ls

docker-machine ssh aws01

docker stack deploy -c docker-compose.yml tomcatOnAWS

Add a rule in security group for port 8080 INBOUND RULE.

docker container ls

eval $(docker-machine env -u)

docker-machine rm aws01

Got error: Error creating machine: Error checking the host: Error checking and/or regenerating the certs: There was an error validating certificates for host "54.224.150.194:2376": dial tcp 54.224.150.194:2376: connect: connection refused

docker-machine regenerate-certs aws01

docker-machine restart aws01

1. **LOGS**

docker container ls 🡪 get the container id

docker logs <<container id>>

1. **Deploy WAR file in Docker Image**

<http://dockerimage.blogspot.com/2017/05/deploy-war-file-to-docker-image.html>

1. **Volumes**

docker volume create tomcatVolum

docker volume ls

docker volume inspect tomcatVolume

**command to log into docker host(VM) on mac:**

screen ~/Library/Containers/com.docker.docker/Data/vms/0/tty

ENTER

1. **Jenkins on AWS**

docker-machine create --driver amazonec2 --amazonec2-access-key AKIAICAAAOOFRHNFONFQ --amazonec2-secret-key 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 aws01

OR

docker-machine create --driver virtualbox vm01

docker-machine env aws01

eval $(docker-machine env aws01)

docker-machine ls

docker swarm init

docker node ls

docker stack deploy -c docker-compose.yml jenkinsStack

hit URL: <http://192.168.99.100:8080/>

adminuser: love/love

Check logs: docker service ls

docker service logs l3o6hzwev81q

eval $(docker-machine env -u)

Specify memory of Virtual Machine in Virtual Box: docker-machine create --driver virtualbox --virtualbox-memory "2048" vm01

Command for help:

docker-machine create -d virtualbox –help

Create a github configuration in Jenkins – Add Githug server and generate token to connect to your github account.

Configure Jenkins to install java and maven on box if it is not installed already.

Restarted docker container without removing the docker host and verified that data remains there.

**Install docker within Jenkins Container**

<https://getintodevops.com/blog/the-simple-way-to-run-docker-in-docker-for-ci>

<https://stackoverflow.com/questions/47854463/got-permission-denied-while-trying-to-connect-to-the-docker-daemon-socket-at-uni>

**Login as root user on Container:**

docker exec -it -u root dde77831804c bash

**Add Jenkins user to docker group**

sudo usermod -a -G docker Jenkins

**Jenkins Pipeline Script**

node {

stage('Initialize'){

def dockerHome = tool 'Love-Docker'

env.PATH = "${dockerHome}/bin:${env.PATH}"

}

stage('docker') {

docker.image('hello-world').run()

}

}

**Useful Links**

<https://serverfault.com/questions/725262/what-causes-the-connection-refused-message>

<https://www.daveperrett.com/articles/2006/08/16/opening-a-port-on-linux/>

Problem: docker: Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post http://%2Fvar%2Frun%2Fdocker.sock/v1.39/containers/create: dial unix /var/run/docker.sock: connect: permission denied.

Solution:

sudo usermod -a -G docker ubuntu

sudo usermod -a -G docker jenkins

Check /etc/group that user has been added to group. Check the group id for docker group is same on host and container.

Restart the Jenkins container.

docker run -d -p 8080:8080 -p 50000:50000 -v /home/docker/jenkins\_home:/var/jenkins\_home -v /var/run/docker.sock:/var/run/docker.sock jenkins/jenkins:lts

sudo chown -R 1000:1000 /home/docker/jenkins\_home

node() {

stage('Run Tomcat Image') {

docker.withRegistry('https://registry.hub.docker.com', 'docker-hub-credentials'){

docker.image('love3001/java-repository:tomcat-9.0.18').run('-p 8080:8080')

}

}

}