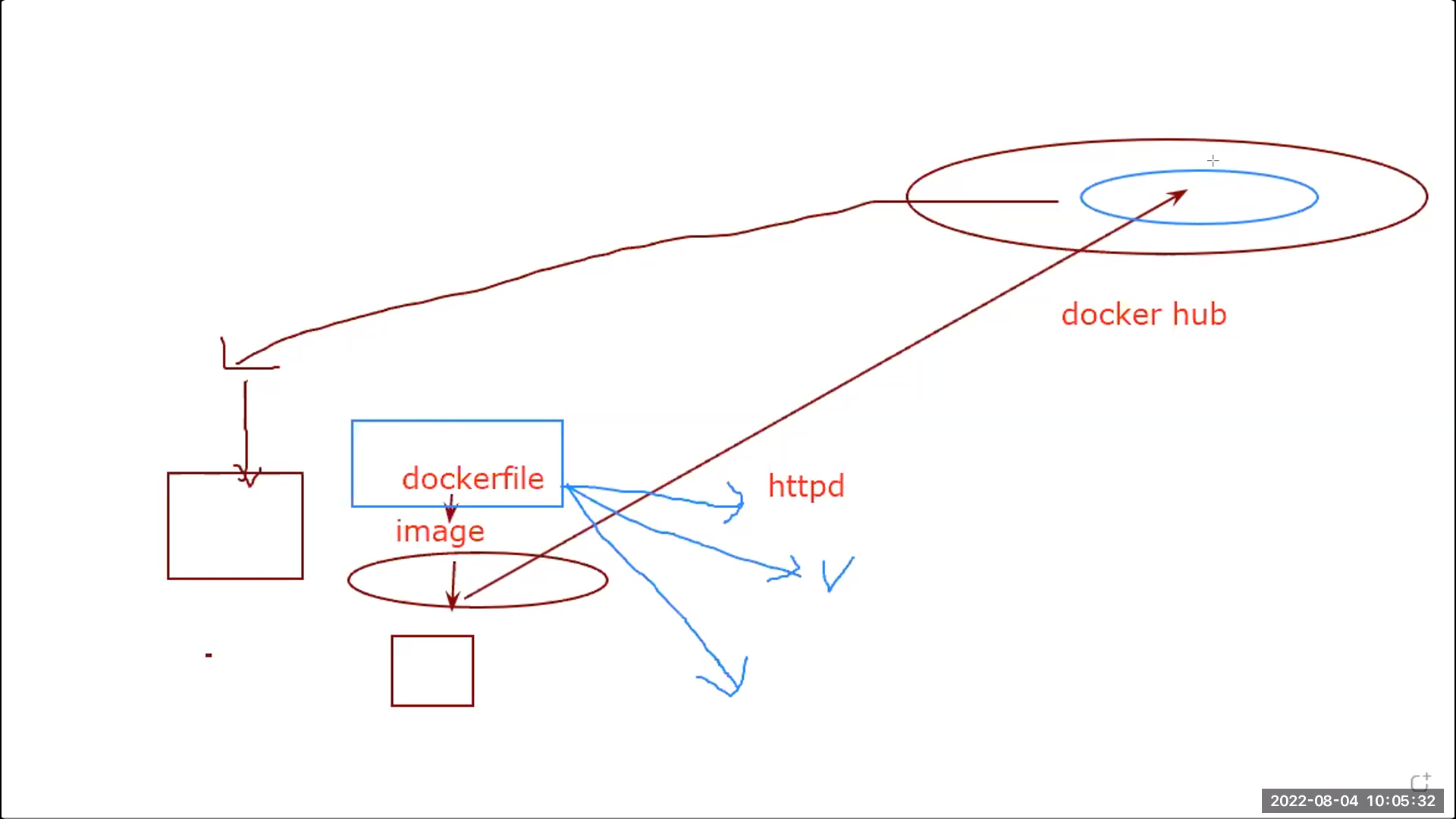
DockerDay-4

Mrng: docker files,creating own images



---------------------------------------------------------

Dockerfile[1.df] creating own httpd image

FROM centos:7

RUN yum install httpd -y

RUN systemctl enable httpd

EXPOSE 80

COPY index.html /var/www/html

CMD ["httpd","-D" ,"FOREGROUND"]

--------------------------------------------------------------------------------------

Explanation:

Preconditions: create a directory where make on index.html file and above 1.df

Using 1.df we created our image so

Firstline: to get the centos repo image where httpd s/w is located

Next: after getting centos repo we install httpd from that repo

Next: we enable the httpd

Next: we make httpd to run on port 80

Next: copy that html created in preconditions

Next: it is the command to behave the container as apache whenever we create the container from the image

Note: have a docker hub account with one created public repositery

Commands:

2 yum install docker

3 systemctl start docker

4 systemctl enable docker

6 mkdir dockerfiles

7 cd dockerfiles

8 vi 1.df

11 cd dockerfiles

12 vi 1.df

13 cat > index.html

14 docker build . -f 1.df -t santhosh127

19 docker run -d --name c1 santhosh127

20 docker ps

21 docker exec -it c1 bash

22 docker inspect c1 |grep -i ipaddr

23 curl 172.17.0.2

24 ls

25 cat 1.df

27 docker login

30 docker tag santhosh127 santhosh127:glogic

31 docker images

35 docker images

36 docker push santhosh127/latest:latest

37 ls

38 ls

Explanation:

14: to build the above docker file 1.df with tagname [-f : forcibly]

27: to login into the dockerhub account

30: to tag the create own image with name of dockeraccount/reponame: something

36: to push the above tagged image into docker hub

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2.df creating directories

FROM centos:7

WORKDIR /A

WORKDIR B

WORKDIR /C

WORKDIR /A/D

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Explanation:

First: it will be base os for containers so that is y we r taking centos you can also use differnet flavours like redhat

Next: creates the ‘A’ folder in container when we create

Next: creates the ‘B’ folder in ‘A’ folder

Next: creates the ‘C’ folder like A

Next: creates the ‘D’ Folder in ‘A’

-------------------------------------------------------------------------------------------------------------------------------

Commands:

44 docker build . -f 2.df -t dir:d

45 docker images

46 docker run -d -it --name c2 dir:d

47 docker ps

48 docker exec -it c2 bash

Explantion:

Task: create a container with above image and check the folder or directories

-------------------------------------------------------------------------------------------------------------------

3.df ADD vs COPY

Create 1.txt 2.txt files, sun directories and tar it into the 1.tar file and then remove 2.txt ,sun from main directory

FROM centos:7

WORKDIR /A

WORKDIR /B

WORKDIR /C

WORKDIR /D

COPY 1.txt /A

COPY 1.tar /B

ADD 1.txt /C

ADD 1.tar /D

------------------------------------------------------------------------------------------------------------------

Explanation:

6thline: it will copy the exact same file to A

Next: it will copy the exact same file to B

Next: it will copy the exact same file to C

Next: it will extarct the files in 1.tar

So COPY and ADD functions are same when come to txt files but for tar file ADD command extracts the files in it so only extracted files are even 1.tar file will get deleted

Commands:

53 dd if=/dev/zero of=/root/dockerfiles/1.txt bs=1M count=2

54 ls

55 cd

56 ls

57 cd dockerfiles

58 ls

59 dd if=/dev/zero of=/root/dockerfiles/2.txt bs=1M count=2

60 mkdir sun

61 tar -cvf 1.tar 1.txt 2.txt sun

62 ls

63 rm -rf sun

64 rm -r 1.txt

65 rm -r 2.txt

66 ls

67 ls -hltr

68 tar -tvf 1.tar

69 ls

70 ls -hltr

71 tar -xvf 1.tar

72 ls

73 rm -rf 2.txt sun

74 ld

75 ls

76 vi 3.df

77 docker build . -f 3.df -t copy:add

78 docker run -d --name c3 copy:add

79 docker exec -it c3 bash

80 docker ps

81 docker start c3

82 docker ps

83 docker run -d -it --name c4 copy:add

84 docker ps

85 docker exec -it c4 bash

Explanation:

53: it will create the text file containes zeros with 2MB

60: creates one directory

61: it will zip all file into 1.tar file

68: it will show the zip files of 1.tar

71: it will unzip the files in 1.tar

Note: use 83 instead of 78 because using 78 the conatier is created but not running

Tast: create a container with above image and see the results of 3.df files

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4.df creating volume

FROM centos:7

VOLUME vol1

Explanation:

Secondline : creates volume vol1

------------------------------------------------------------------------------------------------------------------

Commands:

86 history

87 vi 4.df

90 docker build . -f 4.df -t vol:vol1

93 docker run -d -it --name c7 vol:vol1

94 docker exec -it c7 bash

95 cd /var/lib/dockerfiles/vol1

96 cd /var/lib/dockerfiles

97 cd /var

98 ls

99 cd /lib

100 ls

101 cd /var

102 ls

103 cd lib

104 ls

105 cd docker/

106 ls

107 cd volumes

108 ls

109 cd bbd25cef9903b06f701ae4e0f1b9adef6eb182fb0c6228b9e1a4cc93463bf332/

110 ls

111 cd \_data

112 ls

113 touch b{1..2}

114 ls

115 cd /

116 cd

117 ls

118 cd dockerfiles/

119 ls

Explanation:

113: creates the text files of b1,b2

---------------------------------------------------------------------------------------------------------------------------------------

5.df volumes with cmd

FROM centos:7

VOLUME vol2

CMD sleep 3000

Same as 4.df

But it sleeps for 3secs

--------------------------------------------------------------------------------------------------------------------------------------

Commands:

120 vi 5.df

121 docker build . -f 5.df -t vol:vol2

122 docker run -d -it c8 vol:vol2

123 docker run -d -it --name c8 vol:vol2

124 docker run -d -it --name c9 -v volume:/apple vol:vol2

125 docker exec -it c9 bash

126 cd /var/lib/docker/

127 ls

128 cd voluimes

129 cd volumes

130 ls

131 cd volume

132 ls

133 cd \_data

Explaantion:

123: creates the container with volume becos in 5.df we created the image with volume

124: same as 123 but it will create separete foder volume in /var/lib/docker/volumes/ and also apple in the created container

Samba is protcol which is used to transfer the files from windows to linux vice versa

FileZilla is s/w which is user transfer the files from windows to linux and vice versa

Steps :

Open filezilla

Prequistes:

Click on edit-> setting-> sftp-> add ppk key file

1. Type your public ip at host
2. Type your user name [ec2-user for no passwod]
3. Tye passwor
4. Quick connect

**Evng:**

**------------------------------------------------------------------------------------------------------------**

**6.df (CMD)**

**FROM centos:7**

**CMD ["ping","google.com"]**

**CMD ["ping","facebook.com"]**

**Explanation:**

**Firstline : it exceute the command ping google.com**

**Next: it exceute the command ping facebook.com**

**Note: But in this case ping facebook.com only work because when we use CMD the last command overrides the all previous CMD commands**

**-------------------------------------------------------------------------------------------------------------------**

**Commands:**

**149 vi 6.df**

**150 docker build . -f 6.df -t cmd:cmd1**

**151 vi 6.df**

**152 docker build . -f 6.df -t cmd:cmd1**

**153 docker run --name c1 -dit cmd:cmd1**

**154 dockdr ps**

**156 docker logs c1**

**157 vi 6.df**

**158 docker build . -f 6.df -t cmd:cmd2**

**159 docker run --name c2 -dit cmd:cmd2**

**160 docker logs c1**

**161 nslookup google.com**

**163 docker run --name c2 -dit cmd:cmd2 hostname**

**164 docker run --name c3 -dit cmd:cmd2 hostname**

**Explanation:**

**149:to create or edit the file**

**156: it will show the logs that are in background but in this ping facbook.com is logfile according the created image in dockerfile**

**161: it will show the ipaddress and hostname of google**

**163: it will overrides the CMD coomand in above dockerfile and shows the hostname when execute**

**The 156 command**



**-------------------------------------------------------------------------------------------------------------------------------------------**

**7.df(EntryPoint)**

**FROM centos:7**

**RUN yum update -y**

**ENTRYPOINT ["echo", "HelloWorld"]**

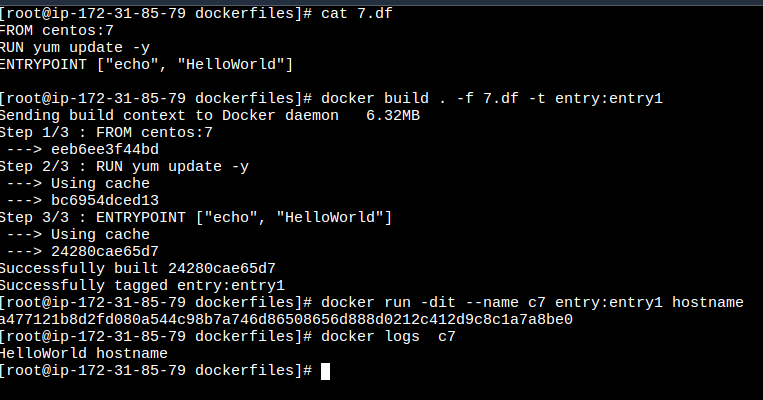
**Explanation:**

**Entry point works on string and appends the string it will run the command**

**Secondline: run the command yum update –y**

**Next: it will exceute the command helloworld**

**Even if have another entry point below hello world it will append to the helloworld it will not override the previous command CMD**



**--------------------------------------------------------------------------------------------------------------------------------------**

**Commands:**

**166 vi 7.df**

**167 docker build . -f 7.df -t entry:e1**

**168 docker run -dit c4 entry:e1 hostname**

**169 docker run -dit --name c4 entry:e1 hostname**

**170 docker logs c4**

**171 ls**

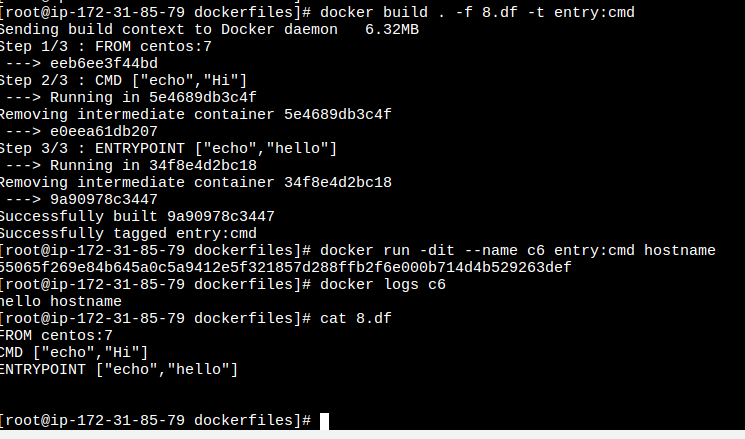
**172 cd**

**173 cd dockerfiles/**

**174 ls**

**Explanation:**

**168: it will append the hello world with hostname [it will not override like CMD ]**



**------------------------------------------------------------------------------------------------------------------------------------**

**8.df[cmd vs entrypoint]**

**FROM centos:7**

**CMD ["echo","Hi"]**

**ENTRYPOINT ["echo","hello"]**

**Explanation:**

**Entrypoint overrides the secondline**

**--------------------------------------------------------------------------------------------------------------------------------**

**Commands:**

**180 vi 8.df**

**181 docker build . -f 8.df -t entry:cmd**

**182 docker run -dit --name c6 entry:cmd hostname**

**183 docker logs c6**

**184 cat 8.df**

**Expalnation:**

**182: it will overrides CMD and prints the hello from entrypoint hostname**

**Result: hello hostname**

**184: It display the 8.df file**

**Assignment:**

**Prequisites:**

**Makde diretory data**

**Backend:**

**Backend.df ,createtable.sql, insertvalues.sql**

**-------------------------------------------------------------------------------------------------------------------**

**Backend.df**

**FROM mysql**

**ENV MYSQL\_DATABASE priar**

**COPY ./createtable.sql /docker-entrypoint-initdb.d/**

**COPY ./insertvalues.sql /docker-entrypoint-initdb.d/**

**------**

**createtable.sql**

**CREATE TABLE priarworld (firstname varchar(20),rollnumber int(5));**

**------------**

**insertvalues.sql**

**INSERT INTO priarworld (firstname,rollnumber) VALUES ('msd',1);**

**INSERT INTO priarworld (firstname,rollnumber) VALUES ('virat',2);**

**INSERT INTO priarworld (firstname,rollnumber) VALUES ('pant',3);**

**INSERT INTO priarworld (firstname,rollnumber) VALUES ('hacker',4);**

**INSERT INTO priarworld (firstname,rollnumber) VALUES ('paras',5);**

**INSERT INTO priarworld (firstname,rollnumber) VALUES ('pari',6);**

**Note:**

**Exceute the command after enter into the container using the backend.df image**

alter user 'root'@'localhost' identified with mysql\_native\_password by '123';

alter user 'root'@'%' identified with mysql\_native\_password by '123';

**------------------------------------------------------------------------------------------------------**

**Forntend.df**

**FROM centos:7**

**#MAINTAINER "priar world"**

**#LABEL "APP"="Devlopement"**

**RUN yum install httpd -y**

**RUN yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm -y**

**RUN yum install http://rpms.remirepo.net/enterprise/remi-release-7.rpm -y**

**RUN yum install yum-utils -y**

**RUN yum-config-manager --enable remi-php72 -y**

**RUN yum install php php-mcrypt php-cli php-gd php-curl php-mysql php-ldap php-zip php-fileinfo -y**

**EXPOSE 80**

**RUN systemctl enable httpd**

**CMD ["httpd", "-D", "FOREGROUND"]**

**---------**

**Index.php**

**<?php**

**$servername = "172.17.0.2";**

**$username = "root";**

**$password = "123";**

**$dbname = "priar";**

**// Create connection**

**$conn = new mysqli($servername, $username, $password, $dbname);**

**// Check connection**

**if ($conn->connect\_error) {**

**die("Connection failed: " . $conn->connect\_error);**

**}**

**$sql = "SELECT firstname,rollnumber FROM priarworld";**

**$result = $conn->query($sql);**

**if ($result->num\_rows > 0) {**

**// output data of each row**

**while($row = $result->fetch\_assoc()) {**

**echo "Name: " . $row["firstname"]. " " . $row["rollnumber"]. "<br>";**

**}**

**} else {**

**echo "0 results";**

**}**

**$conn->close();**

**?>**

**-------------------------------------------------------------------------------------**

**Commands:**

**209 mkdir data**

**210 ls**

**211 cd data**

**212 vi backend.df**

**213 vi createtable.sql**

**214 vi insertvalues.sql**

**215 vi frontend.df**

**216 vi index.php**

**217 ls**

**218 mv backend.df createtable.sql frontend.df index.php insertvalues.sql data**

**219 ls**

**220 cd data**

**221 ls**

**222 docker build . -f backend.df -t backend:b1**

**224 docker run -dit --name b1 -e MYSQL\_ROOT\_PASSWORD=123 -p 3306:3306 backend:b1**

**225 docker exec -it b1 bash**

**226 docker inspect b1 |grep -i ipaddr**

**227 docker build . -f frontend.df -t frontend:f1**

**228 docker run -dit --name f1 -p 81:80 -v /data:/var/www/html frontend:f1**

**Expalanation:**

**218: it will move files to data folder**

**224: it will create the database container with admin root password 123 [-e Environment variables]**