Start Docker Containers Automatically

- 3 MINS

JAVA DOCKER FIELD NOTES ENGLISH

Starting your Docker containers automatically using `systemd`.

Introduction

After your Docker containers are set up and running, you might need to be able to start some of them automatically on a reboot or a crash. There are several ways of getting this done.

One of them is to use restart policies provided by Docker. They can be set to control whether your containers start automatically when they exit, or when Docker restarts.

Alternatively, you can use a process manager such as `upstart`, `systemd`, or `supervisor` instead. In this post, I want to show you how it is done with `systemd`.

Create the Service File

To create a service file that will be used by `systemd` (`systemctl` command), we will first need to get your container name. This can be done by running the following command in your shell:

```
$ docker ps -a
```

The output will look something like this. Select the right container from the list, and note its name in the last column. In this example, we will be using `mywiki` container.

```
CONTAINER ID        IMAGE                          COMMAND                  CREATED
573193cf1d5e        hypriot/rpi-busybox-httpd      "/bin/busybox http..."   2 days ago
e85753d57a67        easypi/dokuwiki-arm            "/bin/sh -c 'php-f..."   1 days ago
```

Now, we will need to create a file (choose an appropriate file name for the service):

```
$ sudo nano /etc/systemd/system/docker-dokuwiki.service
```

Paste the following into the file. Set a proper `Description`, and make sure to update the container name in `ExecStart` and `ExecStop`:

```
[Unit]
Description=DokuWiki Container
Requires=docker.service
After=docker.service

[Service]
Restart=always
ExecStart=/usr/bin/docker start -a mywiki
ExecStop=/usr/bin/docker stop -t 2 mywiki

[Install]
WantedBy=local.target
```

A couple of notes about the script above:

1. This file is called a unit file for `systemd`.
2. Make sure you don't have any extra line brakes within the sections, like `Unit`, or `Service`.
3. The `-a` option in the Docker command for `ExecStart` makes sure it is running in attached mode, i.e., attaching STDOUT/STDERR and forwarding signals.
4. The `-t` option in the Docker command for `ExecStop` specifies seconds to wait for it to stop before killing the container.

Activate the Service

Before we can activate the service we have created, we need to reload the unit file. You will also need to run this command anytime you do any modifications to the unit files:

```
$ sudo systemctl daemon-reload
```

To activate the service run the following commands (*remember to change the service name*):

```
$ sudo systemctl start docker-dokuwiki.service
$ sudo systemctl enable docker-dokuwiki.service
```

To disable the service run the following commands (*remember to change the service name*):

```
$ sudo systemctl stop docker-dokuwiki.service
$ sudo systemctl disable docker-dokuwiki.service
```

Changes will come to effect on a reboot:

```
$ sudo reboot
```

Now you should have a container that will start on a server reboot, Docker restart, or a crash. *Congratulations!*

As a next step, you might want to look at (external documentation links):

- Adding some more parameters to the unit file.
- Available `docker start` options.
- Available `docker start` options.

« Command Line Tools for Your Java Projects

Personal Data Security »