



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

## Running Containers and Troubleshooting

# Table of Contents

<b>Running Containers</b>	
Running Containers	4
<b>Troubleshooting</b>	
Troubleshooting	10

---

Search the table of all articles in this guide, listed in alphabetical order, to find the article you need.

# Running Containers

## Contents

- [1 Running Containers](#)
- [2 Lifecycle](#)
- [3 Starting and Stopping a Container](#)

Instructions to run the Docker containers.

### Warning

The following content has been deprecated and is maintained for reference only.

## Running Containers

**Note:** At the end of this topic, you will be provided with a terminal to an environment that has all the prerequisites (such as Docker and Kubernetes) up and running. You can practice your commands in this tutorial without any need to setup your own environment.

Containers are running instances of an Image. To run containers, follow these steps:

1. Create a container from the base image for the latest version of the Ubuntu that is available.

### Important

- If you do not have an Ubuntu base image installed locally, extract the latest one for your local repository.
- You must start the container in interactive mode attached to the current terminal and running the bash shell.
- After running, make sure you shut down the container by running 'exit'.

```
[user@tcox1 ~]$ sudo docker pull ubuntu:latest
Trying to pull repository docker.io/library/ubuntu ...
latest: Pulling from docker.io/library/ubuntu
ae79f2514705: Pull complete
5ad56d5fc149: Pull complete
170e558760e8: Pull complete
395460e233f5: Pull complete
6f01dc62e444: Pull complete
Digest: sha256:506e2d5852de1d7c90d538c5332bd3cc33b9cbd26f6ca653875899c505c82687
[user@tcox1 ~]$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
docker.io/httpd	latest	c24f66af34b4	5 days ago	177.3 MB
docker.io/ubuntu	latest	747cb2d60bbe	7 days ago	122 MB

```
[user@tcox1 ~]$ sudo docker run -it ubuntu:latest /bin/bash
root@f1d4d12c2c70:/# exit
exit
```

2. Run the appropriate Docker command to obtain the name of the previously run container. Issue the appropriate command to restart the container for which you obtained the name. Do NOT create a new container. Restart the container that was just used.

```
[user@tcox1 ~]$ sudo docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f1d4d12c2c70	ubuntu: latest	"/bin/bash"	About a minute ago	Exited(0)	About a minute ago	jovial_kilby

```
[user@tcox1 ~]$ sudo docker restart jovial_kilby
jovial_kilby
[user@tcox1 ~]$ sudo docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f1d4d12c2c70	ubuntu: latest	"/bin/bash"	2 minutes ago	Upto 7 seconds		jovial_kilby

3. Stop the container, and then remove the container from the system by using the following command.

```
[user@tcox1 ~]$ sudo docker stop jovial_kilby
jovial_kilby
[user@tcox1 ~]$ sudo docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f1d4d12c2c70	ubuntu:latest	"/bin/bash"	3 minutes ago	Exited (0) 10 seconds ago		jovial_kilby

```
[user@tcox1 ~]$ sudo docker rm jovial_kilby
jovial_kilby
[user@tcox1 ~]$ sudo docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

4. Create (not run) a container called "my\_container" by using the parameters that will allow the container to run interactively, and get the terminal attached to the local console running the bash shell. Ensure the container is not running.

```
[user@tcox1 ~]$ sudo docker create -it --name="my-container" ubuntu:latest /bin/bash
c90b35870c09fe63d1bac782342dd734b2edf4ac6abb282690d1585aa259841e
[user@tcox1 ~]$ sudo docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
c90b35870c09	ubuntu:latest	"/bin/bash"	4 seconds ago	Created		my-container

5. Start the container, and ensure the container is running. Run the following command to attach your session to the running container to ensure you are logged on to the shell.

```
[user@tcox1 ~]$ sudo docker start my-container
my-container
[user@tcox1 ~]$ sudo docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
c90b35870c09	ubuntu:latest	"/bin/bash"	4 minutes ago	Up 2 minutes		my-container

```
[user@tcox1 ~]$ sudo docker attach my-container
root@c90b35870c09:/#
root@c90b35870c09:/# exit
exit
```

### Lifecycle

The following commands illustrate the Docker Lifecycle:

- `docker create` creates a container but does not start the container.
- `docker rename` allows the container to be renamed.
- `docker run` creates and starts a container in a single operation.
- `docker rm` deletes a container.
- `docker update` updates a container's resource limits.

Usually, when you run a container without options, it will start and stop immediately. If you want the container to keep running, you can use the command, `docker run -td container_ID`. This command uses the option `-t` to allocate a pseudo-TTY session and option `-d` to detach the container automatically (you can run container in background and print the container ID).

To have a transient container, use the command `docker run --rm`. This command will remove the container after it stops.

To map a directory on the host to a docker container, use the command `docker run -v $HOSTDIR:$DOCKERDIR`.

To remove the volumes associated with the container, the deletion of the container must include the option `-vswitch` like in `docker rm -v`.

There is also a logging driver available for individual containers in docker 1.10. To run docker with a custom log driver (that is syslog), use the command `docker run --log-driver=syslog`.

`docker run --name yourname docker_image` is a useful command because when you specify `--name` inside the run command, you can start and stop a container by calling it with the name that you specified when you created it.

### Starting and Stopping a Container

Commands to start and stop a container:

- `docker start` starts a container so it is running.
- `docker stop` stops a running container.
- `docker restart` stops and starts a container.
- `docker pause` pauses a running container, "freezing" it in place.
- `docker unpause` unpauses a running container.
- `docker wait` blocks until running container stops.
- `docker kill` sends a SIGKILL signal to a running container.
- `docker attach` connects to a running container.

To integrate a container with a host process manager, start the daemon with the commands `-r=false` and then use `docker start -a`.

You can practice the above-mentioned commands using the following widget:





# Troubleshooting

## Contents

- [1 Troubleshooting Docker Containers](#)
- [2 Information on Running Docker Containers](#)

### How to troubleshoot docker containers

#### Warning

The following content has been deprecated and is maintained for reference only.

## Troubleshooting Docker Containers

**Note:** At the end of this topic, you will be provided with a terminal to an environment that has all the prerequisites (such as Docker and Kubernetes) up and running. You can practice your commands in this tutorial without any need to setup your own environment.

1. Get environment settings.

```
docker run --rm ubuntu env
```

2. Kill running containers.

```
docker kill $(docker ps -q)
```

3. Delete all containers (force!! running or stopped containers).

```
docker rm -f $(docker ps -qa)
```

4. Delete old containers.

```
docker ps -a | grep 'weeks ago' | awk '{print $1}' | xargs docker rm
```

5. Delete stopped containers.

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

6. Delete containers after stopping.

```
docker stop $(docker ps -aq) && docker rm -v $(docker ps -aq)
```

7. Delete dangling images.

```
docker rmi $(docker images -q -f dangling=true)
```

8. Delete all images.

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

9. Delete dangling volumes. As of Docker 1.9.0:

```
docker volume rm $(docker volume ls -q -f dangling=true)
```

In 1.9.0, the filter `dangling=false` does not work. It is ignored and lists all volumes.

10. Show image dependencies.

```
docker images -viz | dot -Tpng -o docker.png
```

11. Slim down Docker containers.

**Cleaning APT in a RUN layer:** This must be done in the same layer as that of the other APT commands. If not, the previous layers will still contain the original information and your images will still be large.

```
RUN {apt commands} \  
&& apt-get clean \  
&& rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
```

**Flatten an image:**

```
ID=$(docker run -d image-name /bin/bash)  
docker export $ID | docker import --flat image-name
```

**For backup:**

```
ID=$(docker run -d image-name /bin/bash)  
(docker export $ID | gzip -c > image.tgz)  
gzip -dc image.tgz | docker import --flat image-name
```

## Information on Running Docker Containers

- `docker ps` displays running containers.
- `docker logs` gets logs from the container. (You can use a custom log driver, but logs are available only for `json-file` and `journald` in 1.10).
- `docker inspect` inspects all the information of a container (including the IP address).
- `docker events` gets events from the container.
- `docker port` displays the public facing port of the container.
- `docker top` displays the running processes in container.
- `docker stats` displays the containers' resource usage statistics.
- `docker diff` displays the changed files in the container's FS.
- `docker ps -a` displays running and stopped containers.
- `docker stats --all` displays a running list of containers.
- `docker update` updates a container's resource limits.

To check the CPU, memory, and network I/O usage of a single container:

`docker stats`

For all containers listed by ID:

`docker stats $(docker ps -q)`

For all containers listed by name:

`docker stats $(docker ps --format '{{.Names}}')`

For all containers listed by image:

`docker ps -a -f ancestor=ubuntu`

To remove all untagged images:

`docker rmi $(docker images | grep "^" | awk '{split($0,a," "); print a[3]}')`

To remove container by a regular expression:

`docker ps -a | grep wildfly | awk '{print $1}' | xargs docker rm -f`

To remove all exited containers:

`docker rm -f $(docker ps -a | grep Exit | awk '{ print $1 }')`

You can practice the above-mentioned commands using the following widget:

