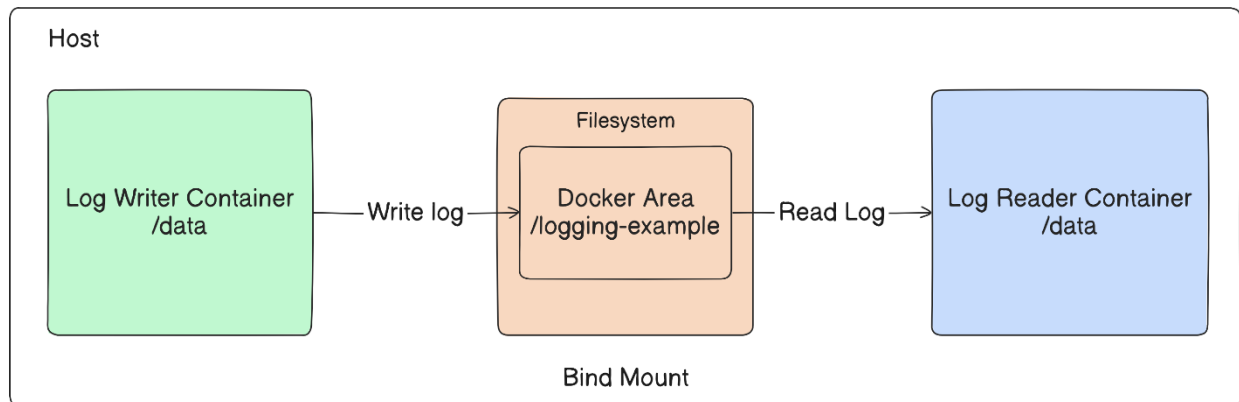


Log Sharing Between Containers

This scenario demonstrates the process of sharing files between multiple Docker containers using two methods: bind mounts and Docker volumes. The goal is to showcase the benefits of Docker volumes over bind mounts, and to illustrate the flexibility and ease of use provided by anonymous volumes and the `--volumes-from` flag.

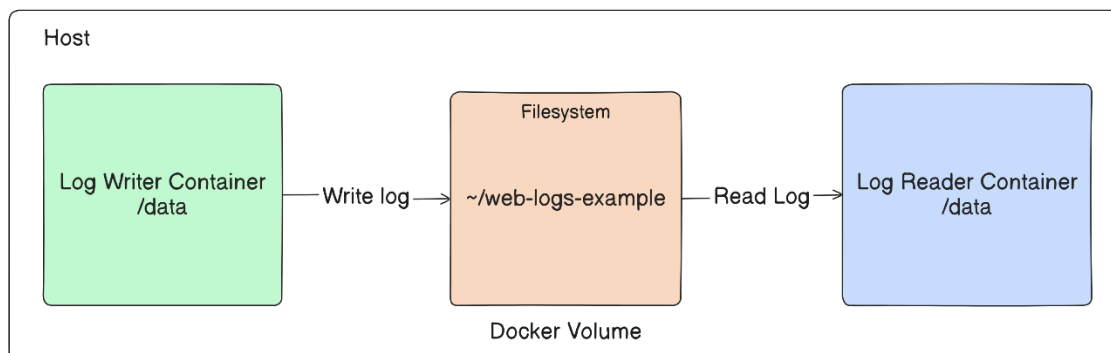
Scenario Overview

Bind Mount Example: We start by setting up a directory on the host and bind-mounting it into two containers—one for writing log files and one for reading them.



Docker Volume Example: We then perform the same operation using Docker volumes, eliminating host-specific dependencies.

Anonymous Volumes and `--volumes-from` Flag: Finally, we demonstrate using anonymous volumes and the `--volumes-from` flag to dynamically share volumes between multiple containers.



Initial setup:

First we will create a docker image that performs simple file writing application in a Docker container. Here's how you can create your own:

1. Create a Dockerfile

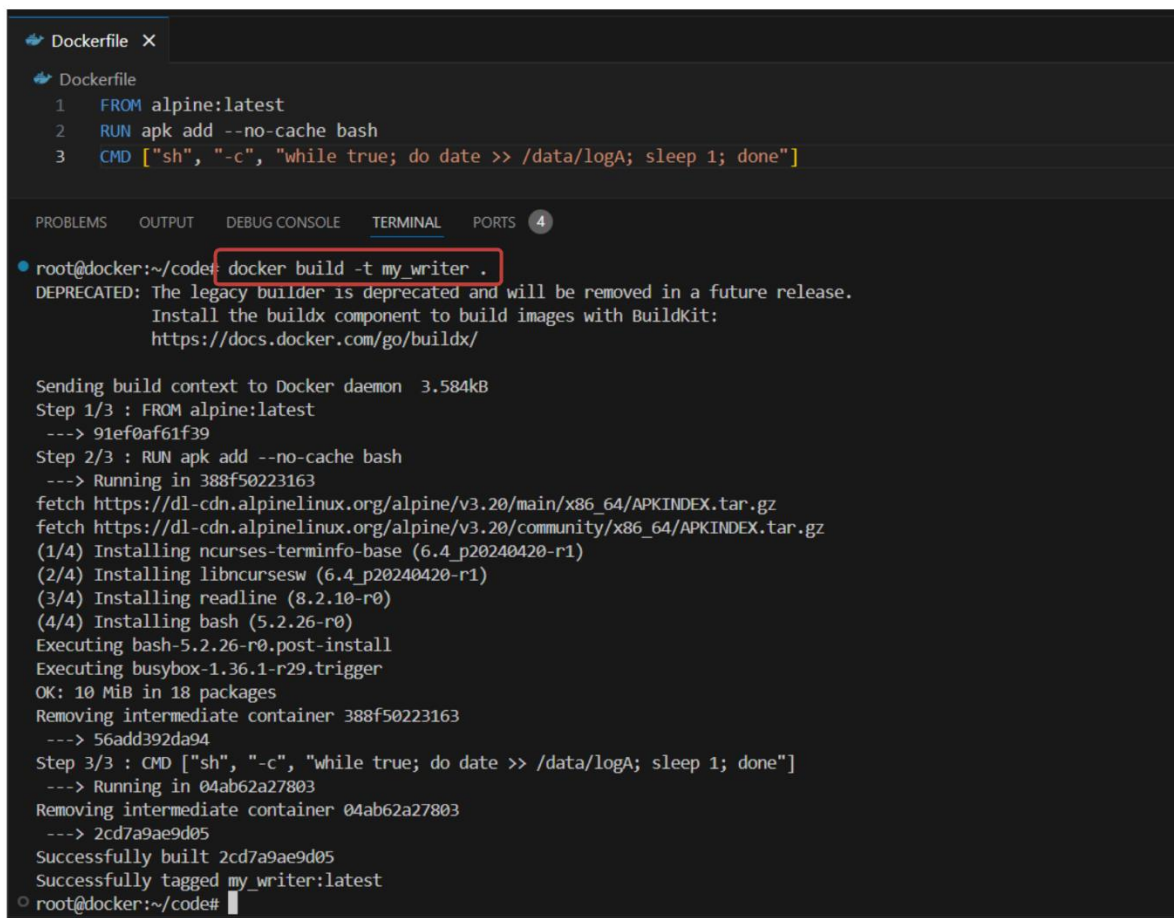
FROM alpine:latest

RUN apk add --no-cache bash

CMD ["sh", "-c", "while true; do date >> /data/logA; sleep 1; done"]

2. Build the Docker image

docker build -t my_writer .



```
Dockerfile
1 FROM alpine:latest
2 RUN apk add --no-cache bash
3 CMD ["sh", "-c", "while true; do date >> /data/logA; sleep 1; done"]

root@docker:~/code# docker build -t my_writer .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/

Sending build context to Docker daemon 3.584kB
Step 1/3 : FROM alpine:latest
--> 91ef0af61f39
Step 2/3 : RUN apk add --no-cache bash
--> Running in 388f50223163
fetch https://dl-cdn.alpinelinux.org/alpine/v3.20/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.20/community/x86_64/APKINDEX.tar.gz
(1/4) Installing ncurses-terminfo-base (6.4_p20240420-r1)
(2/4) Installing libncursesw (6.4_p20240420-r1)
(3/4) Installing readline (8.2.10-r0)
(4/4) Installing bash (5.2.26-r0)
Executing bash-5.2.26-r0.post-install
Executing busybox-1.36.1-r29.trigger
OK: 10 MiB in 18 packages
Removing intermediate container 388f50223163
--> 56add392da94
Step 3/3 : CMD ["sh", "-c", "while true; do date >> /data/logA; sleep 1; done"]
--> Running in 04ab62a27803
Removing intermediate container 04ab62a27803
--> 2cd7a9ae9d05
Successfully built 2cd7a9ae9d05
Successfully tagged my_writer:latest
root@docker:~/code#
```

3. Verify the docker image

docker images

```
root@docker:~/code# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
my_writer	latest	2cd7a9ae9d05	About a minute ago	9.83MB
my_microservice	latest	437c233a44ae	20 minutes ago	125MB
<none>	<none>	b346654d7431	23 minutes ago	125MB
<none>	<none>	56032acb9c7a	31 minutes ago	125MB
python	3.9-slim	397ed8d31636	2 weeks ago	125MB
alpine	latest	91ef0af61f39	2 weeks ago	7.8MB
cassandra	2.2	d941d5b1ca8d	3 years ago	374MB

```
root@docker:~/code#
```

Log sharing using Bind Mount:

1. Setup a Known Location on Host:

LOG_SRC=~/.web-logs-example

mkdir \${LOG_SRC}

LOG_SRC: This is an environment variable that stores the path to the directory where the logs will be stored.

mkdir \${LOG_SRC}: This command creates a new directory at the path specified by LOG_SRC.

2. Create and Run a Log-Writing Container and use bind mounts to share the log directory:

docker run --name plath -d \

--mount type=bind,src=\${LOG_SRC},dst=/data \

my_writer

```

root@docker:~/code# docker run --name plath -d \
> --mount type=bind,src=${LOG_SRC},dst=/data \
> my_writer
572d1d254166ff6ee91ba7e123fdd5638987b92d1da52482b0c9278b93f3e2d
root@docker:~/code# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
572d1d254166   my_writer     "sh -c 'while true; ..." 6 seconds ago  Up 5 seconds  7000-7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp  plath
022a81418262   cassandra:2.2 "docker-entrypoint.s..." 4 minutes ago  Up 4 minutes  7000-7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp  cass2
root@docker:~/code#

```

3. Create and Run a Log-Reading Container and use bind mounts to share:

```

docker run --rm \
--mount type=bind,src=${LOG_SRC},dst=/data \
alpine:latest \
head /data/logA

```

```

root@7fde7b0b91e1531a:~/code# docker run --rm \
> --mount type=bind,src=${LOG_SRC},dst=/data \
> alpine:latest \
> head /data/logA
Thu Sep 26 21:01:10 UTC 2024
Thu Sep 26 21:01:11 UTC 2024
Thu Sep 26 21:01:12 UTC 2024
Thu Sep 26 21:01:13 UTC 2024
Thu Sep 26 21:01:14 UTC 2024
Thu Sep 26 21:01:15 UTC 2024
Thu Sep 26 21:01:16 UTC 2024
Thu Sep 26 21:01:17 UTC 2024
Thu Sep 26 21:01:18 UTC 2024
Thu Sep 26 21:01:19 UTC 2024
root@7fde7b0b91e1531a:~/code#

```

Explanation:

- **docker run:** This command creates and starts a new container.
- **--name plath:** This names the container "plath".
- **-d:** This flag runs the container in detached mode, meaning it runs in the background.
- **--mount type=bind,src=\${LOG_SRC},dst=/data:** This option specifies a bind mount. It maps the host directory (src) to the container directory (dst).
 - **type=bind:** Indicates the type of mount.

- **src=\${LOG_SRC}**: Source directory on the host.
- **dst=/data**: Destination directory inside the container.
- **--rm**: This flag automatically removes the container when it exits.
- **alpine:latest**: The image used to create the container. Alpine is a lightweight Linux distribution.
- **head /data/logA**: This command reads the top part of the log file.

View Logs from Host:

We can also view the logs directly from the host.

`cat ${LOG_SRC}/logA`

cat: This command displays the contents of the file.

\${LOG_SRC}/logA: Path to the log file on the host.

```
root@7fde7b0b91e1531a:~/code#  
root@7fde7b0b91e1531a:~/code# cat ${LOG_SRC}/logA  
Thu Sep 26 21:01:10 UTC 2024  
Thu Sep 26 21:01:11 UTC 2024  
Thu Sep 26 21:01:12 UTC 2024  
Thu Sep 26 21:01:13 UTC 2024  
Thu Sep 26 21:01:14 UTC 2024  
Thu Sep 26 21:01:15 UTC 2024  
Thu Sep 26 21:01:16 UTC 2024  
Thu Sep 26 21:01:17 UTC 2024  
Thu Sep 26 21:01:18 UTC 2024  
Thu Sep 26 21:01:19 UTC 2024  
Thu Sep 26 21:01:20 UTC 2024
```

Stop the Log-Writing Container:

`docker rm -f plath`

```
root@ubuntu-19r7bi-6c84595885-r9psx:~# docker rm -f plath
plath
root@ubuntu-19r7bi-6c84595885-r9psx:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@ubuntu-19r7bi-6c84595885-r9psx:~#
```

Log sharing using Docker Volume:

Now we will achieve the same result using docker volume.

1. Create Docker Volume:

`docker volume create --driver local logging-example`

`docker volume ls`

```
root@ubuntu-19r7bi-6c84595885-r9psx:~# docker volume create --driver local logging-example
logging-example
root@ubuntu-19r7bi-6c84595885-r9psx:~# docker volume ls
DRIVER    VOLUME NAME
local     logging-example
root@ubuntu-19r7bi-6c84595885-r9psx:~#
```

Explanation:

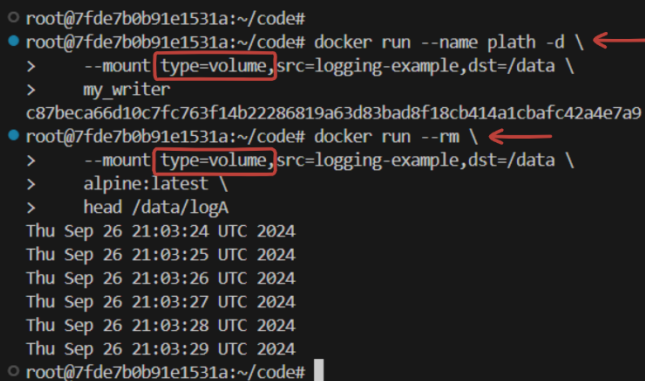
- **docker volume create:** This command creates a new Docker volume.
- **--driver local:** This specifies the volume driver to use. "local" is the default driver.
- **logging-example:** Name of the volume.

2. Create and Run a Log-Writing Container:

```
docker run --name plath -d \  
  --mount type=volume,src=logging-example,dst=/data \  
  my_writer
```

3. Create and Run a Log-Reading Container:

```
docker run --rm \  
  --mount type=volume,src=logging-example,dst=/data \  
  alpine:latest \  
  head /data/logA
```



```
root@7fde7b0b91e1531a:~/code#  
root@7fde7b0b91e1531a:~/code# docker run --name plath -d \  
>   --mount type=volume,src=logging-example,dst=/data \  
>   my_writer  
c87beca66d10c7fc763f14b22286819a63d83bad8f18cb414a1cbafc42a4e7a9  
root@7fde7b0b91e1531a:~/code# docker run --rm \  
>   --mount type=volume,src=logging-example,dst=/data \  
>   alpine:latest \  
>   head /data/logA  
Thu Sep 26 21:03:24 UTC 2024  
Thu Sep 26 21:03:25 UTC 2024  
Thu Sep 26 21:03:26 UTC 2024  
Thu Sep 26 21:03:27 UTC 2024  
Thu Sep 26 21:03:28 UTC 2024  
Thu Sep 26 21:03:29 UTC 2024  
root@7fde7b0b91e1531a:~/code#
```

Explanation:

- **docker run:** This command runs a new container.
- **--mount type=volume,src=logging-example,dst=/data:** This option specifies a volume mount.
 - **type=volume:** Indicates the type of mount.
 - **src=logging-example:** Source volume.
 - **dst=/data:** Destination directory inside the container.

4. View Logs from Host:

`cat /var/lib/docker/volumes/logging-example/_data/logA`

```
root@ubuntu-19r7bi-6c84595885-r9psx:~# cat /var/lib/docker/volumes/logging-example/_data/logA
Sat Jun  8 20:10:55 UTC 2024
Sat Jun  8 20:10:56 UTC 2024
Sat Jun  8 20:10:57 UTC 2024
Sat Jun  8 20:10:58 UTC 2024
Sat Jun  8 20:10:59 UTC 2024
Sat Jun  8 20:11:00 UTC 2024
Sat Jun  8 20:11:01 UTC 2024
Sat Jun  8 20:11:02 UTC 2024
Sat Jun  8 20:11:03 UTC 2024
Sat Jun  8 20:11:04 UTC 2024
Sat Jun  8 20:11:05 UTC 2024
```

5. Stop the Log-Writing Container:

`docker stop plath`

Anonymous Volumes

Now, let's explore using anonymous volumes and the `--volumes-from` flag.

1. Create Containers with Anonymous Volumes:

```
docker run --name fowler \
  --mount type=volume,dst=/library/PoEAA \
  --mount type=bind,src=/tmp,dst=/library/DSL \
  alpine:latest \
  echo "Fowler collection created."

docker run --name knuth \
  --mount type=volume,dst=/library/TAoCP.vol1 \
  --mount type=volume,dst=/library/TAoCP.vol2 \
  --mount type=volume,dst=/library/TAoCP.vol3 \
  --mount type=volume,dst=/library/TAoCP.vol4.a \
```



```
alpine:latest \
```

```
echo "Knuth collection created"
```

```
root@ubuntu-19r7bi-6c84595885-r9psx:~# docker run --name fowler \
--mount type=volume,dst=/library/PoEAA \
--mount type=bind,src=/tmp,dst=/library/DSL \
alpine:latest \
echo "Fowler collection created."

docker run --name knuth \
--mount type=volume,dst=/library/TAoCP.vol1 \
--mount type=volume,dst=/library/TAoCP.vol2 \
--mount type=volume,dst=/library/TAoCP.vol3 \
--mount type=volume,dst=/library/TAoCP.vol4.a \
alpine:latest \
echo "Knuth collection created"
Fowler collection created.
Knuth collection created
root@ubuntu-19r7bi-6c84595885-r9psx:~#
```

Explanation:

- **--mount type=volume,dst=/library/PoEAA:** Creates an anonymous volume mounted at /library/PoEAA.
- **--mount type=bind,src=/tmp,dst=/library/DSL:** Creates a bind mount from /tmp on the host to /library/DSL in the container.

2. Share Volumes with Another Container

Create a container that uses the volumes from the previous containers.

```
docker run --name reader \
```

```
--volumes-from fowler \
```

```
--volumes-from knuth \
```

```
alpine:latest ls -l /library/
```

```

root@ubuntu-19r7bi-6c84595885-r9psx:~# docker run --name reader \ ←
--volumes-from fowler \
--volumes-from knuth \
alpine:latest ls -l /library/
total 0
drwxrwxrwt    1 root    root    6 Jun  8 20:14 DSL
drwxr-xr-x    2 root    root    6 Jun  8 20:25 PoEAA
drwxr-xr-x    2 root    root    6 Jun  8 20:25 TAoCP.vol1
drwxr-xr-x    2 root    root    6 Jun  8 20:25 TAoCP.vol2
drwxr-xr-x    2 root    root    6 Jun  8 20:25 TAoCP.vol3
drwxr-xr-x    2 root    root    6 Jun  8 20:25 TAoCP.vol4.a
root@ubuntu-19r7bi-6c84595885-r9psx:~#

```

Explanation:

- **--volumes-from fowler:** Copies the mount points from the container "fowler".
- **--volumes-from knuth:** Copies the mount points from the container "knuth".
- **ls -l /library/:** Lists the contents of the /library/ directory.

3. Inspect Volumes of the New Container:

Check the volumes of the new container.

```
docker inspect --format "{{json .Mounts}}" reader | jq .
```

Expected Output: (Make sure to install jq command-line tool if you want to format the JSON output in a prettier way)

```
sudo apt-get update
```

```
sudo apt install jq -y
```

```

○ root@7fde7b0b91e1531a:~/code#
● root@7fde7b0b91e1531a:~/code# docker inspect --format "{{json .Mounts}}" reader | jq .
[
  {
    "Type": "volume",
    "Name": "d613364c45c879007d01bae312520887489719d02b369da10b2377e3ad44675b",
    "Source": "/var/lib/docker/volumes/d613364c45c879007d01bae312520887489719d02b369da10b2377e3ad44675b/_data",
    "Destination": "/library/PoEAA",
    "Driver": "local",
    "Mode": "",
    "RW": true,
    "Propagation": ""
  },
  {
    "Type": "bind",
    "Source": "/tmp",
    "Destination": "/library/DSL",
    "Mode": "",
    "RW": true,
    "Propagation": "rprivate"
  },
  {
    "Type": "volume",
    "Name": "8da7206a551e83e3e7922270d4f1288a1c0d29dd24345a6c10c8a8b811535a5a",
    "Source": "/var/lib/docker/volumes/8da7206a551e83e3e7922270d4f1288a1c0d29dd24345a6c10c8a8b811535a5a/_data",
    "Destination": "/library/TAoCP.vol1",
    "Driver": "local",
    "Mode": "",
    "RW": true,
    "Propagation": ""
  },
  {
    "Type": "volume",
    "Name": "f392f1e6d24369b1bb0dc1de67cd7a41c8b026b3f1d23f115fec8d294d211f84",
    "Source": "/var/lib/docker/volumes/f392f1e6d24369b1bb0dc1de67cd7a41c8b026b3f1d23f115fec8d294d211f84/_data",
    "Destination": "/library/TAoCP.vol2",
    "Driver": "local",
    "Mode": "",
    "RW": true,
    "Propagation": ""
  },
  {
    "Type": "volume",
    "Name": "1d836acae0581c754de87e0bccf481a1b89f19bd4e4200adbab27f0f4908782",
    "Source": "/var/lib/docker/volumes/1d836acae0581c754de87e0bccf481a1b89f19bd4e4200adbab27f0f4908782/_data",
    "Destination": "/library/TAoCP.vol3",
    "Driver": "local",
    "Mode": "",
    "RW": true,
    "Propagation": ""
  },
  {
    "Type": "volume",
    "Name": "6594c7a14ac4ff082fc0d5c9da89050f252add1c75816438061fe7ff3904edd3",
    "Source": "/var/lib/docker/volumes/6594c7a14ac4ff082fc0d5c9da89050f252add1c75816438061fe7ff3904edd3/_data",
    "Destination": "/library/TAoCP.vol4.a",
    "Driver": "local",
    "Mode": "",
    "RW": true,
    "Propagation": ""
  }
]
○ root@7fde7b0b91e1531a:~/code#

```

docker inspect: Provides detailed information about Docker objects.

--format "{{json .Mounts}}": Formats the output to show the mounts of the container.

Cleaning up volumes

Removing a Specific Volume

```
docker volume ls
```

```
docker volume rm <volume_name>
```

Pruning Unused Volumes

```
docker volume prune
```

Forcefully Removing All Volumes

```
docker stop $(docker ps -aq)
```

```
docker rm $(docker ps -aq)
```

```
docker volume rm $(docker volume ls -q)
```

```
total reclaimed space: 0B
root@7fde7b0b91e1531a:~/code# docker stop $(docker ps -aq)
- aq)
docker volume rm $(docker volume ls -q)d5370d9a5af3
c36f8983a7f0
10d35cfdc5b4
c87beca66d10
root@7fde7b0b91e1531a:~/code# docker rm $(docker ps -aq)
d5370d9a5af3
c36f8983a7f0
10d35cfdc5b4
c87beca66d10
root@7fde7b0b91e1531a:~/code# docker volume rm $(docker volume ls -q)
1d836acaf0581c754de87e0bccf481a1b89f19bd4e4200adbab27f0f4908782
8da7206a551e83e3e7922270d4f1288a1c0d29dd24345a6c10c8a8b811535a5a
6594c7a14ac4ff082fc0d5c9da89050f2524dd1c75816438061fe7ff3904edd3
d613364c45c879007d01bae312520887489719d02b369da10b2377e3ad44675b
f392f1e6d24369b1bb0dc1de67cd7a41c8b026b3f1d23f115fec8d294d211f84
logging-example
root@7fde7b0b91e1531a:~/code# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@7fde7b0b91e1531a:~/code# docker volume ls
DRIVER         VOLUME NAME
root@7fde7b0b91e1531a:~/code#
```

By following these procedures, you can efficiently manage and remove Docker volumes, ensuring your Docker environment remains clean and optimized.

Conclusion

This scenario highlights the advantages of using Docker volumes over bind mounts for sharing files between containers. Docker volumes offer better portability, simplified management, and improved security. Additionally, using anonymous volumes and the `--volumes-from` flag provides dynamic and flexible data sharing, making it easier to manage complex containerized applications.