- d -detach from terminal & run

Date
Page

# systmctl status docker.
# systemctl restart docker.
# systemctl enable docker ← enable the service in booting sequence.

Docker Commands.

1. run: Start a Container.

# docker run image
OR
# docker run inge: latest
# docker run image:1.1.0

# docker run -d image :1.1.0
↗ run's in background.

-d - detach.

Search image in local first if not present it downloads from dockerhub.

2. ps: list of running containers.

# ~~dockerps~~ docker ps
# docker ps-a [previously installed]
running → # docker ps-q [Quite: lis the ID of container.]
all the containers # docker ps-aq.( container id).

-It → interactive terminal.

Date ____
Page ____

3) **Stop** : Stop a container.

```
# docker stop Name OR container_ID.
# docker start container_ID.
```

stop multiple running containers in onego.

```
# docker stop $ (docker ps -pq).
```

4) **rm** : Remove a container.

```
# docker rm name/container ID
# docker rm -f name/container ID.
```

eg

```
# docker rm $(docker ps -aq).
# docker rm -f $(docker ps -aq).
```

5) **images** : list images.

```
# docker images.
# docker images -q.
```

6) **rmi** : Remove image [-f forcefully].

```
# docker rmi <image_id> .
# docker rmi -f <image_id> .
```

eg:

```
docker rmi $(docker images -q)
docker rmi -f $(docker images -q).
```

/usr/local/apache2/htdocs

Date
Page

7) pull : Only download image
# docker pull imagename.

8) exec - execute command.
# docker exec <name> cat /etc/passwd
# docker exec -it <name/id> bash.
spawn a shell bash and give interactive console.

9.) Rename the container name.
docker run -d --name "Web1" httpd

10) Display the docker host information with.
# docker info

Jason format.
Array [ ].
Obj { }.
Key: Value..
NULL ← if no key & values present.

```
[{
   "cdac" : {"name" : "       ",
             "class" : "       "
            }.
 }.
```

[{"cdac": {"name":"       ", "class":   }}]
Scanned by CamScanner

apt install jq

# cat data.json | jq ← Beautify.
# cat data.json | jq. ← Dot represents first Object and next should be key.

# cat data.json/jq.name ← Key name.

# cat data.json / jq.cdac.narm

1st Obj

key key.

# cat data.json/jq.[] ← first Array.
# cat dat.json / jq.[].cdac.love

entire code array

https://mmumshad.github.io/json-path-quiz/

docker inspect -f "{{json.Id}}" container_id.

<u>Port Mapping</u>:

# docker run -d --name "web1" -p 8080:80 httpd

publish.

host port. container port image name

-m → module.
-t → tag

Date
Page

**Volume Mapping:**

container httpd : /usr/local/apache2/htdocs

Host os : /web1

```
# mkdir /web1
# cat > /web1/index.html
ULALA
```

```
# docker run -d --name "server1" -p 80:80 -v
/web1:/usr/local/apache2/htdocs httpd:latest
```

—— × —— × —— × —— × ——

- Create Acc on docker.
- Create repository
- Docker Push:

```
# docker login
username:
password:
```

```
FROM ubuntu
RUN apt update
RUN apt install python3 -y
COPY ./prog.py /mnt/prog.py
ENTRYPOINT python3 /mnt/prog.py
```

docker build . -t repo/image name
babadai/8pss.

docker push . repo/~~dot~~ image name.

- input from user and place.

Date ______
Page ______

```
COPY ./index.html /usr/local/apache2/htdocs.
EXPOSE 80
```

2) Expoot Docker Images:

```
# docker ps
# docker export webappl >webapp.tar.
```

You could commit this containee as a new image locally, but you could also use the Docker import.

Command:

```
# docker import - mywebapp <webapp.tar.
```

View: tar -tvf demo.tar

Extract: tar -xvf demo.tar -C demo.

extract view forcefully.

extract in

Docker composte.

YAML:

1. [Key - Value] pair:
Key : value

eg:
fruit : Apple
Veg : Carrot.

Esc. :q - exit
:wq - save.

2. Array / list :.
Fruit :
    - Orange
    - Apple.

3. Dict .
It a set properties that group together.

Banana :
    Calories : 102
Grapes :
    Calories : 99 .

[4] key value / dict / list :

Fruits :
- Banan :
    Calories : 102 .
- Grapes :
    Calories : 99 .

Install
```
apt install docker-compose -y
```
Make folder : & write
Create a file named as docker-compose.yml
```
 services :
    web1 :
       image : "httpd : latest"
       ports :
          - "80 : 80"
```

control &

Date
Page

```
    volumes:
      - /web1 : /usr/local/apache2/htdocs
  web2:
    image: "nginx : latest"
    ports:
      - "801:80"
```

# docker-compose config ← verify syntax & yaml version.
# docker-compose up ← forground
# docker-compose up -d. ← Background
#. docker-compose ps. ← ~~composed~~ To see list of docker compose.
# docker-compose down ← container stop & remove.

*

```
version : '3.9'
services:
  kids :
    image : "python:3"
    ports :
      - "80:80"
    command : python3 -m http.server 80
    working_dir : /mnt
    volumes :
      - ./lol : /mnt
```

Scanned by CamScanner

ifconfig docker0
systemctl restart docker.

vim Dockerfile.

```
FROM ~~UB~~ ubuntu
RUN apt update
~~ap~~RUN apt install -y . . . .
RUN pip3 install flask
RUN pip3 install flask-mysql.
COPY ./app.py /
ENTRYPOINT ~~/app.py/~~ FLASK_APP=/app.py flask
                                    run -- host 0.0.0.0
(cat > app.py
import os.
:
 host 0.0.0.0.)
```

```
                     babadai
docker build . -t ~~pgdai~~/cdac
            present
```

deploy

build: .

```
docker-compose-yml
services:
   cdacdai:
      image: "babadai/cdac"
      ports:
        - "5000:8080"
```

```
docker-compose up -d.
  "        "     ps
```

## Link Container:

By linking containers, you provide a secure channel via which Docker containers can communicate to each other.

first : NoSQL Data Structure Server Redis.

```
# docker pull redis.
# docker run -d --name db redis.
# docker ps.
```

second : Now let us run a another container, named as "ubuntu" container.

```
# docker run -d --link db:redis --name myserver
  unbuntu:latest .
# cat /etc/hosts.
# ping redis.
```

- link flag is sourcecontainername:containeraliasname.