

Docker

Docker ???

Docker?? Docker??

??

- [Secure Docker Network](#)
- [MySQL Backup](#)
- [??????](#)
- [Docker Compose ??](#)
- [Installation](#)
- [Dockerfile](#)
- [Learning](#)
- [SWAG - Secure Web Application Gateway](#)
- [Transfer/Move Docker](#)
- [Docker Management](#)
- [Watchtower](#)
- [????](#)
- [Linux Distros for Running Container](#)
- [Dockerize Custom Application](#)

Secure Docker Network

With SWAG

[SWAG - Secure Web Application Gateway](#)

With iptables

- [? Linux ??? Docker Bridge ???????????? \(6\) - iT ???](#)

?? Container ???????? iptables ????????????

- docker-compose: ??????????(br-*) ? DOCKER-USER ????
- docker: ?? PRE_DOCKER ????

for docker-compose)

```
#!/usr/bin/env bash

# Purpose: Restrict the access to the container from external IPs.
# Author: Alang, 2019/8/14
#

CONTAINER="nginx_mysql_web_1"
EXT_IF="eth0"
BRG_IF="$(ip a | awk '/br-[a-z0-9]+:/{print $0}' | awk '{print $2}' | sed 's://g')"
```

```
if [ ! -x /usr/bin/docker ]; then
    echo "Abort: Unable to run the docker command!"
    echo "Recommendation: Check if the Docker was installed."
    exit 1
fi

if [ -z "$BRG_IF" ]; then
    echo "Abort: The network interface (br-????) with container not found. "
    exit 1
fi
```

```

# Check the network interfaces
for i in $EXT_IF $BRG_IF;do
    if ! (ip a show $i >/dev/null 2>&1); then
        echo "Abort: The network interface $i doesn't exist"
        exit 1
    fi
done

# Check if the DOCKER-USER chain exists, if it does flushing the rules.
iptables -C FORWARD -j DOCKER-USER 2> /dev/null

if [ $? -eq 0 ]; then
    # Flush all existing rules
    iptables -F DOCKER-USER
else
    echo "Abort: The chain DOCKER-USER not found!"
    echo "Recommendation: "
    echo " - Is the Docker daemon running?"
    echo " - Is the version of Docker 17.06+?"
    exit 1
fi

## Docker container named www-nginx public access policy
## If using docker-compose, change FORMAT to
## --format '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}'
##
$WWW_IP="$(/usr/bin/docker inspect --format '{{.NetworkSettings.IPAddress}}' $CONTAINER 2>/dev/null)"
WWW_IP="$(/usr/bin/docker inspect --format '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}'
$CONTAINER 2>/dev/null)"

if [ -z "$WWW_IP" ];then
    echo "Abort: The docker IP isn't detected!"
    echo "Recommendation:"
    echo " - Is the specified container UP?"
    echo " - Adjust the variable \$WWW_IP."
    exit 1
fi

# Default action

```

```

iptables -I DOCKER-USER -i $EXT_IF -j DROP

## Insert web server container filter rules
## Allow All IPs to access 80 & 443
#iptables -I DOCKER-USER -i $EXT_IF -p tcp -d $WWW_IP --dport 80 -j ACCEPT
#iptables -I DOCKER-USER -i $EXT_IF -p tcp -d $WWW_IP --dport 443 -j ACCEPT
## Allow Cloudflare IPs to acces 443
## Check the IP list of Cloudflare at the URL https://www.cloudflare.com/ips-v4
iptables -I DOCKER-USER -o $BRG_IF -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
iptables -I DOCKER-USER -i $BRG_IF ! -o $BRG_IF -j ACCEPT
iptables -I DOCKER-USER -m state --state RELATED -j ACCEPT
iptables -I DOCKER-USER -i $BRG_IF -o $BRG_IF -j ACCEPT

iptables -I DOCKER-USER -i $EXT_IF -p tcp -d $WWW_IP -s 173.245.48.0/20 --dport 443 -j ACCEPT
iptables -I DOCKER-USER -i $EXT_IF -p tcp -d $WWW_IP -s 103.21.244.0/22 --dport 443 -j ACCEPT
iptables -I DOCKER-USER -i $EXT_IF -p tcp -d $WWW_IP -s 103.22.200.0/22 --dport 443 -j ACCEPT
iptables -I DOCKER-USER -i $EXT_IF -p tcp -d $WWW_IP -s 103.31.4.0/22 --dport 443 -j ACCEPT
iptables -I DOCKER-USER -i $EXT_IF -p tcp -d $WWW_IP -s 141.101.64.0/18 --dport 443 -j ACCEPT
iptables -I DOCKER-USER -i $EXT_IF -p tcp -d $WWW_IP -s 108.162.192.0/18 --dport 443 -j ACCEPT
iptables -I DOCKER-USER -i $EXT_IF -p tcp -d $WWW_IP -s 190.93.240.0/20 --dport 443 -j ACCEPT
iptables -I DOCKER-USER -i $EXT_IF -p tcp -d $WWW_IP -s 188.114.96.0/20 --dport 443 -j ACCEPT
iptables -I DOCKER-USER -i $EXT_IF -p tcp -d $WWW_IP -s 197.234.240.0/22 --dport 443 -j ACCEPT
iptables -I DOCKER-USER -i $EXT_IF -p tcp -d $WWW_IP -s 198.41.128.0/17 --dport 443 -j ACCEPT
iptables -I DOCKER-USER -i $EXT_IF -p tcp -d $WWW_IP -s 162.158.0.0/15 --dport 443 -j ACCEPT
iptables -I DOCKER-USER -i $EXT_IF -p tcp -d $WWW_IP -s 104.16.0.0/12 --dport 443 -j ACCEPT
iptables -I DOCKER-USER -i $EXT_IF -p tcp -d $WWW_IP -s 172.64.0.0/13 --dport 443 -j ACCEPT
iptables -I DOCKER-USER -i $EXT_IF -p tcp -d $WWW_IP -s 131.0.72.0/22 --dport 443 -j ACCEPT

# My IPs
#iptables -I DOCKER-USER -i $EXT_IF -p tcp -d $WWW_IP -s 219.70.88.133 --dport 443 -j ACCEPT

[ $? -eq 0 ] && echo "Done."
## EOL

```

for docker)

```

#!/usr/bin/env bash

# Usage:

```

```
# timeout 10 docker_iptables.sh

#

# Use the builtin shell timeout utility to prevent infinite loop (see below)

CONTAINER="raida17"

if [ ! -x /usr/bin/docker ]; then
    exit
fi

# Check if the PRE_DOCKER chain exists, if it does there's an existing reference to it.
iptables -C FORWARD -o docker0 -j PRE_DOCKER 2> /dev/null

if [ $? -eq 0 ]; then
    # Remove reference (will be re-added again later in this script)
    iptables -D FORWARD -o docker0 -j PRE_DOCKER
    # Flush all existing rules
    iptables -F PRE_DOCKER
else
    # Create the PRE_DOCKER chain
    iptables -N PRE_DOCKER
fi

# Default action
iptables -I PRE_DOCKER -j DROP

# Docker Containers Public Admin access (insert your IPs here)
#iptables -I PRE_DOCKER -i eth0 -s 192.184.41.144 -j ACCEPT
#iptables -I PRE_DOCKER -i eth0 -s 120.29.76.14 -j ACCEPT

# Docker Containers Restricted LAN Access (insert your LAN IP range or multiple IPs here)
#iptables -I PRE_DOCKER -i eth1 -s 192.168.1.101 -j ACCEPT
#iptables -I PRE_DOCKER -i eth1 -s 192.168.1.102 -j ACCEPT

# Docker internal use
iptables -I PRE_DOCKER -o docker0 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
iptables -I PRE_DOCKER -i docker0 ! -o docker0 -j ACCEPT
iptables -I PRE_DOCKER -m state --state RELATED -j ACCEPT
iptables -I PRE_DOCKER -i docker0 -o docker0 -j ACCEPT
```

```

# Docker container named www-nginx public access policy
# If using docker-compose, change format to
# --format '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}'
WWW_IP_CMD="/usr/bin/docker inspect --format '{{.NetworkSettings.IPAddress}}' $CONTAINER"
WWW_IP=$( $WWW_IP_CMD )

# Double check, wait for docker socket (upstart docker.conf already does this)
while [ ! -e "/var/run/docker.sock" ]; do echo "Waiting for /var/run/docker.sock..."; sleep 1; done

# Wait for docker web server container IP
while [ -z "$WWW_IP" ]; do echo "Waiting for $CONTAINER IP..."; WWW_IP=$( $WWW_IP_CMD ); done

## Insert web server container filter rules
## Allow All IPs to access 80 & 443
#iptables -I PRE_DOCKER -i eth0 -p tcp -d $WWW_IP --dport 80 -j ACCEPT
#iptables -I PRE_DOCKER -i eth0 -p tcp -d $WWW_IP --dport 443 -j ACCEPT
## Allow Cloudflare IPs to access 443
iptables -I PRE_DOCKER -i eth0 -p tcp -d $WWW_IP -s 103.21.244.0/22 --dport 443 -j ACCEPT
iptables -I PRE_DOCKER -i eth0 -p tcp -d $WWW_IP -s 103.22.200.0/22 --dport 443 -j ACCEPT
iptables -I PRE_DOCKER -i eth0 -p tcp -d $WWW_IP -s 103.31.4.0/22 --dport 443 -j ACCEPT
iptables -I PRE_DOCKER -i eth0 -p tcp -d $WWW_IP -s 104.16.0.0/12 --dport 443 -j ACCEPT
iptables -I PRE_DOCKER -i eth0 -p tcp -d $WWW_IP -s 108.162.192.0/18 --dport 443 -j ACCEPT
iptables -I PRE_DOCKER -i eth0 -p tcp -d $WWW_IP -s 131.0.72.0/22 --dport 443 -j ACCEPT
iptables -I PRE_DOCKER -i eth0 -p tcp -d $WWW_IP -s 141.101.64.0/18 --dport 443 -j ACCEPT
iptables -I PRE_DOCKER -i eth0 -p tcp -d $WWW_IP -s 162.158.0.0/15 --dport 443 -j ACCEPT
iptables -I PRE_DOCKER -i eth0 -p tcp -d $WWW_IP -s 172.64.0.0/13 --dport 443 -j ACCEPT
iptables -I PRE_DOCKER -i eth0 -p tcp -d $WWW_IP -s 173.245.48.0/20 --dport 443 -j ACCEPT
iptables -I PRE_DOCKER -i eth0 -p tcp -d $WWW_IP -s 188.114.96.0/20 --dport 443 -j ACCEPT
iptables -I PRE_DOCKER -i eth0 -p tcp -d $WWW_IP -s 190.93.240.0/20 --dport 443 -j ACCEPT
iptables -I PRE_DOCKER -i eth0 -p tcp -d $WWW_IP -s 197.234.240.0/22 --dport 443 -j ACCEPT
iptables -I PRE_DOCKER -i eth0 -p tcp -d $WWW_IP -s 198.41.128.0/17 --dport 443 -j ACCEPT
iptables -I PRE_DOCKER -i eth0 -p tcp -d $WWW_IP -s 199.27.128.0/21 --dport 443 -j ACCEPT

# Finally insert the PRE_DOCKER table before the DOCKER table in the FORWARD chain.
iptables -I FORWARD -o docker0 -j PRE_DOCKER

```

With iptables + ipset

- [Limit Docker Container Access to Certain IP Addresses](#)

With ufw

- [ufw + Docker](#)

With Firewallld

```
# Removing DOCKER-USER CHAIN (it won't exist at first)
firewall-cmd --permanent --direct --remove-chain ipv4 filter DOCKER-USER

# Flush rules from DOCKER-USER chain (again, these won't exist at first; firewallld seems to remember these
even if the chain is gone)
firewall-cmd --permanent --direct --remove-rules ipv4 filter DOCKER-USER

# Add the DOCKER-USER chain to firewallld
firewall-cmd --permanent --direct --add-chain ipv4 filter DOCKER-USER

# Add rules (see comments for details)
firewall-cmd --permanent --direct --add-rule ipv4 filter DOCKER-USER 0 -m conntrack --ctstate
RELATED,ESTABLISHED -j ACCEPT -m comment --comment "This allows docker containers to connect to the
outside world"

## Change the Docker Subnet address to your network settings (Could be 172.18.0.0/16)
firewall-cmd --permanent --direct --add-rule ipv4 filter DOCKER-USER 0 -j RETURN -s 172.17.0.0/16 -m comment
--comment "allow internal docker communication"

## Add an ip to allow access to the docker exposed port
firewall-cmd --permanent --direct --add-rule ipv4 filter DOCKER-USER 0 -p tcp -m multiport --dports 9392 -s
10.18.109.20/32 -j ACCEPT -m comment --comment "my allowed ip address to openvas port"

# Add as many ip or other rules and then run this command to block all other traffic
firewall-cmd --permanent --direct --add-rule ipv4 filter DOCKER-USER 0 -j REJECT -m comment --comment "reject
all other traffic"

# restart the services
systemctl stop docker
systemctl stop firewallld
systemctl start firewallld
systemctl start docker
```


MySQL Backup

? Host ??

backup-db.sh

```
#!/bin/bash

BKDIR="db_backups"
BKFILE="cloudcoin_raida#17.`date +%y%m%d`.sql"
DBUSER="ThisDBUser"
DBNAME="cloudcoin_raida"
CONTNAME="raida17"
CONTDIR="/opt/data/$BKDIR"
HOSTDIR="/docker_vol/raida/data/$BKDIR"
KEEP=2

[ -d $HOSTDIR ] || mkdir $HOSTDIR
cd $HOSTDIR
start_s=$(date +%s)
echo "***** START `date "+%F@%T"` *****"

# Full Backup
echo "-> Back up Full DB "
docker exec $CONTNAME bash -c "mysqldump --single-transaction --add-drop-table -u $DBUSER $DBNAME > $CONTDIR/$BKFILE"
retval=$?

if [ $retval -eq 0 ];then
    echo "... [success]"
else
    echo "... [failure]**"
    exit 1
fi

# Compress the files
```

```
echo "-> Compress the dump file "
gzip -f $HOSTDIR/$BKFILE

retval=$?
if [ $retval -eq 0 ];then
    echo "... [success]"
    BKFILE="$BKFILE.gz"
else
    echo "... [*failure*]"
    exit 1
fi

# Purge the old files
echo "-> Purge the old files"
ls_files=$(ls -lt cloudcoin_raida* | awk -F ' ' '{print $9}')
len=${#ls_files[@]}
i=$KEEP
while ((i < $len));do
    rm -vf "${ls_files[i]}"
    let i++
done

end_s=$(date +%s)
elapsed=$(( end_s - start_s ))
echo
echo "Elapsed: $elapsed seconds"
echo "Output File: $HOSTDIR/$BKFILE"
echo "***** END `date +%F@%T` *****"
echo
```

??????

????

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}}"
```

?? Volumes

Docker ? Data Volume ?????????????????? containers ??????????????????

?????

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

```
// □ volume
```

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

“ TIP:

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

??????????

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

```
# [?????] volume [????????????] --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
```

Log Rotation for Container

- ????

Edit `/etc/docker/daemon.json`

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

Restart Docker daemon

```
sudo systemctl restart docker
```

- ????

```
# Disable logging
docker run --log-driver=none

# Limit logging
docker run --log-driver=json-file --log-opt max-size=10m --log-opt max-file=3
```

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
jrcls/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  jrcls/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-entrypt.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

```

Limit CPU & Memory

```
docker run --memory=512m --cpus=1
```

Docker Compose

```

services:
  app:
    image: myimage
    deploy:
      resources:
        limits:
          cpus: '0.50'
          memory: 256M

```

FAQ

- ????

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

Solution: ????

```

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 ???

```

Docker Compose ??

NOTE: ?????? `docker compose` ?

NOTE: ??????????docker-compose.yml ?????????????????????? docker-compose ???
container??

??????????????

```
# For all services
docker-compose up -d

# For specified service
docker-compose up -d <service-name>
```

Build the image of the service

```
docker-compose build <service-name>
docker-compose up --rebuild <service-name>
```

????????????

```
docker-compose ps
```

???????????????????? container

```
docker-compose stop
docker-compose start
```

“ ?? container ???

???????????????????? container

For all services

docker-compose down

For a specified service

docker-compose stop <service-name>

docker-compose rm -f <service-name>

“ NOTE:

- ?????? docker-compose.yml ?????? down ?????? up -d ??????

- ?????????????????? docker-compose up -d

??????????????

docker-compose logs

????????????????

docker-compose exec <service-name> sh -c "pwd"

????? --env-file

docker compose --env-file .env up -d

docker-compose.yml

environment + healthcheck

services:

db:

image: pgvector/pgvector:0.6.2-pg15

restart: always

ports:

- '5432:5432'

environment:

POSTGRES_USER: talkdai

POSTGRES_PASSWORD: talkdai

POSTGRES_DB: talkdai

```
volumes:
  - ./etc/db-extensions.sql:/docker-entrypoint-initdb.d/db-extensions.sql
healthcheck:
  test: ["CMD", "pg_isready", "-d", "talkdai", "-U", "talkdai"]
  interval: 10s
  timeout: 5s
  retries: 5
dialog:
  image: ghcr.io/talkdai/dialog:latest
  volumes:
    - ./data:/app/data/
  ports:
    - '8000:8000'
  depends_on:
    db:
      condition: service_healthy
  environment:
    - PORT=8000
    - DATABASE_URL=postgresql://talkdai:talkdai@db:5432/talkdai
    - OPENAI_API_KEY=sk-your-openai-api-key
    - STATIC_FILE_LOCATION=/app/static
    - DIALOG_DATA_PATH=./data/your.csv
    - PROJECT_CONFIG=./data/your.toml
```

healthcheck(web) + build + networks

```
networks:
  net:
    driver: bridge

services:
  server:
    image: server
    build:
      context: .
      dockerfile: Dockerfile
    volumes:
      # Be aware that indexed data are located in "/chroma/chroma/"
      # Default configuration for persist_directory in chromadb/config.py
      # Read more about deployments: https://docs.trychroma.com/deployment
```

- chroma-data:/chroma/chroma

command: "--workers 1 --host 0.0.0.0 --port 8000 --proxy-headers --log-config chromadb/log_config.yml --
timeout-keep-alive 30"

environment:

- IS_PERSISTENT=TRUE
- CHROMA_SERVER_AUTHN_PROVIDER=\${CHROMA_SERVER_AUTHN_PROVIDER}
- CHROMA_SERVER_AUTHN_CREDENTIALS_FILE=\${CHROMA_SERVER_AUTHN_CREDENTIALS_FILE}
- CHROMA_SERVER_AUTHN_CREDENTIALS=\${CHROMA_SERVER_AUTHN_CREDENTIALS}
- CHROMA_AUTH_TOKEN_TRANSPORT_HEADER=\${CHROMA_AUTH_TOKEN_TRANSPORT_HEADER}
- PERSIST_DIRECTORY=\${PERSIST_DIRECTORY:-/chroma/chroma}
- CHROMA_OTEL_EXPORTER_ENDPOINT=\${CHROMA_OTEL_EXPORTER_ENDPOINT}
- CHROMA_OTEL_EXPORTER_HEADERS=\${CHROMA_OTEL_EXPORTER_HEADERS}
- CHROMA_OTEL_SERVICE_NAME=\${CHROMA_OTEL_SERVICE_NAME}
- CHROMA_OTEL_GRANULARITY=\${CHROMA_OTEL_GRANULARITY}
- CHROMA_SERVER_NOFILE=\${CHROMA_SERVER_NOFILE}

restart: unless-stopped # possible values are: "no", "always", "on-failure", "unless-stopped"

ports:

- "8000:8000"

healthcheck:

Adjust below to match your container port

test: ["CMD", "curl", "-f", "http://localhost:8000/api/v1/heartbeat"]

interval: 30s

timeout: 10s

retries: 3

networks:

- net

volumes:

chroma-data:

driver: local

Installation

Docker Compose

Updated: ?? Docker Compose ????? Docker ????

????

```
sudo apt-get install docker-compose-plugin
```

????

```
docker compose version
```

The latest download: <https://docs.docker.com/compose/install/>

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

If you can't run the command, creating a sybolic link as follows.

```
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

```
docker-compose --version
```

ARM64 Linux

Download: <https://github.com/docker/compose/releases>

```
sudo curl -L --fail https://github.com/docker/compose/releases/download/v2.2.3/docker-compose-linux-aarch64 -o /usr/local/bin/docker-compose
```

```
sudo chmod 0755 /usr/local/bin/docker-compose
```

Docker

Latest: <https://docs.docker.com/engine/install/>

Fedora 41

```
sudo dnf -y install dnf-plugins-core
sudo dnf-3 config-manager --add-repo https://download.docker.com/linux/fedora/docker-ce.repo

sudo dnf install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Ubuntu 20.04/22.04

```
sudo apt-get update
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg \
    lsb-release

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-
archive-keyring.gpg

echo \
    "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

Debian 9/10

```
sudo apt-get update

sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl gnupg \
    lsb-release

# Add Docker's GPG key
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o
```

```
/etc/apt/trusted.gpg.d/docker.gpg
```

```
# Set up the repository
```

```
echo \
```

```
"deb [arch=$(dpkg --print-architecture)] https://download.docker.com/linux/debian \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
# Install Docker Engine
```

```
sudo apt-get update
```

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

```
# Verify that Docker is installed correctly.
```

```
sudo docker version
```

```
sudo docker run hello-world
```

RedHat 8

```
“ RHEL 8 ?? IBM Z(s390x) ?????? Docker????? x86_64 ????? CentOS ??????
????? docker-ce ??????????????????
```

???? (??????????? RHN)

```
dnf update
```

???? docker-ce ?????

?? /etc/yum.repos.d/docker-ce.repo

```
[docker-ce-stable]
```

```
name=Docker CE Stable - $basearch
```

```
baseurl=https://download.docker.com/linux/centos/$releasever/$basearch/stable
```

```
enabled=1
```

```
gpgcheck=1
```

```
gpgkey=https://download.docker.com/linux/centos/gpg
```

?????

```
dnf config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

```
dnf update
```

????????

?? docker-ce

??????

?? docker ??

```
[root@mydocker ~]# docker version

Client: Docker Engine - Community
Version:      20.10.14
API version:  1.41
Go version:   go1.16.15
Git commit:   a224086
Built:        Thu Mar 24 01:47:44 2022
OS/Arch:      linux/amd64
Context:      default
Experimental:  true

Server: Docker Engine - Community
Engine:
Version:      20.10.14
```

```
API version:    1.41 (minimum version 1.12)
Go version:     go1.16.15
Git commit:     87a90dc
Built:          Thu Mar 24 01:46:10 2022
OS/Arch:        linux/amd64
Experimental:   false
containerd:
  Version:      1.5.11
  GitCommit:    3df54a852345ae127d1fa3092b95168e4a88e2f8
runc:
  Version:      1.0.3
  GitCommit:    v1.0.3-0-gf46b6ba
docker-init:
  Version:      0.19.0
  GitCommit:    de40ad0
```

[Optional] non-root ????

```
usermod -aG docker $USER
```

Raspberry Pi OS

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
sudo usermod -aG docker Pi
docker version
docker info
docker run hello-world
```

Rootless Mode

Ubuntu 22.04

Stop system-wide Docker daemon

```
sudo systemctl disable --now docker.service docker.socket
```

Install

```
sudo apt-get install docker-ce-rootless-extras uidmap
dockerd-rootless-setup.sh install
docker info
```

Optional: Configure the systemd

```
systemctl --user start docker
systemctl --user enable docker
sudo loginctl enable-linger $(whoami)
```

- The socket path is set to `$XDG_RUNTIME_DIR/docker.sock` by default. `$XDG_RUNTIME_DIR` is typically set to `/run/user/$UID`.
- The data dir is set to `~/.local/share/docker` by default. The data dir should not be on NFS.
- The daemon config dir is set to `~/.config/docker` by default. This directory is different from `~/.docker` that is used by the client.

Optional: Client

```
# Optional #1: with socket path
export DOCKER_HOST=unix://$XDG_RUNTIME_DIR/docker.sock
docker run -d -p 8080:80 nginx
```

```
# Optional #2: with CLI context
docker context use rootless
docker run -d -p 8080:80 nginx
```

Dockerfile

Build Image

- Docker image ????: [dive](#)

Basic build command

```
cd /path/to/Dockerfile
docker build -t your-tag/your-name .
```

Set the version

More details: [docker-bookstack/Dockerfile at master · linuxserver/docker-bookstack · GitHub](#)

```
FROM ghcr.io/linuxserver/baseimage-alpine-nginx:3.17

# set version label
ARG BUILD_DATE
ARG VERSION
ARG BOOKSTACK_RELEASE
LABEL build_version="Linuxserver.io version:- ${VERSION} Build-date:- ${BUILD_DATE}"
LABEL maintainer="homerr"
```

????:

Build the image

```
docker build \
  --build-arg BUILD_DATE=$( date +"%FT%T%z" ) \
  --build-arg VERSION=v22.07.3-ls36 \
  --no-cache \
  --pull \
  -t lscr.io/linuxserver/bookstack:latest .
```

Get the version of container

```
docker inspect -f '{{ index .Config.Labels "build_version" }}' <container-name>
```

Examples

Chroma DB

```
FROM python:3.11-slim-bookworm AS builder
ARG REBUILD_HNSWLIB
RUN apt-get update --fix-missing && apt-get install -y --fix-missing \
    build-essential \
    gcc \
    g++ \
    cmake \
    autoconf && \
    rm -rf /var/lib/apt/lists/* && \
    mkdir /install

WORKDIR /install

COPY ./requirements.txt requirements.txt

RUN pip install --no-cache-dir --upgrade --prefix="/install" -r requirements.txt
RUN if [ "$REBUILD_HNSWLIB" = "true" ]; then pip install --no-binary :all: --force-reinstall --no-cache-dir --
    prefix="/install" chroma-hnswlib; fi

FROM python:3.11-slim-bookworm AS final

RUN mkdir /chroma
WORKDIR /chroma

COPY --from=builder /install /usr/local
COPY ./bin/docker_entrypoint.sh /docker_entrypoint.sh
COPY ./ /chroma

RUN apt-get update --fix-missing && apt-get install -y curl && \
    chmod +x /docker_entrypoint.sh && \
    rm -rf /var/lib/apt/lists/*

ENV CHROMA_HOST_ADDR "0.0.0.0"
ENV CHROMA_HOST_PORT 8000
```

```
ENV CHROMA_WORKERS 1
ENV CHROMA_LOG_CONFIG "chromadb/log_config.yml"
ENV CHROMA_TIMEOUT_KEEP_ALIVE 30

EXPOSE 8000

ENTRYPOINT ["/docker_entrypoint.sh"]
CMD [ "--workers ${CHROMA_WORKERS} --host ${CHROMA_HOST_ADDR} --port ${CHROMA_HOST_PORT} --
proxy-headers --log-config ${CHROMA_LOG_CONFIG} --timeout-keep-alive ${CHROMA_TIMEOUT_KEEP_ALIVE}"]
```

Learning

- [Writing An Optimized Dockerfile](#)
- [What is the Difference Between COPY and ADD Instructions in Dockerfile](#)
- [Best practices for writing Dockerfiles](#)

Learning

????

Basic

- [??] [Docker in Production](#)
- [The Docker Handbook – Learn Docker for Beginners](#)
- [Introduction to Docker eBook](#)

Advanced

- [How to SSH into a Docker Container](#)
- [Monitoring Docker Containers With Grafana Using Dockprom](#)
- [How to Set Up Remote Access to Docker Daemon \[Detailed Guide\]](#)
- [How to deploy on remote Docker hosts with docker-compose](#)
- [Where are Docker Images, Containers and Volumes Stored on the Linux Host System?](#)
- [How To Analyze And Compare Container Images Using Container-diff](#)
- [How to Set Docker Memory and CPU Usage Limit](#)
- [?????? Docker ????????. ???? Docker/Kubernetes... | by Ian Chen | Starbugs Weekly ??????? | Medium](#)
- [Top Docker Performance Tweaks on Low-Power Hardware - Virtualization Howto](#)

Backup

Resilio Sync

- [Backup Docker Containers using Resilio Sync](#)
- [Video] [Backup Docker Container Files with Resilio Sync](#)

Network

- [How to Create and Use MacVLAN Network in Docker](#)
- [Gluetun??Docker???VPN????????????????? · Ivon???? \(ivonblog.com\)](#)

Container

- [What you need to know about containers for Python](#)

Docker Security Essentials

Download: [Linode_eBook_HackerSploit_DockerSecurityEssentials.pdf](#)

Content is structured and organized as follows:

- In **The Docker Platform** section, we will begin the process by explaining the various components that make up the Docker platform.
- In the **Auditing Docker Security** section, we will explore the process of performing a security audit of the Docker platform. An audit identifies vulnerabilities in the configuration of the components that make up the platform.
- In the next two sections, we will begin the process of securing the Docker host and the Docker daemon to ensure that we have a secure base to operate from:
 - **Securing the Docker Host**
 - **Securing the Docker Daemon**
- The remaining sections of the guide will conclude by taking a look at the various ways of securing containers and the process of building secure Docker images:
 - **Container Security Best Practices**
 - **Controlling Container Resource Consumption with Control Groups (cgroups)**
 - **Implementing Access Control with AppArmor**
 - **Limiting Container System Calls with seccomp**
 - **Vulnerability Scanning for Docker Containers**
 - **Building Secure Docker Images**

Docker Monitoring

- [Video] [Best Docker Container Monitoring Tools - Free and open source](#)
- [Lazydocker](#) - Docker UI
 - [Video] [Your Minimalist Docker UI - Lazydocker](#)
- [Dozzle](#) - Real-time logging and monitoring for Docker in the browser
 - GitHub: <https://github.com/amir20/dozzle>

Other Tools

- [dockerc](#) - compile docker images to standalone portable binaries

Container Registry Providers

- DockerHub
- Google Container Registry (GCR)
- Azure Container Registry (ACR)
- AWS Elastic Container Registry (ECR)

SWAG - Secure Web Application Gateway

[SWAG](#) is a rebirth of the [letsencrypt docker image](#), a full fledged web server and reverse proxy that includes Nginx, Php7, Certbot (Let's Encrypt client) and Fail2ban.

Tutorials

- [Introducing SWAG - Secure Web Application Gateway](#)
- [SWAG setup](#)

Transfer/Move Docker

Sample: osslab-dekiwiki

```
docker run -d -p 8880:80 \  
-v /etc/localtime:/etc/localtime:ro \  
-v $PWD/vol/data:/data \  
-v $PWD/vol/var-lib-mysql:/var/lib/mysql \  
-v $PWD/vol/var-log-httpd:/var/log/httpd \  
-v $PWD/vol/var-log-dekiwiki:/var/log/dekiwiki \  
-v $PWD/vol/var-www-dekiwiki-attachments:/var/www/dekiwiki/attachments \  
-v $PWD/vol/var-www-dekiwiki-bin-cache:/var/www/dekiwiki/bin/cache \  
--name $NAME \  
osslab-dekiwiki \  
/bin/bash
```

Using Save Command

Save Image

```
# osslab-dekiwiki is image name  
docker save osslab-dekiwiki > osslab-dekiwiki.tar
```

```
ls -l
```

```
-rw-r--r-- 1 root root 2519701504 Feb 27 19:19 osslab-dekiwiki.tar
```

Load Image

```
docker load < osslab-dekiwiki.tar
```

Using Export Command

Run the instance in detach mode

```
docker run -it --detach --name osslab-mig osslab-dekiwiki
```

```
docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS	NAMES			
673f45c27a8b	osslab-dekiwiki	"/bin/sh -c /startup..."	5 seconds ago	Up 4 seconds
80/tcp	osslab-mig			

Export Docker Container

```
docker export 673f45c27a8b > osslab-mig.tar
```

```
ls -l
```

```
-rw-r--r-- 1 root root 1143085568 Feb 27 14:21 osslab-mig.tar
```

Import Docker Container to the target machine

```
tar -c osslab-mig.tar | docker import - osslab-dekiwiki
```

Move Container

```
## Stop the container
docker stop <container-name>

## Save container image
docker commit <container-name> mycontainerimage
docker save mycontainerimage | gzip > mycontainerimage.tar.gz

## Load container image to destination host
gunzip -c mycontainerimage.tar.gz | docker load

## Transfer image without creating a file
docker save mycontainerimage | gzip | ssh root@203.0.113.1 'gunzip | docker load'
```

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

```
## Save container image
```

```
docker commit <container-name> mycontainerimage
```

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

```
## Load container image to destination host
```

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

```
## Transfer image without creating a file
```

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

“ TIP:

- exit container Linux
- container commit image?

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

```
???? container???????????????????????????????? commit ??????
image?
```

Backup & Restore Volume

Backup volume

```
# Backup
cd /root/osslab
docker stop osslab-dekiwiki
docker rm osslab-dekiwiki
tar czf osslab-val.tar.gz vol

# Restore to target machine
cd /root/osslab
tar xpf osslab-val.tar.gz
```

FAQ

Q: Import ???????? container?

“ Error response from daemon: OCI runtime create failed: container_linux.go:380: starting container process caused: exec: "/bin/sh": stat /bin/sh: no such file or directory: unknown
Error: failed to start containers: osslab

Solution: ???? Export-Import ?? Save-Load ?????

Q: ?? docker load ?????

“ open /var/lib/docker/tmp/docker-import-863867335/bin/json: no such file or directory

Solution: ?? Save ? image ????? `docker load` ?Export ? image ???? `docker import` ?

Docker Management

Portainer

<https://www.portainer.io/>

A centralized service delivery platform for containerized apps.

Portainer can be deployed on top of any K8s, Docker or Swarm environment. It works seamlessly in the cloud, on prem and at the edge to give you a consolidated view of all your containers.

Portainer is available in 2 versions: Business Edition (BE) and Community Edition (CE). Portainer BE is a fully supported, service delivery platform suitable for use organization-wide, whilst CE is open source and suitable for use by small, self-supporting teams.

Installation

- <https://docs.portainer.io/v/ce-2.11/start/install>
- [Video] <https://www.youtube.com/watch?v=DGw6P5Lkj-U>

Yacht

[Yacht](#) - Yacht is a container management UI with a focus on templates and 1-click deployments.

- [Video] [Yacht - an Open Source, Self Hosted, Modern, Web GUI for Docker Management similar to Portainer.](#)
- [Yacht ? ??Docker????????????docker-compose · Ivon???? \(ivonblog.com\)](#)

Dockge

Self-hosted - Docker compose.yaml - Stack-oriented Manager

- [Dockge](#)
- GitHub: <https://github.com/louislam/dockge>

Watchtower

Setup

run.sh

```
docker stop watchtower-myst-presearch-storj
docker rm watchtower-myst-presearch-storj

docker pull containrrr/watchtower
docker run -d \
  --name watchtower-myst-presearch-storj \
  -v /var/run/docker.sock:/var/run/docker.sock \
  containrrr/watchtower \
  myst presearch-node storagenode
docker ps
```

Tutorials

- [How To Automatically Update Running Docker Containers Using Watchtower](#)

```
# □ Nginx Web □
$ docker run -d --name target nginx

$ TARGET_IP=$(
  docker inspect \
    -f '{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' \
    target
)
```

```
$ NETWORK=$(
  docker container inspect \
    -f '{{range $net,$v := .NetworkSettings.Networks}}{{printf "%s" $net}}{{end}}' \
    target
)

$ docker run -d \
  --publish ${HOST_PORT}:${TARGET_PORT} \
  --network ${NETWORK} \
  --name forwarder nixery.dev/socat \
  socat TCP-LISTEN:${TARGET_PORT},fork TCP-CONNECT:${TARGET_IP}:${TARGET_PORT}

$ curl localhost:${TARGET_PORT}
```

???? Container ??

- [Updating Docker Containers With Zero Downtime \(linuxhandbook.com\)](https://linuxhandbook.com/zero-downtime-docker/)

Linux Distros for Running Container

- [5 Best Lightweight Linux Distros for Running Containers \(tecmint.com\)](#)

Photon OS

- <https://vmware.github.io/photon/docs/>
- [Downloading Photon OS · vmware/photon Wiki · GitHub](#)

Default account credentials:

- username:
- password:

Run Docker

```
# To run Docker from the command prompt
systemctl start docker

# To ensure Docker daemon service runs on every subsequent VM reboot
systemctl enable docker
```

Fedora CoreOS

- [The container optimized OS | The Fedora Project](#)
- Doc: [Fedora CoreOS Documentation :: Fedora Docs](#)
- [Proxmox Containers with Fedora CoreOS Install - Virtualization Howto](#)

Dockerize Custom Application

Tutorials

- [Dockerize a Python application - DEV Community](#)
- [DEV Community](#)