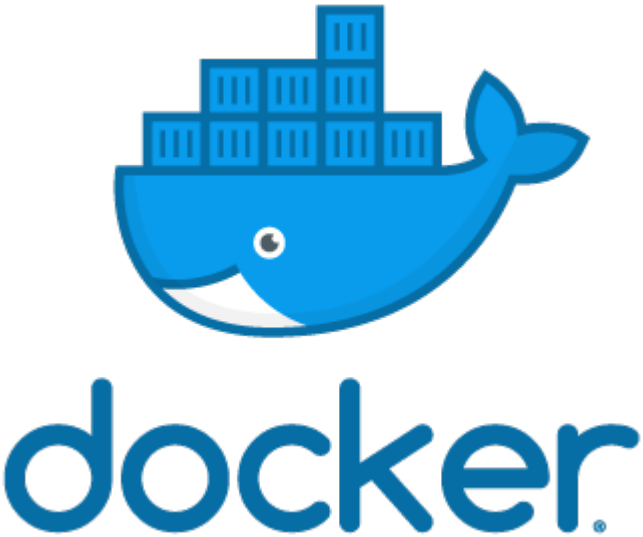
Restart Stopped Containers in Docker



Sometimes an issue on one of your servers may interrupt your Docker based development and stop all the containers that you haven't fully configured to be auto-started just yet. In such cases, it will be useful for you to know how to find stopped containers and restart them all using a single command.

List Stopped Containers in Docker

Using the filtering functionality of the docker ps command, we can quickly get all the necessary information for the stopped containers:

root@xps:~# **docker ps -a -f status=exited**
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
014a746dbb9d wordpress "docker-entrypoint.s…" 21 hours ago Exited (0) 21 hours ago romantic_fermi
080cf6412ac4 hello-world "/hello" 3 days ago Exited (0) 3 days ago modest_mestorf

Since we want to restart of these containers, we'll probably need to pass their docker container IDs to another command, like docker start.

Hence the command above should be run with the -q parameter, which skips all the non-essential info and only returns the list of docker containers:

```
root@xps:~# docker ps -a -q -f status=exited
014a746dbb9d
080cf6412ac4
```

Restart all the Stopped Containers in Docker

Now all we have left to do is pass the above command to the docker start, like shown below. One by one, all the container IDs will appear as Docker restarts them:

```
root@xps:~# docker start $(docker ps -a -q -f status=exited)
014a746dbb9d
080cf6412ac4
```

Sure enough, when we do docker ps now, we should see these containers:

```
root@xps:~# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
9e7115e34496 wordpress "docker-entrypoint.s…" 19 hours ago Up 19 hours 127.0.0.1:80->80/tcp, 127.0.0.1:443->443/tcp wordpress
014a746dbb9d wordpress "docker-entrypoint.s…" 21 hours ago Up 2 seconds 80/tcp romantic_fermi
c397a72fbd58 mariadb:latest "docker-entrypoint.s…" 21 hours ago Up 21 hours 3306/tcp db
```

I can see the 014a746dbb9d container, but the other one is not running. Want to know why? It’s because this was a Hello, world Docker container – it’s not mean to stay running in background. Instead, it shows Hello, world and exits. It’s usually run like this:

```
root@xps:~# docker run hello-world
```

```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(amd64)
3. The Docker daemon created a new container from that image which runs the
executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/


That's it for today. Enjoy!
```

See Also

- Docker
- Docker – Stop Containers
- Docker – List containers
- Migrate Docker container
- Remove Unused Volumes in Docker
- Inspect Docker containers