# Docker

**What is docker** :

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package.

Dockers Provide a standard runtime , meaning developers can code apps in the docker containers on their docker host.

Before Containers there was virtualization concept . Where virtiual servers were hosted on a physical server .But each of the virtual server has a different OS and this used to occupy a lot of space and used to have a high impact on the cpu and memory unit of the Physical server.

To over come this disadvantage Containers concept has being introduced.

App 
os 
App 
os 
App 
os 
App 
os 
Hypervisor 
0000 
Server 
Hypervisor 
Architecture 
Operating System 
0000 
Server 
container 
Architecture 

# Containers

Containers are like virtual machines and they are the application runtime environment we run run applications inside it . These are light weighted , they occupy less cpu space .

How do they work ?

Containers are runned on a physical server . Whre the operating system manages userspace to each container. Container is an isolated instance of user space

App 
App 
App 
App 
Operating System 
Physical Machine 

Each Container has its own Root file system.and has its own view of its root file system.

( L meds«ueu Juuı) 
1 ıauıeıuo) 
mapa ptuaısÂs eps 
9 

Each Container has its own PID1 and its own indepented process hirachy inside it .

5.0 
Container 1 
(pki nanpspace I ) 
3.1 
4.0 
Container 3 
(pid nanpspace 3) 
Container 2 
(pid nanpsJ»ce 2) 

5.0 
Container 1 
(pki nanpspace I ) 
3.1 
4.0 
Container 3 
(pid nanpspace 3) 
Container 2 
(pid nanpsJ»ce 2) 

Each Container has its network configuration settings like its own ip address and port number to reach it

Namespaces : The Kernel namespaces allow us to Isolated partition the various aspects of the system . Where it maintians the every process of one conatiner and stops it from getting interacted with another container.

1)The pid Namespace.

1. The Net Namespace.
2. The MNT Namespace.
3. The User Namespace.

Another kernel feature container uses is

Control Groups (Cgroups): Let us to group together the resources and apply limits.

We set limits on container to how much cpu , memory that the container has the acceess to.

we can adjust the c group limits to a container.

Capabilities: The Capabilities Gives us the fine grain control over what privilages a user or a process gets.

Capabilities takes the root privilages and break them into smaller privilages and we can assign them when ever we needed. These are important in the security point of view.

|  |  |  |
| --- | --- | --- |
| **Installing docker on** | **EC2 instance** |  |

Type :1) sudo yum update -y

2)sudo yum install -y docker

3) sudo service docker start

4)sudo usermod -a -G docker ec2-user

1. docker info

**Major docker components**:

Docker engine is the shiiping yard .

Docker images are the manifests contains the details of the containers (what it contains and where to deliver)

Docker containers are the shipping containers.

Docker Engine 
(Shipping Yard) 
Docker Images 
(Manifests) 
Docker Containers 
(Shipping Containers) 

The same goes for the application you deploy in a server.You code an app in a docker container on your workstation and you run it in a docker container

Then it is deployed very faseter and easier way.

**Docker Engine**:

It’s a docker program we install on each docker host to provide us the with a docker environment and access to all docker services.

It contains the application infrastructe and runtime dependencies ,like process hirachies, variables ,acess to kenrel and its features , resource allocation.

And its all standardized .it all looks same frim docker engine to another. So we can install docker engine on any laptop or any host and run our containers on it.

[ec2-user@ip-172-31-29-252 etc]$ docker run -it ubuntu /bin/bash

Unable to find image 'ubuntu:latest' locally

latest: Pulling from library/ubuntu

d5c6f90da05d: Pull complete

1300883d87d5: Pull complete

c220aa3cfc1b: Pull complete

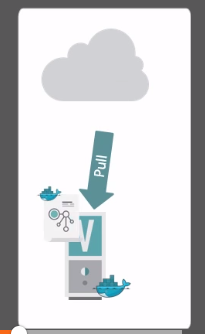
2e9398f099dc: Pull complete

dc27a084064f: Pull complete

Digest: sha256:34471448724419596ca4e890496d375801de21b0e67b81a77fd6155ce001edad

Status: Downloaded newer image for ubuntu:latest

root@dcb261a8884e:/#



That docker engine -> consists of docker Daemon and Docker client

When ever an Image is run.

Docker Client makes an API calls to Docker Daemon . And Docker daemon searches for that image if that image is available locally it renders it to docker client. Or else it pulls that image from the Docket hub Registry .

Once the Images are Pulled they are stored under : /Var/Lib/docker/aufs

Aufs is the standard storage docker driver.

DOCKER CONTAINERS:

A container is a running Linux instance. These are the run time constructs. It’s a bare bone linux machine.

We use docker run command to launch a container.

We use the command :

Docker run -it ubuntu /bin/bash to launch a ubuntu container with a shell terminal.

Here we are specifiying the process to pid1 . It may not be /bin/bash , we can specify Apache, Redis ,whatever as long as they are installed on our containers.

Containers exits untill the process inside them exists.

Docker containers terminate when the process pid1 inside them is terminated.

**Images**

Images are Instances of the containers . Where we can launch the containers

We generally see when a container is launched from a image , we find different image layers

These are stacked on one on each layer . Each layers are called images , together they form a single image

Image 
Layer 2 (Image 2) 
Layer 1 (Image 1) 
Layer O (Image O) 

-> single image is shared between multiple containers.

-> the bottom level image is called the root image or the base image .

-> Each image has a id.

Uninon Mounts :

The abitlity to mount the multiple files systems on the top of each other.

->image layering is accomplished through union mounts.

->They combine all of the each layer into a single view giving us or the application or the operating system.

->The only writable layer is added to the layer above all the layers.

->Ech and every container gets thin writable layerthat slapped right on the top.

->where we make changes where all container state is stored. And sapce is consumed on this layer.

-> AUFS Is the default union mount implementation used on the ubuntu hosts running docker

-> Images are stored in the AUFS locally .

Registries

Tuesday, October 10, 2017

8:32 PM

Public and private Registries.

Registries are like services and with in them,they contain repositories .

Pushing images to the public docker hub repository>

-> we need to tag our image before we push into the repository.

->docker tag (image id ) ( repository name)

-> docker push (repository name ).

-> docker pull (repository name)

Docker Hub Enterprise;

is the another form of the docker hub .

**Docker Commands**

#####################################################################3

Docker Container Commands and Run commands

##############################################################

Running a Docker image fedora to launch a container containning a fedora OS

->Docker Run -it fedora /bin/bash .

Here -It = interactive mode which opens up the terminal and user can type the input .

Fedora = it is the image ,to launch a fedora container.

/bin/bash = to run shell inside it .

->Docker Run -it fedora :latest /bin/bash : gets fedora latest image

->Docker Run -it fedora :Vfhjdfsjk /bin/bash : gets the specified fedora vesion.

->Docker version : displays the version of the client and the server

**Docker Attach and detach commands** :

->Docker Attach (container name)

-> Docker run -d Image name (To run the container in detached mode in background)

->docker run --cpu -shares =256 (to control how many cpus shares a container gets)

-> docker run memory =1g (docker run givess 1gig to container)

->docker top

->docker inspect

->docker attach

**Commands to stop and start the container:**

->Docker stat <container>

->Docker stop<container>

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

->Getting out of a container : Cntrl P +Q

-----------

To stop all the containers and return the containers ids which are stopped.

->Docker stop $(docker ps -aq ) /

To start :

->Docker start $(docker ps -aq

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

-> docker ps -a (to know the id of the process )

############################################################################----

Docker Images Commands

####################################################-------------------

Docker images :

Displays the images that are installed .

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

Docker pull ubuntu : 14.0.4 { (version) : which verion need to be pulled . If you need thelatest one just enter the ubuntu with no version.

This will pull the images from the hub.

->Docker pull -a ubuntu ( pulls all versions of ubuntu )

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

->Docker run images :

Docker run is the docker client command which makes api calls to daemon.If daemon doesn't have the image it will pull from the docker hub.

Running the image.. This will search for the hello-world and the the daemon will pull this from the docker hub if it is not available .

-> Docker commit : command

Docker commit (image id ) message

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

->Docker rmi images :

Removes the pulled images .

-> Docker commands to copying a image from one host to another.

On docker host run

Image of host 1:

Docker save -o /tmp/files containername(image)

-> on host 2

Tar -tf /tmp/files

->then run

Docker load -I (location of the file store)

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

Dokcer file is runned and buid to form a docker Image -> and the docker image is run to build a docker container.

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

**DOCKER Networking commands**:

Docker run –d –p 5001:80 --name =web1 apache image ( example command for running the web1 container on apache image , exposing ports ie indication to container that 5001 on docker host will be connected to 80 on the container.)

Docker run –d –p 5001:80/udp --name =web1 apache image (command to have udp connection)

Docker port (container name ) (to know the port type connection)

Use Capital P for exposing all the ports in the contaniner.

Docker run –d –P --name =web1(container namer) apache image (name of the image )…(this will show the ports exposed on the docker file for a container).

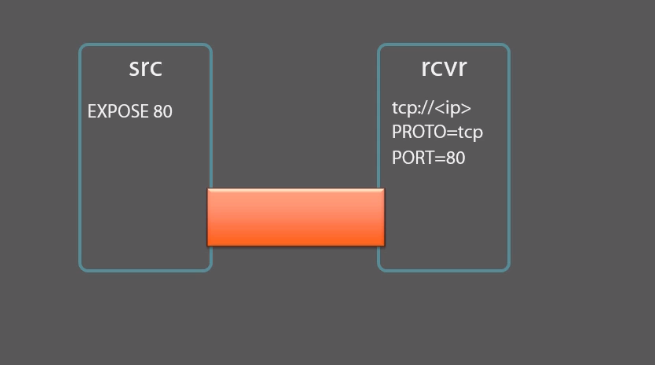
**Docker Expose** :

Expose 80 on docker file means exposing the container on port 80 .

**Linking Containers :**

**Linking containers** is more secure than exposing the ports.

When speaking with the linking the containers we have a source , and a recipient or a receiver.



Source container has the exposed ports listed in the docker file . and it is not explicity exposed to outside world at run time it is only linked with onter container (receiver) at runtime.

To linka src and receiver use

* build docker file with port 80 on dock file , build it and run
* Docker run --name=src –d (image name )
* Then run another container receiver
* Docker run –name=rcvr --link =src:alias-src -it Ubuntu /bin/bash (running the receiver container and linking with the source container with alias name also )

Use the docker inspect command to see in the recierver container.

**Getting a shell in a container**

Docker attach is not the best way to get the shell in a container .

Running ssh inside the containers:

Most of the containers dosen't support the ssh inside the containers.

So there is an option to get the shell access inside the container its by using NSENTER

nsenter:

nsenter -> allow us to enter the namespaces

Requires the containers pid.

nsenter command:

Nsener -m -u -n -p -I -t (pid of container) /bin/bash

-m = mount namespace -

-u = unioin mount namespace

-n=network namespace

-p = process namespace

-I = ipc namespace

-t = tty

Docker enter -> to enter into the container

Docker exec -It (container name ) /bin/bash : to enter into a container and run executing the commands inside the container (another way )

**Docker File:**

Docker file is used to build the docker images .

It consits of instructions to how to build an image. These instructions are run from top to bottom. Left to right.

Run-> used to run the commands against our images that we are building.

Every run isturction adds a layer to our image

CMD -> this short for command

This will run our command described in the docker file

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

Building an Image from adocker file

docker build -t (imagename)

Running a docker Images to create containers:

[ec2-user@ip-172-31-29-252 etc]$ docker run -it ubuntu /bin/bash

Unable to find image 'ubuntu:latest' locally

latest: Pulling from library/ubuntu

d5c6f90da05d: Pull complete

1300883d87d5: Pull complete

c220aa3cfc1b: Pull complete

2e9398f099dc: Pull complete

dc27a084064f: Pull complete

Digest: sha256:34471448724419596ca4e890496d375801de21b0e67b81a77fd6155ce001edad

Status: Downloaded newer image for ubuntu:latest

root@dcb261a8884e:/#

After running the instace and the doker image on our amazon ec2 instance , we are actually inside the command line of the container . And It is denoted by root@dcb261a8884e:/# . The number beside the root is the short form of containers unique ID.

Volumes

Sunday, September 24, 2017

Volumes in containers are decoupling the data in volumes from containers and sharing the data within the containers .

Specifing the volumes command

Docker run -it -v (/mount point location) --name = voltaniner (name of the container .) ubuntu:15.04 /bin/bash

\*To mount the data directory on the docker host to inside the container then the command is

->Docker run -v /data:/data

\*to remove the volume of the container :

Then Docker rm -v <container>

Mentioning the volumes In the docker file is by writing

VOLUME /data

* we can’t mount the host file directory into the container by defining the mount location in the docker file using Volumes , it can be done by using the docker run command.

**Trouble Shooting**

**Docker Daemon logging :**

We can start daemon in 4 differnet ways :

1. Debug : debug logs ifno +error +fatal
2. Info: info logs error+fatal
3. Error : error logs error +fatal
4. Fatal : fatal logs only fatal

To debug a container

1. Stop the service : service docker stop
2. Run : docker –d -l debug &

We can also edit the docker config file to see the logs in/etc/default/docker:

1. DOCKER\_OPTS=”debug”

**Container Logging** :

Docker logs container\_name

**Intermediate images** :

When the docker image created with no name or no tag , go to the build instructions and go through the steps and get the image id where it got failed and run it in interactive mode to trouble shoot it.

**Docker0 Bridge :**

To check the bridgeipaddress is already using by any other container

1)Stop docker service.

1. Get the docker 0 brige ip address
2. Now brig the link down
3. Ip link del docker0
4. We edit the docker config file (add ip value to docker \_opts = --bip=ipaddress

**IpTables:**

Command to check the iptables : iptables -L –V

--icc = inter container communication (default =true)

--iptables = will determine any modifications required for the ip tables rules default (ture)

If need to change the configurations chage in the config file

DOCKER\_OPTS= --icc=true //// or (-- iptables=false) ///