

# ??????

## ????

# List running containers

```
docker ps
```

# ssh into the container

```
docker exec -it <container-name> /bin/sh
```

# Restart a container

```
docker restart <container-name>
```

# Show running container stats

```
docker stats
```

# Check docker daemon disk space usage

```
docker system df
```

# Purge those unused images, networks, containers and volumes

```
docker system prune
```

# Check the container log

```
docker logs <container-name>
```

# Search docker registry for image

```
docker search <image-name>
```

# Create and start a container

```
docker run -it <image-name> /bin/bash
```

# Check container's exposed ports

```
docker port {container-name}
```

## ????

- [Docker Cheat-Sheet](#)

```
man docker <command>
man docker build
man docker rmi
```

## ?? Images

```
## [] Docker Hub [] image name
docker search lamp

## [] image name
docker images

## [] image []
docker inspect <image-name>

## [] image
docker pull ubuntu:13.10

## [] image
docker rmi <image-name>

## [] images
docker rmi $(docker images -q)

## [] images[] my-images []
docker rmi $(docker images | grep -v 'ubuntu\|my-image' | awk {'print $3'})

## [] <none> [] images
docker rmi $(docker images -f "dangling=true" -q)

## [] myapp/myimage [] <none> [] images
docker rmi $(docker images myapp/myimage -f "dangling=true" -q)

## [] images []
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock nate/dockviz images -t
```

## ?? Containers

```
## [ ] container [ ] console
docker run -it <image-name> /bin/bash
docker run -it --name <container-name> <image-name> /bin/bash
```

```
## [ ] daemon [ ] container
docker run -d -p 11180:80 <image-name>
docker run -d --name web <image-name>
TIP: [ ] container [ ]
```

```
docker run -d -p 80:80 --rm <image-name>
[ ] --rm [ ] container [ ]( [ ] docker rm [ ] )
(docker start) [ ] docker run [ ]
```

```
## [ ] containers
docker ps
docker ps -a
```

```
## [ ] container [ ] Volumes [ ] IP [ ] Hostname [ ]
docker inspect <container-id>
```

```
## [ ] container
docker rm <container-id>
```

```
## [ ] containers
NOTE: [ ] container
docker ps -a -q | xargs -n 1 docker rm
docker rm $(docker ps -aq)
```

```
## [ ] container
docker ps -a | grep "Exited" | awk '{print $1}' | xargs docker rm
docker rm $(docker ps --all -q -f status=exited)
NOTE: [ ] container [ ] rebuild image [ ]
```

```
## [ ] container
docker stop <container-id>
```

```
## Stop all containers
docker stop $(docker ps -aq)
docker rm $(docker ps -aq)
```

```
docker ps -aq | xargs docker stop
```

```
## [ ] container
```

```
docker export <container-id> > ubuntu-mysql.tar
```

```
## [ ] container
```

```
cat ubuntu-mysql.tar | docker import - <image-name>
```

```
## [ ] container
```

```
[ ] Ctrl P [ ] Ctrl Q
```

```
NOTE: [ ] Ctrl+P [ ] Bash [ ]
```

```
## [ ] container
```

```
docker attach <container-id>
```

```
[ ]
```

```
docker attach <container-name>
```

```
[ ] container [ ] daemon [ ]
```

```
docker exec -it <container-id/name> /bin/bash
```

```
## [ ] container [ ]
```

```
docker commit <container-id> <image-name>
```

```
## [ ] container [ ] IP
```

```
docker inspect <container-id> | grep IPAddress | cut -d '"' -f 4
```

## Check Container CPU and RAM Usage

```
docker stats
```

```
docker stats --no-stream
```

```
docker stats --no-stream -a
```

```
docker stats <container-name>
```

```
docker stats --format "table {{.Container}}\t{{.CPUPerc}}\t{{.MemPerc}}"
```

```
docker ps --no-trunc --format "{{.Names}}\t{{.ID}}"
```

## ??? container ?????

```
## Stop the container
```

```
docker stop <container-name>
```

```
docker save mycontainerimage | gzip > mycontainerimage.tar.gz
```

```
gunzip -c mycontainerimage.tar.gz | docker load
```

```
docker save mycontainerimage | gzip | ssh root@203.0.113.1 'gunzip | docker load'
```

?? exit ?????? container?????? Linux

```
-p ? Host ? port 11180 ??? container ? port 80
```

- ? container ????volume ?????????? base image ????? volume  
????????????????????????????
- volume ????????????????
- ? image ???? (commit)?volume ??????????????
- ?? container ????volume ??????????????

```
docker run -t -i -p 80:80 -v ${PWD}/webapp:/webapp alang/centos5-lamp_php51
```

??-v <host-dir>:<container-dir>

? container ?????????? /webapp????????????????????

```
????????? host ?????????? container ??????????????????????:
```

```
docker inspect -f {{.Volumes}} <container-id>
```

```
??????
```

```
/var/lib/docker/vfs/dir/bfebd8cb6.....
```

## Docker Network

```
# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
7ccaf6119fa8	nginx:latest	"nginx -g 'daemon of..."	2 days ago	Up 39 hours	0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp	nginx_mysql_web_1
81a920bb51a6	nginx_mysql_php	"docker-php-entrypoi..."	2 days ago	Up 2 days	9000/tcp	nginx_mysql_php_1
437a7501198f	mariadb:10.3	"docker-entrypoint.s..."	2 days ago	Up 2 days	3306/tcp	nginx_mysql_db_1

```
# docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
852eff02220e	bridge	bridge	local
334d2b8571a4	host	host	local
b97cae66a977	nginx_mysql_default	bridge	local
40d15afb34b4	none	null	local

```
# docker network inspect -f '{{.IPAM.Config}}' bridge | jq -r .[].Subnet
```

```
# docker network inspect -f '{{.IPAM.Config}}' bridge | jq -r .[].Gateway
```

```
# brctl show
```

bridge name	bridge id	STP enabled	interfaces
br-b97cae66a977	8000.0242569e79ff	no	veth3ce8cbd veth5129652 veth55dcdf7
docker0	8000.0242faff70bb	no	

?? container IP

```
## Method #1: By inspecting the container
docker inspect <container_id> | grep -i ipaddr
docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' <container_id>

# get an IP address associated with a specific network
# docker container inspect -f '{{.NetworkSettings.Networks.<NETWORK NAME>.IPAddress }}'
<CONTAINER_ID_OR_NAME>
docker container inspect -f '{{.NetworkSettings.Networks.bridge.IPAddress }}' ubuntu-ip

## Method #2: Using the container's shell
docker exec -it <container-name> sh
> ip
or
> ifconfig

# if you get the errors with 'command not found', following the below steps to install the relevant packages.
> apt update -qq
> apt install iproute2 -yqq

## Method #3: By inspecting the network itself
# docker network inspect <NETWORK NAME>
docker network inspect bridge | jq .[].Containers
docker network inspect bridge | jq '.[].Containers."<CONTAINER ID>".IPv4Address'

docker network inspect -f '{{json .Containers}}' bridge | \
jq '..|if type == "object" and has("Name") then select(.Name=="<CONTAINER NAME>") | .IPv4Address else
empty end' -r
```

## ?? Docker

??????

```
# ?? Docker ??
docker version

# Docker ???
docker info
```

host ? container ?????

```
docker cp <container-name>:/etc/nginx/nginx.conf /data/web/conf
docker cp host_source_path my_container:destination_path
docker cp -a host_source_path my_container:destination_path
```

?????????

```
# [image] container[docker] <none> [image]
docker images --quiet --filter "dangling=true"
docker system prune

# [volume] volume [volumes] --volumes
docker system prune -a --volumes

# For volumes only
docker volume ls -f dangling=true
docker volume prune
```

## Restart Policy

- [Beginner's Guide to Docker Restart Policy](#)

???? container

```
# Add --restart=unless-stopped
docker run -d -p 4449:4449 --name myst --restart=unless-stopped
```

## Docker Logging

- [Complete Beginner's Guide to Docker Logging](#)

```
docker logs {container-name}
docker logs --tail 50 {container-name}
docker logs -f {container-name}
docker logs -f --tail 20 {container-name}

# View timestamp in Docker logs
docker logs -t {container-name}
docker -n=10 -t {container-name}

# Viewing Docker logs in a specified time period
```



```
docker logs --since 1440m -t {container-name}
docker logs --until 1440m -t {container-name}
docker logs --since 2021-07-28 -t {container-name}
```

## Docker system service logs

```
sudo journalctl -u docker
```

## Where are Docker logs stored

```
sudo ls -lh /var/lib/docker/containers
```

## Enabling Log Rotation for Docker (JSON)

Edit `/etc/docker/daemon.json`

```
{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "10m",
    "max-file": "3"
  }
}
```

## Restart Docker daemon

```
sudo systemctl restart docker
```

## Disk Space Usage

```
avimanyu@iborg-desktop:~$ docker system df
```

TYPE	TOTAL	ACTIVE	SIZE	RECLAIMABLE
Images	4	4	1.065GB	0B (0%)
Containers	4	4	5.705kB	0B (0%)
Local Volumes	7	7	1.108GB	0B (0%)
Build Cache	0	0	0B	0B

```
avimanyu@iborg-desktop:~$ docker system df -v
```

Images space usage:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE	SHARED SIZE	UNIQUE SIZE
CONTAINERS						
ghost	4.32.0	b40265427368	8 weeks ago	468.8MB	0B	468.8MB 1
jrcs/letsencrypt-nginx-proxy-companion	latest	037cc4751b5a	13 months ago	24.35MB	0B	24.35MB 1
jwilder/nginx-proxy	latest	509ff2fb81dd	15 months ago	165MB	0B	165MB 1
mariadb	10.5.3	f5d2bcaf057b	20 months ago	407MB	0B	407MB 1

Containers space usage:

CONTAINER ID	IMAGE	COMMAND	LOCAL VOLUMES	SIZE	CREATED
STATUS	NAMES				
899cc90e85d9	ghost:4.32.0	"docker-entrypoint.s..."	1	0B	8 weeks ago Up 8 weeks
ghost_ghost_6					
17b58fdafbce	jrcs/letsencrypt-nginx-proxy-companion	"/bin/bash /app/entr..."	4	571B	3 months ago Up 2 months
letsencrypt-proxy-companion					
58f99f46ee03	jwilder/nginx-proxy	"/app/docker-entryp..."	5	5.13kB	3 months ago Up 2 months
jwilder-nginx-proxy					
fb907286b60e	mariadb:10.5.3	"docker-entrypoint.s..."	1	2B	3 months ago Up 2 months
ghost_db_1					

Local Volumes space usage:

VOLUME NAME	LINKS	SIZE
ghostdb	1	434.7MB
jwilder-nginx-with-ssl_acme	2	36.09kB
jwilder-nginx-with-ssl_certs	2	25.12kB
jwilder-nginx-with-ssl_dhparam	1	1.525kB
jwilder-nginx-with-ssl_html	2	1.106kB
jwilder-nginx-with-ssl_vhost	2	556B
ghost	1	674MB

Build cache usage: 0B

CACHE ID	CACHE TYPE	SIZE	CREATED	LAST USED	USAGE	SHARED
----------	------------	------	---------	-----------	-------	--------

avimanyu@iborg-desktop:~\$ docker image ls

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
busybox	latest	beae173ccac6	6 weeks ago	1.24MB
ubuntu	latest	fb52e22af1b0	5 months ago	72.8MB

```
alpine      latest  49f356fa4513  10 months ago  5.61MB
hello-world latest  d1165f221234  11 months ago  13.3kB
```

```
avimanyu@iborg-desktop:~$ docker ps --size
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES	SIZE
1171dcfb7e06	alpine	"sleep 10"	10 months ago	Up 9 seconds			

```
# Overlay2 is the default Docker storage driver on Ubuntu.
```

```
# You can confirm this by running the 'docker info' command and looking for the Storage Drive
```

```
# To get the <<hash-named-directory> by the command 'docker inspect <image-name>'
```

```
sudo du -sh /var/lib/docker/overlay2/<hash-named-directory>/
```

```
# Specific Volume Disk Usage
```

```
$ docker volume ls
```

DRIVER	VOLUME NAME
--------	-------------

local	d502589845f7ae7775474bc01d8295d9492a6c26db2ee2c941c27f3cac4449d1
-------	--

local	e71ee3960cfef0a133d323d146a1382f3e25856480a727c037b5c81b5022cb1b
-------	--

local	test-data
-------	-----------

```
$ sudo du -sh /var/lib/docker/volumes/test-data/_data
```

```
4.0K /var/lib/docker/volumes/test-data/_data
```

## FAQ

### ???? image

“ rror response from daemon: conflict: unable to delete dd78a816fb76 (must be forced) - image is referenced in multiple repositories

Solution: ????? image id ????? image ???????? image id ?????????????????????? image ??????

```
root@greencloud-us-1TB:~/watchtower# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
mysteriumnetwork/myst	latest	5c613786d102	39 hours ago	53.3MB
presearch/node	latest	27216957eb08	10 days ago	69.8MB
storjlabs/storagenode	latest	0ac3b4808897	3 weeks ago	124MB
lscr.io/linuxserver/transmission	latest	8cad68f9dac4	7 months ago	95.7MB

```
containrrr/watchtower          latest  333de6ea525a  8 months ago  16.9MB
jellyfin/jellyfin              latest  0aa773b67433  13 months ago  717MB
presearch/auto-updater         latest  dd78a816fb76  17 months ago  16.4MB  <===
containrrr/watchtower          <none>  dd78a816fb76  17 months ago  16.4MB  <===
storjlabs/watchtower           latest  6af6621e20c1  2 years ago   14.3MB
nate/dockviz                    latest  93b5259c1e18  4 years ago   6.61MB
```

```
root@greencloud-us-1TB:~/watchtower# docker rmi dd78a816fb76
```

Error response from daemon: conflict: unable to delete dd78a816fb76 (must be forced) - image is referenced in multiple repositories

```
root@greencloud-us-1TB:~/watchtower# docker rmi presearch/auto-updater containrrr/watchtower
```

```
Untagged: presearch/auto-updater:latest
```

```
Untagged: presearch/auto-
```

```
updater@sha256:3283e0b5be326d77ff4f4e8b7a91d46aaa1d511c74877b5a32f161548812d00c
```

```
Untagged: containrrr/watchtower:latest
```

```
Untagged:
```

```
containrrr/watchtower@sha256:bbf9794a691b59ed2ed3089fec53844f14ada249ee5e372ff0e595b73f4e9ab3
```

```
Deleted: sha256:333de6ea525af9137e1f14a5c1bfaa2e730adca97ab97f74d738dfa99967f14f
```

```
Deleted: sha256:f493af3d0a518d307b430e267571c926557c85222217a8707c52d1cf30e3577e
```

```
Deleted: sha256:62651dc7e144aa8c238c2c2997fc499cd813468fdbc491b478332476f99af159
```

```
Deleted: sha256:83fe5af458237288fe7143a57f8485b78691032c8c8c30647f8a12b093d29343
```

## ???? localhost ??

?? container ?????????????? host ???????? `http://localhost:XXX` ????????

`http://host.docker.internal:XXX` ???

Revision #32

Created 19 June 2020 11:46:01 by Admin

Updated 14 January 2025 17:30:05 by Admin