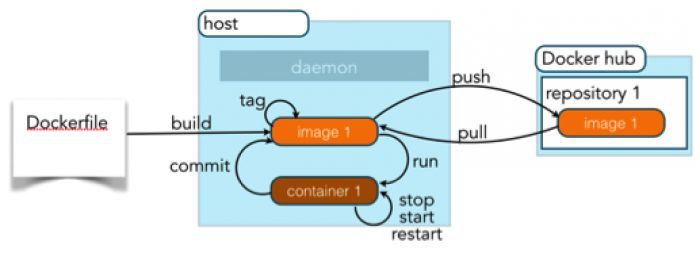
[https://docs.docker.com/network/network-tutorial-standalone/#use-the-default-bridge-network](https://docs.docker.com/network/network-tutorial-standalone/" \l "use-the-default-bridge-network) //Reffer good information about networking concepts



**DOCKER on UBUNTU**

require only X86\_64bit and kenel-.3.10 higher

wget -q0- https://docker.com/ | ssh

docker run hello world

**DOCKER on CENTOS**

require only X86\_64bit and kenel-2.5 higher

curl -0 -sSL https://get.docker.com/rpm/1.7.0/centos-6/RPMS/X86\_64/docker-engine-1.7.0.X86\_64.rpm

yum localinstall --nogpgcheck docker-engine-1.7.0.X86\_64.rpm

**USER**

usermod -aG docker user1

**MEMORY**

adjust new memory

nano /etc/default/grub

grub\_cmdline\_linux="cgroup\_enable=memory swapaacont=1"

exit and login

update-grub

**FIREWALL**

UFW forwarding (default bridge to inter containers traffic)

ufw status

nano /etc/default/ufw

default\_forwarding\_policy="ACCEPT"

ufw reload

ufw allow 2375/tcp --> forwarding port

**DNS**

nano /etc/default/docker --> preffered

dns dnsip

restrat docker

nano /etc/NetworkManager/NetworkManager.conf

comment dns entry

**DOCKER DAEMON**

docker pull centos (image)

docker run -t -i centos /bin/bash

will see container id like [root@09600]

**DOCKER TEMPLATE IMAGES**

docker images

list all images

docker run -t -i centos:10.5 /bin/bash

will see container id like [root@09600]

**RUN CONTAINER**

**[Interactive -i option]**

docker run -t -i centos:10.5 /bin/bash

**[Non-Interactive]**

docker run ubuntu:14.04 /bin/echo "Hello Docker"

kill container once echo done.

**ACCESS IMAGES from REGISTRY Interactive MODE**

docker images

docker pull ubuntu:14.04

docker run -t -i ubuntu:14.04 /bin/bash

run app and commands

List Containers

Docker ps 🡪 Docker host

**DOCKERIZATION APPLICATIONS**

Package and run applications within Containers………

Pull 🡪 Build 🡪 Run

RUN Containers

Docker –it centos:latest /bin/bash

Echo “hello world”

**Daemonizing a Container**

JOB=$(docker run –d ubuntu /bin/sh –c “while true; do echo hello world; sleep 1; done”)

-d means DEAMON

Docker ps 🡪 to list all

Docker logs $JOB 🡪 see the logs

**Access Daemon**

JOB=$(docker run –d Ubuntu /bin/sh –c “while true; do echo hello world; sleep 1; done”)

Docker stop $JOB 🡪 stop the container job

Docker start $JOB 🡪 Strat the container job

Docker restart $JOB 🡪 Restart the container job

Docker kill $JOB 🡪 kill the container job

Docker rm $JOB 🡪 Remove the container job

Docker system prune 🡪 remove All stopped containers

* All stopped containers
* All networks
* All images
* Build chache

**Working with Containers**

**Commands**

Docker 🡪 help

Docker help

Docker run –help

Docker version

Docker info 🡪 list all

**Client Commands**

docker run –d ubuntu /bin/sh –c “while true; do echo hello world; sleep 1; done”

docker ps

docker images 🡪 local images from registry

**Running APPS on Containers**

**Single thread (Container Die)**

Docker run Ubuntu /bin/echo “hello world”

**Interactive (Container available until EXIT)**

Docker run –it Ubuntu /bin/bash

**As Deamon (Container available until stop/kill/rm)**

docker run –d ubuntu /bin/sh –c “while true; do echo hello world; sleep 1; done”

**Review the Application Container**

Its like Multiple VM’s on Host

Multiple Containers with Apps

**Ports**

Docker run –d p 80:5000 traning/webapp python script.py

Docker port <Container ID/Name> 5000

**Name/Inspecting Container**

Docker run –name CName

Docker inspect CName 🡪 info of Container with facts

**Docker Images**

List Images:

Docker images

🡪 list images in local registry

Pull Images:

Docker pull centos

🡪 list images from both (local registry/docker hub)

If not in local, will download from Docker Hub registry

Find Images:

Docker search usernam

🡪 List images from (local registry) if username exists.

**Login to Docker HUB**

Docker login

Username: xxxx

Password:xxxx

Email:xxxxx@xxx

**Modify Images:**

**Push and Remove Images from Host**

**## List Docker CLI commands**

docker

docker container --help

## Display Docker version and info

docker --version

docker version

docker info

## Execute Docker image

docker run hello-world

## List Docker images

docker image ls

**## List Docker containers (running, all, all in quiet mode)**

docker container ls

docker container ls --all

docker container ls -aq

docker build -t friendlyhello .  # Create image using this directory's Dockerfile

docker run -p 4000:80 friendlyhello  # Run "friendlyname" mapping port 4000 to 80

docker run -d -p 4000:80 friendlyhello         # Same thing, but in detached mode

docker container ls                                # List all running containers

docker container ls -a             # List all containers, even those not running

docker container stop <hash>           # Gracefully stop the specified container

docker container kill <hash>         # Force shutdown of the specified container

docker container rm <hash>        # Remove specified container from this machine

docker container rm $(docker container ls -a -q)         # Remove all containers

docker image ls -a                             # List all images on this machine

docker image rm <image id>            # Remove specified image from this machine

docker image rm $(docker image ls -a -q)   # Remove all images from this machine

docker login             # Log in this CLI session using your Docker credentials

docker tag <image> username/repository:tag  # Tag <image> for upload to registry

docker push username/repository:tag            # Upload tagged image to registry

docker run username/repository:tag                   # Run image from a registry

Child commands

Command        Description

docker attach   Attach local standard input, output, and error streams to a running container

docker build     Build an image from a Dockerfile

docker checkpoint        Manage checkpoints

docker commit Create a new image from a container’s changes

docker config   Manage Docker configs

docker container          Manage containers

docker cp         Copy files/folders between a container and the local filesystem

docker create   Create a new container

docker deploy   Deploy a new stack or update an existing stack

docker diff        Inspect changes to files or directories on a container’s filesystem

docker events   Get real time events from the server

docker exec      Run a command in a running container

docker export   Export a container’s filesystem as a tar archive

docker history   Show the history of an image

docker image   Manage images

docker images  List images

docker import   Import the contents from a tarball to create a filesystem image

docker info       Display system-wide information

docker inspect  Return low-level information on Docker objects

docker kill         Kill one or more running containers

docker load      Load an image from a tar archive or STDIN

docker login     Log in to a Docker registry

docker logout   Log out from a Docker registry

docker logs       Fetch the logs of a container

docker manifest            Manage Docker image manifests and manifest lists

docker network Manage networks

docker node     Manage Swarm nodes

docker pause    Pause all processes within one or more containers

docker plugin   Manage plugins

docker port      List port mappings or a specific mapping for the container

docker ps         List containers

docker pull       Pull an image or a repository from a registry

docker push     Push an image or a repository to a registry

docker rename Rename a container

docker restart   Restart one or more containers

docker rm         Remove one or more containers

docker rmi        Remove one or more images

docker run        Run a command in a new container

docker save      Save one or more images to a tar archive (streamed to STDOUT by default)

docker search   Search the Docker Hub for images

docker secret    Manage Docker secrets

docker service   Manage services

docker stack     Manage Docker stacks

docker start      Start one or more stopped containers

docker stats      Display a live stream of container(s) resource usage statistics

docker stop      Stop one or more running containers

docker swarm   Manage Swarm

docker system  Manage Docker

docker tag        Create a tag TARGET\_IMAGE that refers to SOURCE\_IMAGE

docker top        Display the running processes of a container

docker trust      Manage trust on Docker images

docker unpause            Unpause all processes within one or more containers

docker update  Update configuration of one or more containers

docker version  Show the Docker version information

docker volume  Manage volumes

docker wait       Block until one or more containers stop, then print their exit codes

**How to install Docker/Handle the service**

==========================

yum install docker

docker version //Give the version of the docker

Deamon : dockerd

Handle the service : sevice docker stop

                                     sevice docker start //Works on RHEL 7 also

Note : Docker configuration files located under : /etc/sysconfig

[root@centosbox1 sysconfig]# pwd;ls -lrt docker\*

/etc/sysconfig

-rw-r--r--. 1 root root  415 Jul  3  2017 docker-storage

-rw-r--r--. 1 root root   56 Jul  3  2017 docker-network

-rw-r--r--. 1 root root 1983 Jul  3  2017 docker

-rw-r--r--. 1 root root  179 Jul  3  2017 docker-storage-setup

[root@centosbox1 sysconfig]#

========================

1.docker --help

2.docker ps --->List running containers

3.docker ps -a //List all containers include runing and exit containers

4.docker images --->List the avilable images

5.dcoker run --help -->Doker run options

6.docker run -it --name <Name of container> <image>

Ex: docker run -it --name myubuntu docker.io/ubuntu

7.docker run -d -t --name <Name of container> <image> <process> //To run the docker in background

Ex: docker run -d -t --name myubuntu docker.io/ubuntu /bin/bash

8.docker run -it --name <Name of container> -h <hostnameof the container> -v <Path to store conatiner> <image> <Procees to run in container>

Ex:docker run -it --name vol-test -h CONTAINER -v /data debian /bin/bash

9.docker attach <Container ID/Name>/<Container Name> connect to docker contatiner

10.docker start <Container ID/Name> //Run the exit container

11.docker stop <Container ID/Name> //to stop the running container

12.docker kill <Container ID/Name> //To kill the container goto exit state doing it forcefully

13.docker exec -it <nameof the container> <process name> //To start the new process on running Container

Ex:  docker exec -it myubuntu /bin/bash

14.docker rm <nameof the container/conainerID> //To remove the exit container

15. docker rm $(docker ps -aq) //Remove the all exit containers in single shot

16. docker rmi <Image name> //To remove the docker image

17.docker rmi $(docker images -q) //TO remove the all docker images

18. docker inspect <ContainerId> //Give all infro---------mation about container

19. docker logs <name/id of the container> //To see the container logs

20.docker commit <containerID>  <Tagname>       //Commit container and make image

21. docker tag <image ID> name    //Tag the images //Tag the allready existing images

22 .docker diff <conatinerID> //it will compare with base images and give the difference

23 .Docker info //Give all configuration infromation of docker & list of the containers

=================================

Container Installation commands:

===================================

apt-get update //To update the ubuntu container

apt-get install net-tools //To install the network tools like ifconfig /ip

apt-get install ssh //Install openssh

dpkg -l //List the packages in Ubuntu

ctrl+pq //To comeout from container without exit

=================================

Building & Commiting & Shiping  the docker images

================================

Make file name as Dockerfile write the commands to run the coontainer in file

use : docker build -t <tag of the Docker> <Path of the Dockerfile>

Docker history <Nameof the Build Image> //Gives the information about Imaages involved for Docker Build

Example of docker file build & run to get the IP adress of your internet/Pubilc  IP

===========================

FROM ubntu

Maintainer Ram [ramchirumalla@gmail.com](mailto:ramchirumalla@gmail.com)

RUN apt-get update

RUN apt-get install -y openssh-server

RUN apt-get install -y curl && apt-get install -y vim && apt-get install -y net-tools

VOLUME /data

CMD curl ipinfo.io/ip

Note : Always dockerfile name should be "Dockerfile" under name of respective directiroy and time of Building ou have to give the directory Name:

Exapmple :

mkdir /var/tmp/publicippull/

vi Dockerfile //Write the code inside

docker build -t mypubippull /var/tmp/publicippull/

ENTRYPOINT : Is Keyworkd we uses the docker file to run certain command and options will be passsed the time of container exicution.

Commit the changes :

========================

docker commit <containerID>  <Tagname>       //Commit container and make image,Remeber always you have to save or commit the images with username/desiredname to push to docker hub

Ex: docker commit chirumal/myubuntu

Note : Even you can commit and make the images from  exist containers also

Shiping/Pushing the images:

============================

docker login -u chirumal -p <password>

docker push <image name>  //Remeber always you have to save or commit the images with username/desiredname

Docker Volumes :

=======================

DockerStorage Configuration files :

/etc/sysconfig/docker-storage

/etc/sysconfig/docker-storage-setup

/usr/lib/docker-storage-setup/docker-storage-setup  //Main config file

By default the volumes will be created under /var/lib/docker/volumes

Note : By default, Docker uses a loopback-mounted sparse file in /var/lib/docker.

The loopback makes it slower, and there are some restrictive defaults, such as 100GB max storage.

Dcoker run --help

  -v, --volume value                Bind mount a volume (default [])

      --volume-driver string        Optional volume driver for the container

      --volumes-from value          Mount volumes from the specified container(s) (default [])

Bind mount a volume:

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

docker run -it --name -v <sourcedir/volume>:<destinationdir> --name <name of the container>  <image> <process name> //Bind Mounting

Ex :

docker volume create --name volume1

dcker volume ls

docker run -it -v volume1:/vol1 --name datacontainer chirumal/ramubuntu /bin/bash

Give the volume to container:

=============================

docker run -it --name <name of the container> -v <mountdirectory> <image>

EX: docker run -it -v volume5 --name myvolcontainer2 chirumal/ramubunt //to mount the directory as a volume5 in container on docker engine machine volume under /var/lib/docker/volumes

Give the mount poitn as a volume you can proceed below way :

==================================================================

docker run -it -v /vol1/:/vol1 --name datacontainer chirumal/ramubuntu /bin/bash //Hear i mapped the /vol1 filesystem to the container.

for this we encounter permission issues its working selinux=premisive mode . these kind of volumes never handled by docker deamon

share the volume from one container to another container :

===============================================================

docker run -it --name master -v backup:/backup -v logs:/logs chirumal/ramubuntu  //This will create the direcltory under /var/lib/docker/volumes and map to container

docker run -it --name slave --volumes-from master chirumal/ramubuntu

Handling the Volumes:

======================

docker volume ls //List the voulmes

docker volume rm <VOlume Name>  //Remove the volume

docker volume inspect <VOlume Name> //to inspect the volume

docker volume prune    // Remove all unused volumes

docker volume create --name <name of the volume> //Create the volume this will create the direcltory under /var/lib/docker/volumes and map to container

Ex : docker volume          create --name my-vol1

docker run -it -v data:/data5 --name myvolcontainer2 chirumal/ramubunt //to map the directroy in side the continer with target name

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

Follow the below blog for more info on docker volumes:

======================================================

<http://container-solutions.com/understanding-volumes-docker/>

Docker Networking :

===========================

4 Types of netwokring support  in Docker : Bridge,overlay,Host,Macvlan

Network Drivers Summary:

1.User-defined bridge networks are best when you need multiple containers to communicate on the same Docker host.

2.Host networks are best when the network stack should not be isolated from the Docker host, but you want other aspects of the container to be isolated.

3.Overlay networks are best when you need containers running on different Docker hosts to communicate, or when multiple applications work together using swarm services.

4.Macvlan networks are best when you are migrating from a VM setup or need your containers to look like physical hosts on your network, each with a unique MAC address.

5.Third-party network plugins allow you to integrate Docker with specialized network stacks.

Bridge networks:  Bridge networks are usually used when your applications run in standalone containers that need to communicate.

================  Two types of brdige networks 1.Default bridge network .2 User defined bridge Network

Docker Engine and containers communicate containers with  "default bridge network" Defaultbrdige network not recomanded for production.

docker networks ls //To check the existing docker network

[root@centosbox1 ~]# docker network ls

NETWORK ID          NAME                DRIVER              SCOPE

352ccba0f798        bridge              bridge              local

e6a78bf74b68        host                host                local

5f3bb90398cb        none                null                local

The default Brdige network listed above.

The latter two are not fully-fledged networks, but are used to start a container connected directly to the Docker daemon

host’s networking stack, or to start a container with no network devices.

docker network inspect bridge //To insepect the bridge network and IPS infromation of the running containers

Note : The containers running under default network will comunicate with IPS but not with hostname

User defined bridge Network:Recomnded for production because

============================

1.its does the DNS resolution this capability is called automatic service discovery.

2.User-defined networks can attached/detacched on the fly,

3.Each user-defined network creates a configurable bridge means you can configure/EDIT your network as per requirements

Tools to Handle user defined networks:

=======================================

1.docker network create <networkname> //To create the user Bridge network

Ex: docker network create my-net

Docker netwroks ls

docker network inspect my-net

2.docker network rm <networkname> //To remove the bridge network

Ex: docker network rm my-net

3.docker network connect <network> <Running Container> //Add the network to the running Container

Ex :docker network connect my-net my-nginx

4.docker network disconnect network> <Running Container> //Remove the network to the running Container

Ex: docker network disconnect my-net my-nginx

5.Docker run -dit --name <name of Conatiner> --network <Network name> <image> <Process>

Ex: Docker run -dit --name alpine1 --network alpine-net alpine /bin/bash

Note :You can only connect to one network during the docker run command, so you need to use docker network attach for ading more networks

User below test Case try to ping other container with hostnames & Ip adress you understand better

=========================================

docker network create alpine-net

docker run -dit --name alpine1 --network alpine-net alpine ash

docker run -dit --name alpine2 --network alpine-net alpine ash

docker run -dit --name alpine3 alpine ash

docker run -dit --name alpine4 --network alpine-net alpine ash

docker network connect bridge alpine4

Host Networking :

==================

Deals with networking standalone containers which bind directly to the Docker host’s network, with no network isolation.

Nothing Container direclty maing the port to Docker.

Example below with nginx image hoe we can do this port maping we can see .

docker run -itd --network host --name my\_nginx nginx //Run the container map the container with docker host port number 80

netstat -tulpn | grep :80 //Check the port lising

Check the Nginx by browsing to <http://localhost:80/>

or you can expose the port like below:

docker run --name some-nginx -d -p 8080:80 some-content-nginx

then you can access from Browser : <http://localhost:8080>

**Docker Swarm:-**

===========================

docker swarm (enable Swarm mode; join a Swarm; adjust cluster parameters)

docker node (view nodes; promote/demote managers; manage nodes)

docker service (create and manage services)

docker node ls (to list nodes in the swarm cluster)

docker swarm init --advertise-addr 192.168.146.147  --listen-addr 192.168.146.147 (create the cluster)

docker swarm join-token manager (to know about the key information to make the node to join the cluster)

docker swarm join-token -q manager (to know about the key information to make the node to join the cluster)

docker info (daemon info)

below commands will open the required firewall ports to make the nodes to join the swarm

sudo firewall-cmd --add-port=2376/tcp --permanent

sudo firewall-cmd --add-port=2376/tcp --permanent

sudo firewall-cmd --add-port=2377/tcp --permanent

sudo firewall-cmd --add-port=7946/tcp --permanent

sudo firewall-cmd --add-port=7946/udp --permanent

sudo firewall-cmd --add-port=4789/udp --permanent

service firewalld restart (restart the firewall service after opening the firewall ports)

/bin/systemctl restart firewalld.service

docker node promote 03kp0mkhbsaj0yhfpqf1zxr7j <node ID> (to promote the worker node to manager)

docker node demote 03kp0mkhbsaj0yhfpqf1zxr7j  <node ID> (to Demote the Manager node to worker)

####Create a service featuring an Alpine container pinging Google resolvers:

docker service create --name pingpong alpine ping 8.8.8.8