Docker Volumes

Volumes

Volumes are the preferred mechanism for persisting data generated by and used by Docker containers. While [bind mounts](https://docs.docker.com/engine/storage/bind-mounts/) are dependent on the directory structure and OS of the host machine, volumes are completely managed by Docker. Volumes have several advantages over bind mounts:

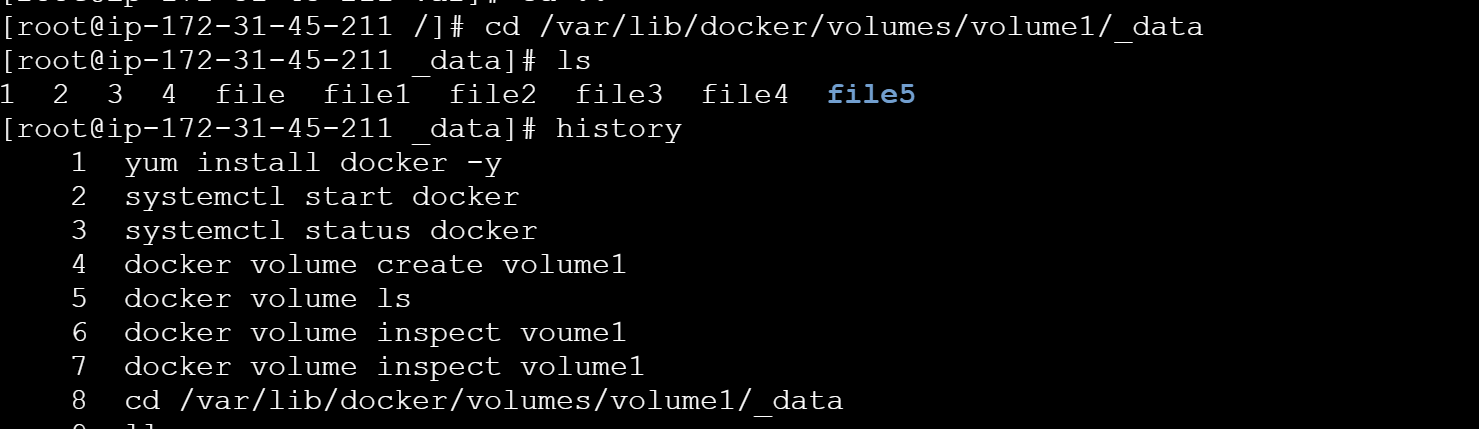
* Volumes are easier to back up or migrate than bind mounts.
* You can manage volumes using Docker CLI commands or the Docker API.
* Volumes work on both Linux and Windows containers.
* Volumes can be more safely shared among multiple containers.
* Volume drivers let you store volumes on remote hosts or cloud providers, encrypt the contents of volumes, or add other functionality.
* New volumes can have their content pre-populated by a container.
* Volumes on Docker Desktop have much higher performance than bind mounts from Mac and Windows hosts.

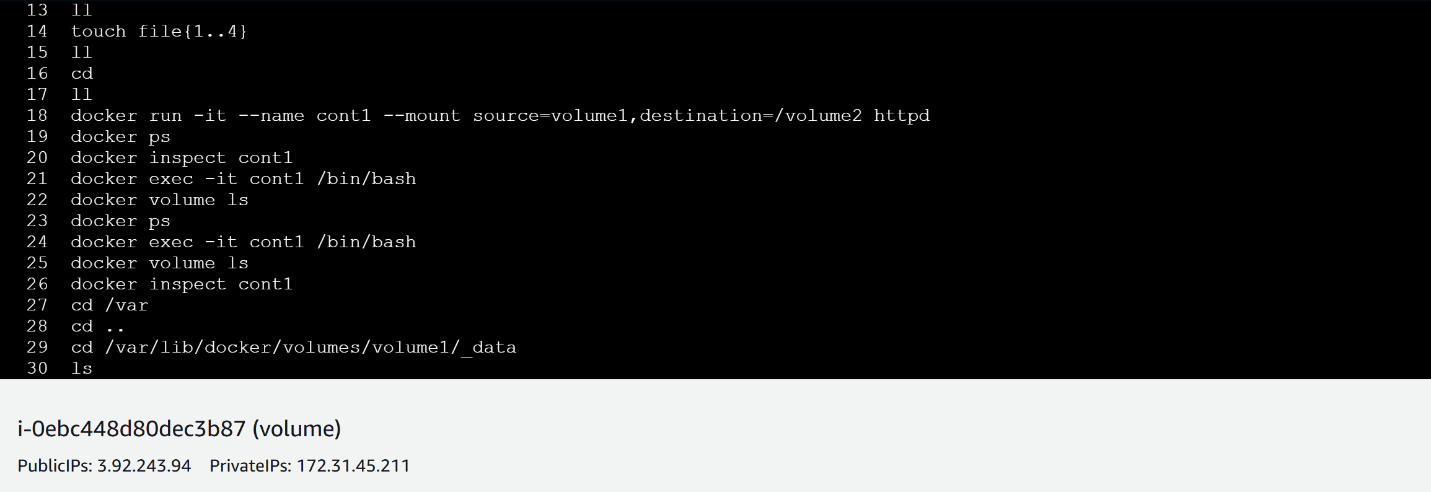
Choose the -v or –mount:

In general, --mount is more explicit and verbose. The biggest difference is that the -v syntax combines all the options together in one field, while the --mount syntax separates them. Here is a comparison of the syntax for each flag.

If you need to specify volume driver options, you must use --mount.

* -v or --volume: Consists of three fields, separated by colon characters (:). The fields must be in the correct order, and the meaning of each field isn't immediately obvious.
  + In the case of named volumes, the first field is the name of the volume, and is unique on a given host machine. For anonymous volumes, the first field is omitted.
  + The second field is the path where the file or directory is mounted in the container.
  + The third field is optional, and is a comma-separated list of options, such as ro. These options are discussed below.
* --mount: Consists of multiple key-value pairs, separated by commas and each consisting of a <key>=<value> tuple. The --mount syntax is more verbose than -v or --volume, but the order of the keys isn't significant, and the value of the flag is easier to understand.
  + The type of the mount, which can be [bind](https://docs.docker.com/engine/storage/bind-mounts/), volume, or [tmpfs](https://docs.docker.com/engine/storage/tmpfs/). This topic discusses volumes, so the type is always volume.
  + The source of the mount. For named volumes, this is the name of the volume. For anonymous volumes, this field is omitted. Can be specified as source or src.
  + The destination takes as its value the path where the file or directory is mounted in the container. Can be specified as destination, dst, or target.
  + The volume-subpath option takes a path to a subdirectory within the volume to mount into the container. The subdirectory must exist in the volume before the volume is mounted to a container. See [Mount a volume subdirectory](https://docs.docker.com/engine/storage/volumes/#mount-a-volume-subdirectory).
  + The readonly option, if present, causes the bind mount to be [mounted into the container as read-only](https://docs.docker.com/engine/storage/volumes/#use-a-read-only-volume). Can be specified as readonly or ro.
  + The volume-opt option, which can be specified more than once, takes a key-value pair consisting of the option name and its value.





How to create volume in Local:

. docker volume create volume name

. now you can see in local using docker volume ls command

. docker inspect volume name

. you can see one path is there i.e. /var/lib/docker/volumes/volumename/\_data

. now you can convert to that path cd /var/lib/docker/volumes/volumename/\_data

. you want to attach that volume to container we use

. docker run -it -d –name container name –mount source=volume name destination=another path(ex:/volume) image name

. Now you can go into the container using this command

Docker exec -it container name /bin/bash

How to use Docker volumes in dockerfile?

Volumes can be declared in your Dockerfile using the VOLUME statement. This statement declares that a specific path of the container must be mounted to a Docker volume. When you run the container, Docker will create an anonymous volume (volume with a unique id as the name) and mount it to the specified path.

