**What is Docker?**

Docker is a set of coupled software-as-a-service and platform-as-a-service products that use operating-system-level virtualization to develop and deliver software in packages called containers. The software that hosts the containers is called Docker Engine.



Docker image is package of software(java, os and python) and application(Technology).

Place where the docker image run is docker container.

Docker is a container Platform. Docker is tool designed to make it easier to deploy and run applications by using container.

Containers allow a developer to package up the application with all the parts it needs, such as libraries and other dependencies, and ship it all out as one package.

docker images

sc849m@zldcmtn23adce1dokr03:~$ docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

dockercentral.it.att.com:5100/com.att.dcae.controller/dcae-controller-mibpoller 19.04-001 90b83870116b 9 weeks ago 405.8 MB

docker exec -it 277627fa1e5a /bin/bash

sc849m@zldcmtn23adce1dokr03:~$ docker exec -it 277627fa1e5a /bin/bash

bash-4.4#

bash-4.4# ls -alrt

total 44

drwxr-xr-x 3 root root 4096 Apr 11 21:32 infrastructure

drwxr-xr-x 4 root root 4096 Apr 11 21:32 etc

drwxr-xr-x 3 root root 4096 Apr 11 21:32 lib

drwxr-xr-x 4 root root 4096 Apr 11 21:32 logs

drwxr-xr-x 3 root root 4096 Apr 11 21:42 archive

drwxr-xr-x 8 root root 4096 Apr 11 21:42 ..

drwxr-xr-x 3 root root 4096 Apr 11 21:42 tmp

drwxr-xr-x 7 root root 4096 Apr 11 21:42 config

drwxr-xr-x 7 root root 4096 May 7 19:53 bin

drwxr-xr-x 13 root root 4096 May 7 19:53 .

-rw------- 1 root root 578 May 15 20:20 .bash\_history

Shell scripting location:

bash-4.4# cd bin/

bash-4.4# ls

common configManager dockerSupplement tools

bash-4.4# cd dockerSupplement/

bash-4.4# ls

nohup.out startDockerSupplement.sh

bash-4.4#

Configuration Location: meta data and json

bash-4.4# cd config/

bash-4.4# ls

configManager dockerSupplement logger tasks

bash-4.4# cd dockerSupplement/

bash-4.4# ls

docker\_supp\_hp\_rule.json hpVesMapper.json metaDataMRBH.txt metaDataVGDF.txt

hpMainConf.json mainConf.json metaDataSCAS.txt metaDataVSILB.txt

Process and jar location :

bash-4.4# ps -ef| grep java

root 89 1 0 Jul16 ? 00:58:35 java -cp /opt/app/vcc/lib/vccHighlandParkPlugins.jar:/opt/app/vcc/lib/dockerSupplement.jar:/opt/app/vcc/lib/attMaven/\* -DHOSTNAME=8be304fdffe14d82a1368aad5acffbfb\_dcae.collectors.dockersupplement.pm -DVCC\_HOME=/opt/app/vcc -Dlogback.configurationFile=/opt/app/vcc/config/logger/logback\_vcc.xml -DisThreadContextMapInheritable=true -Dsubcomponent=DOCKERSUPPLEMENT-VCC -DPROJECTHOME=/opt/app/vcc -DAPPLNAME=dcae com.att.sa.highlandPark.HpStandalone -server -DdockerSupplement -c /opt/app/vcc/config/dockerSupplement/hpMainConf.json,/opt/app/vcc/config/dockerSupplement/hpVesMapper.json -f true --factory com.att.sa.highlandPark.events.HpJsonEventFactory -XX:+UnlockExperimentalVMOptions -XX:+UseCGroupMemoryLimitForHeap -Xms256m -Xmx2G

bash-4.4#

bash-4.4# cd ../../

bash-4.4# cd lib/

bash-4.4# ls

attMaven common.jar configManager.jar dockerSupplement.jar vccHighlandParkPlugins.jar

bash-4.4# ls -alrt

total 144

-rw-r--r-- 1 root root 19181 Apr 11 21:31 common.jar

-rw-r--r-- 1 root root 14460 Apr 11 21:31 configManager.jar

-rw-r--r-- 1 root root 68911 Apr 11 21:31 vccHighlandParkPlugins.jar

-rw-r--r-- 1 root root 18211 Apr 11 21:32 dockerSupplement.jar

drwxr-xr-x 2 root root 12288 Apr 11 21:32 attMaven – pom.xml jar files

drwxr-xr-x 3 root root 4096 Apr 11 21:32 .

drwxr-xr-x 13 root root 4096 May 7 19:53 ..

log location :

cd logs

bash-4.4# ls

application.1.log.zip application.log dockerSupplement\_2019-04-23.log dockerSupplement\_2019-07-19.log

application.10.log.zip audit.log dockerSupplement\_2019-04-24.log dockerSupplement\_2019-07-20.log

application.11.log.zip debug.log dockerSupplement\_2019-04-25.log dockerSupplement\_2019-07-21.log

application.12.log.zip dockerSupplement.log.2019\_04\_11 dockerSupplement\_2019-04-26.log dockerSupplement\_2019-07-22.log

application.13.log.zip dockerSupplement.log.2019\_04\_15 dockerSupplement\_2019-04-27.log dockerSupplement\_2019-07-23.log

application.14.log.zip dockerSupplement.log.2019\_05\_07 dockerSupplement\_2019-04-28.log dockerSupplement\_2019-07-24.log

application.15.log.zip dockerSupplement.log.2019\_07\_02 dockerSupplement\_2019-04-29.log dockerSupplement\_2019-07-25.log

application.16.log.zip dockerSupplement.log.2019\_07\_11 dockerSupplement\_2019-04-30.log dockerSupplement\_2019-07-26.log

application.17.log.zip dockerSupplement.log.2019\_07\_16 dockerSupplement\_2019-07-02.log dockerSupplement\_2019-07-27.log

application.18.log.zip dockerSupplement\_2019-04-11.log dockerSupplement\_2019-07-03.log dockerSupplement\_2019-07-28.log

application.19.log.zip dockerSupplement\_2019-04-12.log dockerSupplement\_2019-07-04.log dockerSupplement\_2019-07-29.log

application.2.log.zip dockerSupplement\_2019-04-13.log dockerSupplement\_2019-07-05.log dockerSupplement\_2019-07-30.log

application.20.log.zip dockerSupplement\_2019-04-14.log dockerSupplement\_2019-07-06.log error.log

application.2019-04-14\_0247.log dockerSupplement\_2019-04-15.log dockerSupplement\_2019-07-07.log metrics.log

application.3.log.zip dockerSupplement\_2019-04-16.log dockerSupplement\_2019-07-08.log performance.log

application.4.log.zip dockerSupplement\_2019-04-17.log dockerSupplement\_2019-07-09.log policy.log

application.5.log.zip dockerSupplement\_2019-04-18.log dockerSupplement\_2019-07-10.log security.log

application.6.log.zip dockerSupplement\_2019-04-19.log dockerSupplement\_2019-07-11.log server.log

application.7.log.zip dockerSupplement\_2019-04-20.log dockerSupplement\_2019-07-16.log

application.8.log.zip dockerSupplement\_2019-04-21.log dockerSupplement\_2019-07-17.log

application.9.log.zip dockerSupplement\_2019-04-22.log dockerSupplement\_2019-07-18.log

bash-4.4# pwd

/opt/app/vcc/logs/DCAE/DOCKERSUPPLEMENT-VCC

bash-4.4# crontab -l

# do daily/weekly/monthly maintenance

# min hour day month weekday command

\*/15 \* \* \* \* run-parts /etc/periodic/15min

0 \* \* \* \* run-parts /etc/periodic/hourly

0 2 \* \* \* run-parts /etc/periodic/daily

0 3 \* \* 6 run-parts /etc/periodic/weekly

0 5 1 \* \* run-parts /etc/periodic/monthly

MicroService Deployment Platform :

EOM Docker – which is deploy on plain docker container

EOM K8 ( Kubernetes ) – It deployed on any docker container in cluster.(how to check which container?)

**Rajasingh Nagarathinam(7:37:51 PM):** <https://codecloud.web.att.com/projects/ST_CCSASDC/repos/design_spec/browse> -- Component Specification

**Rajasingh Nagarathinam(7:38:09 PM):** If you don't have access contact Lisa

**Rajasingh Nagarathinam(7:38:20 PM):** REVEL, LISA <[lr0306@att.com>](mailto:lr0306@att.com);

**Rajasingh Nagarathinam(7:50:14 PM):** <http://dcae-onboardingtoolbox-dev2-dyh1b-edge-s5.ecomp.idns.cip.att.com:30998/web/generate_blueprint> - For BP Generation

<https://docker-curriculum.com/>

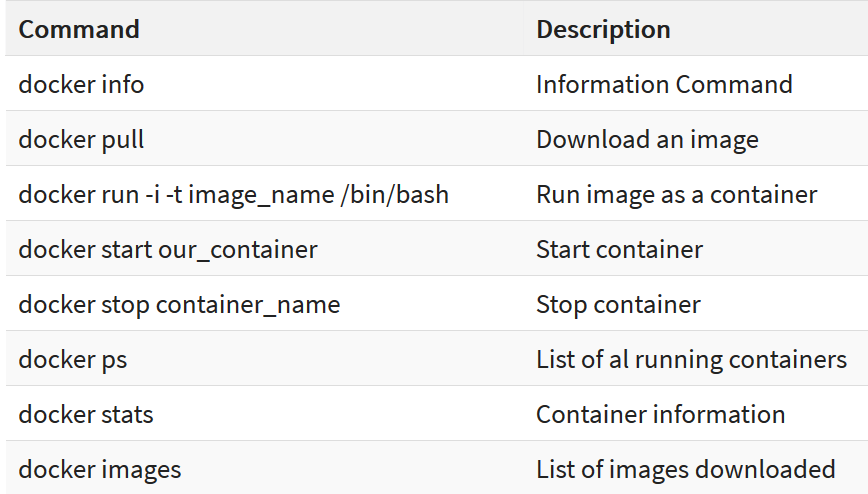
 *mages* - The blueprints of our application which form the basis of containers. In the demo above, we used the docker pull command to download the **busybox** image.

 *Containers* - Created from Docker images and run the actual application. We create a container using docker run which we did using the busybox image that we downloaded. A list of running containers can be seen using the docker ps command.

 *Docker Daemon* - The background service running on the host that manages building, running and distributing Docker containers. The daemon is the process that runs in the operating system to which clients talk to.

 *Docker Client* - The command line tool that allows the user to interact with the daemon. More generally, there can be other forms of clients too - such as [Kitematic](https://kitematic.com/) which provide a GUI to the users.

 *Docker Hub* - A [registry](https://hub.docker.com/explore/) of Docker images. You can think of the registry as a directory of all available Docker images. If required, one can host their own Docker registries and can use them for pulling images.



Start the docker

service docker start

service docker stop

Docker commands:

Basic:

docker ps –a

docker version

docker –v

docker info

docker –help

Images :

docker login (login the docker hub)

docker images

docker pull (get the images from docker hub) docker pull openjdk

docker pull ubuntu:18.04

docker rmi <images\_ID> (to remove the images) –f flag to remove forcefully.

docker inspect <image\_name>

docker images -f “dangling=false”

https://hub.docker.com/

Container :

docker ps

docker run <image\_name/ID>

docker run –name <container\_name> -it busybox sh

docker run -p 8080:8080 -p 50000:50000 -v /your/home:/var/jenkins\_home jenkins

docker stop <container\_ID/Name>

docker start <container\_ID/Name>

docker rm 305297d7a235 ff0a5c3750b9 (remove the container)

docker rm $(docker ps -a -q -f status=exited)

docker container ls

docker exec –it <container ID> /bin/bash

docker pause <container\_ID/Name>

docker unpause <container\_ID/Name>

docker top <container\_ID/Name>

docker stats <container\_ID/Name>

docker kill <container\_ID/Name>

docker history <container\_ID/Name>

docker volumn create <container\_ID/Name>

docker volumn ls

System :

docker stats

docker system df

docker system prune (remove all the images, container and caches which are not used)

**History of Virtualization**

Earlier, the process for deploying a service was slow and painful. First, the developers were writing code; then the operations team would deploy it on bare metal machines, where they had to look out for library versions, patches, and language compilers for the code to work. If there were some bugs or errors, the process would start all over again, the developers would fix it, and then again the operational team was there to deploy.

There was an improvement with the creation of Hypervisors. Hypervisors have multiple Virtual machines or VMs on the same host, which may be running or turned off. VMs decreased the waiting time for deploying code and bug fixing in a big manner, but the real game changer was Docker containers.

## What is Docker?

Docker is an open-source lightweight containerization technology. It has gained widespread popularity in the cloud and application packaging world. It allows you to automate the deployment of applications in lightweight and portable containers.

Docker is computer software used for Virtualization in order to have multiple Operating systems running on the same host. Unlike Hypervisors which are used for creating VM (Virtual machines), virtualization in Docker is performed on system-level in so-called Docker containers.

## Docker Engine

Docker is the client-server type of application which means we have clients who relay to the server. So the Docker daemon called: dockerd is the Docker engine which represents the server. The docker daemon and the clients can be run on the same or remote host, and they communicate through command line client binary, as well as a full RESTful API to interact with the daemon: dockerd.

### Docker Images

Docker images are the "source code" for our containers; we use them to build containers. They can have software pre-installed which speeds up deployment. They are portable, and we can use existing images or build our own.

### Registries

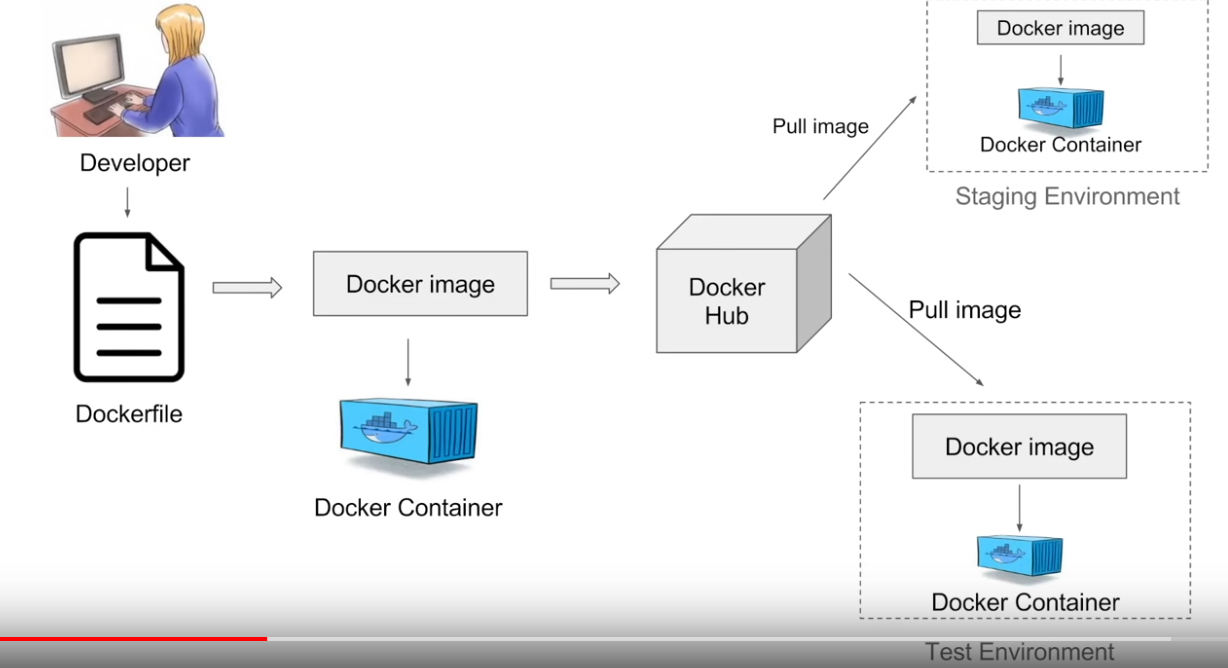
Docker stores the images we build in registries. There are public and private registries. Docker company has public registry called [Docker hub](https://hub.docker.com/), where you can also store images privately. Docker hub has millions of images, which you can start using now.

### Docker Containers

Containers are the organizational units of Docker. When we build an image and start running it; we are running in a container. We can move it, in other words, "ship" the software, modify, manage, create or get rid of it, destroy it, just as cargo ships can do with real containers.

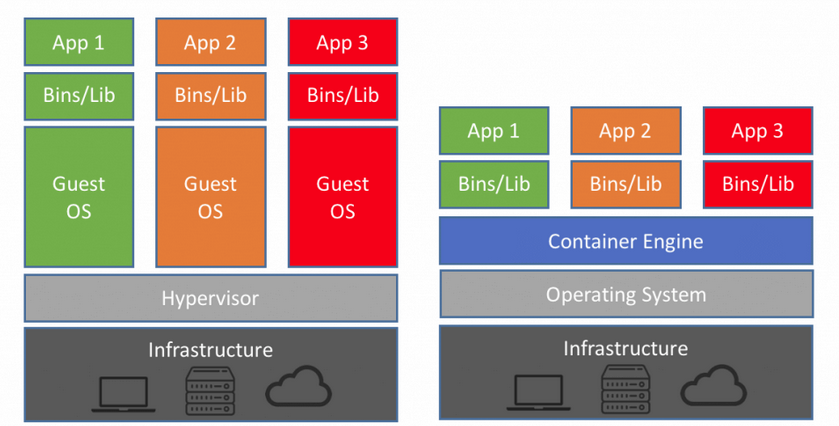
In simple terms, an image is a template, and a container is a copy of that template. You can have multiple containers (copies) of the same image.

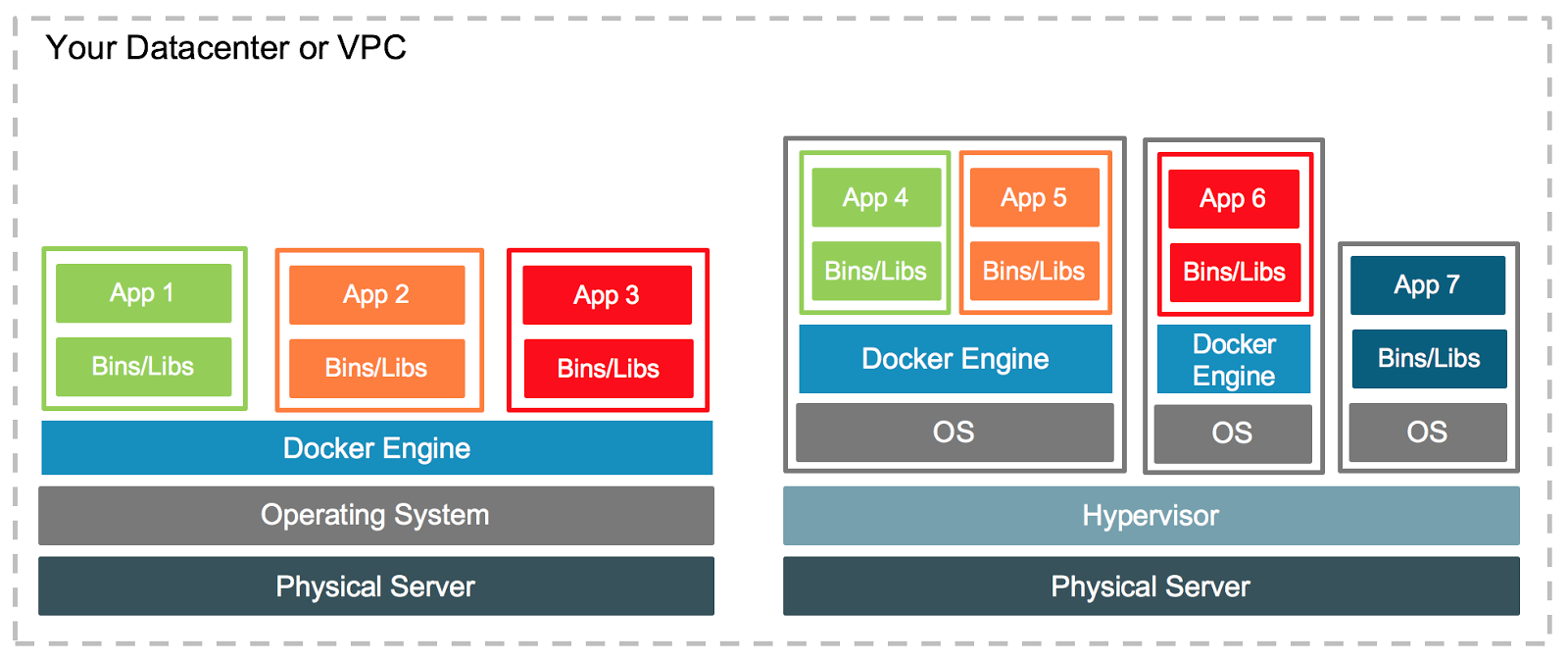
How the docker works?



Developer 🡪 DockerFile 🡪 docker image 🡪 Docker Hub 🡪 pull to docker container

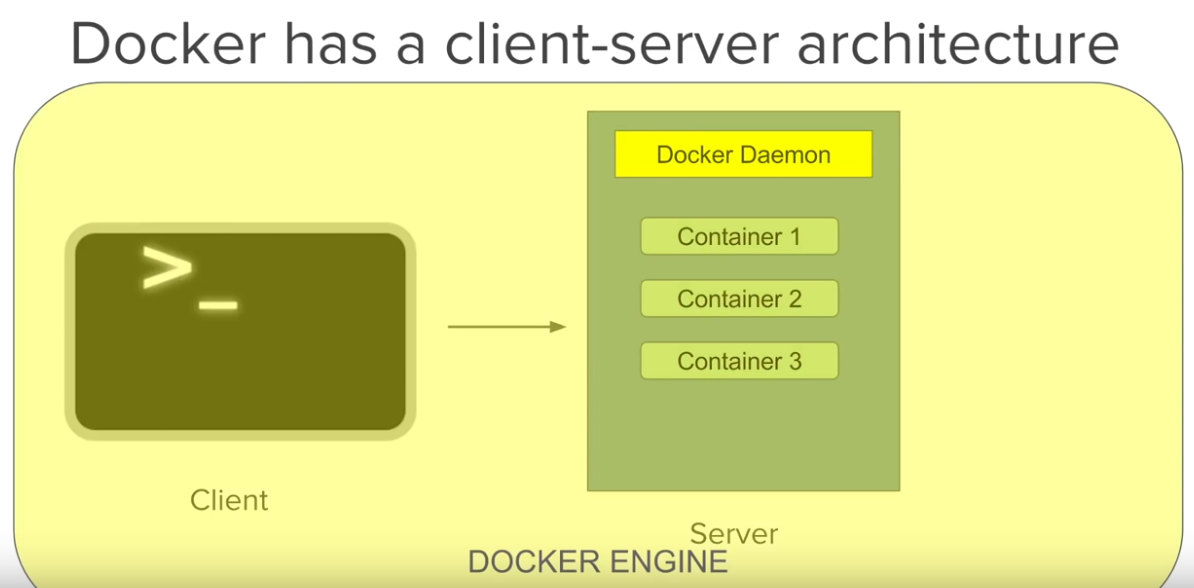
VM vs Container: Docker containers run on top of the host's Operation system. This helps you to improves efficiency. Moreover, we can run more containers on the same infrastructure than we can run Virtual machines because containers use fewer resources.

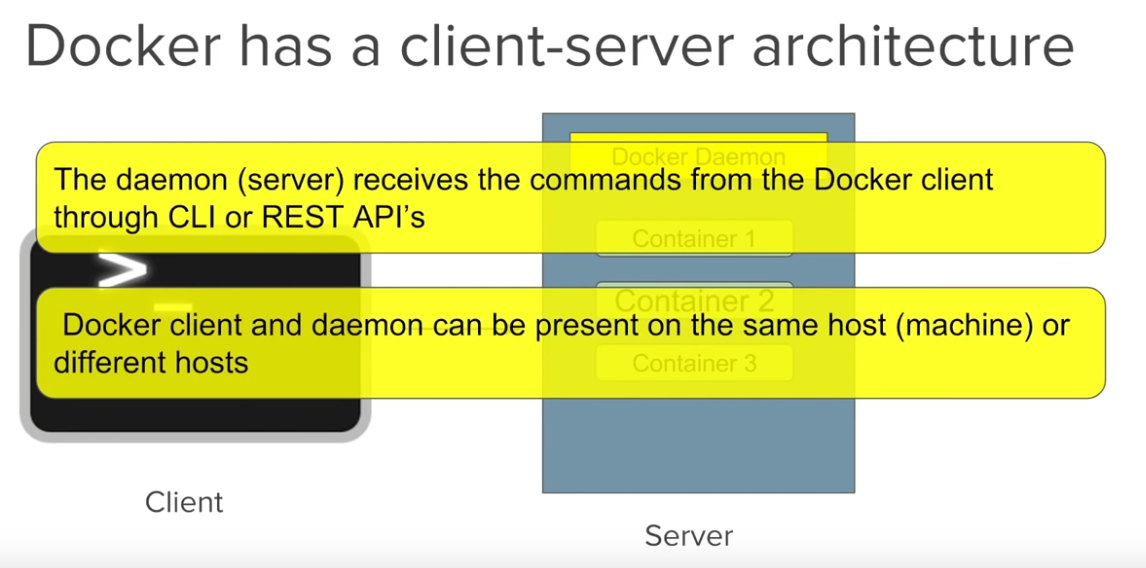




<https://www.guru99.com/docker-interview-questions.html>

Docker Architecture :





**Benefits of Docker :**

1. It resolve code working in one system and not working on another system.(Build app only once)

No need to configure on multiple system. Docker image created pushed in docker central/Hub can be pulled on any docker container/ any ENV.

1. Portability is easy, Docker container can run on any platform.(VB, Amazon ec2).
2. Version Control, Docker has inbuilt version control system as Git.
3. Isolation, With docker every application works in isolation in its own container.
4. Productivity, With docker , developer can package all the software and its dependencies in to docker container and docker will make sure works in every other platform.

Installing the docker.

Linux :

yum install docker

yum remove docker

Windows:

Install the docker tool, with install Hyper –v and VM.

Let's get you started: you will need to download and install Docker to use the Docker command line interface (CLI).

This repository contains everything you need to create your first container.

git clone <https://github.com/docker/doodle.git>

A Docker image is a private filesystem, just for your container. It provides all the files and code your container will need. Running the docker build command creates a Docker image using the Dockerfile. This built image is in your machine's local Docker image registry.

cd doodle\cheers2019 ; docker build -t sachinchippalkatti/cheers2019 .

Running a container launches your software with private resources, securely isolated from the rest of your machine.

docker run -it --rm sachinchippalkatti/cheers2019

Once you're ready to share your container with the world, push the image that describes it to Docker Hub.

docker login ; docker push sachinchippalkatti/cheers2019

Play online : <https://labs.play-with-docker.com/p/bl20tpttcgkg00bft3ig>

<https://automationstepbystep.com/>

Today we will learn : How to install Docker on Linux

Agenda: Prerequisites Connect to Linux Install Docker Start Docker Stop Docker Uninstall Docker Prerequisite OS should be 64 bit Linux kernel ver 3.10 or greater command to check : uname –r

STEP 1 - Connect to Linux system

STEP 2 - Install DOCKER sudo yum -y update sudo yum install -y docker docker docker --version

STEP 3 - start DOCKER

sudo service docker start sudo usermod -a -G docker "user" docker info docker run hello-world : to run hello-world image docker images : to get list of images present locally docker ps : to get list of running containers docker ps -a . : to get list of all containers

STEP 4 - stop DOCKER sudo service docker stop uninstall DOCKER sudo yum remove docker

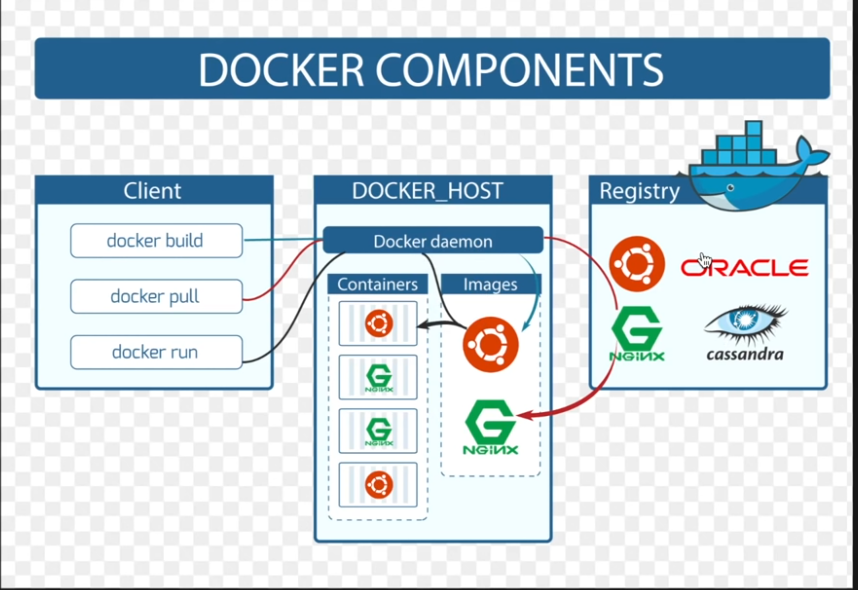
HELPFUL TIPS You can visit - [https://get.docker.com/](https://www.youtube.com/redirect?redir_token=9-dyPEqoCfxgIjybmfOSL83sIId8MTU2NDgyODA1NUAxNTY0NzQxNjU1&q=https%3A%2F%2Fget.docker.com%2F&event=video_description&v=KCckWweNSrM) for more installation related help

To install docker from binaries [https://docs.docker.com/engine/instal...](https://www.youtube.com/redirect?redir_token=9-dyPEqoCfxgIjybmfOSL83sIId8MTU2NDgyODA1NUAxNTY0NzQxNjU1&q=https%3A%2F%2Fdocs.docker.com%2Fengine%2Finstallation%2Fbinaries%2F&event=video_description&v=KCckWweNSrM) Installation steps for amazon ec2 [http://docs.aws.amazon.com/AmazonECS/...](https://www.youtube.com/redirect?redir_token=9-dyPEqoCfxgIjybmfOSL83sIId8MTU2NDgyODA1NUAxNTY0NzQxNjU1&q=http%3A%2F%2Fdocs.aws.amazon.com%2FAmazonECS%2Flatest%2Fdeveloperguide%2Fdocker-basics.html&event=video_description&v=KCckWweNSrM) References: Linux free instance - [https://aws.amazon.com/free/](https://www.youtube.com/redirect?redir_token=9-dyPEqoCfxgIjybmfOSL83sIId8MTU2NDgyODA1NUAxNTY0NzQxNjU1&q=https%3A%2F%2Faws.amazon.com%2Ffree%2F&event=video_description&v=KCckWweNSrM) Docker Manuals - [https://docs.docker.com/manuals/](https://www.youtube.com/redirect?redir_token=9-dyPEqoCfxgIjybmfOSL83sIId8MTU2NDgyODA1NUAxNTY0NzQxNjU1&q=https%3A%2F%2Fdocs.docker.com%2Fmanuals%2F&event=video_description&v=KCckWweNSrM) [https://get.docker.com/](https://www.youtube.com/redirect?redir_token=9-dyPEqoCfxgIjybmfOSL83sIId8MTU2NDgyODA1NUAxNTY0NzQxNjU1&q=https%3A%2F%2Fget.docker.com%2F&event=video_description&v=KCckWweNSrM) [https://docs.docker.com/engine/instal...](https://www.youtube.com/redirect?redir_token=9-dyPEqoCfxgIjybmfOSL83sIId8MTU2NDgyODA1NUAxNTY0NzQxNjU1&q=https%3A%2F%2Fdocs.docker.com%2Fengine%2Finstallation%2Fbinaries%2F&event=video_description&v=KCckWweNSrM) [http://docs.aws.amazon.com/AmazonECS/...](https://www.youtube.com/redirect?redir_token=9-dyPEqoCfxgIjybmfOSL83sIId8MTU2NDgyODA1NUAxNTY0NzQxNjU1&q=http%3A%2F%2Fdocs.aws.amazon.com%2FAmazonECS%2Flatest%2Fdeveloperguide%2Fdocker-basics.html&event=video_description&v=KCckWweNSrM)

**How to create Images?**

Images are the template used to create the docker container.

Container is running instance of images.



Docker files: Is the simple text file with the instruction to build images. Automation of docker image creation.(Dockerfile).

Step 1: create the docker file with Dockerfile.

Step 2: Add the instruction in DockerFile.

Using FROM, RUN, CMD

Step 3: Build dockerfile to create image .

docker build .

docker build –t myimage:1.0 .

Step 4 : Run the image to create container.



Dockerfile

FROM

RUN

CMD [“echo”,”<comment>”]

docker build .

docker build –t myimage:1.0 .

docker run myimage:1.0

<https://github.com/wsargent/docker-cheat-sheet#dockerfile>

<https://docs.docker.com/engine/reference/builder/#environment-replacement>

**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 the docker-compose

docker-compose –-version

docker-compose v

docker-compose version

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

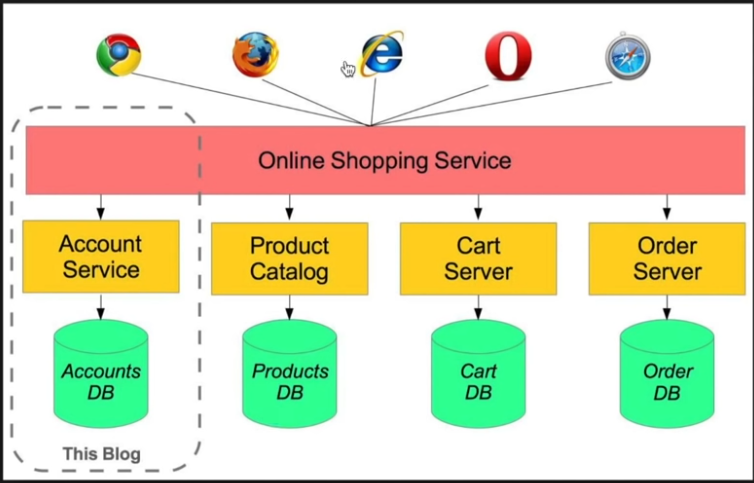
Steps 5 : Bring down application by command docker-compose down

TIPS

How to scale services

—scale

docker-compose up -d --scale database=4



Docker Volumn:

Use of Volumes ===========

Decoupling container from storage

Share volume (storage/data) among different containers

Attach volume to container

On deleting container volume does not delete

Volumes are the preferred mechanism for persisting data generated by and used by Docker containers

: docker volume //get information

: docker volume create

: docker volume ls

: docker volume inspect

: docker volume rm

: docker volume prune

Docker Swarm: (Manager and worker)

A swarm is a group of machines that are running Docker and joined into a cluster

Docker Swarm is a tool for Container Orchestration

Let’s take an example

You have 100 containers

You need to do –

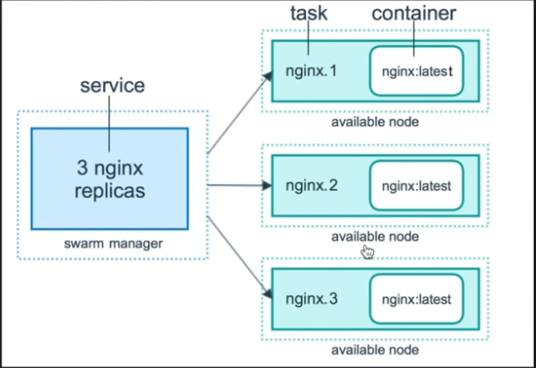
Health check on every container

- Ensure all containers are up on every system

- Scaling the containers up or down depending on the load

- Adding updates/changes to all the containers Orchestration

- managing and controlling multiple docker containers as a single service Tools available - Docker Swarm, Kubernetes, Apache Mesos



Step 1 : Create Docker machines (to act as nodes for Docker Swarm)

Create one machine as manager and others as workers

docker-machine create --driver hyperv manager1

docker-machine create --driver virtualbox manager1

docker-machine:Error with pre-create check: “exit status 126”

[https://stackoverflow.com/questions/3...](https://www.youtube.com/redirect?q=https%3A%2F%2Fstackoverflow.com%2Fquestions%2F38696164%2Fdocker-machineerror-with-pre-create-check-exit-status-126&v=bU2NNFJ-UXA&event=video_description&redir_token=hsWYyxG5rNC8wrqZSkSMtKNOBfp8MTU2NTA4NjMyMUAxNTY0OTk5OTIx) brew cask install virtualbox;

Create one manager machine and other worker machines

Step 2 : Check machine created successfully

docker-machine ls

docker-machine ip manager1

Step 3 : SSH (connect) to docker machine

docker-machine ssh manager1

Step 4 : Initialize Docker Swarm

docker swarm init --advertise-addr MANAGER\_IP

docker node ls

(this command will work only in swarm manager and not in worker)

Step 5 : Join workers in the swarm

Get command for joining as worker In manager node run command

docker swarm join-token worker

This will give command to join swarm as worker

docker swarm join-token manager

This will give command to join swarm as manager SSH into worker node (machine) and run command to join swarm as worker In Manager Run command –

docker node ls

to verify worker is registered and is ready Do this for all worker machines

Step 6 : On manager run standard docker commands

docker info

check the swarm section no of manager, nodes etc

Now check docker swarm command options docker swarm

Step 7 : Run containers on Docker Swarm

docker service create --replicas 3 -p 80:80 --name serviceName nginx

Check the status:

docker service ls

docker service ps serviceName

Check the service running on all nodes Check on the browser by giving ip for all nodes

Step 8 : Scale service up and down On manager node

docker service scale serviceName=2

Inspecting Nodes (this command can run only on manager node)

docker node inspect nodename

docker node inspect self

docker node inspect worker1

Step 9 : Shutdown node

docker node update --availability drain worker1

Step 10 : Update service

docker service update --image imagename:version web

docker service update --image nginx:1.14.0 serviceName

Step 11 : Remove service

docker service rm serviceName

docker swarm leave : to leave the swarm

docker-machine stop machineName : to stop the machine

docker-machine rm machineName : to remove the machine

**4. What are the main drawbacks of Docker?**

Some notable drawbacks of Docker are:

* Doesn't provide a storage option
* Offer a poor monitoring option.
* No automatic rescheduling of inactive Nodes
* Complicated automatic horizontal scaling set up

**13. Explain Docker Swarm?**

Docker Swarm is native gathering for docker which helps you to a group of Docker hosts into a single and virtual docker host. It offers the standard docker application program interface.

**14. How can you monitor the docker in production environments?**

Docker states and Docker Events are used to monitoring docker in the production environment.

**18. What is Hypervisor?**

The hypervisor allows you to create a virtual environment in which the guest virtual machines operate. It controls the guest systems and checks if the resources are allocated to the guests as necessary.

**23. How does communication happen between Docker client and Docker Daemon?**

You can communicate between Docker client and Docker Daemon with the combination of Rest API, socket.IO, and TCP.

**24. Explain Implementation method of Continuous Integration(CI) and Continues Development (CD) in Docker?**

You need to do the following things:

* Runs Jenkins on docker
* You can run integration tests in Jenkins using docker-compose

**29. Explain the process of scaling your Docker containers**

The Docker containers can be scaled to any level starting from a few hundred to even thousands or millions of containers. The only condition for this is that the containers need the memory and the OS at all times, and there should not be a constraint when the Docker is getting scaled.

Below are the steps for Docker life cycle:

* Build
* Pull
* Run

**32. How can you run multiple containers using a single service?**

By using docker-compose, you can run multiple containers using a single service. All docker-compose files uses yaml language.