Docker

Introduction

Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications

There are two big pieces to Docker: **The Docker Engine**, which is the Docker binary that's running on your local machine and servers and does the work to run your software. **The Docker Hub**, which is a website and cloud service that makes it easy for everyone to share their docker images.

Docker is available in two editions: **Community Edition (CE)** and **Enterprise Edition (EE)**.
Docker Community Edition (CE) is ideal for developers and small teams looking to get started with Docker and experimenting with container-based apps. Docker CE has two update channels, **stable** and **edge**:
Stable gives you reliable updates every quarter
Edge gives you new features every month
Docker Enterprise Edition (EE) is designed for enterprise development and IT teams who build, ship, and run business critical applications in production at scale.

Dockerfile (Docker builder):

- Dockerfile is a file, it will describe the steps to create an image quickly.
- The Dockerfile uses a basic DSL (Domain Specific Language) with instructions for building Docker images.
- The Docker daemon runs the instructions in the Dockerfile are processed from the top to down, so you should order them accordingly.
- **The Dockerfile is the source code of the Image.**

Docker image

- **An image is a lightweight, stand-alone, executable package that includes everything needed to run a piece of software, including the code, a runtime, libraries, environment variables, and config files.**

Container

- **A container is a runtime instance of an image**—what the image becomes in memory when actually executed. It runs completely isolated from the host environment by default, only accessing host files and ports if configured to do so.

Docker Commands

#docker --version (OR) #docker version
Docker version 17.03.1-ce, build c6d412e
#docker-compose --version
docker-compose version 1.11.2, build dfed245
 # docker-machine --version
docker-machine version 0.10.0, build 76ed2a6

```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker --version
Docker version 17.03.1-ce, build c6d412e
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker-compose --version
docker-compose version 1.11.2, build dfed245
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker-machine --version
docker-machine version 0.10.0, build 76ed2a6
```

#docker info: It will give status of weather docker is running or not
It will display the detailed information about docker service.

```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker info
Containers: 7
 Running: 0
 Paused: 0
 Stopped: 7
Images: 15
Server Version: 17.06.0-ce
Storage Driver: overlay2
 Backing Filesystem: extfs
 Supports d_type: true
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host ipvlan macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file logentries splunk syslog
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: cfb82a876ecc11b5ca0977d1733adbe58599088a
runc version: 2d41c047c83e09a6d61d464906feb2a2f3c52aa4
init version: 949e6fa
Security Options:
 seccomp
  Profile: default
Kernel Version: 4.9.36-moby
Operating System: Alpine Linux v3.5
OSType: linux
Architecture: x86_64
CPUs: 2
Total Memory: 1.952GiB
Name: moby
ID: UK7C:BJPU:EEIW:I2MT:4FDN:FNB4:BTIN:LFAQ:CCJT:Q6WE:GVCL:SCRH
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): true
 File Descriptors: 17
 Goroutines: 28
 System Time: 2017-07-20T15:10:41.157640585Z
 EventsListeners: 1
Registry: https://index.docker.io/v1/
Experimental: true
Insecure Registries:
 127.0.0.0/8
Live Restore Enabled: false

Bhaskars-MacBook-Air:~ BhaskarReddy$
```

Once you install the docker software it will insatll the docker client and daemon by default.

Client makes API calls to docker daemon.

Through client we can run the docker commands.
Suppose if you run the docker run <<Image Name>> in docker client, docker daemon will search in your local docker engine,if it not able to find it will download from the docker hub (http://hub.docker.com/).

#docker run <Docker Image Id / Docker Image Name>: To run any docker image, we will use run command.

Note: If you give wrong image name, you will get the below error.

```
Bhaskars-MacBook-Air:0 bhaskarreddyl$ docker pull mithuntechnologies-centos
Using default tag: latest
Error response from daemon: pull access denied for mithuntechnologies-centos, repository does not exist or may require 'docker login'
Bhaskars-MacBook-Air:0 bhaskarreddyl$
```

#docker run hello-world: This command will download the hello-world image, if it is not already present, and run the hello-world as a container.

Here hello-world is image name.

Note: While executing above command, if hello-world image is not there in local docker engine, and if your machine is not connected to internet, you will get below error.

```
Bhaskars-MacBook-Air:~ bhaskarreddyl$ docker run hello-world
Unable to find image 'hello-world:latest' locally
docker: Error response from daemon: Get https://registry-1.docker.io/v2/: Service Unavailable.
See 'docker run --help'.
Bhaskars-MacBook-Air:~ bhaskarreddyl$
```

```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/

Bhaskars-MacBook-Air:~ BhaskarReddy$
```

#docker images: List all images that are locally stored with the Docker engine
(OR)
#docker image ls

```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker images
REPOSITORY                                    TAG                 IMAGE ID            CREATED             SIZE
hello-world                                   latest              1815c82652c0        4 weeks ago         1.84 kB
mongo                                         latest              34ba9aead272        7 weeks ago         360 MB
registry.ng.bluemix.net/registryname/mongodbimage   latest              34ba9aead272        7 weeks ago         360 MB
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker image ls
REPOSITORY                                    TAG                 IMAGE ID            CREATED             SIZE
hello-world                                   latest              1815c82652c0        4 weeks ago         1.84 kB
mongo                                         latest              34ba9aead272        7 weeks ago         360 MB
registry.ng.bluemix.net/registryname/mongodbimage   latest              34ba9aead272        7 weeks ago         360 MB
Bhaskars-MacBook-Air:~ BhaskarReddy$
```

#docker images <<Image Name>>: It will display the information about image.

```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker images mongodbimage
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mongodbimage        latest              27d783d4d74d        24 hours ago        377MB
Bhaskars-MacBook-Air:~ BhaskarReddy$
```

#docker images -q: It will display only the images IDs.

```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker images -q
a46b6956e1ef
776068c05b0c
e4e6d42c70b3
c035090355f5
d355ed3537e9
a2ff708b7413
1815c82652c0
34ba9aead272
34ba9aead272
Bhaskars-MacBook-Air:~ BhaskarReddy$
```

#docker rmi << Image ID/Image Name>> (OR) docker image rm << Image ID/Image Name>>: It will delete an image from the local image store.

#docker rmi nging

```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker rmi nginx
Untagged: nginx:latest
Untagged: nginx@sha256:423210a5903e9683d2bc8436ed06343ad5955c1aec71a04e1d45bd70b0d68460
Deleted: sha256:e4e6d42c70b3f79c5d57c170526592168992eb3303a6594c439302fabd92d9a3
Deleted: sha256:67f1adf25899acf829e72865ec039f1b07c7ad84c9f57a77730618433fea3d80
Deleted: sha256:3f9e7d79387e7c0bedc20907cb1408074bb56145313cb9a5712858e691a5ebb9
Deleted: sha256:54522c622682789028c72c5ba0b081d42a962b406cbc1eb35f3175c646ebf4dc
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker images
REPOSITORY                                    TAG                 IMAGE ID            CREATED             SIZE
hello-world                                   latest              1815c82652c0        4 weeks ago         1.84 kB
registry.ng.bluemix.net/registryname/mongodbimage   latest              34ba9aead272        7 weeks ago         360 MB
mongo                                         latest              34ba9aead272        7 weeks ago         360 MB
Bhaskars-MacBook-Air:~ BhaskarReddy$
```

#docker rmi -f 27d783d4d74d: With image id also we can remove docker image.

```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker rmi 27d783d4d74d
Error response from daemon: conflict: unable to delete 27d783d4d74d (must be forced) - image is referenced in multiple repositories
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker rmi -f 27d783d4d74d
Untagged: bhaskar0504/get-mangodb:newmongodb
Untagged: mongodbimage:latest
Deleted: sha256:27d783d4d74d43008b61808055dfc0584a3e4f42385619ddec7dac8916f7ceaa
Deleted: sha256:55dfdcdaa2ae7209f216627f30ffafa6d867510c9fb47b85f514a762f250bce0
Deleted: sha256:9db2f463f6ccb7b5abc8e8ea5ef1ec1f16ea25009c803478540a62738abe7ed1
Deleted: sha256:ea0b67fe74abf9d12d058a52237985afb1d9d9ebdf1396ff294f846bfa5c0baf
Deleted: sha256:fcb7a4e81615d5dce2795d7fc9b489fd7be62e3b42ac5e333117959b305acc74
Deleted: sha256:f9c0f9fbd92cad1b0587c3393271e4a4e0a2e8f17f953c1087c4c9ecc9cbe8f5
Bhaskars-MacBook-Air:~ BhaskarReddy$
```

#docker rmi $(docker images -q) (OR) docker rmi -f $(docker images -q) : It will remove all the images from docker engine.

#docker run --name "hello-world-container" helloworld: Start the hello-world image with "**hello-world-container**" container name.

#docker create "hello-world-cont" helloworld : It will create a container called "hello-world-cont" from the image hello-world and it won't start the container.

Note: If the image is not found locally, Docker will pull it from Docker Hub.

#docker ps (OR) docker container ls: Lists running containers
(It will not display the stopped containers)
#docker ps –a (OR) docker container ls --all (OR) docker container -a: Lists all containers
(It will display the stopped containers along with running containers.)

```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
19ce01f02a5a        nginx               "nginx -g 'daemon ..."   15 minutes ago      Up 15 minutes       0.0.0.0:80->80/tcp  webserver
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS                     NAMES
19ce01f02a5a        nginx               "nginx -g 'daemon ..."   23 minutes ago      Up 23 minutes               0.0.0.0:80->80/tcp        webserver
849c0fb09af6        hello-world         "/hello"                 38 minutes ago      Exited (0) 38 minutes ago                             hardcore_nightingale
9982ebcc5610        mongo               "docker-entrypoint..."   4 weeks ago         Exited (255) 43 minutes ago 0.0.0.0:27017->27017/tcp  infallible_agnesi
0c09bbf1c36b        hello-world         "/hello"                 4 weeks ago         Exited (0) 4 weeks ago                                wizardly_newton
f7ee8295302a        hello-world         "/hello"                 4 weeks ago         Exited (0) 4 weeks ago                                dazzling_tesla
```

docker ps -a
#docker stop <container name|id> (OR) docker container stop <container name|id>: It will stop the docker container.
#docker stop webserver: It will stop the container called webserver.

```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker stop webserver
webserver
Bhaskars-MacBook-Air:~ BhaskarReddy$
```

#docker start <Container name|id>
#docker start webserver: It will start the webserver.

```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker start webserver
webserver
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
19ce01f02a5a        nginx               "nginx -g 'daemon ..."   38 minutes ago      Up 6 seconds        0.0.0.0:80->80/tcp  webserver
Bhaskars-MacBook-Air:~ BhaskarReddy$
```

#docker pause CID/CNAME: It will pause the container.
#docker unpause CID/CNAME: It will unpause the container.
Docker Container status are One of below
created, restarting, running, removing, paused, exited, or dead

#docker ps -a --filter "name=mithun16thjune": It will display all the containers with name mithun16thjune name.

#docker ps -a --filter 'exited=0' :

#docker ps --filter status=running:
#docker ps --filter status=paused:

#docker logs <container name>: It will display the logs for that container.
#docker logs --tail 100 <<Container Name>>: Print the last 100 lines of a container's logs.

#docker top <<Container ID>>: This will shows the top processes in within in a container.
#docker top 1486dbea7b5e

```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker top 1486dbea7b5e
PID                 USER                TIME                COMMAND
3011                root                0:00                nginx: master process nginx -g daemon off;
3035                101                 0:00                nginx: worker process
Bhaskars-MacBook-Air:~ BhaskarReddy$
```

#docker reanme <<Container Old Name>> <<Container New Name>>: It will rename the conatiner.
#docker rm -f <<Container Name>>: It will remove the container.

#docker rm -f webserver: It will stop and remove the running container with a single command.

```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker rm -f webserver
webserver
Bhaskars-MacBook-Air:~ BhaskarReddy$
```

#docker stop: It will restrt the container.
#docker stop $(docker ps -a -q): It will stop all the containers.
#docker rm -f $(docker ps -aq): Delete all running and stopped containers.

#docker kill <<CID/C Name>>: It will kill the containe.

#docker container prune: Delete all stopped containers.

```
Last login: Thu Jul 26 16:41:44 on console
bhaskars-air:~ bhaskarreddy1$ docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
a1890257439cbfaf3cdb5b422a728870629717fe6228fed99fd880cbfddcf211
be89bad4218f24ecad41478ca43a743b44b64019afb5e62e72df60d94f6e02a9

Total reclaimed space: 0B
```

#docker ps -a|awk '$2=="hello-world" {print $1}' |xargs docker rm : It will delete all the containers related hello-world image.

#docker ps --filter "status=paused"

#docker search <<Image Name>>: It will search all of the publicly available images on Docker Hub(https://hub.docker.com).

Note: This command will not work if your machine is not connect to internet.

```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker search ubuntu
Error response from daemon: Get https://index.docker.io/v1/search?q=ubuntu&n=25: dial tcp: lookup index.docker.io on 192.168.65.1:53: no such host
Bhaskars-MacBook-Air:~ BhaskarReddy$
```

#docker search puppet: It will search puppet images from Dockerhub.

```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker search puppet
NAME                                DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
puppet/puppetserver                 A Docker Image for running Puppet Server. ...   27
devopsil/puppet                     Dockerfile for a container with puppet ins...   24                   [OK]
puppet/puppetdb                     A Docker image for running PuppetDB             15
puppet/puppetserver-standalone      An image for running a Puppet Server stand...   10
puppet/puppet-agent-alpine          Puppet Agent as a Docker Image. Based on t...   10
puppet/puppet-agent-ubuntu          Puppet Agent as a Docker Image. Based on t...   9
puppet/puppetdb-postgres            An image based on the upstream Postgres im...   7
puppet/puppetexplorer               The Puppet Explorer dashboard for PuppetDB      6
puppet/puppetboard                  The Puppet Board dashboard for PuppetDB         6
puppet/puppet-agent                 Puppet Agent as a Docker Image.                 4
puppet/puppet-agent-centos          Puppet Agent as a Docker Image. Based on t...   3
puppet/facter                       Facter as a Docker Image                        3
puppet/puppet-agent-debian          Puppet Agent as a Docker Image. Based on t...   2
vladgh/puppet                       Ubuntu 16.04 LTS Base image with Puppet         2                    [OK]
jumanjiman/puppet                   Use Puppet to configure CoreOS hosts            2
solict/provisionous-puppet-centos   CentOS provisions with Puppet included          2                    [OK]
bigtop/puppet                       Images build by ci.bigtop.apache.org jobs ...   1
thekevjames/puppet-moduletest       Matrix test and validation tool for puppet...   1
vshn/puppet-puppetserver            VSHN Puppet Environment in Docker - Puppet...   1                    [OK]
coigovpl/puppet-forge-server        Docker image to run a private Puppet Forge...   0                    [OK]
capd/puppet                         puppet                                          0                    [OK]
whatsaranjit/puppet-agent           Used for Puppet Kubernetes demo                 0
mfsysops/puppet                     Puppet build                                    0
itherz/puppet                       Dockerized Puppet Server with MCO (ActiveM...   0                    [OK]
samaelson/puppet                    Puppet lint image for CI                        0                    [OK]
Bhaskars-MacBook-Air:~ BhaskarReddy$
```

This command will returns the below details.

- Repository names
- Image descriptions
- Stars - these measure the popularity of an image
- Official - an image managed by the upstream developer (e.g., the ubuntu image managed by the Ubuntu team)
- Automated - an image built by the Docker Hub's Automated Build process

#docker search --no-trunc puppet

```
bhaskars-air:~ bhaskarreddy1$ docker search --no-trunc puppet
NAME                                DESCRIPTION                                                                                STARS     OFFICIAL   AUTOMATED
puppet/puppetserver                 A Docker Image for running Puppet Server. Will try and connect to a linked PuppetDB instance.   43
macadmins/puppetmaster              Simple puppetmaster based on CentOS 6                                                      25                   [OK]
devopsil/puppet                     Dockerfile for a container with puppet installed. Based on CentOS 6.x                      24                   [OK]
puppet/puppetdb                     A Docker image for running PuppetDB                                                        16
puppet/puppetserver-standalone      An image for running a Puppet Server standalone (ie. not connected to a PuppetDB instance)  14
puppet/puppet-agent-ubuntu          Puppet Agent as a Docker Image. Based on the latest Ubuntu image.                          10
puppet/puppet-agent-alpine          Puppet Agent as a Docker Image. Based on the latest Alpine image.                          10
camptocamp/puppetserver             Puppetlabs's puppetserver                                                                  7                    [OK]
puppet/puppetboard                  The Puppet Board dashboard for PuppetDB                                                    7
puppet/puppetdb-postgres            An image based on the upstream Postgres image with the PuppetDB additions                  7
alekzonder/puppeteer                GoogleChrome/puppeteer image and screenshots scripts                                       6                    [OK]
puppet/puppetexplorer               The Puppet Explorer dashboard for PuppetDB                                                 6
puppet/puppet-agent                 Puppet Agent as a Docker Image.                                                            4
jumanjiman/puppet                   Use Puppet to configure CoreOS hosts                                                       3
vladgh/puppet                       Ubuntu 16.04 LTS Base image with Puppet                                                    2                    [OK]
vladgh/puppetserver                 Vlad's Puppet Server                                                                       2                    [OK]
vladgh/puppetserverdb               Vlad's Puppet Server configured for PuppetDB                                               1                    [OK]
vladgh/puppetdb                     Vlad's PuppetDB                                                                            1                    [OK]
thekevjames/puppet-moduletest       Matrix test and validation tool for puppet modules.                                        1
vpgrp/puppet                        Docker images of Puppet.                                                                   1                    [OK]
vshn/puppet-puppetserver            VSHN Puppet Environment in Docker - Puppetserver                                           1                    [OK]
samaelson/puppet                    Puppet lint image for CI                                                                   0                    [OK]
camptocamp/puppetdb                 Puppetlabs's puppetdb                                                                      0                    [OK]
coigovpl/puppet-forge-server        Docker image to run a private Puppet Forge Server (https://github.com/unibet/puppet-forge-server)   0                    [OK]
itherz/puppet                       Dockerized Puppet Server with MCO (ActiveMQ), Hiera, R10k                                  0                    [OK]
bhaskars-air:~ bhaskarreddy1$
```

#docker search --filter=stars=10 puppet

```
bhaskars-air:~ bhaskarreddy1$ docker search --filter=stars=10 puppet
NAME                                DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
puppet/puppetserver                 A Docker Image for running Puppet Server. ...   43
macadmins/puppetmaster              Simple puppetmaster based on CentOS 6           25                   [OK]
devopsil/puppet                     Dockerfile for a container with puppet ins...   24                   [OK]
puppet/puppetdb                     A Docker image for running PuppetDB             16
puppet/puppetserver-standalone      An image for running a Puppet Server stand...   14
puppet/puppet-agent-ubuntu          Puppet Agent as a Docker Image. Based on t...   10
puppet/puppet-agent-alpine          Puppet Agent as a Docker Image. Based on t...   10
bhaskars-air:~ bhaskarreddy1$
```

#docker pull <<Image Name>>: Pull an image from Docker Hub

Note: If you give wrong image name, you will get the below error.

```
Bhaskars-MacBook-Air:0 bhaskarreddyl$ docker pull mithuntechnologies-centos
Using default tag: latest
Error response from daemon: pull access denied for mithuntechnologies-centos, repository does not exist or may require 'docker login'
Bhaskars-MacBook-Air:0 bhaskarreddyl$
```

#docker pull ubuntu: It will download the ubuntu image from docker hub.
If no tag is provided, Docker Engine uses the :latest tag as a default.
This command pulls the ubuntu:latest image.
#docker pull ubuntu:16.04 : It will download the ubuntu 16.04 version.
#docker run -i -t ubuntu:16.04 /bin/bash: It will conenct to the ubuntu. (By default user is root)

```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker run -i -t ubuntu  /bin/bash
root@333c854108f0:/#
```

#docker inspect <<CID>> : It will give information for container.
Note: You can use “docker inspect “ command to get information for containers, netwroks.., not only for containers.

#docker inspect --format '{{ .State.Status }}' <<CID>> :

#docker inspect --format '{{ .State.Pid }}' <<CID>> :

#docker inspect --format '{{ .NetworkSettings.IPAddress }}' <<CID>> :

For More info: https://docs.docker.com/engine/reference/commandline/inspect/#get-an-instances-mac-address

#docker attach <<CID>> : It will conenct to running container.

Note: If the container is in stop/pause state while executing docker attach command, you will get below error.

```
Bhaskars-MacBook-Air:0 bhaskarreddyl$ docker attach 2cfd7585fdfd
You cannot attach to a stopped container, start it first
```

```
Bhaskars-MacBook-Air:~ bhaskarreddyl$ docker attach 708cf881340d
You cannot attach to a paused container, unpause it first
```

#docker exec <<CID>> : Run a linux command in a running container.

#docker exec <<CID>> ls :

#docker exec <<CID>> hostname :

#docker exec <<CID>> touch /tmp/mithuntechnologies.txt :

#docker stats <<CID>> : It will display a live stream of container resource usage statics.

Note: If you want to come out form console, type Ctrl + C.

Execute docker stats and then execte below command in another command prompt and see the result.
#docker exec <<CID>> yum update

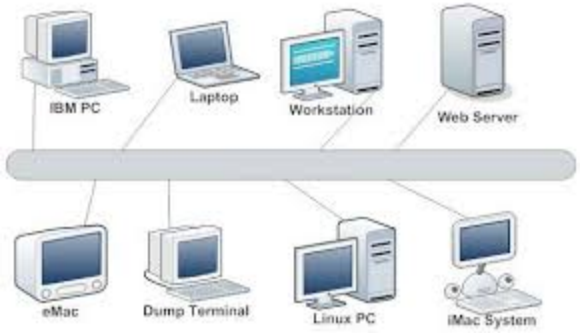
If the container is paused, then the docker exec command will fail with an error:
#docker pause <<CID|CNAME>>

#docker exec <<CID>> ls : It will give error, because the container is in paused state.

#docker network --help : It will display the all options for network command.

#docker network ls: List the networks.

Network is a group of two or more devices that can communicate.
In practice, a network is comprised of a number of different computer systems connected by physical and/or wireless connections.




```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
16d3fc470e85        bridge              bridge              local
b2b646cb327e        host                host                local
6f285ce058f5        none                null                local
Bhaskars-MacBook-Air:~ BhaskarReddy$
```

#docker network create mithuntechnologies : It will create the network with name called as mithuntechnologies with default type is bridge.

#docker network rm mithuntechnologies : It will delete the network with name called as mithuntechnologies.

#docker network inspect bridge : Display detailed information on one or more networks.

#docker network connect : Connect a container to a network

docker network connect [OPTIONS] NETWORK CONTAINER

Options:

--alias strings	Add network-scoped alias for the container
--ip string	IPv4 address (e.g., 172.30.100.104)
--ip6 string	IPv6 address (e.g., 2001:db8::33)
--link list	Add link to another container

--link-local-ip strings Add a link-local address for the container

#docker network prune: Remove all unused networks.

Docker Volumes

By default, all files created inside a container.
The data doesn't persist when that container is no longer running, and it can be difficult to get the data out of the container if another process needs it.

Volume is a specially designed directory.
Volumes are created and managed by Docker, if you don't create explicitly and set.
Volumes are stored in a part of the host filesystem which is managed by Docker (/var/lib/docker/volumes/ on Linux).

We can create a volume explicitly using the **'docker volume create'** command
docker voulume --help:
Create a container with volume mithuntechnoVol1
#docker run -it -v /Users/bhaskarreddyl/Desktop/dockervolumes:/mithuntechnoVol1 ubuntu

Create the Read only Volume
docker run -it -v /Users/bhaskarreddyl/Desktop/dockervolumes:/mithuntechnoVol1:ro --name ubuntucontainer16 ubuntu

```
bhaskars-air:dockervolumes bhaskarreddyl$ docker run -it -v /Users/bhaskarreddyl/Desktop/dockervolumes:/mithuntechnoVol1:ro ubuntu
root@bee454b5947d:/# touch mithuntechnoVol1/test2.txt
touch: cannot touch 'mithuntechnoVol1/test2.txt': Read-only file system
root@bee454b5947d:/#
```


#docker run -it --volumes-from ubuntucontainer16 --name ubuntucontainer1604 ubuntu:16.04:
Create a container ubuntucontainer1604 that uses the same volumes as ubuntucontainer16

Can I mount same volume to multiple docker containers?
Ans) Yes you can add same location as a volume to many docker containers.

#docker login: To sign into the Docker Hub.

```
Bhaskars-MacBook-Air:~ bhaskarreddyl$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: mithuntechnologies
Password:
Login Succeeded
Bhaskars-MacBook-Air:~ bhaskarreddyl$
```

If no server is specified, then the default is used
Default is https://index.docker.io/v1/
https://registry-1.docker.io/v2/

#docker logout: To logout from the Docker Hub. If no server is specified, then the default is used.

```
Bhaskars-MacBook-Air:~ BhaskarReddy$ docker logout
Removing login credentials for https://index.docker.io/v1/
Bhaskars-MacBook-Air:~ BhaskarReddy$
```

Note: In docker hub we can create only one private repository. We can create many public repositories.

Dockerfile have various keywords.

FROM: This keyword indicates the base image from which the container is built.

MAINTAINER: One of the commands that can be set anywhere in the file - although it would be better if it was declared on top - is MAINTAINER. This non-executing command declares the author, hence setting the author field of the images. It should come nonetheless after FROM.

RUN: This keyword indicates the command that needs to be executed on the image.

EXPOSE: The EXPOSE command is used to associate a specified port to enable networking between the running process inside the container and the outside world (i.e. the host).

ENV: The ENV command is used to set the environment variables (one or more). These variables consist of “key = value” pairs which can be accessed within the container by scripts and applications alike.

COPY: Copying local files into container.

CMD: The command CMD, similarly to RUN, can be used for executing a specific command. However, unlike RUN it is not executed during build, but when a container is instantiated using the image being built.

ENTRYPOINT : ENTRYPOINT has two forms:

ENTRYPOINT ["executable", "param1", "param2"] (exec form, preferred)
ENTRYPOINT command param1 param2 (shell form)
An ENTRYPOINT allows you to configure a container that will run as an executable.

VOLUME : The VOLUME command is used to enable access from your container to a directory on the host machine (i.e. mounting it).
WORKDIR : The WORKDIR directive is used to set where the command defined with CMD is to be executed.

What is the difference between ADD and COPY?

COPY and ADD are both Dockerfile instructions that serve similar purposes. They let you copy files from a specific location into a Docker image.

COPY command will copy files/directories from your host machine to your image.
ADD command will copy files/directories from you host machine to your image and also it will copy tar files and will extract that file and remove the tar file once it extract. In this case if you use COPY command you need to use layers like below.

COPY mithuntechnologies.tar /tmp/
RUN tar -xvf /tmp/mithuntechnologies.tar -C /usr/local/
RUN rm /tmp/mithuntechnologies.tar

If you user ADD

ADD mithuntechnologies.tar /usr/local/

And also ADD is used for getting files from remotely and copy into image as follows.

ADD http://mithuntechnologies.com/mithuntechnologies.tar /usr/src/mithun/

What is the difference between RUN and CMD?

RUN is an image build step, the state of the container after a RUN command will be committed to the docker image. A Dockerfile can have many RUN steps that layer on top of one another to build the image.

CMD is the command the container executes by default when you launch the built image. A Dockerfile can only have one CMD. The CMD can be overridden when starting a container with docker run $image $other_command.

ENTRYPOINT is also closely related to CMD and can modify the way a container starts an image.

With help of below command we can create the docker image from Dockerfile.

docker build

Syntax: docker build -t ImageName:TagName dir

Here

- -t is to mention a tag to the image
- ImageName – This is the name you want to give to your image
- TagName – This is the tag you want to give to your image
- Dir – The directory where the Dockerfile is present.

Example: docker build -t mongodbimage:1.0 .

In this example **mongodbimage** is the image name
1.0 is the tag name
. is the current directory for the Dockerfile.

Since the Docker File is in the present working directory, we used "." at the end of the command to signify the present working directory.

Example: docker build -f /path/to/a/Dockerfile.

Task: Create the one Dockerfile to install httpd and copy index.html file into /usr/local/apache2/htdocs/ directory.

Step 1: Create the file name called Dockerfile and put the below contents.

```
FROM httpd:2.4
MAINTAINER DevOps Training <devopstrainingblr@gmail.com>
COPY ./index.html /usr/local/apache2/htdocs/
#EXPOSE 80
```

Step 2: docker build -t mithuntechnohttpserver:1.0 .

Step 3: docker run -p 8081:80 httpd (**Note:** If you use EXPOSE 80 in Docker file, while running image no need to mention -p 80)

Once you ran the above command successfully, open the browser and type the localhost:8081, it will display the contents of index.html.

Note: If it will not work, use IP address instead of localhost

Step 4: docker images

Step 5: Tag the image
docker tag <<Image ID>> mithuntechnologies/httpdimage:tag
Note: Here mithuntechnologies is username/organisation name and httpdimage is the repository name.
To find the image id use the **docker images** command.
Step 6: Publish the image to docker hub

docker push mithuntechnologies/httpdimage

```
bhaskars-air:~ bhaskarreddyl$ docker push mithuntechnologies/http_customimge
The push refers to repository [docker.io/mithuntechnologies/http_customimge]
c91d78dac84e: Preparing
bb7e751144a8: Preparing
2131c7c317ea: Preparing
00ab72b0018f: Preparing
9d38604080fc: Preparing
25639e024724: Waiting
3d35e332eca0: Waiting
1618a71a1198: Waiting
denied: requested access to the resource is denied
bhaskars-air:~ bhaskarreddyl$
```

If you see this error you have to login into docker hub with below command.

docker login

docker diff <Container ID/Container Name> : Inspect changes to files or directories on a container's filesystem
A ---> A file or directory was added
D ---> A file or directory was deleted
C ---> A file or directory was changed

Task: Create the one Dockerfile to install one war on Tomcat server.

Step 1: Create the file name called **Dockerfile** and put the below contents.

```
FROM tomcat:8.0.20-jre8

COPY ./SampleAntProject.war /usr/local/tomcat/webapps/SampleAntProject.war
```

Note: Make sure executing below commands , need to have SampleAntProject.war file and Dockefile in same directory.

Follow below commands.

docker build -t tomcatimage:8.0 .

docker run -p 8088:8080 tomcatimage

Once it started open the browser and type http://localhost:8088/SampleAntProject

---

 # docker save -o ~/Desktop/backup_vnc_container.tar vnc_container:

```
Bhaskars-MacBook-Air:~ bhaskarreddy1$ docker images
REPOSITORY                                                    TAG                 IMAGE ID            CREATED             SIZE
vnc_container                                                 latest              08c4bbae15b0        12 days ago         1.98GB
registry.ng.bluemix.net/gaia_automation_namespace/vnc_container   latest              08c4bbae15b0        12 days ago         1.98GB
dockervnc                                                     latest              2a5933dfb10b        13 days ago         1.12GB
ubuntu                                                        16.04               dd6f76d9cc90        2 weeks ago         122MB
hello-world                                                   latest              725dcfab7d63        2 weeks ago         1.84kB
Bhaskars-MacBook-Air:~ bhaskarreddy1$ docker save -o ~/Desktop/backup_vnc_container.tar vnc_container
Bhaskars-MacBook-Air:~ bhaskarreddy1$
```

 # docker load -i ~/Desktop/vnc_container.tar

```
Bhaskars-MacBook-Air:~ bhaskarreddy1$ docker load -i ~/Desktop/backup_vnc_container.tar
Loaded image: vnc_container:latest
```

What are the deployment strategies you followed in docker swarm?
blue/green technique.
https://container-solutions.com/deployment-strategies/

DockerHub URL
https://hub.docker.com

Create the Org
Create the Repo
Create Teams

Errors

1) Sending build context to Docker daemon 3.072kB
Error response from daemon: No build stage in current context
Cause: Issue with Dockerfile, something is wrong with Dockerfile syntax.
Solution: Check Dockerfile and fix it.

Before the Docker daemon runs the instructions in the Dockerfile, it performs a preliminary validation of the Dockerfile and returns an error if the syntax is incorrect:

Resources:
https://docs.docker.com/get-started/
https://www.digitalocean.com/community/tutorials/docker-explained-using-dockerfiles-to-automate-building-of-images ---> Dockerfile Commands (Docker Keywords)
https://github.com/wsargent/docker-cheat-sheet ---> Docker Commands Cheat sheet

https://training.play-with-docker.com/

https://docs.docker.com/engine/reference/builder/ --->Docker Keywords

http://containertutorials.com/volumes/volume_from_image.html ---> Docker Network.

https://docs.docker.com/engine/docker-overview/