day-9 [Container images] (1)

image —Run As a→ container

* Container -

Light weight Container ← has dependencies ←

Run on a host OS and get Network resource & all.

[Hard disk
Network resources
everything is taken from host machine]

> docker images
> docker run -d -p (New container) ngix (new image) → Run image as a container.
(-d = detached)

> docker PS (List of Running con.)
> docker PS -a (Stopped & Runn. containers)

> Started in two ways

(Ready made)	(Add something)
1. Detached mode	2. Interactive mode.
[Ready made - shirt]	[Buy material, lots of customization involve]
→ image	→ CentOS Image.
→ with war file	→ install Req. Software
→ serves's end-users.	→ Launch as contain
→ Production ready	add java
– Run immediately.	→ add dependencies.

[detached container]

> docker -d (detached mode)
 ↳ Run → Get - Response.

Detached mode → (production Ready) (hosting)
Interactive mode (development purpose)

> docker run -d httpd (Running)
docker run -d centos (Not running)
 It can't serve anything
 - as far some process Running, container is running (httpd, tomcat)
 - other centos (dead)

> docker run -it centos
root@xyz # cat etc/os-release

> yum update. (Add - whater you want)

> ~~Exit~~ [Control + PQ (Exit out from container) other wise contair may die)]

> docker ps (terminal process running inside)
(list of cont.)
> ~~doc~~ docker stop containerId (unique Id for comm. with container)
> ~~dock~~ docker Rm containerId

> docker rm -f id (Forcefully Removed containers)

> docker ~~ps (stop~~ run -d ngix (webServer software)
(Car started) started ↑ Container only

> docker PS

Conta-id UniqueId	image (which imaage used)	command) "ngix -g 'daemon..'	Created	Status

RAN ← image creation

CMD - ContainerRun (which proc. Start.) ran command auto matically

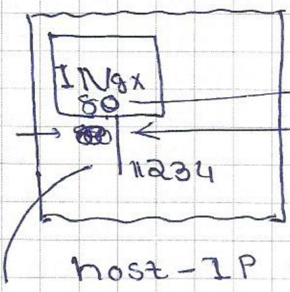
← Process → Run, (Run the command when some body launch) (default command Executed)

→ design (when somebody start cont. then start or execute this command).

Ports ~~nginx -g~~
80 | tcp

Names
diff_names (dynamic name)
funny-kare
feeling ↑ name of Pers.

(



inside container.
Berongs to container

host-IP (

[publishing Ports]

- put portno. outside)

{map container port with host port} (publishing, port forwarding, port mapping)

→ host IP: portno. (on host)
↳ [directed to contaire port.]

- docker run -d [-P] nginx --name ravi
kapital
Port
[0:0.0.00 : 32774] → 80 | tcp
host port ↗ From machine
↑ container)

- routes 80 portno.

→ -P (publish portnumber]
↑ random port number (32774)
32768 - XYZ ← Random
start

- docker run --name ravi1 -d -p 1234:80 nginx
should be unique ↑ small p ↑ ↑ con. Port

→ accessible from host's port (host: Port)

> docker run ... -p 1234:80 ngix
port failed because of host reserved port)

> curl Localhost: 80 (~~docker~~ host) connect refu
curl " : 1234 (will return html code)
65 - 1 65535 (default range)

> -P starts from 32764 - 65535
-P 32764-65535 (

> interactive mode container

> docker exec (Enter into a running container)

> docker exec containerId touch /tmp/ravi
ls /tmp.

> ravi fire (ls tmp directory) [Enter in container)

> docker exec -it 60x4z (interactive mode, execute command
→ bash (any command you can give)

root@x4z :# cat /etc/os-release
Name: ...
" : vi r4 (not found) (not required for nginx)
: apt-get update
> apt-get install vi
> cd ../usr/share/nginx/html
vi index.html (Ctrl Pq)
> read escape sequence. (?)

> docker ps

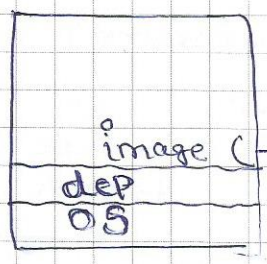
> docker logs 6078x4z

> docker logs -f 6078q2 (-f = follow the live logs)

> new image creation (new version, new update)

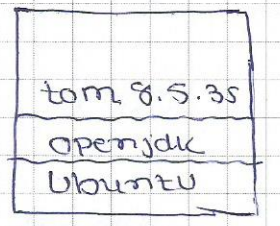
Bash (Sh works for every container)

* Image Creation



(Software as a image)

Build our own image



(Our own Spies)

→ Container OS (Ubuntu)
- pull Ubuntu OS
- install all your software.

↑ if container dies then all changes dead.

→ (image is immutable)

1. UBuntu (Base OS, Base layer)
2. run it as Container, interactive mode.
3. Make change
4. Save as a new image

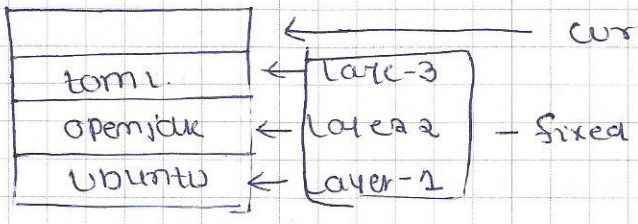
↳ docker run -it Centos
touch /tmp/ravi
touch /tmp/abcd
} ← changes made to container not

docker commit -m 'Centos ravi' 8n85 (container Id) newimage (name)
Image Id

> docker images.
newimage Latest

> images → immutable by nature (run in container
- make changes
- create new images)

* Docker Image Automations.



← writtable Layer, it's possible to create New image.

- Dockerfile (Script or set of line to create an image automatically)

From nginx (Base Image)
[docker directive (capital letter)] COPY index.html /var/www/html (Run inside container)
EXPOSE 80
→ CMD /usr/sbin/nginx -g 'daemon off;' (
meta data
From Ubuntu:16 ← interactive mode
(!) Run apt-get update (Run inside container (creating container
Expose 80 (inside Port numer i container)
← ngix port.
(Mendatory ← Future'
CMD /usr/sbin/nginx -g 'dam." ← what you want to run when

> docker built -t ravinginx -f dockerfile . (?)
(name) of image
↑ context base folder.
→ Create an image.

> docker image
ravinginx ...

> more dockerfile_tomcat1.
file_name
> docker built -t ravitomcat -f dockerfile_tom . base Folder Path.