

# How To Share Data Between the Docker Container and the Host



Updated July 11, 2018 © 249.9k DOCKER UBUNTU UBUNTU 18.04

By: Melissa Anderson

## Introduction

In general, Docker containers are ephemeral, running just as long as it takes for the command issued in the container to complete. By default, any data created inside the container is only available from within the container and only while the container is running.

Docker volumes can be used to share files between a host system and the Docker container. For example, let's say you wanted to use the official Docker Nginx image and keep a permanent copy of Nginx's log files to analyze later. By default, the `nginx` Docker image will log to the `/var/log/nginx` directory *inside* the Docker Nginx container. Normally it's not reachable from the host filesystem.

In this tutorial, we'll explore how to make data from inside the container accessible on the host machine.

## Prerequisites

To follow this article, you will need an Ubuntu 18.04 server with the following:

- A non-root user with sudo privileges, following the [Initial Server Setup with Ubuntu 18.04](#) guide.
- Docker installed with the instructions from **Step 1** and **Step 2** of [How To Install and Use Docker on Ubuntu 18.04](#).

If you're new to Docker, [The Docker Ecosystem](#) series provides a detailed overview of key concepts.

**Note:** Even though the Prerequisites give instructions for installing Docker on Ubuntu 18.04, the `docker` commands for Docker data volumes in this article should work on other operating systems as long as Docker is installed.

## Step 1 — Bindmounting a Volume

The following command will create a directory called `nginxlogs` in your current user's home directory and bindmount it to `/var/log/nginx` in the container:

```
$ docker run --name=nginx -d -v ~/nginxlogs:/var/log/nginx -p 5000:80 nginx
```

Let's take a moment to examine this command in detail:

- `--name=nginx` names the container so we can refer to it more easily.
- `-d` detaches the process and runs it in the background. Otherwise, we would just be watching an empty Nginx prompt and wouldn't be able to use this terminal until we killed Nginx.
- `-v ~/nginxlogs:/var/log/nginx` sets up a bindmount volume that links the `/var/log/nginx` directory from inside the Nginx container to the `~/nginxlogs` directory on the host machine. Docker uses a `:` to split the host's path from the container path, and the host path always comes first.
- `-p 5000:80` sets up a port forward. The Nginx container is listening on port `80` by default. This flag maps the container's port `80` to port `5000` on the host system.
- `nginx` specifies that the container should be built from the Nginx image, which issues the command `nginx -g "daemon off"` to start Nginx.

**Note:** The `-v` flag is very flexible. It can bindmount or name a volume with just a slight adjustment in syntax. If the first argument begins with a `/` or `~/`, you're creating a bindmount. Remove that, and you're naming the volume.

- `-v /path:/path/in/container` mounts the host directory, `/path` at the `/path/in/container`
- `-v path:/path/in/container` creates a volume named `path` with no relationship to the host.

For more on named volumes, see [How to Share Data Between Docker Containers](#)

## Step 2 — Accessing Data on the Host

We now have a copy of Nginx running inside a Docker container on our machine, and our host machine's port `5000` maps directly to that copy of Nginx's port `80`.

Load the address in a web browser, using the IP address or hostname of your server and the port number: `http://your_server_ip:5000`. You should see:

# Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](http://nginx.org).  
Commercial support is available at [nginx.com](http://nginx.com).

*Thank you for using nginx.*

More interestingly, if we look in the `~/nginxlogs` directory on the host, we'll see the `access.log` created by the container's `nginx` which will show our request:

```
$ cat ~/nginxlogs/access.log
```

This should display something like:

## Output

```
203.0.113.0 - - [11/Jul/2018:00:59:11 +0000] "GET / HTTP/1.1" 200 612 "-"  
"Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36  
(KHTML, like Gecko) Chrome/54.0.2840.99 Safari/537.36" "-"
```

If you make any changes to the `~/nginxlogs` folder, you'll be able to see them from inside the Docker container in real time as well.

## Conclusion

In this tutorial we demonstrated how to create a Docker data volume to share information between a container and the host file system. This is helpful in development environments, where it is necessary to have access to logs for debugging. To learn more about sharing persistent data between containers, take a look at [How To Share Data between Docker Containers](#).

By: Melissa Anderson

♡ Upvote (23)

📄 Subscribe

🔗 Share



We just made it easier for you to deploy faster.

[TRY FREE](#)

Related Tutorials

[How To Install Docker Compose on Ubuntu 18.04](#)

[How To Remove Docker Images, Containers, and Volumes](#)

[Naming Docker Containers: 3 Tips for Beginners](#)

[How to Manually Set Up a Prisma Server on Ubuntu 18.04](#)


[How To Use Traefik as a Reverse Proxy for Docker Containers on Debian 9](#)

6 Comments

Leave a comment...

[Log In to Comment](#)

 [henscu](#) December 5, 2016

1  Excellent posts on Docker, Melissa. The way you break each command down in detail and explain the flags you use makes it easy to understand and learn.

---

^ [MelissaAnderson](#) MOD February 22, 2017

1 Thank you for the feedback! I'm glad it was helpful.

---

^ [inanc](#) May 14, 2017

0 It doesn't work if the host is on the digitalocean and the local directory is on my laptop.

---

^ [greggman](#) January 4, 2018

0 If I ran the container without -v do I need to recreate the container after if I want to add the share?

---

^ [vartanjan7](#) February 3, 2018

0 <https://kitematic.com>

All you want

---

^ [testjpatel](#) February 21, 2018

0 Excellent Post. Good post if we have any doubt about how docker is running internally. Keep it up.

Regards

Vipul



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



