+++++++++++++++++++++++++++++++++++++++++++++++++++

Creating customized docker images

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

Whenever docker container is deleted,

all the softwares that we have installed within the container will also be deleted.

If we can save the container as an image, then we can preserve the softwares.

This creation of customized docker images can be done in two ways.

1) using docker commit command

2) using docker file

Using docker commit

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

# docker run --name c11 -it ubuntu

Update apt repository

# apt-get update

# apt-get install git

TO check the git

# git --version

# exit

TO save the container as image (snapshot )

# docker commit c11 myubuntu

To see the list of images

# docker images ( you can see the image which you have created )

++++++++++++++++++++++

Now lets run the image which we have created

# docker run --name c22 -it myubuntu

# git --version ( git is pre installed )

+++++++++++++++++++++++++++++++++++++++++++++++++

Using docker file

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

This is a simple text file, which uses predefinied keywords for creating customized docker images.

Key words used in docker file ( case sensitive )

1) FROM -- used to specify the base image from which the docker file has to be created.

2) MAINTAINER -- This represents name of the organization or the

author who created this docker file.

3) CMD -- This is used to specify the initial command that should be executed when the contain

starts.

4) ENTRYPOINT - used to specify the default process that should be executed when container starts.

It can also be used for accepting arguments from the CMD instruction.

5) RUN -- Used for running linux commands within the container. It is generally helpful for installing

the software in the container.

6) USER -- used to specify the default user who should login into the container.

7) WORKDIR -- Used to specify default working directory in the container

8) COPY -- Copying the files from the host machine to the container.

9) ADD -- Used for copying files from host to container, it can also be used for downloading files

from remote servers.

10) ENV -- used for specifying the environment variables that should be passed to the container.

EXPOSE -- Used to specify the internal port of the container

VOLUME -- used to specify the default volume that should be attached to the container.

LABEL -- used for giving label to the container

STOPSIGNAL -- Used to specify the key sequences that have to be passed in order to stop the container.

+++++++++++++++++++++

+++++++++++++++++++++++++++++++++

Create a dockerfile by taking nginx as the base image

and specify the maintainer as logiclabs. Construct an image from the dockerfile.

Creating customized docker images by using docker file.

$ sudo su -

# vim dockerfile

FROM nginx

MAINTAINER logiclabs

:wq

TO build an image from the dockerfile

# docker build -t mynginx .

( t stands for tag,

. stands for current working dir

mynginx is the new image name )

TO see the image

# docker images

++++++++++++++++++++++++++++++++++++++++++++++

When ever i start my container, i want a program to get executed.

# vim dockerfile

FROM centos

MAINTAINER logiclabs

CMD ["date"]

:wq

TO build an image from the dockerfile

# docker build -t mycentos .

TO see the image

# docker images

Running conainer from the image

# docker run -it mycentos

++++++++++++++++++++++++++

In one docker file, we can have one CMD instruction.

If we give two CMD instruction, it executes the latest one

Lets try

# vim dockerfile

FROM centos

MAINTAINER logiclabs

CMD ["date"]

CMD ["ls", "-la"]

:wq

# docker build -t mycentos .

# docker run -it mycentos

( Observation, we get ls -la output )

++++++++++++++++++++++++++++++++++++++++++++++

In ubuntu container, I want to install git in it.

Lets remove the docker file

# rm dockerfile

# vim dockerfile

FROM ubuntu

MAINTAINER logiclabs

RUN apt-get update

RUN apt-get install -y git

:wq

Note: CMD -- will run when container starts.

RUN -- will executed when image is created.

# docker build -t myubuntu .

Lets see the images list and space consumed by our image

# docker images

# docker run -it myubuntu

# git --version

# exit

++++++++++++++++++++++++++++++++++

Cache busting

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

Whenever an image is build from a dockerfile, docker reads its memory and checks which instructions were

already executed. These steps will not be reexecuted.

It will execute only the latest instructions. This is a time saving mechanism provided by docker.

But, the disadvantage is, we can end up installing software packages from a repository which is updated

long time back.

Ex:

# cd docker

# vim dockerfile

Lets just add one more instruction

FROM ubuntu

MAINTAINER logiclabs

RUN apt-get update

RUN apt-get install -y git

RUN apt-get install -y tree

:wq

Lets build an image

# docker build -t myubuntu .

( Observe the output, Step 2, 3, 4 is using cache. Only step 5 is executed freshly )

Advantage: time saving mechanism

Disadvantage : Lets say, you are running after 4 months, We are installing tree from apt which is

updated long time back. )

TO avoid this disadvanatge we use cache busting

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

Note: cache busting is implemented using && symbol.

Which ever statement in the docker file has && will be re-executed.

# vim dockerfile

FROM ubuntu

MAINTAINER logiclabs

RUN apt-get update && apt-get install -y git tree

:wq

Lets build an image

# docker build -t myubuntu .

( Observe the output, step 3 - It is not using cache )

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Working on docker registry

Registry is a location where docker images are saved.

Types of registry

1) public registry

2) private registry

public registry is hub.docker.com

Images uploaded here are available for everyone.

Usecase: Create a customized ubuntu image, by installing tree in it.

Save this container as an image, and upload this image in docker hub.

Step 1: Create a new account in hub.docker.com

Step 2: Creating our own container

# docker run --name c5 -it ubuntu

Lets install tree package in this container

/# apt-get update

/# apt-get install tree

/# exit

Step 3: Save the above container as an image

# docker commit c5 sunildevops77/ubuntu\_img26

( sunildevops77/ubuntu\_img15 -- is the image name )

Note: Image name should start with docker\_id/

To see the list of images

# docker image ls ( we can see the new image )

TO upload the image to hub.docker.com ( docker login command is used )

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

# docker login ( provide docker\_id and password )

To upload the image

# docker push <image\_name>

# docker push sunildevops77/ubuntu\_img26

login to docker hub to see your image

+++++++++++++++++++++++++++++++++++++++++++++++++++

+++++++++++++++++++++++++++++++++++++++++++++++++++