How to Access a Container in Docker?

written by sysadmin | 17 May 2025

After you install Docker and learn some basic Docker commands to set up a container, this article will explain how to access a container in Docker.

Problem

How to access a container in Docker?

Solution

There are 2 ways to access a container in Docker:

A. Via CLI

If you want to access a container in Docker, use the format below:

```
docker exec -it container_id/container_name shell
```

where the **-i** option is interactive to keep the input active, the **-t** option is an argument for pseudo -tty (terminal access) allocation, and **shell** is the program contained in the container and it can be different to be different to the code used in the container likes bash or sh shell, but for more details, please look at the documentation of each image). For example, there is an nginx container that is running, and if you want to access the nginx container, you can use the command below:

```
docker exec -it nginx bash
```

After that, you should be able to access the container as shown below:

```
sysadmin@docker:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS     NAMES
e6d61413d2af   nginx     "/docker-entrypoint.…"   10 minutes ago   Up 10 minutes   80/tcp    nginx
sysadmin@docker:~$
sysadmin@docker:~$ docker exec -it nginx bash
root@e6d61413d2af:/#
root@e6d61413d2af:/# ls
bin   dev                  docker-entrypoint.sh  home  lib64  mnt  proc  run   srv  tmp  var
boot  docker-entrypoint.d  etc                   lib   media  opt  root  sbin  sys  usr
root@e6d61413d2af:/#
```

Access into the container

If you want the results of the command in the container to be shown on the server, then use the format below:

```
docker exec container_name/container_id bash -c "your-linux-commands"
```

For example, use the command below:

```
docker exec webapp1 bash -c "ls -al /bin/sync; echo; ls -al /usr"
```

```
sysadmin@docker:~$ docker exec nginx bash -c "ls -al /bin/sync; echo; ls -al /usr"
-rwxr-xr-x 1 root root 39824 Sep 20  2022 /bin/sync

total 48
drwxr-xr-x 1 root root 4096 Apr  7 00:00 .
drwxr-xr-x 1 root root 4096 Apr 15 04:10 ..
drwxr-xr-x 1 root root 4096 Apr  8 01:46 bin
drwxr-xr-x 2 root root 4096 Mar  7 17:30 games
drwxr-xr-x 2 root root 4096 Mar  7 17:30 include
drwxr-xr-x 1 root root 4096 Apr  8 01:46 lib
drwxr-xr-x 2 root root 4096 Apr  7 00:00 lib64
drwxr-xr-x 4 root root 4096 Apr  7 00:00 libexec
drwxr-xr-x 1 root root 4096 Apr  7 00:00 local
drwxr-xr-x 1 root root 4096 Apr  8 01:46 sbin
drwxr-xr-x 1 root root 4096 Apr  8 01:46 share
drwxr-xr-x 2 root root 4096 Mar  7 17:30 src
sysadmin@docker:~$
```

Run some Linux commands in the container

Or you can also use the format below if you only run a command in the container:

```
docker exec container_name/container_id linux-command
```

For example, use the command below:

```
docker exec nginx ls -al /usr
```

```
sysadmin@docker:~$ docker exec nginx ls -al /usr
total 48
drwxr-xr-x 1 root root 4096 Apr  7 00:00 .
drwxr-xr-x 1 root root 4096 Apr 15 04:10 ..
drwxr-xr-x 1 root root 4096 Apr  8 01:46 bin
drwxr-xr-x 2 root root 4096 Mar  7 17:30 games
drwxr-xr-x 2 root root 4096 Mar  7 17:30 include
drwxr-xr-x 1 root root 4096 Apr  8 01:46 lib
drwxr-xr-x 2 root root 4096 Apr  7 00:00 lib64
drwxr-xr-x 4 root root 4096 Apr  7 00:00 libexec
drwxr-xr-x 1 root root 4096 Apr  7 00:00 local
drwxr-xr-x 1 root root 4096 Apr  8 01:46 sbin
drwxr-xr-x 1 root root 4096 Apr  8 01:46 share
drwxr-xr-x 2 root root 4096 Mar  7 17:30 src
sysadmin@docker:~$
```

Run a command in the container

B. Via website

You can access a container through the website, but this method only produces applications that run in the container on the website and do not run commands in the container. Usually, these containers use a web server image such as Apache or Nginx, or applications made by developers. If you want to run these applications and access the application in the browser, use the format below:

```
docker container run -d --name container_name -p port_server:port_container image_name:tag
```

For example, you want to run an nginx container whose application can be seen by using port 8080 on the server, then use the command below:

```
docker container run -d --name webapp1 -p 8080:80 nginx
```

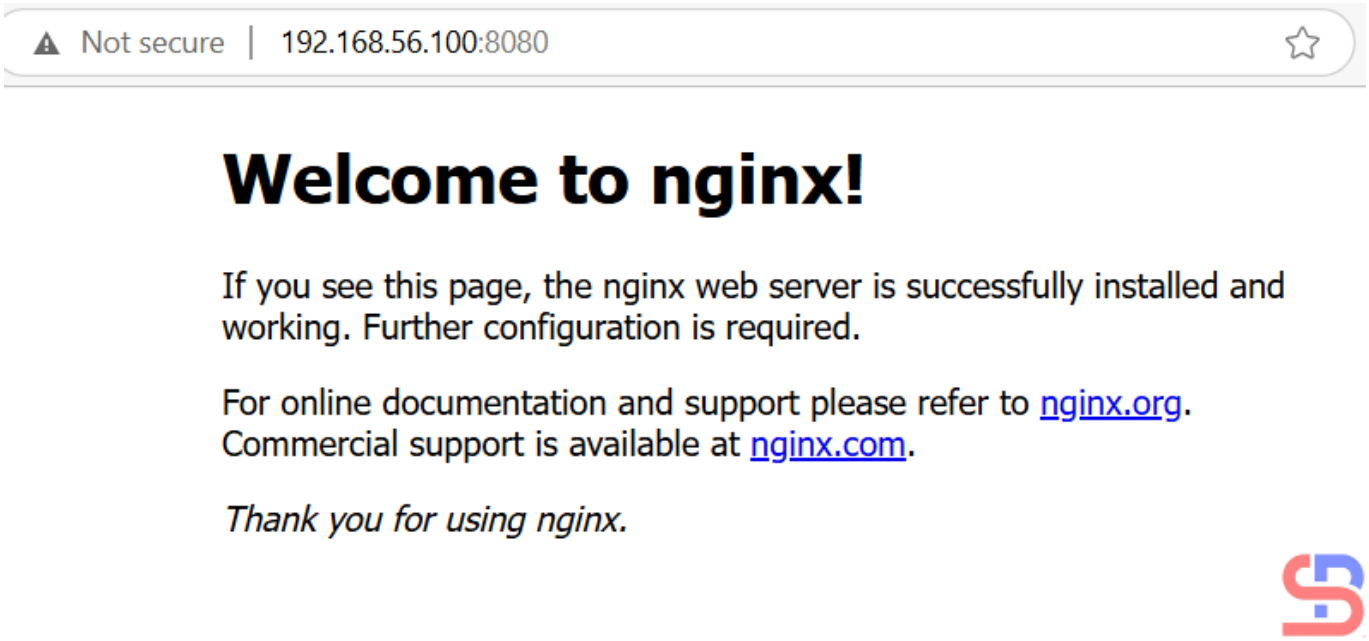
```
sysadmin@docker:~$ docker container run -d --name webapp1 -p 8080:80 nginx
2a4eadafffcd4899dce3201f8e110489e77d5c0f6d4a9bac8af91f48a06adf35
sysadmin@docker:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                                   NAMES
2a4eadafffcd   nginx     "/docker-entrypoint.…"   5 seconds ago    Up 4 seconds    0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   webapp1
e6d61413d2af   nginx     "/docker-entrypoint.…"   42 minutes ago   Up 42 minutes   80/tcp                                  nginx
sysadmin@docker:~$
```

Run the container with the accessed port

Open your browser and type the url below:

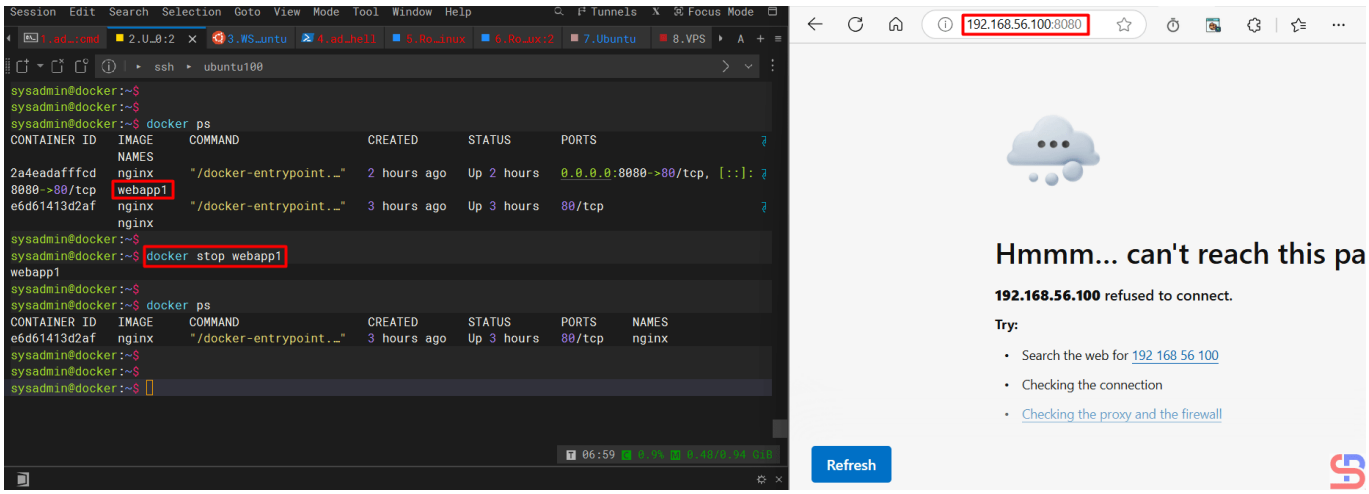
http://your_ip_server:8080

There should be a display as below:



Display the application on the website

If you stop the container, then the application cannot be accessed through a browser as below:



Turn off the container

Note

You can also access a database installed in a container using the commands in this article.

References

docs.docker.com
spacelift.io
youtube.com