DOCKER

* Dockerisanopensourcecentralizedplatformdesignedtocreate,deployandrun applications.
* DockerusescontaineronthehostOStorunapplications.Itallowsapplicationstousesame Linux kernel as a system on the host computer rather than creating a whole virtual OS.
* Wecaninstalldockeron anyOS butdocker enginerunsnativelyonLinuxdistributions.
* Dockerwrittenin“GO”programminglanguage.
* Dockerisatoolthatperforms OSlevelvirtualization alsoknownas Containerization.
* Beforedockermanyusersfacetheproblemthataparticularcodeisrunninginthe developer’s system but not in the user’s system.
* Docker was first release in march 2013. It is developed by Solomon Hykesand Sebastian Pahl.
* DockerisasetofPlatform-as-a-ServicethatusesOSlevelvirtualizationwhereasVMWare uses hardware level of virtualization.

AdvantagesofDocker:

* NopreallocationofRAM.
* CI efficiency: - docker enables you to build a container image and use that same imageacross every step of the deployment process.
* Less cost.
* Itislightinweight.
* Itcan runon physicalh/w /virtual h/woron cloud.
* Youcanreusethisimage.
* Ittookverylesstimetocreatecontainer. Disadvantages of Docker:
* Dockerisnotagoodsolution forapplicationthatrequiresrich GUI.
* Difficultto managelarge numberof containers.
* Dockerdoesn’tprovidecrossplatformcompatibilitymeansifanapplicationisdesignedto run in a docker container in windows than it can’t run in Linux or vice-versa.
* DockerissuitablewhenthedevelopmentO.SandtestingO.Saresame.IftheO.Sare different then we should use VM.
* Nosolutionfordatarecoveryand backup.

# Componentsof Docker:

1. DockerDaemon:
   * Dockerdaemonrunsonhost O.S.
   * Itisresponsible forrunningcontainerstomanagesdockerservices.
   * Dockerdaemoncancommunicatewithother daemons.
2. DockerClient:
   * Dockeruserscan interactwithdockerthrougha client.
   * DockerclientusescommandsandRESTAPItocommunicatewiththedocker daemon.
   * When a client runs any server command on the docker client terminal, the client terminal sends these docker commands to the docker daemon.
   * Itispossiblefordocker clientto communicatewithmorethanone daemon.
3. Docker Host:
   * Dockerhostisusedtoprovideanenvironmenttoexecute andrunapplications.
   * Itcontainsthedockerdaemon,images,containers,networksandstorages.
4. DockerHub/Registry:
   * Dockerregistrymanagesandstoresthedockerimage.
   * Therearetwo typesofregistries inthedocker:
     1. PublicRegistry:it isalsocalled asdocker hub.
     2. PrivateRegistry: itis used toshareimagewithin theenterprise.
5. Docker Image:
   * Dockerimagesarethereadonlybinarytemplatesusedtocreatedockercontainers.

or

* + Singlefilewith allthedependenciesandconfigurationrequiredto runa program.

WaystoCreateanImage:

* + 1. Takeimage fromthedockerhub.
    2. Createimagefromdocker file.
    3. Createimagefromexistingdockercontainers.

1. DockerContainers:
   * Containersholdtheentirepackagesthatisneededtoruntheapplication. Or
   * Inotherwords,wecansaythattheimageisatemplateandthecontainer isacopy if that template.
   * Containeris likeavirtual machine.
   * Imagesbecomescontainerwhenthey runondockerengine.

# BasicDocker Commands:

Toseeallimagespresentinyourlocalrepo: # docker images

Tofindoutimagesindockerhub # docker search image\_name

Todownloadimagefromdockerhubtolocalmachine # docker pull image\_name

Togive aname to container

#dockerrun-it--namenew\_nameimage\_name/bin/bash i - (interactive mode) t – ( terminal )

Tocheckservicestartornot(status) # docker service status

Tostart:#dockerservicestart Tostop:#dockerservicestop

Tostartcontainer

#dockerstart container\_name

To go insidecontainer

#dockerattachcontainer\_name

Toseeallcontainers # docker ps –a ps (process status)

Toseerunningcontainers # docker ps

To stop container

#docker stop container\_name

Todeleteacontainer

#docker rm container\_name

Createcontainerfrom ourownImage:

LoginintoAWSaccountandstartyourEC2instance,accessitfrom putty.

Nowwehavetocreatecontainerfromourownimage.Therefore,createonecontainerfirst: #docker run -it –name container\_name image\_name /bin/bash

#cdtmp/

Nowcreateonefileinsidethistmpdirectory

# touch myfile

Nowifyouwanttoseethedifferencebetweenthebasicimageandthechangesonit

# docker diff container\_name

Nowcreate imageof this container

# docker commit container- name new-imagename (for new image which we need to make )

#docker images

Nowcreate containerfrom this image

#docker run -it --name contanier-name image name ( which we had made ) /bin/bash

# ls

#cd tmp

#ls (you will get all of yourfiles)

# Dockerfile:

Dockerfileisbasicallyatextfile.Itcontainssomesetofinstructions.Automationofdockerimage creation.

Dockerfilecomponents:

FROM:forbaseimage,thiscommandmustbeonthetopofthedockerfile. RUN: to execute commands, it will create a layer in image MAINTAINER: author/ owner/ description

COPY:copyfilesfromlocalsystem(dockervm)weneedtoprovidesource,destination(wecan’t download file from internet and any remote repo.)

ADD:similarto copybutitprovidesa feature to downloadfiles frominternet,alsoextract fileat docker image side.

EXPOSE:toexposeportssuchasport8080fortomcat,port80fornginxetc. CMD: execute commands but during container creation.

ENTRYPOINT:similartoCMDbuthashigherpriorityoverCMD,firstcommandswillbe executed by ENTRYPOIN only.

ENV:environmentvariables

Dockerfile

* CreateafilenamedDockerfile
* Addinstructionsin Dockerfile
* Builddockerfiletocreate image
* Runimagetocreatecontainer # vi Dockerfile

FROM ubuntu

RUN echo “Subscribe” > /tmp/testfile

Esc -> :wq!

To create image come out of Dockerfile

#docker build -t new-image-name .

#docker ps -a

#dockerimage

Now create container from the above image #docker run -it --name container-name new-image-name /bin/bash

#cat /tmp/testfile

Method -2

#vi Dockerfile

FROM ubuntu WORKDIR /tmp

RUN echo “thankyou” > /tmp/testfile ENV myname Sandeep

COPY testfile1 /tmp ADD test.tar.gz /tmp

# DockerVolume:

* Volumeissimply adirectory insideourcontainer.
* Finally,wehavetodeclarethis directory as avolumeandthensharevolume.
* Evenifwestop thecontainer still,wecanaccessvolume.
* Volumewill becreated inonecontainer.
* Youcan declareadirectory as avolumeonly whilecreating container.
* Youcan’tcreatevolume fromexistingcontainer.
* You can shareonevolumeacrossany number of containers.
* Volumewill notbe includedwhen youupdateanimage.
* You can mapvolume in twoways:
  1. Containerto container
  2. Hosttocontainer Benefits of Volume:
* Decouplingcontainerfromstorage.
* Sharevolumeamongdifferentcontainers.
* Attachvolumetocontainers.
* Ondeletingcontainervolumedoesn’t delete.

# CreatingVolumefrom Dockerfile:

CreateaDockerfileandwrite FROM ubuntu

VOLUME“myvolume”

ThencreateimagefromthisDockerfile #docker build -t myimage

Nowcreatea containerfromthis imageand run

#dockerrun-it--namecontainer1myimage/bin/bash Now do ls, you can see myvolume.

# Nowsharevolumewith another container

Containerto container

#dockerrun-it--namecontainer2(new)--privileged=true–volumesfromcontainer1ubuntu

/bin/bash

Nowaftercreatingcontainer2,myvolumeisvisible.Whateveryoudoinonevolume,canseefrom other volume.

#touch/myvolume/samplefile #docker start container1

#dockerattachcontainer1 #ls/myvolume

Youcanseesamplefileherethenexit.

# Nowcreatevolumebyusingcommand:

#dockerrun-it--namecontainer3-v/volume2ubuntu/bin/bash # ls

#cd/volume2

Nowcreateone filecont3fileand exit

Nowcreate onemorecontainerand sharevolume2

#dockerrun -it--name container4--privileged=true--volumefromcontainer3ubuntu/bin/bash Now you re inside container do ls you can see volume2

Nowcreateone fileinsidethisvolume andthencheck incontainer3 youcan seethat file.

Volumes(Hostto Container)

Verifyfilesin /home/ec2-user

#dockerrun-it--namehostcontainer-v/home/ec2-user:/container--privileged=trueubuntu

/bin/bash

#cd/container

Dols,nowyoucanseeallfilesofhostmachine. #touch contanerfile (in container) and exit

Nowcheck in EC2 machineyoucan seethis abovefile.

# Someothercommands:

#dockervolumels

#dockervolumecreate<volumename> #docker volume rm <volumename>

#dockervolumeprune(itremovesallunuseddockervolume) #docker volume inspect <volumename>

#dockercontainerinspect<containername>

# DockerPortExpose:

LoginintoAWS account,createonelinux instance.

Nowgotoputty->loginas->ec2-user #sudo su

#yum update-y

#yuminstalldocker-y

#servicedockerstart

#dockerrun-td--nametechserver-p80:80ubuntu # docker ps

#dockerporttechservero/p-80/tcp–0.0.0.0/80 # docker exec -it techserver /bin/bash

#apt-getupdate

#apt-getinstallapache2-y # cd /var/www/html

#echo“writesomemsg”>index.html #service apache2 start

#dockerrun-td--name myjenkins -p8080:8080 jenkins

Differencebetweendockerattach anddockerexec:

* Docker ‘exec’ creates anew process in the container’s environment while docker ‘attach’ just connect the standard input/output of the main process inside the container to corresponding standard input/output error of current terminal.
* Docker ‘exec’is specifically forrunningnewthingsinanalreadystartedcontainerbeita shell or some other process.

Whatis thedifferencebetweendockerexposeandpublish:

Basicallyyouhavethree options:

1. Neitherspecifyexposenor-p
2. Onlyspecifyexpose
3. Specifyexposeand-p
4. If you specify neither expose nor -p, the service in the container will only be accessiblefrom inside the container itself.
5. Ifyou expose aport,the servicein the containeris not accessiblefrom outsidedockerbut from inside other docker containers so this is good for inter-container communication.
6. If you expose and -p a port, the service in the container is accessible from anywhere even outside docker.

Ifyoudo–pbutdonotexposedockerdoesanimplicitexpose.Thisisbecauseifaportisopento the public, it is automatically also open to theother docker containers. Hence-p includes expose.

# Howto pushdockerimageindockerhub:

GotoAWSaccount–selectAmazonlinux Now go to putty – login as – ec2-user #sudo su

#yumupdate-y

#yuminstalldocker-y #service docker start

#dockerrun-itubuntu/bin/bash

Nowcreatesomefilesinsidecontainer,nowcreateimageofthiscontainer #docker commit container1 image1

Nowcreateaccountinhub.docker.com Now go to EC2 instance

#docker login

Enteryourusernameandpassword Now give tag to your image

#dockertagimage1dockerid/newimage #docker push dockerid/newimage

Nowyou can seethis imagein docker hubaccount

Nowcreateoneinstanceinanotherregionandpillimagefromhub #docker pull dockerid/newimage

#dockerrun-it--namemycondockerid/newimage/bin/bash

Someimportantcommands:

Stop all running containers: # docker stop $(docker ps -a -q) Deleteallstoppedcontainers:#dockerrm$(dockerps-a-q) Delete all images: docker rmi -f $(docker images -q)