Docker Volumes: How to Create & Get Started

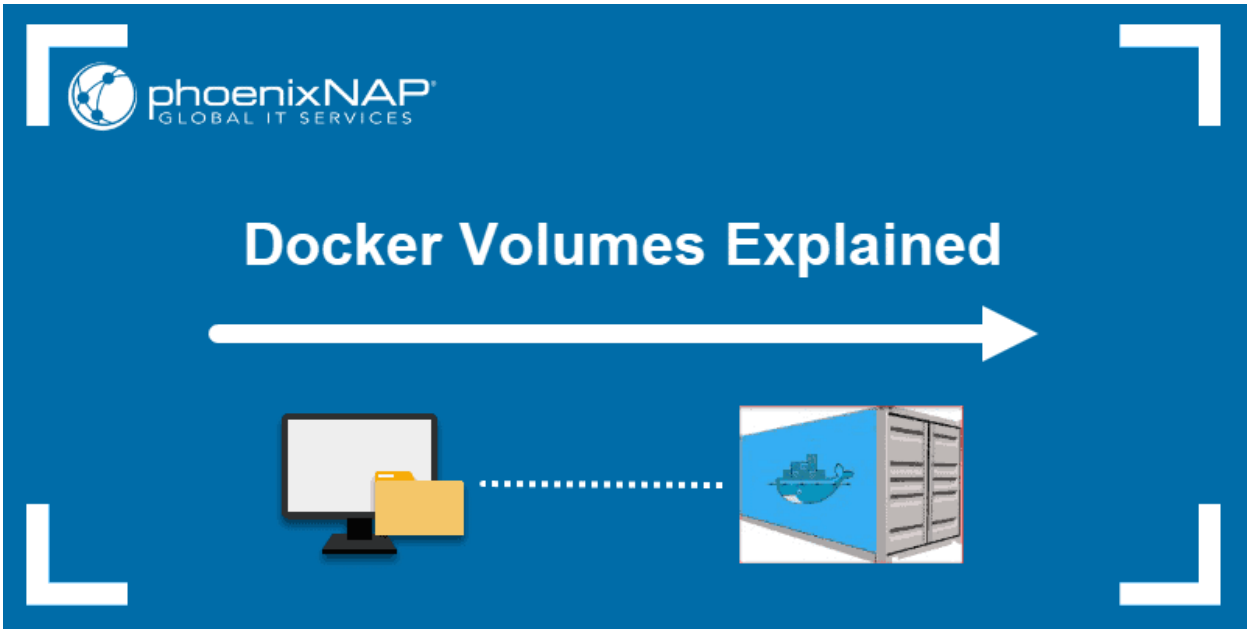
July 27, 2020

DOCKER VIRTUALIZATION

Introduction

Docker volumes are widely used and useful tools for ensuring data persistence while working in containers. They are a better alternative than compiling additional writable layers, which increase Docker image size.

In this tutorial, learn how to use Docker Volumes with practical examples.



What are Docker Volumes?

Docker volumes are file systems mounted on Docker containers to preserve data generated by the running container.

The volumes are stored on the host, independent of the container life cycle. This allows users to back up data and share file systems between containers easily.

Getting Started With Docker Volumes

There are different ways to mount a Docker volume while launching a container. Users can decide between the `-v` and the `--mount` flags, which are added to the `docker run` command.

This article shows examples of both flags in use.

How to Create a Docker Volume

To create a Docker Volume use the command:

```
docker volume create [volume_name]
```

Docker automatically creates a directory for the volume on the host under the **/var/lib/docker/volume/** path. You can now mount this volume on a container, ensuring data persistence and data sharing among multiple containers.

For example, to create a volume under the name **data**, you would run the command:

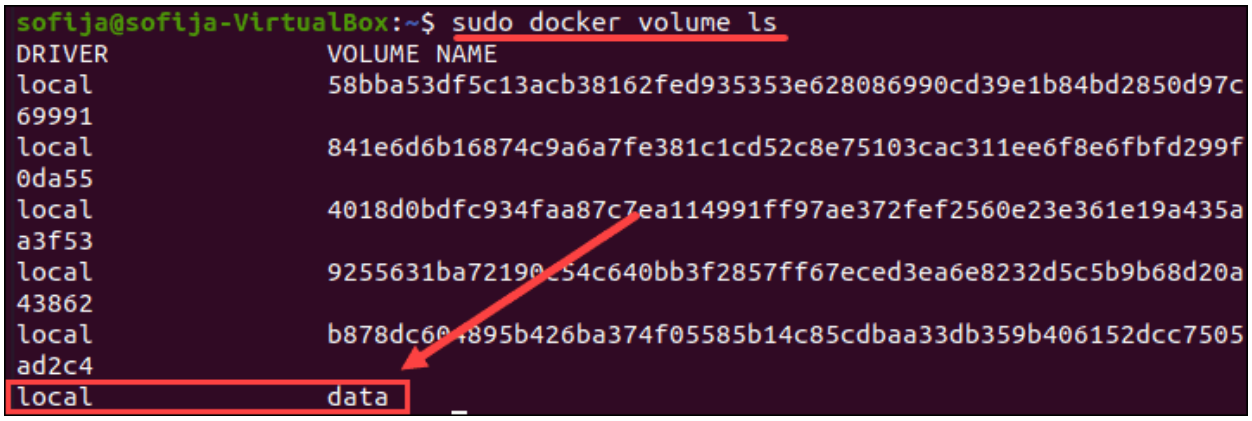
```
docker volume create data
```

List Docker Volumes

To verify you have successfully created a Docker volume, prompt Docker to list all available volumes with:

```
docker volume list
```

The output displays a list of volumes, specifying their location (**DRIVER**) and their **VOLUME NAME**. In the image below, you can see the volume **data** created in the previous section.

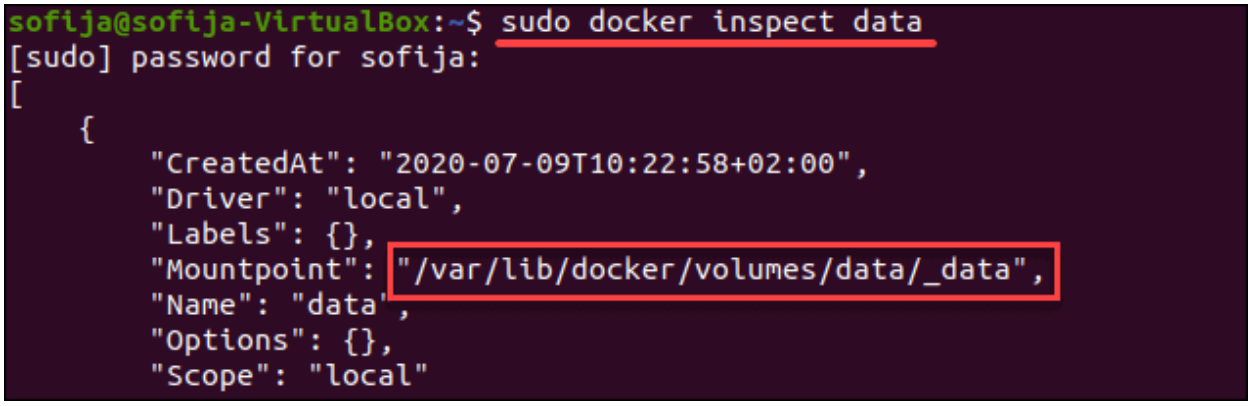


Inspecting Docker Volumes

To see more information about a Docker volume, use the **`inspect`** command:

```
docker volume inspect [volume_name]
```

It lists details of a volume, including its location on the host file (**Mountpoint**). Everything stored within the data volume can also be found in the directory listed under the mountpoint path.



Mounting a Data Volume

To mount a data volume to a container add the **`--mount`** flag to the **`docker run`** command. It adds the volume to the specified container, where it stores the data produced inside the virtual environment.

To run a container and mount a data volume to it, follow the basic syntax:

```
docker run --mount source=[volume_name],destination=[path_i
n_container] [docker_image]
```

Replace **`[path_in_container]`** with the path where you want to place the data volume in the container. Everything stored in that directory automatically gets saved on the data volume on the host as well.

For example, to launch an Ubuntu container and mount the **data** volume to it, run:

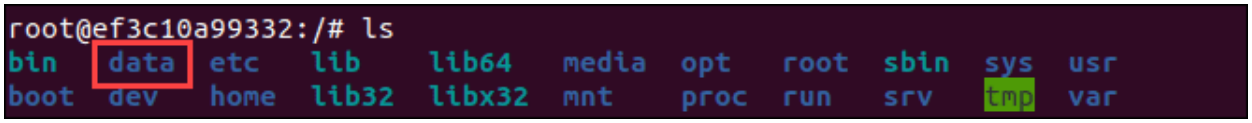
```
docker run -it --name=example1 --mount source=data,destinat
ion=/data ubuntu
```

The command instructs Docker to run a container in interactive mode (**`-it`**) from the Ubuntu image, under the name **`example1`**, while mounting the volume data in the **`/data`** directory inside the container.

Then, check to verify the volume was successfully mounted by listing the content of the container:

```
ls
```

Find the Docker volume under the name defined while launching the container. In this example, it is **data**.



Copying Files Between Containers From a Shared Volume

Let's see how Docker volumes allow you to share files between containers.

To do so, we use the volume and container created in the previous section. This included running the commands:

- **`docker volume create data`**
- **`docker run -it --name=example1 --mount source=data,destination=/data ubuntu`**

1. Once you have switched to the container command prompt, move to the data volume directory:

```
cd data
```

2. Create an empty sample file using the touch command:

```
touch sample1.txt
```

3. Now, exit the container:

```
exit
```

4. Then, launch a new container **example2** with the same data volume:

```
docker run -it --name=example2 --mount source=data,destinat
ion=/data ubuntu
```

5. List the content of the container. You should find the data directory, as in **example1**:

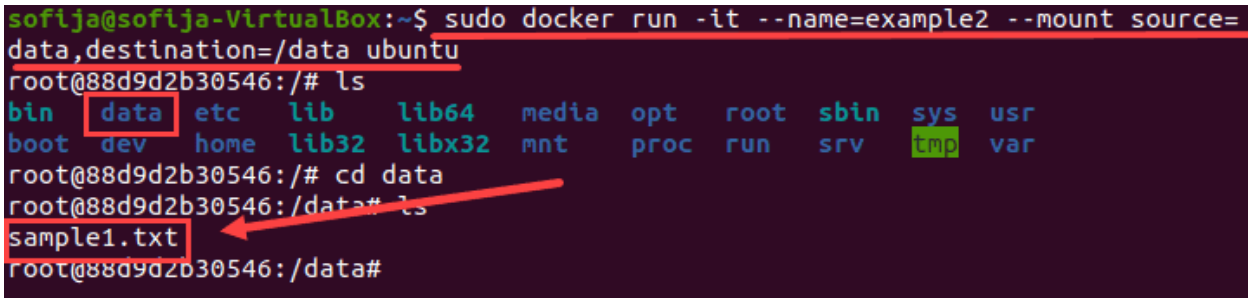
```
ls
```

6. Move to the data directory and list the content of it:

```
cd data
```

```
ls
```

The output should list the **sample1.txt** file you created in the previous container (**example1**).



Note: You might be interested in finding out how Kubernetes persistent volumes work.

Mounting a Host Directory as a Data volume

You can also mount an existing directory from the host machine to a container. This type of volume is called **Host Volumes**.

You can mount host volumes by using the `-v` flag and specifying the name of the host directory.

Everything within the host directory is then available in the container. What's more, all the data generated inside the container and placed in the data volume is safely stored on the host directory.

The basic syntax for mounting a host directory is:

```
docker run -v "$(pwd)":[volume_name] [docker_image]
```

The `"$(pwd)"` attribute instructs Docker to mount the directory the user is currently in.

The following example outlines how this is done.

1. First, create a sample directory on the host under the name **tmp** and move into it:

```
mkdir tmp
```

```
cd tmp
```

2. Once inside the directory, create a test file to see whether it will be available from the container:

```
touch file.txt
```

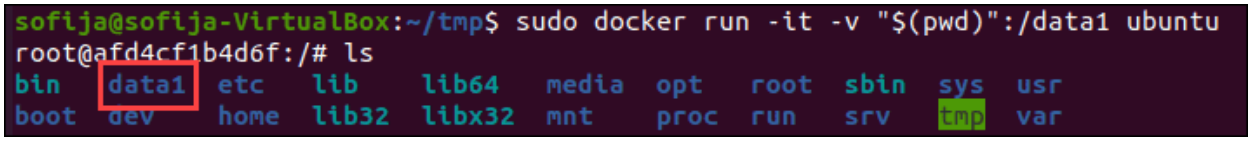
2. Then, use the **`docker run`** command to launch an Ubuntu container with the host directory attached to it:

```
docker run -it -v "$(pwd)":/data1 ubuntu
```

This launches the container in interactive mode and mounts a volume under the name **data1**.

3. List the content of the container and verify there is a **data1** directory:

```
ls
```



4. Open the mounted directory and list the content. The output should display the file you created on the host:

```
cd data1
```

```
ls
```

Volume Permission and Ownership

Volume permissions van be changed by configuring the ownership within the Dockerfile.

Use the **`RUN`** instruction and the **`chown`** command to set the Docker volume permission. Make sure this instruction precedes the line defining the **`VOLUME`**.

Alternatively, you can change the ownership of the directory used as the host volume.

For instructions on how to use the **`chown`** command, refer to Linux File Permission Tutorial: How To Check And Change Permissions.

How to Delete Docker Volumes

To delete a Docker volume, you need to specify its name.

The basic syntax for removing a Docker volume in the command line is:

```
docker volume rm [volume_name]
```

Docker removes volumes only if they are not in use at the moment. If there is a container with the specified volume, it responds with an error. To proceed, stop and remove the container and then rerun the **`docker volume rm`** command.

Note: If you don't know the name of the volume, use the **`docker volume ls`** command.

How to Delete All Volumes at Once

To delete all unused Docker volumes with a single command:

```
docker volume prune
```

The output warns you it will remove all local volumes not used by at least one container. Press **y** to continue.

Conclusion

After reading this article, you should have a better understanding of how Docker volumes work, as well as how to mount volumes to containers.

Was this article helpful? Yes No

Sofija Simic

Sofija Simic is an experienced Technical Writer. Alongside her educational background in teaching and writing, she has had a lifelong passion for information technology. She is committed to unscrambling confusing IT concepts and streamlining intricate software installations.

Next you should read