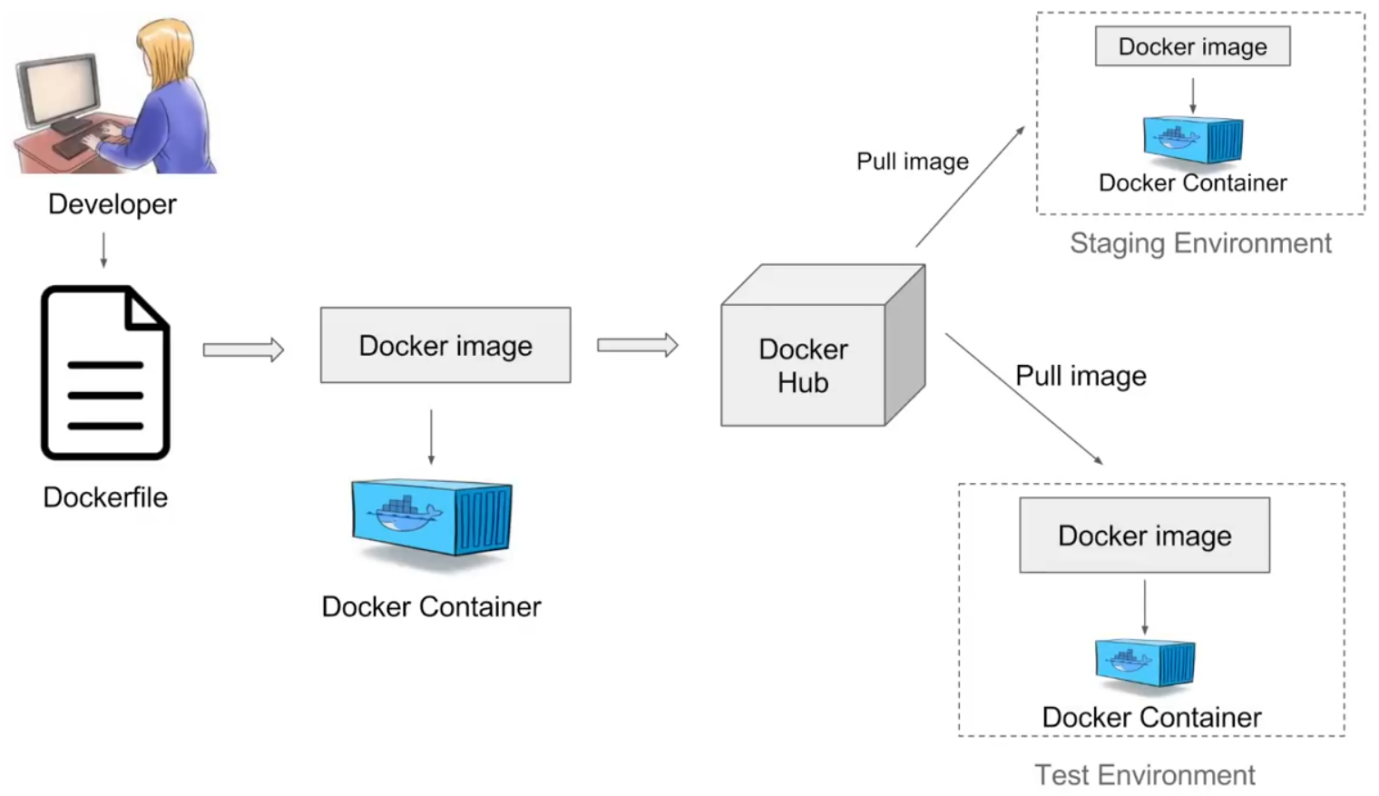
**Docker:**

**What is Docker:**

Docker is a tool designed to make it easier to 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.

**Workflow of Docker:**

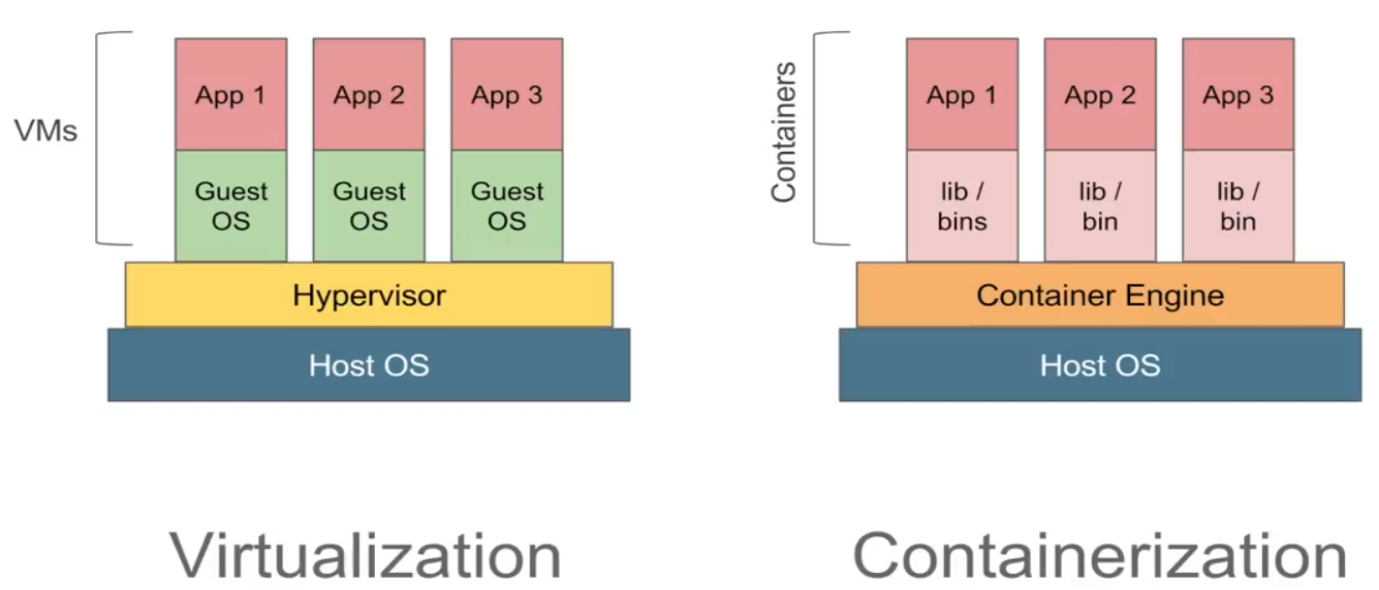
--------------------------------------------------------------------------------------------------------------------------------------

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

Docker is Container platform:

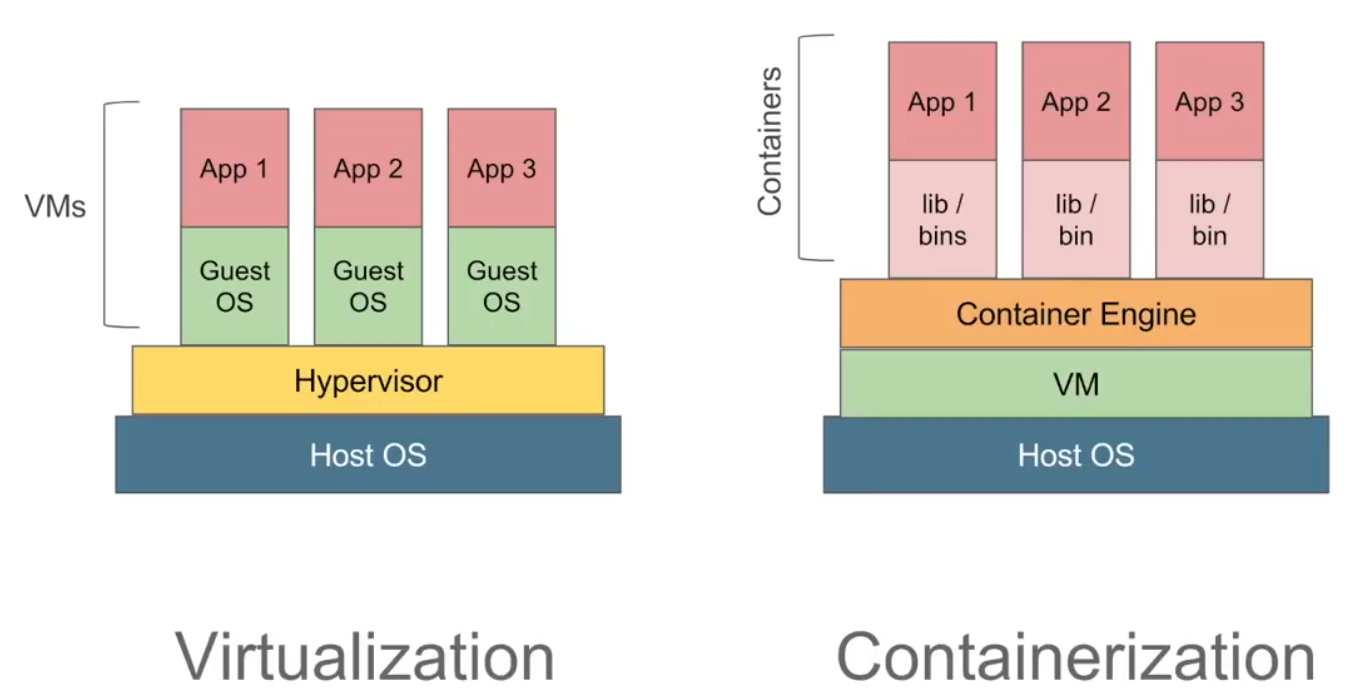
**Virtualization VS Containerization**

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



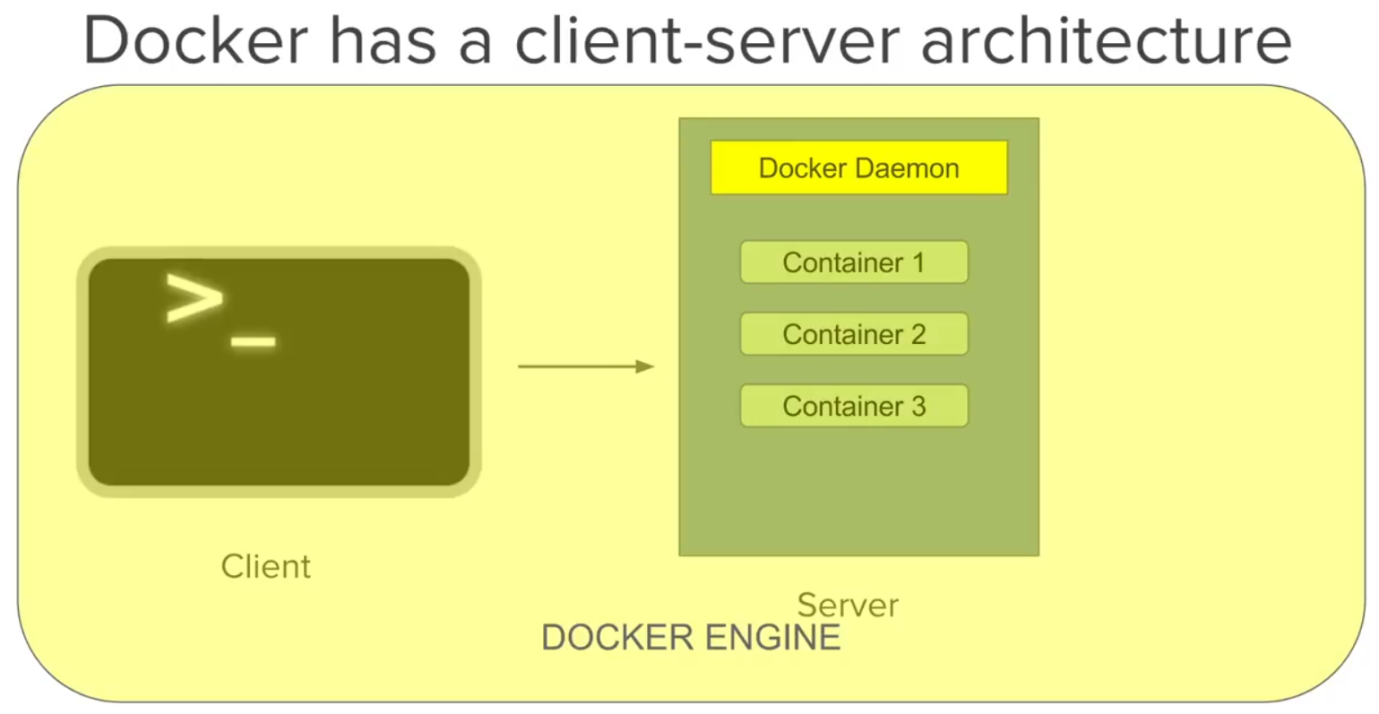
If u want run windows application in Linux machine so in that case, we need to have VM first and then we can run our application through containers.

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



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

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



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

The daemon (Server) receives the command from the Docker client through CLI or REST API’s.

Docker client and daemon can be present on the same host (machine) or different host.

**Build ap only Once:** An Application inside a container can run on any system that has docker installed. So, there is no need to build and configure app multiple times on different platform.

**More sleep and less worry:** With Docker you test your application inside a container and ship it inside a container.

This means the environment in which you test is identical to the one on which the app will run in production.

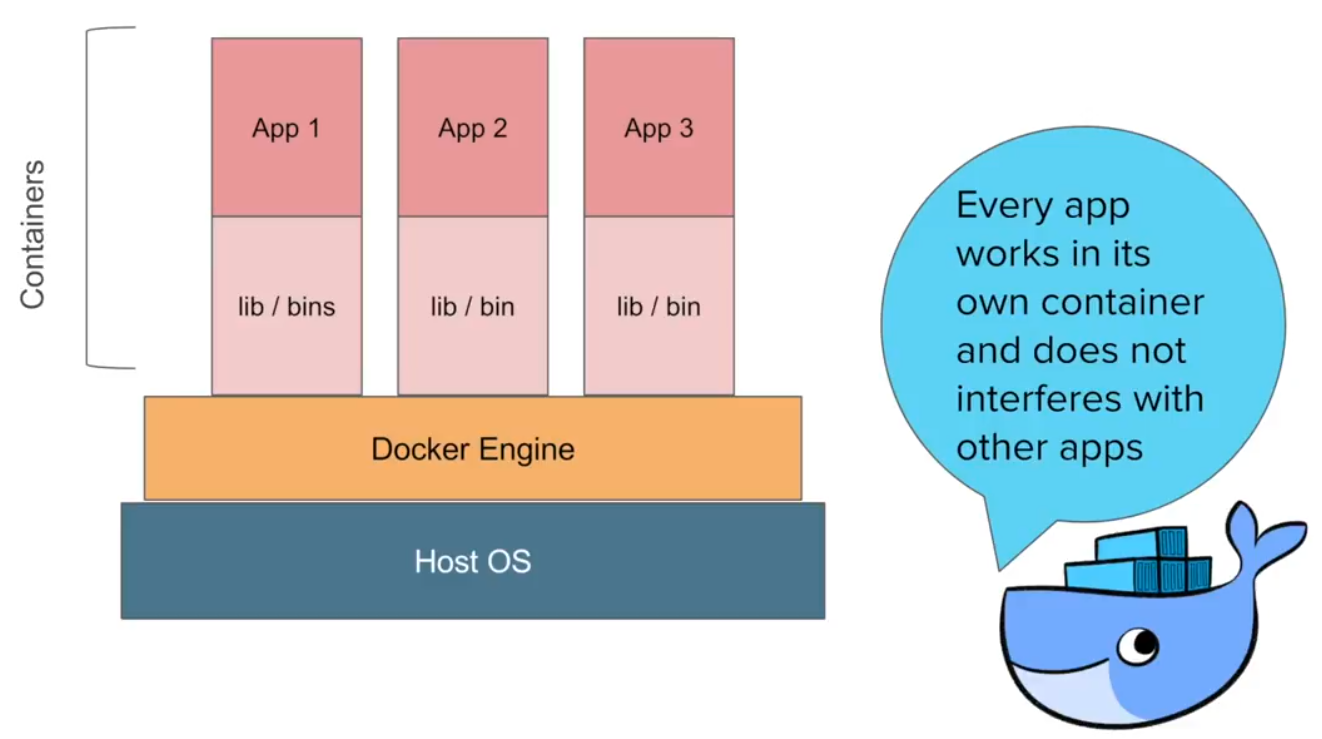
Docker container can run on any platform. It can run on your local system, Amazon EC2, Google Cloud platform, Rackspace Server, Virtual Box and etc….

**Version Control:** Like Git, Docker has in-built version control system.

Docker containers work just like GIT repositories, allowing you to commit changes to your docker images and version control them.

**Feature of Docker:**

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



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

**Isolation:** With Docker every application works in isolation in its own container and does not interface with other applications running on the same system.

So Multiple containers we can run on the same system without interference.

For Removal also you can simply delete the container and it will not leave behind any files or traces on the system.

**Productivity:** Docker allows faster and more efficient deployments without worrying about running your app on different platform.

It increases productivity many folds.

**Docker installation in Linux machines**

**Prerequisites:**

* Connect to Linux
* Install Docker
* Start Docker
* Stop Docker
* Uninstall Docker

**Connect to Linux:** Login to any of the Linux Machine.

**Install Docker:** sudo yum -y update and sudo yum install -y docker and command “docker” will give the options.

**Start Docker:** sudo service docker start and sudo usermod -a -G docker <system username> (no need to use sudo), docker images, docker info, docker ps, docker ps -a, docker run <image name>

**Stop Docker:** sudo service docker stop

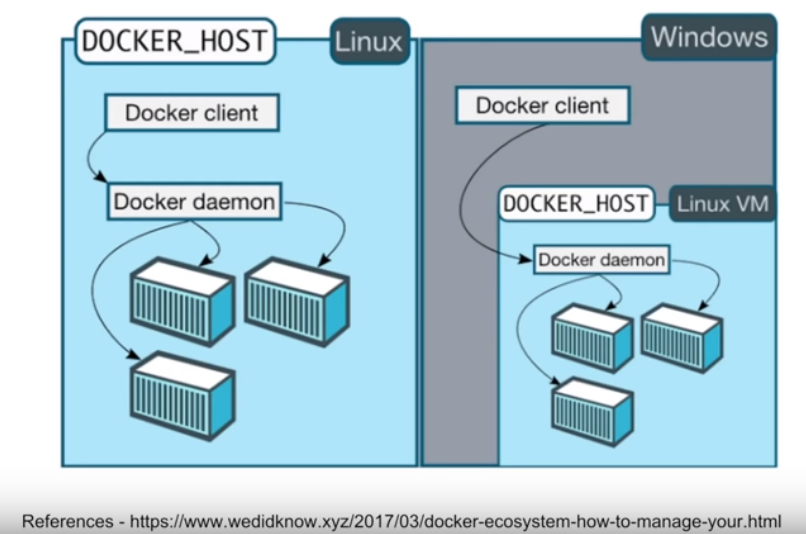
**UnInstall Docker:** sudo yum remove docker

HELPFUL TIPS You can visit - [https://get.docker.com/](https://www.youtube.com/redirect?event=video_description&v=KCckWweNSrM&redir_token=bV_OxCbYW_sp6N2aKrNpZtDDEWF8MTU4NTkwODI5OUAxNTg1ODIxODk5&q=https%3A%2F%2Fget.docker.com%2F) for more installation related help To install docker from binaries [https://docs.docker.com/engine/instal...](https://www.youtube.com/redirect?event=video_description&v=KCckWweNSrM&redir_token=bV_OxCbYW_sp6N2aKrNpZtDDEWF8MTU4NTkwODI5OUAxNTg1ODIxODk5&q=https%3A%2F%2Fdocs.docker.com%2Fengine%2Finstallation%2Fbinaries%2F) Installation steps for amazon ec2 [http://docs.aws.amazon.com/AmazonECS/...](https://www.youtube.com/redirect?event=video_description&v=KCckWweNSrM&redir_token=bV_OxCbYW_sp6N2aKrNpZtDDEWF8MTU4NTkwODI5OUAxNTg1ODIxODk5&q=http%3A%2F%2Fdocs.aws.amazon.com%2FAmazonECS%2Flatest%2Fdeveloperguide%2Fdocker-basics.html)

**Docker installation in Windows machines**

**Prerequisites:**

* Install Docker
* Troubleshooting tips
* Validate Installation
* Uninstall Docker



**Install Docker:** STEP 1: Install Docker [https://docs.docker.com/toolbox/toolb...](https://www.youtube.com/redirect?event=video_description&v=ymlWt1MqURY&q=https%3A%2F%2Fdocs.docker.com%2Ftoolbox%2Ftoolbox_install_windows%2F&redir_token=Ac1zDB1AvmmMnsc_qzS5M1RtChF8MTU4NTkwOTU0MEAxNTg1ODIzMTQw) [https://docs.docker.com/docker-for-wi...](https://www.youtube.com/redirect?event=video_description&v=ymlWt1MqURY&q=https%3A%2F%2Fdocs.docker.com%2Fdocker-for-windows%2Finstall%2F&redir_token=Ac1zDB1AvmmMnsc_qzS5M1RtChF8MTU4NTkwOTU0MEAxNTg1ODIzMTQw)

**Troubleshooting tips:** STEP 2: Troubleshooting (if any issue occurs) Convenient way to enable/disable Hyper-V [https://superuser.com/questions/54005...](https://www.youtube.com/redirect?event=video_description&v=ymlWt1MqURY&q=https%3A%2F%2Fsuperuser.com%2Fquestions%2F540055%2Fconvenient-way-to-enable-disable-hyper-v-in-windows-8&redir_token=Ac1zDB1AvmmMnsc_qzS5M1RtChF8MTU4NTkwOTU0MEAxNTg1ODIzMTQw)

Installing Docker Toolbox on Windows with Hyper-V Installed [https://jayvilalta.com/blog/2016/04/2...](https://www.youtube.com/redirect?event=video_description&v=ymlWt1MqURY&q=https%3A%2F%2Fjayvilalta.com%2Fblog%2F2016%2F04%2F28%2Finstalling-docker-toolbox-on-windows-with-hyper-v-installed%2F&redir_token=Ac1zDB1AvmmMnsc_qzS5M1RtChF8MTU4NTkwOTU0MEAxNTg1ODIzMTQw)

**Validate Installation:** STEP 3: validate INSTALLATION run some docker commands – docker, docker –version, docker run "image name"

**Uninstall Docker:** control panel – uninstall program feature.

**Docker installation in Mac machines**

**Prerequisites Docker** –

Mac OS Yosemite 10.10.3 or above

Docker Toolbox - mac OS 10.8 “Mountain Lion” or newer

STEP 1 Install DOCKER

STEP 2 validate INSTALLATION run some docker commands

STEP 3 uninstall DOCKER

Note: “kitematic” tool for accessing the Docker. It is a front and tool.

References: [https://www.docker.com/](https://www.youtube.com/redirect?redir_token=HMgIqqUx35dHuci5ibsBTjSBAnp8MTU4NTkxMTUzMkAxNTg1ODI1MTMy&event=video_description&v=MU8HUVlJTEY&q=https%3A%2F%2Fwww.docker.com%2F)

<https://www.docker.com/docker-mac>

<https://store.docker.com/editions/com...>

<https://www.docker.com/products/docke...>

<https://docs.docker.com/toolbox/overv...>

<https://docs.docker.com/toolbox/toolb...>

**Basic Commands of Docker:**

**Basic:**

* docker version – it will give the docker information. It will give the docker client and server related information like docker client/server version, api version, os information, etc…
* docker -v/docker --version – it will give the docker version.
* docker info – it will give some details about docker like how many running, stopped, paused containers and how many images, storage driver info and etc…

# docker images --help – it will give options of all docker image command. (Ex: docker run help, etc...

* docker login – it will for the docker hub username and password. (we need to create account in the docker hub site. [https://hub.docker.com/](https://www.youtube.com/redirect?redir_token=A9UewwVzuPHHQxXT5p7nEEENsrd8MTU4NTkxNDUwNUAxNTg1ODI4MTA1&event=video_description&v=HqBMEmoAd1M&q=https%3A%2F%2Fhub.docker.com%2F))

**Images:**

* docker images – it will give the Existing images list. And “docker images -q” – it will give image id
* docker pull – docker pull <image name> - it will download the images.
* docker rmi – docker rmi <image name> - it will remove the images.

**Containers:**

* docker ps – it will give the running container id.
* docker start <container id> - it will start container.
* docker stop <container id> - it will stop container.
* docker run <image> - it will start the images and it will create container and it will give the container id.

**System:**

* docker stats – it will give all the details about running container details like mem usage, id, CPU load, etc...
* docker system df – it will give the memory usage of docker images.
* docker system prune – it will remove the unused data in the docker, and it will remove all unused images not just dangling (images which are not associate with the container) ones. (docker system prune --help), it will remove the stopped container also.

**Docker file**

* What is Docker file
* How to create Docker file
* How to build image from Docker file
* Basic commands

**What is Docker file:**

A text file with instructions to build image Automation of Docker Image Creation

FROM, RUN, CMD

**How to create Docker file:**

1st step – create a file called “Dockerfile” without any extension.

2nd step – add instruction in docker file.

3rd step – build docker file to create image.

4th step – run image to create container.

“Vim dockerfile”

Inside the file –

# getting base image <image name>

FROM <image name>

MAINTAINER <the person name, who is the maintainer of the image> ##### name of the person

RUN apt-get update #### it will execute the during during building the image

CMD [“echo”, “hello world…! From my first docker image”] ##### it will display, when you created container out of the image and while running the image.

Save and close the file.

**How to build image from docker file and Commands:**

“docker build <path of the file (mention only path of the file, don’t mention file name)>” - To execute the created file

“docker build -t <tag image name:tag> <path of the file>”

References:

<https://github.com/wsargent/docker-ch...>

<https://docs.docker.com/engine/refere...>

<https://www.google.co.in/search?q=doc...>

**Docker Compose file**

Docker compose:

tool for defining & running multi-container docker applications

use yaml files to configure application services (docker-compose.yml)

can start all services with a single command: docker compose up

can stop all services with a single command: docker compose down

can scale up selected services when required

Step 1: install docker compose (already installed on windows and mac with docker)

docker-compose -v

2 Ways

* [https://github.com/docker/compose/rel...](https://www.youtube.com/redirect?q=https%3A%2F%2Fgithub.com%2Fdocker%2Fcompose%2Freleases&event=video_description&v=HUpIoF_conA&redir_token=n42MPyARWkRD5Mdyq8YYfvyiSs98MTU4NTk4ODQ2NkAxNTg1OTAyMDY2)
* Using PIP pip install -U docker-compose

Step 2: Create docker compose file at any location on your system “docker-compose.yml”

Step 3: Check the validity of file by command “docker-compose config”

Step 4: Run docker-compose.yml file by command “docker-compose up -d”

Step 5: Bring down application by command “docker-compose down”

**Inside compose file:**

Vim docker-compose.yml

version: ‘3’

services:

web:

image: nginx

ports:

* 9090:80

database:

image: redis

save and close the file.

**TIPS**

How to scale services

—scale “docker-compose up -d --scale database=4” (duplicates the instances)

(nginx is web application server and redis is a database server. If u want replication of any services, then need to use the “--scale <service name>=number of replication”. Ex: --scale database=4 – it will create 4 duplicate instance of the database server(redis))

References:

<https://hub.docker.com>

<https://github.com/docker/compose/rel...>

<https://docs.docker.com/compose/compo...>

<https://www.google.co.in/search?q=mic...>

**Persistent Storage Volumes:**

It is external storage disc feature. Whenever container goes dies we can retrieve the data from external disc.

1. What are Volumes
2. How to create /list /delete volumes
3. How to attach volume to a container
4. Hoe to share volume among containers
5. What are bind mounts

**Commands:**

docker volume --help

docker volume create <volume name> - it will create new volume.

docker volume ls – list the exists volumes.

docker volume inspect <volume name> - it will give details about volume.

docker volume rm <volume name> - it will remove the volume.

docker volume prune – it will remove all unused volumes.

**Commands:**

docker pull <image name>

docker pull <image name:tag>

docker image –help

docker pull image

docker images

docker images -q – it will display container id.

docker images -f – it will give filter option.

docker images -f “dangling=false” – it will give the dangling false images info.

docker images -f “dangling=true” – it will give the dangling images info.

docker images -f “dangling=false” -q

docker images -a

docker run -it <image name> - it will give interactive mode and it will go to inside of the container.

docker run -it <image name> bash - it will give interactive mode and it will go to inside of the container and it will give the shell.

docker run –name <name of the container> -it <image name> bash – we can assign name to the container.

docker inspect – it will give complete system level image details.

docker rmi image – to remove the docker images.

docker rmi -f image – force fully we can remove the images.

docker attach <container id> - it will make the container up always until unless we need to exit the container.

docker exec -it <container id> - it is used for login to container and which should be active one.

docker exec <container id> touch <filename> - it will create file inside the container without login.

docker commit <container id> <new image name:tag> - it will create docker image from the container id.

docker rm <container id> - it will remove the container.

docker start <container id> - it will start the container.

docker stop <container id> - it will stop the container.

docker restart <container id> - it will stop and starts the container.

docker kill <container id> - it will stop the container forcefully.

docker rm $(docker ps -aq) – it will remove all containers at a time. (a – it will display all containers, q – it will display all container id’s only)

docker run -itd -p 8080:80 <image name/id> - it will map the port numbers with host machine. (-p – port publish, 8080 – it is host machine port, 80 – it is container machine port)

(after this u can check it ports are available or not through the command and command is “netstat -ntlp” – it will give the available ports in the machine. (n-numerical, t-tcp, l-listening, and p-program/pid))

docker ps

docker run <image name>

docker start container id/name

docker stop container id/name

docker pause container id/name

docker unpause container id/name

docker top container name/id – it will give top running process in the docker.

docker stats container name/id – it will give all details about running process like Linux top command.

docker attach container name/id – it will attach the container.

docker rm container name/id – it will remove the container.

docker history image name/id – it will give all recently executed command on docker.

docker container diff <container id> - it will give the what and all changes we have done in that container.

docker volume --help – it will give the option of the docker volume command.

docker volume create <volume name> - it will create new volume.

docker volume ls – list the volumes.

docker volume inspect <volume name> - it will give all the details about volume, like mount point, name.

docker volume rm <volume name> - it will delete the volume.

docker volume prune – it will remove all unused volumes.

“docker container cp <file name of host machine> <container name>:/path/of/the/container/folder”

“docker container create --name <container name> -it --mount source=volume\_name,target=/path/of/the/container/folder”

**Docker compose file commands:**

docker-compose build

docker-compose build web

docker-compose up -d

docker-compose up -d web

docker-compose down

docker-compose down web

docker-compose ps

docker-compose exec web ls -l

docker-compose start

docker-compose start web

docker-compose stop

docker-compose stop web

docker-compose restart

docker-compose restart web

docker-compose pause - it will pause and un-pause the container.

docker-compose pause web

docker-compose rm

docker-compose rm web

**Docker network commads:**

docker network connect

docker network create

docker network disconnect

docker network inspect

docker network ls

docker network prune

docker network rm

docker network create --subnet 172.20.0.0/16 --ip-range 172.20.240.0/24 <network name>

docker network connect --ip 172.20.128.2 <network name> <container name>

docker network disconnect <network name> <container name>

docker network connect <network name> <container name>

**Converting the container to new image:**

docker container commit <container id> - it will create new image of our existing container and it is not having any name, it is having only new image id.

docker image tag <new image id> <name of the new image with tag> - for giving name to the new image, which is having the without name.

**Uploading Image to the docker hub:**

docker login – it is for login to the docker hub command and it will ask username and password.

docker push <image name> - it is command for uploading our new image to the docker hub.

(Note: While removing any image it should not be associate with any container. If it is associated also, we can remove the images, 1st we can remove the container and we can remove the images, or another way is we can remove the images directly by using “-f” option)

**Docker file creation commands:**

* FROM – The base images for building a new image and it should be top of the file.
* MAINTAINER – Optional, it contains the name of the maintainer.
* RUN – Used to execute the command during the build process of docker image.
* ADD – it is used for extracting the zip file of your local machine and put into the container machine and also, we can give the direct url to download the files. (ADD <zip file of local machine> <destination of container machine>, ex1: ADD /root/file.tar.gz /home/, ex2: ADD <url> /hom/)
* ENV – Define an Environment variable. (Ex: ENV work=/root/, WORKDIR $work)
* CMD – Used for executing commands when we build a new container from the docker image.
* ENTRYPOINT – it is 1st command to execute after starting container. It is not overwriting this command but in “CMD” we can overwrite the command.
* WORKDIR – is is used for current working directory. Means if you have so many commands to run in same directory that time we can set as our working directory.

(Ex: mkdir /root/test1, touch /root/file1, touch file2 in this time we can make “WORKDIR /root/” and after that we can use the instruction as like “mkdir ./test1, touch ./file1, touch ./file2)

* USER – Set the user or UID for the container created with the image.
* VOLUME – Enable access/linked directory between the container and the host machine. ( VOLUME ["/etc/mysql", "/var/lib/mysql"])
* EXPOSE – it is for exporting the port number.
* COPY – it is used to copy the folders/files from local machine to the container machine. (COPY <source path of local machine> <destination path in container machine>)

**Git Commands:**

git --version

git config --global user.email "yourGitHub@email.com"

git config --global user.name "yourGitHubusername"

git init – it will create “.git” folder in the current directory.

git status – it will check the current folder files.

git add <file name> - it will add the files to the local repository, but it will not commit.

git commit -m <message of file and it will display in git after committing> - commit the changes to local repository.

git commit -a -m <message of file and it will display in git after committing> - commit the all changes to local repository.

git add \*.\* - it will add all files to the local repo.

git add \*.txt – it will add all txt files to the local repo.

git add . – it will add all current folder files to the local repo.

git remote add origin <location of remote repo> - it will add the remote location of github repo.

git push -u origin master – it will push committed code to the github repo.

**Commit the code to existing git repo:**

git init

git add . or git add -A

git commit -m “comment of the commit”

git remote add origin <git repo url>

git remote -v

git push -f origin master