**Setup your own personal lab if you can using the below document - a free google cloud account with 300$ Credit**

<https://github.com/lerndevops/docker/blob/master/01-install/GCP-Account-Setup.pdf>

441 sudo su -

442 docker -v

443 sudo wget https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installDocker.sh -P /tmp

444 sudo chmod 755 /tmp/installDocker.sh

445 sudo bash /tmp/installDocker.sh

446 docker -v

447 docker version

448 clear

449 history

450 clear

451 cd /var/lib/docker/

452 ls -l

453 ps -ef |grep dockerd

454 systemctl status docker.service **# press q on your keyboard to come out**

455 systemctl restart docker.service

<https://github.com/lerndevops/docker/tree/master/01-install>

ps -ef | grep dockerd

docker image ls # to list images on your server

docker pull nginx:1.21

docker pull nginx

clear

docker pull tomcat:latest

docker pull mongo:latest

clear

docker image ls

docker image inspect tomcat:latest

clear

docker image rm tomcat:latest

docker image ls

History

docker image ls

docker run nginx:1.21

docker run -d nginx:1.21

docker run -d tomcat:latest

clear

docker ps -a # to see the containers running/exited

docker ps # to see only the runnin containers

history

clear

docker ps -a

docker start 0144124cef5c

docker ps -a

docker stop cfdddf218479

docker ps -a

docker restart 49fed9ddc24b

docker ps -a

clear

docker ps -a

docker kill bab9261d91ef

docker ps -a

docker start bab9261d91ef

docker ps -a

clear

docker ps

docker image ls

docker run -d nginx:latest

docker ps -a

docker rm e8cf4408db94

docker rm --force e8cf4408db94

docker rm --force 0b1fd22169fd bab9261d91ef 0144124cef5c

docker ps -a

517 docker logs 2e7dc04c2308

518 docker logs cfdddf218479

519 clear

520 docker ps

521 docker inspect cfdddf218479

522 curl 172.17.0.4:8080

523 history

524 docker ps

525 docker stats

526 curl 172.17.0.4:8080

538 docker ps

539 docker run -d -P 3f8a00f137a0 ## -P will generate port dynamically

540 docker ps

541 docker run -d -P 3f8a00f137a0

542 docker ps

543 docker run -d -p 9080:8080 tomcat:8.5.40 ## -p allows you to configure your own port

547 echo $SHELL

548 ls -l /bin/bash

549 docker ps

550 docker exec -it fd4baa8eae99 /bin/bash

551 docker ps

552 docker exec fd4baa8eae99 ls -l

553 docker exec fd4baa8eae99 java -version

554 java -version

555 docker exec fd4baa8eae99 vi

583 cd

584 ls -l

585 sleep 10

586 docker run -d ubuntu:18.04 sleep 30

587 docker ps -a

588 docker run -d ubuntu:18.04 sleep 30

589 docker ps -a

590 clear

591 docker ps

592 docker top fd4baa8eae99

593 ps -ef

594 clear

595 docker ps

596 docker exec fd4baa8eae99 ps -ef

597 docker exec 2e7dc04c2308 ps -ef

598 docker exec 2e7dc04c2308 top

599 docker top fd4baa8eae99

600 history

583 cd

584 ls -l

585 sleep 10

586 docker run -d ubuntu:18.04 sleep 30

587 docker ps -a

588 docker run -d ubuntu:18.04 sleep 30

589 docker ps -a

590 clear

591 docker ps

592 docker top fd4baa8eae99

593 ps -ef

594 clear

595 docker ps

596 docker exec fd4baa8eae99 ps -ef

597 docker exec 2e7dc04c2308 ps -ef

598 docker exec 2e7dc04c2308 top

599 docker top fd4baa8eae99

600 history

604 docker rm -f `docker ps -qa`

605 clear

606 docker ps -a

607 docker image ls

608 docker pull tomcat:8.5.40

609 docker run -d tomcat:8.5.40

610 docker ps

611 docker exec -it 9ec9eeb9c6d4 /bin/bash

Inside the container

1 ls -l

2 cd /usr/local/tomcat/webapps/

3 ls -l

4 wget https://github.com/lerndevops/code/raw/main/sampleapp.war

5 ls -l

6 history

7 cat /etc/os-release

8 vi hello.txt

9 apt-get update

10 apt-get install -y vim

11 vim --version

12 clear

13 exit

612 docker ps

613 docker image ls

614 docker ps -s

615 docker commit -m "deployed sampleapp" 9ec9eeb9c6d4 sampleapp:v1

616 docker image ls

617 docker rm -f 9ec9eeb9c6d4

618 docker ps

619 docker run -d -P sampleapp:v1

620 docker ps

621 docker logs f47fcfc45c99

622 docker ps

vi abc.txt -- opens the file in read only mode

to edit the file, press 'i' on keyboard for INSERT mode & then you can write anything

once required text is entered press 'ESC' on keyboard to back readonly mode

press ':wq' on keyboard -- to save & come out of the file

press ':q' on keyboard -- to come out of the file without saving

press ':wq!' on keyboard -- to save forcefully & come out of the file

press ':q!' on keyboard -- to come out of the file forcefully without saving

( note: vi is an file editor, you can use any editor you like by installing it on your machine )

vi Dockerfile # the file name can be anything

FROM docker.io/library/tomcat:8.5.45

# FROM is a pre defined dockerfile instruction to set base image on top of which we would like to make our customizations

# FROM statement will pull the base image / run it as cont / logs into cont

RUN cd /usr/local/tomcat/webapps/ ; wget https://github.com/lerndevops/code/raw/main/sampleapp.war

RUN apt-get update

RUN apt-get install -y vim

RUN touch /tmp/hello.txt

# RUN instruction helps to execute any command to make changes on top of base image during the image build process

628 cat Dockerfile

631 docker build --file Dockerfile --tag sampleapp:v2 /root

632 docker image ls

633 docker run -d -P sampleapp:v2

634 docker ps

639 wget https://github.com/lerndevops/code/raw/main/jdk-8u331-linux-x64.tar.gz

640 wget https://github.com/lerndevops/code/raw/main/apache-tomcat-9.0.63.tar.gz

641 wget https://github.com/lerndevops/code/raw/main/myapp.war

FROM ubuntu:18.04

#COPY source-file-on-the-vm path-inside-cont-where-to-copy

# COPY just copies the file as they are

COPY jdk-8u331-linux-x64.tar.gz /tmp

RUN tar -xzf /tmp/jdk-8u331-linux-x64.tar.gz -C /opt

RUN rm /tmp/jdk-8u331-linux-x64.tar.gz

RUN mv /opt/jdk1.8.0\_331 /opt/java

ENV JAVA\_HOME /opt/java

ENV JAVA\_VERSION 1.8

#ADD source-file-on-the-vm path-inside-cont-where-to-copy

# ADD extracts the tar in the mmeory & copies only the extracted content to container

ADD apache-tomcat-9.0.63.tar.gz /opt

RUN mv /opt/apache-tomcat-9.0.63 /opt/tomcat

COPY server.xml /opt/tomcat/conf

ENV TOMCAT\_HOME /opt/tomcat

ENV TOMCAT\_VERSION 9.0.63

COPY myapp.war $TOMCAT\_HOME/webapps/

EXPOSE 9090

#CMD ["executable"]

#CMD ["executable", "arg1", "arg..n"]

CMD ["/opt/tomcat/bin/catalina.sh", "run"]

651 cat myappdfile

652 docker build --file /root/myappdfile --tag myapp:v1

653 docker build --file /root/myappdfile --tag myapp:v1 /tmp

654 docker build --file /root/myappdfile --tag myapp:v1 /root

655 docker image ls

656 docker run -d -P myapp:v1

657 vi myappdfile

658 docker build --file /root/myappdfile --tag myapp:v2 /root

659 docker image ls

660 docker run -d -P myapp:v2

661 docker ps

662 docker ps -a

663 docker logs 4a3fa0f4ec98

664 vi myappdfile

665 docker build --file /root/myappdfile --tag myapp:v2 /root

666 docker run -d -P myapp:v2

667 docker ps -a

668 vi myappdfile

669 docker build --file /root/myappdfile --tag myapp:v3 /root

670 docker image ls

671 docker run -d -P myapp:v3

672 docker ps -a

673 cat myappdfile

674 tar -xzf apache-tomcat-9.0.63.tar.gz

675 ls -l

676 cd apache-tomcat-9.0.63/

677 ls -l

678 cd conf/

679 ls -l

680 vi server.xml

681 cp server.xml /root/

682 cd

683 ls -l

684 vi myappdfile

685 docker build --file /root/myappdfile --tag myapp:v4 /root

686 docker image ls

687 docker run -d -P myapp:v4

688 docker ps

689 cat myappdfile

DAY3:

FROM ubuntu:18.04

RUN useradd -ms /bin/bash naresh

USER naresh

RUN touch /tmp/hello.txt

CMD ["sleep", "500"]

754 vi user-dfile

755 docker build --file user-dfile --tag myubuntu:user .

756 docker image ls

757 docker run -d myubuntu:user

761 docker exec -it 799a20247004 /bin/bash

FROM ubuntu:18.04

CMD ["sleep", "6000"]

702 vi cmd-dfile

703 docker build --file cmd-dfile --tag myubuntu:cmd .

704 docker image ls

705 docker image inspect myubuntu:cmd

706 clear

707 docker run -d myubuntu:cmd

708 docker ps

709 docker run -d myubuntu:cmd sleep 900

710 docker ps

711 docker run -d myubuntu:cmd ls -l

712 docker run -d myubuntu:cmd pwd

713 docker ps -a

FROM ubuntu:18.04

ENTRYPOINT ["sleep", "6000"]

716 vi entrypoint-dfile

717 docker build --file entrypoint-dfile --tag myubuntu:entry .

718 docker image ls

719 clear

720 docker run -d myubuntu:entry

721 docker ps -a

722 docker run -d myubuntu:entry ls -l

723 docker run -d myubuntu:entry pwd

724 docker ps -a

FROM ubuntu:18.04

ENTRYPOINT ["sleep"]

CMD ["600"]

727 vi cmd-entry-dfile

728 docker build --file cmd-entry-dfile --tag myubuntu:entry-cmd .

729 docker image ls

730 docker run -d myubuntu:entry-cmd

731 docker ps

732 docker run -d myubuntu:entry-cmd 900

733 docker ps

FROM alpine:latest

RUN mkdir -p /home/naresh

WORKDIR /home/naresh

RUN touch hello.txt

CMD ["sleep", "800"]

781 vi workdir-dfile

782 docker build --file workdir-dfile --tag myalpine:workdir .

783 docker run -d myalpine:workdir

784 docker ps

785 docker exec -it 2a0ac01c8693 /bin/sh

FROM maven:3.6.2-jdk-8 AS stage1

# stage1 is an alias name for build stage

RUN git clone https://github.com/lerndevops/samplejavaapp

WORKDIR /samplejavaapp

RUN mvn package

FROM tomcat:8.5.45

WORKDIR /usr/local/tomcat/webapps/

COPY --from=stage1 /samplejavaapp/target/sampleapp.war .

# importing the .war from build stage

790 vi multi-dfile

791 docker build --file multi-dfile --tag sampleapp:multi-stage .

792 docker image ls

793 cat multi-dfile

794 docker image inspect sampleapp:multi-stage

801 docker login

802 docker push docker.io/library/sampleapp:multi-stage

803 docker push sampleapp:multi-stage

804 docker build --file multi-dfile --tag docker.io/lerndevops/slearnapp:sampleapp-multi-stage .

805 docker image ls

806 docker pull gcr.io/google-containers/busybox:1.27

807 docker image ls

808 docker push docker.io/lerndevops/slearnapp:sampleapp-multi-stage

809 history

810 docker image ls

811 docker tag myapp:v4 docker.io/lerndevops/slearnapp:myapp-v4

812 docker image ls

813 docker push lerndevops/slearnapp:myapp-v4

829 docker image ls

830 docker save --output myimages.tgz myubuntu:cmd myalpine:workdir alpine:latest

831 ls -l

832 scp myimages.tgz labuser@172.31.1.97:/tmp

833 docker image ls

834 docker image prune -a

835 docker image ls

836 docker ps -a

837 docker container prune -a

838 docker container prune

839 docker ps

840 docker ps -a

841 docker rm -f 9e4a37fbd70d e5766d0d525c

842 docker image prune -a

843 docker image ls

844 docker ps -a

845 docker system prune

846 history

847 clear

848 docker image ls

849 ls -l

850 docker load --input myimages.tgz

851 docker image ls

854 docker run -d -p 5000:5000 --restart always --name registry registry:2

855 docker ps

856 docker image ls

857 docker tag myalpine:workdir localhost:5000/lerndevops/myapp:v1

858 docker image ls

859 docker push localhost:5000/lerndevops/myap:v1

860 docker push localhost:5000/lerndevops/myapp:v1

861 docker image ;s

862 clear

863 docker ps

864 docker image prune -a

865 docker image ls

866 docker pull localhost:5000/lerndevops/myapp:v1

867 docker image ls

868 docker volume ls

**DAY4**

39 docker network ls

40 ifconfig

41 docker network inspect bridge

42 docker run -d --name cont1 lerndevops/samples:netshoot

43 docker run -d --name cont2 lerndevops/samples:netshoot

44 docker ps

45 docker inspect cont1

46 docker inspect cont2

47 docker ps

48 docker exec cont1 ifconfig

49 docker exec cont2 ifconfig

50 docker exec cont2 ping 172.17.0.2

51 docker exec cont1 ping 172.17.0.3

52 docker exec cont1 curl 172.17.0.3:80

53 docker exec cont1 ip route

54 docker ps

55 docker network ls

56 docker network --help

57 docker network create --help

58 docker network create mynet --driver bridge --subnet 23.24.0.0/16 --gateway 23.24.0.1

59 docker network ls

60 docker network inspect mynet

61 docker run -d --name cont3 --network mynet lerndevops/samples:netshoot

62 docker run -d --name cont4 --network mynet lerndevops/samples:netshoot

63 docker ps

64 docker exec cont3 ifconfig

65 docker exec cont4 ifconfig

66 clear

67 docker exec cont1 hostname -i

68 docker exec cont2 hostname -i

69 docker exec cont3 hostname -i

70 docker exec cont4 hostname -i

71 docker exec cont3 ping 23.24.0.3

72 docker exec cont3 ping 172.17.0.2

73 docker exec cont3 ping cont4

74 docker exec cont4 ping cont3

75 docker exec cont2 ping cont1

81 docker exec cont1 ifconfig

82 docker network connect mynet cont1

83 docker exec cont1 ifconfig

84 docker exec cont1 ping cont3

85 docker exec cont1 ping cont4

86 docker network disconnect mynet cont1

87 docker exec cont1 ping cont3

88 docker exec cont1 ifconfig

90 docker network connect bridge cont3

98 docker network ls

99 docker run -d --name cont5 --network none lerndevops/samples:netshoot

100 docker exec cont5 ifconfig

101 docker run -d --name cont6 --network host lerndevops/samples:netshoot

102 docker ps

103 curl localhost:80

104 docker run -d --name cont7 --network host lerndevops/samples:netshoot

105 docker ps -a

106 docker logs 6ead1c24e868

Docker volumes concept

123 docker volume ls

124 docker volume create applogs

125 docker volume ls

126 docker volume inspect applogs

127 cd /var/lib/docker/volumes/applogs/\_data

128 ls -l

129 cd

130 docker run -d -P --mount type=volume,src=applogs,target=/usr/local/tomcat/logs tomcat:latest

131 docker ps

132 cd /var/lib/docker/volumes/applogs/\_data

133 ls -l

134 touch hello.txt

135 ls -l

136 docker exec -it bce67532ed60 /bin/bash

137 ls -l

138 docker ps

139 docker rm -f bce67532ed60

140 docker ps

141 ls -l

142 history

143 cd

144 docker run -d -P --volume applogs:/usr/local/tomcat/logs tomcat:latest

145 dockerdocker volume inspect applogs

146 docker volume inspect applogs

147 ls -l /var/lib/docker/

148 mkdir /opt/nginxlogs

149 mkdir /opt/nginxconf

150 docker run -d -P --mount type=bind,src=/opt/nginxlogs,target=/var/log/nginx --mount type=bind,src=/opt/nginxconf,target=/usr/share/nginx/html nginx:latest

151 docker ps

152 docker run -d -P --volume /opt/nginxlogs:/var/log/nginx -v /opt/nginxconf:/usr/share/nginx/html nginx:latest

153 cd /opt/nginxlogs/

154 ls -l

155 cat error.log

156 docker ps

157 ls -l

158 cat access.log

159 tail -f access.log

160 ls -l

161 cd /opt/nginx

162 cd /opt/nginxconf/

163 ls -l

164 echo "hi there from vm" > index.html

168 docker run -d -it --name tmptest --mount type=tmpfs,destination=/app nginx:latest

169 cd

170 df -h

171 ls -l

172 docker ps

173 docker exec -it 78640479c4ae /bin/bash

187 docker image ls

188 docker image history nginx:latest

189 cd /var/lib/docker/

190 ls -l

191 cd image/

192 ls -l

193 cd overlay2/

194 ls -l

195 cd layerdb/

196 ls -l

197 cd sha256/

198 ls -l

199 cd fbb6f6171fb4fd604480ac82d038457520c107eff25a5cf8a242476b03bdd233

200 ls -l

201 cd ..

202 docker image ls

203 cd ..

204 ls -l

205 cd ..

206 ls -l

207 cd imagedb/

208 ls -l

209 cd content/

210 ls -l

211 cd sha256/

212 ls -l

213 docker image ls

214 cd e86a46dc6799f01353c5d623884d6e9ad6994dba4ef5308c339aa48675bf7cfd

215 cat e86a46dc6799f01353c5d623884d6e9ad6994dba4ef5308c339aa48675bf7cfd

216 clear

217 cd

218 docker ps

219 docker exec -it 078ae8e9ae02 /bin/bash

220 docker ps -s

221 docker image ls

222 docker commit -m "alkdj" 078ae8e9ae02 test:v1

223 docker image ls

224 docker image history tomcat:latest

225 docker image history test:v1

226 clear

227 docker ps

228 docker export 078ae8e9ae02 myimage.tgz

229 docker export 078ae8e9ae02

230 docker export 078ae8e9ae02 -o myimage.tgz

231 ls -l

232 docker import -i myimage.tgz

233 docker import myimage.tgz

234 docker image ls

235 docker import myimage.tgz test:v2

236 docker image ls

237 docker history test:v2

DAY 5

sudo vi /lib/systemd/system/docker.service

ExecStart=/usr/bin/dockerd -H fd:// **-H tcp://0.0.0.0:4243**

systemctl daemon-reload

sudo service docker restart

test: curl http://localhost:4243/version

<https://github.com/lerndevops/docker/blob/master/02-daemon/enable-remote-API.txt>

7 systemctl status docker.service

8 clear

9 docker version

10 ps -ef|grep dockerd

11 cat /etc/os-release

12 sudo vi /lib/systemd/system/docker.service

13 systemctl daemon-reload

14 sudo service docker restart

15 sudo service docker status

16 docker version

17 curl http://localhost:9908/version

18 docker info

19 clear

20 docker ps

21 docker run -d -P nginx:latest

USE CLI on Different VM to connect to a remote docker daemon / server

6 docker version

7 systemctl stop docker.socket

8 docker version

9 clear

10 docker version

11 ping 172.31.5.135

12 export DOCKER\_HOST=172.31.5.135:9908

13 docker version

14 ps -ef|grep dockerd

15 docker run -d -P nginx:latest

Disable the remote docker host

18 unset DOCKER\_HOST

19 docker version

20 systemctl start docker.socket

21 docker ps

22 docker version

root@vm1:/var/log# cat /etc/docker/daemon.json

{

"data-root": "/mnt/docker",

"log-driver": "syslog",

"storage-driver": "devicemapper"

}

39 mkdir /mnt/docker

40 ls -l /mnt/docker/

41 docker ps

42 docker rm -f `docker ps -qa`

43 docker image ls

44 docker image rm nginx:latest snyk/snyk

45 clear

46 cd /etc/docker/

47 vi daemon.json

48 df -h /

49 cat daemon.json

50 systemctl restart docker.service

51 ls -l /mnt/docker/

52 docker pull nginx:latest

53 docker run -d -P nginx:latest

54 docker image inspect nginx:latest

55 docker ps

56 docker inspect 494039c19e49

57 cat daemon.json

58 ls -l /var/lib/

59 clear

60 ls -l /var/lib/docker/

61 ls -l /mnt/docker/

62 docker ps

63 docker logs 494039c19e49

64 docker inspect 494039c19e49

65 cat /mnt/docker/containers/494039c19e494021fbc513e8e66f9b0b0d34d459d07c3e2ba81c2233fc64bf52/494039c19e494021fbc513e8e66f9b0b0d34d459d07c3e2ba81c2233fc64bf52-json.log

66 docker logs 494039c19e49402

67 docker info | grep -i logging

68 docker info

69 cd /var/log/

70 ls

71 vi syslog

72 docker info

73 vi /etc/docker/daemon.json

74 docker info | grep -i logging

75 systemctl restart docker.service

76 docker info | grep -i logging

77 docker run -d -P nginx:latest

78 docker ps

79 docker inspect 2882be896a27

80 docker ps

81 docker logs 2882be896a27

82 vi /var/log/syslog

83 docker logs 2882be896a27

84 ls -l /etc/docker/

85 docker info

86 vi /etc/docker/daemon.json

87 systemctl restart docker.service

88 docker ps -a

89 docker info | grep -i storage

90 cat /etc/docker/daemon.json

<https://docs.docker.com/storage/storagedriver/select-storage-driver/>

<https://github.com/lerndevops/docker/blob/master/09-docker-ee/About%20daemon.json>

109 docker pull jelastic/tomcat:11.0.0-M1-temurinjdk-17.0.6

110 export DOCKER\_CONTENT\_TRUST=1

111 docker pull jelastic/tomcat:11.0.0-M1-temurinjdk-19.0.2

112 docker pull alpine

114 unset DOCKER\_CONTENT\_TRUST

115 docker pull jelastic/tomcat:11.0.0-M1-temurinjdk-19.0.2

<https://raw.githubusercontent.com/lerndevops/docker/master/07-security/docker-content-trust.txt>

127 vi compose.yaml

128 cat compose.yaml

129 docker compose --file compose.yaml up -d

130 docker ps

131 docker compose --file compose.yaml up --scale myapp=5 --scale mydb=3 -d

132 docker ps

133 docker compose --file compose.yaml up --scale myapp=3 -d

134 docker ps

135 docker compose --file compose.yaml down

136 history

volumes:

applogs: # docker volume create applogs

networks:

mynet: # docker network create mynet --driver bridge

services:

myapp: # this is the service name can be any string

image: nginx:latest

ports:

- 80 # -P / 8080:80 # -P

#- 443

volumes:

#- systempath:contpath

- applogs:/var/log/nginx # docker managed volumes

- /opt/nconf:/usr/share/nginx/html # bind mount

networks:

- mynet

mydb:

image: mongo

ports:

- 27017

networks:

- mynet

Set hostname on a vm

sudo hostnamectl set-hostname <anyname>

sudo hostnamectl set-hostname master-node

sudo hostnamectl set-hostname worker-node01

sudo hostnamectl set-hostname worker-node02

Logout & login back (use exit command)

On master node

18 docker info | grep -i swarm

19 docker swarm init

20 docker swarm join-token worker

21 docker node ls

On worker nodes

3 docker info | grep -i swarm

4 docker swarm join --token SWMTKN-1-1pwp4wy86k73ch5bg90zx9aktffo7zey0vfr1qkkk4n4iiqdxg-9a0q5maha7e871g4aumwidjuv 10.128.0.13:2377

**DAY6**

25 docker node ls

26 docker service create --name myapp **--mode replicated** --replicas 7 -p 9080:3000 lerndevops/samples:pyapp-v1

27 docker service create --name mydb **--mode replicated** --replicas 3 -p 9081:27017 mongo:latest

28 docker service ls

29 docker service ps myapp

30 docker service ps mydb

31 docker service scale myapp=12

32 docker service ps myapp

33 docker service scale myapp=3

34 docker service ps myapp

35 docker service logs myapp

36 docker service ls

37 docker service rm mydb

38 docker service ls

40 docker service create --name testgm **--mode global** -p 9099:80 nginx:latest

41 docker node ls

42 docker service ls

43 docker service ps testgm

44 docker service scale testgm=10

50 docker service ls

51 docker ps

52 docker service scale myapp=10

53 docker service logs --follow myapp

54 docker service scale myapp=1

55 docker service logs --follow myapp

56 docker service scale myapp=10

56 docker service scale myapp=10

57 history

58 docker service ls

59 docker service update --help

60 docker service ls

61 docker service update myapp --image lerndevops/samples:pyapp-v2 --update-parallelism 2 --update-delay 30s --update-failure-action rollback --update-order start-first

62 docker service ls

63 docker service rollback myapp

volumes:

data:

data-bkp:

networks:

myoverlay:

services:

springbootapp:

image: lerndevops/samples:springboot-app

deploy:

replicas: 2

placement:

constraints:

- node.labels.role==app

restart\_policy:

condition: on-failure

resources:

limits:

cpus: "0.2"

memory: 512M

ports:

- "9090:8080"

depends\_on:

- mongo

networks:

- myoverlay

mongo:

image: lerndevops/samples:mongodb

deploy:

replicas: 1

placement:

constraints:

- node.labels.role==db

restart\_policy:

condition: on-failure

ports:

- "27017:27017"

volumes:

- data:/data/db

- data-bkp:/data/bkp

networks:

- myoverlay

65 vi app.yaml

66 ls -l

68 docker stack deploy -c app.yaml myappstack

69 docker stack ls

70 docker stack ps myappstack

71 docker stack services myappstack

72 docker service ls

73 docker service ps myappstack\_springbootapp

74 docker inspect c8v47q9m1qj1

75 clear

76 docker node ls

77 docker node inspect sworker-node01

78 clear

79 docker node ls

80 docker node update sworker-node01 --label-add role=app

81 docker node inspect sworker-node01 | grep -iA 1 labels

82 docker node inspect sworker-node01

83 docker service ls

84 docker service ps myappstack\_springbootapp

85 docker node update sworker-node02 --label-add role=db

86 docker service ps myappstack\_springbootapp

87 docker service ls

88 docker service ps myappstack\_mongo

89 docker service scale myappstack\_mongo --replicas 0

90 docker service scale myappstack\_mongo=0

91 docker service ps myappstack\_mongo

92 docker service ls

93 docker service logs -f myappstack\_springbootapp

94 docker service scale myappstack\_mongo=

95 docker service scale myappstack\_mongo=1

97 docker stack ls

98 docker stack rm myappstack

99 docker service ls

100 docker service rm myapp testgm

**Uninstall Docker on a VM / Node**

wget <https://raw.githubusercontent.com/lerndevops/labs/master/scripts/uninstallDocker.sh> -P /tmp

sudo bash /tmp/uninstallDocker.sh

33 docker ps | grep apiserver

34 kubectl get nodes

35 wget https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installKubectl.sh -P /tmp

36 sudo bash /tmp/installKubectl.sh

37 kubectl get nodes

38 sudo apt-get install -y jq

39 sudo apt-get install -y unzip

40 AUTHTOKEN=$(curl -sk -d '{"username":"admin","password":"admin@1234"}' https://10.142.0.5/auth/login | jq -r .auth\_token)

41 curl -k -H "Authorization: Bearer $AUTHTOKEN" https://10.142.0.5/api/clientbundle -o $HOME/client-bundle.zip

42 ls -l

43 unzip $HOME/client-bundle.zip -d $HOME/client-bundle

44 cd client-bundle && eval "$(<env.sh)"

45 kubectl get nodes

46 kubectl get nodes -o wide

47 kubectl get pods -o wide

48 clear

49 cd

50 kubectl get nodes -o wide

51 kubectl run pod1 --image=nginx

52 kubectl get pods -o wide

53 kubectl describe pod pod1

54 kubectl get pods -o wide

55 kubectl logs pod1

56 kubectl describe pod pod1

57 clear

58 kubectl logs pod1 -c pod1

59 kubectl logs pod1 -c cont2

60 kubectl exec -it pod1 -c pod1 -- /bin/bash

61 kubectl get pods -o wide

62 curl 192.168.2.67:80

63 kubectl get nodes

64 kubectl get pod

65 kubectl delete pod pod1

66 history

67 kubectl logs pod1 -c cont2

68 clear

69 kubectl create deployment pyapp --image=lerndevops/samples:pyapp-v1

70 kubectl get deployment -o wide

71 kubectl get pods -o wide

72 kubectl scale deployment pyapp --replicas 4

73 kubectl get pods -o wide

74 kubectl scale deployment pyapp --replicas 3

75 kubectl get pods -o wide

76 kubectl get deployment -o wide

77 kubectl describe deployment pyapp

78 kubectl get pods -o wide

79 kubectl delete pod pyapp-799d6f579d-j6pc4 pyapp-799d6f579d-qv648 pyapp-799d6f579d-wktfl

80 kubectl describe deployment pyapp

81 kubectl get pods -o wide

82 clear

83 kubectl get pod -o wide

84 kubectl get pods -o wide --show-labels

85 vi deployment.yaml

86 kubectl apply --filename deployment.yaml

87 kubectl get deployment -o wide

88 kubectl get pods -o wide

89 cat deployment.yaml

90 kubectl get pods -o wide

91 curl 192.168.2.72:3000 ; echo

92 kubectl get pods -o wide --show-labels

93 curl 192.168.139.22:3000 ; echo

94 vi service.yaml

95 kubectl apply -f service.yaml

96 kubectl get services -o wide

97 kubectl get pods -o wide --show-labels

98 kubectl describe service myapp-svc

99 curl 192.168.139.21:3000 ; echo

100 curl 10.96.238.176:80 ; echo

101 kubectl scale deployment mydeployment --replicas 1

102 kubectl get pods -o wide --show-labels

103 kubectl describe service myapp-svc

104 kubectl scale deployment mydeployment --replicas 5

105 kubectl get pods -o wide --show-labels

106 kubectl describe service myapp-svc

107 clear

108 kubectl get services

109 kubectl delete service myapp-svc

110 vi np-svc.yaml

111 kubectl apply -f np-svc.yaml

112 vi np-svc.yaml

113 kubectl apply -f np-svc.yaml

114 kubectl get services -o wide

115 kubectl get namespaces

116 kubectl get pods -o wide

117 kubectl get pods -o wide --namespace default

118 kubectl get pods -o wide --namespace kube-public

119 kubectl get pods -o wide --namespace kube-system

120 kubectl create namespace teama

121 kubectl create namespace teamb

122 kubectl create namespace dev

123 kubectl create namespace qa

124 kubectl create namespace uat

125 kubectl create namespace app1

126 kubectl create namespace app2

127 kubectl get namespaces

128 clear

129 ls -l

130 cat np-svc.yaml

131 kubectl apply -f np-svc.yaml

132 kubectl get services --namespace default

133 kubectl apply -f np-svc.yaml --namespace dev

134 kubectl apply -f deployment.yaml --namespace dev

135 kubectl apply -f deployment.yaml --namespace qa

136 kubectl apply -f deployment.yaml --namespace teama

137 kubectl apply -f deployment.yaml --namespace teamb

138 kubectl get deployment -o wide --namespace dev

139 kubectl get deployment -o wide --namespace qa

140 kubectl get deployment -o wide --all-namespaces

141 kubectl get pod -o wide --all-namespaces

142 kubectl delete deployment mydeployment

143 kubectl get deployment -o wide --all-namespaces

144 kubectl get deployment -o wide --namespace qa

apiVersion: apps/v1

kind: Deployment

metadata:

name: mydeployment

namespace: default

labels: # are like tags # are optional

app: myapp

spec:

replicas: 4 # number of pods to be created

selector: # is mandatory # helps the controller to select the pods it created & manage them

matchLabels:

app: myapp

template: # what pod to be created

metadata:

labels: # like tags, helps to identify a group, in kube we must/mandatoryly have assing a label to every pod

app: myapp # here key & value are you choice

spec:

containers:

- name: myapp-cont

image: lerndevops/samples:pyapp-v1

ports:

- containerPort: 3000

- name: cont2

image: nginx

ports:

- containerPort: 80

kind: Service

apiVersion: v1

metadata:

name: myapp-svc

namespace: defult

#labels: # are optional

spec:

#type: ClusterIP # it is an internal virtual Load Balancer (kube will create a dynamica LB IP) that can forward a req into a single/group of pods

selector:

app: myapp

ports:

- name: cont1 # name can be anything

port: 80 # this is the port to be used along with virutal lb created by kube

targetPort: 3000 # this is port of the app inside the cont always

- name: cont2

port: 81

targetPort: 80

kind: Service

apiVersion: v1

metadata:

name: myapp-svc

#labels: # are optional

spec:

type: NodePort # it will publish a nodeport/hostport on every node/vm in the cluter & also creates

# an internal virtual Load Balancer (kube will create a dynamica LB IP) that can forward a req into a single/group of pods then

# then maps the nodeport to clusterip:port

selector:

app: myapp

ports:

- name: cont1 # name can be anything

port: 80 # this is the port to be used along with virutal lb created by kube

targetPort: 3000 # this is port of the app inside the cont always

nodePort: 30001 # -p

- name: cont2

port: 81

targetPort: 80

#nodePort: 30004 # -P

apiVersion: apps/v1

kind: Deployment

metadata:

name: hp-vol-dep

namespace: default

labels: # are like tags # are optional

app: app1

spec:

replicas: 2 # number of pods to be created

selector: # is mandatory # helps the controller to select the pods it created & manage them

matchLabels:

app: app1

template: # what pod to be created

metadata:

labels: # like tags, helps to identify a group, in kube we must/mandatoryly have assing a label to every pod

app: app1 # here key & value are you choice

spec:

volumes:

- name: hpvol # the name can be any string

hostPath:

path: /applogs # this is path on node/worker where the pod will run

containers:

- name: myapp-cont

image: tomcat:latest

ports:

- containerPort: 3000

volumeMounts:

- name: hpvol

mountPath: /usr/local/tomcat/logs # this is the path inside the cont

197 kubectl get namespaces

198 kubectl get pods --all-namespaces

199 vi deployment.yaml

200 kubectl apply -f deployment.yaml

201 kubectl get deployment -o wide

202 kubectl get services

203 vi np-svc.yaml

204 kubectl apply -f np-svc.yaml

205 kubectl get services

206 kubectl get pods -o wide

207 kubectl describe service myapp-svc

208 kubectl get deployment -o wide

209 vi deployment.yaml

210 kubectl apply -f deployment.yaml

211 kubectl rollout status deployment pyapp

212 vi deployment.yaml

213 kubectl rollout undo deployment pyapp

214 kubectl rollout status deployment pyapp

215 kubectl get pods

216 kubectl scale deployment pyapp --replicas 2

217 vi hp-vol-dep.yaml

218 kubectl apply -f hp-vol-dep.yaml

219 kubectl get pods -o wide

220 cd /applogs/

221 ls -l

222 touch hello.txt

223 ls -l

224 docker exec -it hp-vol-dep-bd66c69-4vwtc -- /bin/bash

225 kubectl exec -it hp-vol-dep-bd66c69-4vwtc -- /bin/bash

226 ls -l

227 kubectl delete deployment hp-vol-dep

228 kubectl get pods -o wide

kind: PersistentVolume

apiVersion: v1

metadata:

name: pv-hp

spec:

hostPath:

path: /applogs

capacity:

storage: 2Gi

accessModes:

- ReadWriteOnce # only one node/vm can read/write the data into the volume - ReadWriteMany/ReadOnlyMany

kind: PersistentVolumeClaim

apiVersion: v1

metadata:

name: app1-pvc

spec:

volumeName: pv-hp

resources:

requests:

storage: 2Gi

accessModes:

- ReadWriteOnce

apiVersion: apps/v1

kind: Deployment

metadata:

name: pv-hp-vol-dep

namespace: default

labels: # are like tags # are optional

app: app1

spec:

replicas: 2 # number of pods to be created

selector: # is mandatory # helps the controller to select the pods it created & manage them

matchLabels:

app: app1

template: # what pod to be created

metadata:

labels: # like tags, helps to identify a group, in kube we must/mandatoryly have assing a label to every pod

app: app1 # here key & value are you choice

spec:

volumes:

- name: pvhpvol # the name can be any string

persistentVolumeClaim: #hostPath:

claimName: app1-pvc #path: /applogs # this is path on node/worker where the pod will run

containers:

- name: myapp-cont

image: tomcat:latest

ports:

- containerPort: 3000

volumeMounts:

- name: pvhpvol

mountPath: /usr/local/tomcat/logs # this is the path inside the cont

232 df -h /

233 clear

234 cd

235 vi pv.yaml

236 kubectl apply -f pv.yaml

237 kubectl get pv -o wide

238 kubectl describe pv pv-hp

239 vi pvc.yaml

240 kubectl apply -f pvc.yaml

241 kubectl get pvc -o wide

242 kubectl get pv -o wide

243 cd pa

244 cd /applogs/

245 ls -l

246 rm -rf \*

247 cd

248 clear

249 vi pv-hp-vol-dep.yaml

250 kubectl apply -f pv-hp-vol-dep.yaml

251 kubectl get pods -o wide

252 cd /applogs/

253 ls -l

254 kubectl get pv,pvc

258 docker node ls

259 ls

260 vi devdb.props

261 docker config create devdbconf2 devdb.props

262 docker config ls

263 docker config inspect devdbconf

264 docker config inspect devdbconf2

265 base64 --help

266 echo -n "nareshadbadminlalkdjf" | base64

267\* echo "REJIT1NUOiAiMTkuNS42LjciCkRCUE9SVDogIjEzNDUiCkRCTkFNRTogImRldmRiIgpEQlVSTDogImpkYmM6dGhpbkAxOS41LjYuNzoxMzQ1L2RldmRiIg==" | base64 --de

268 clear

269 docker service create --name testsvc --replicas 2 -p 9080:80 --config src=BHOST: "19.5.6.7"

270 DBPORT: "1345"

271 DBNAME: "devdb"

272 DBURL: "jdbc:thin@19.5.6.7:1345/devdb"

273 docker service create --name testsvc --replicas 2 -p 9080:80 --config src=devdbconf,target=/opt/dbconf/db.props nginx:latest

274 docker service ls

275 docker service ps testsvc

276 docker ps | grep testsvc

277 docker exec -it 130e4d13d1ee /bin/bash

281 docker secret ls

282 docker secret inspect testsec

283 docker secret create testsec2 catalina.2023-03-19.log

284 docker secret ls

285 docker secret inspect testsec2

286 docker service create --name testsecsvc --replicas 2 --secret src=testsec2,target=/etc/secrets/dbsec.prop nginx:latest

287 docker service ps testsecsvc

288 docker ps | grep testsecsvc

289 docker exec -it f00a16433558 /bin/bash

160 docker node ls

161 docker swarm join-token

162 docker swarm join-token worker

163 docker swarm join-token manager

164 docker node rm sworker-node01

165 docker node rm sworker-node01 --force

166 docker node ls

167 docker node promote sworker-node02

168 docker node ls

169 docker node demote sworker-node02

170 docker node ls

171 docker node demote sworker-node01

172 docker node ls

173 docker node promote sworker-node02

174 docker node promote sworker-node01

175 docker node ls

176 ps -ef|grep dockerd

177 systemctl stop docker.socket

178 systemctl status docker.socket

179 systemctl status docker.service

180 ps -ef|grep dockerd

181 systemctl start docker.socket

182 docker node ls

183 exit

184 history

185 docker swarm leave --force