Creating Containers

The **`docker run`** command creates a new container from an image and starts a process inside the new container. If the container image is not available, this command also tries to download it:

```
$ docker run rhscl/httpd-24-rhel7
Unable to find image 'rhscl/httpd-24-rhel7:latest' locally
Trying to pull repository infrastructure.lab.example.com:5000/r
hscl/httpd-24-rhel7 ...
latest: Pulling from infrastructure.lab.example.com:5000/rhscl/
httpd-24-rhel7
3436c67883ad: Pull complete
c85416a3d375: Pull complete
aa724488ad02: Pull complete
b70312066a34: Pull complete
Digest: sha256:6c92ae9f2cd45923a6e8013282028263529a7d29cd1a3477
4abf3ee5614fec5e
...
[Thu Mar 30 13:58:16.375095 2017] [core:notice] [pid 1] AH0009
4: Command line: 'httpd -D FOREGROUND'
$ ^C
```

Whatever output **`docker run`** shows is generated by the process inside the container, which is just a regular process from the host OS perspective. Killing that process stops the container. In the previous output sample, the container was started with a noninteractive process, and stopping that process with **Ctrl**+**C** (**`SIGINT`**) also stops the container.

The management docker commands require an ID or a name. The **`docker run`** command generates a random ID and a random name that are unique. The **`docker ps`** command is responsible for displaying these attributes:

```
$ docker ps
CONTAINER ID        IMAGE                    COMMAND
CREATED             STATUS                   PORTS
NAMES
347c9aad6049 (1)       rhscl/httpd-24-rhel7   "httpd -D FOREGROU
ND"     31 seconds ago      Up 30 seconds           80/tcp
focused_fermat (2)
```

(1) This ID is generated automatically and must be unique.

(2) This name can be generated automatically or manually specified.

If desired, the container name can be explicitly defined. The **`--name`** option is responsible for defining the container name:

```
$ docker run --name my-httpd-container do180/httpd
```

Important

The name must be unique. An error is thrown if another container has the same name, including containers that are stopped.

Another important option is to run the container as a daemon, running the containerized process in the background. The **`-d`** option is responsible for running in detached mode. Using this option, the container ID is displayed on the screen:

```
$ docker run --name my-httpd-container -d do180/httpd
77d4b7b8ed1fd57449163bcb0b78d205e70d2314273263ab941c0c371ad5641
2
```

The container image itself specifies the command to run to start the containerized process, but a different one can be specified after the container image name in **`docker run`**:

```
$ docker run do180/httpd ls /tmp
anaconda-post.log
ks-script-1j4CXN
yum.log
```

The specified command must exist inside the container image.

> **Note**
>
> Since a specified command was provided in the previous example, the HTTPD service does not start.

Sometimes it is desired to run a container executing a Bash shell. This can be achieved with:

```
$ docker run --name my-httpd-container -it do180/httpd /bin/bash
 bash-4.2#
```

Options **`-t`** and **`-i`** are usually needed for interactive text-based programs, so they get a proper terminal, but not for background daemons.