Docker volumes

Not everything can be in a container. The whole idea is that you can start, stop and delete the containers without losing data.

So if you need to persist data, do it outside of the containers.

You have two different ways of mounting data from your container `bind mounts` and `volumes`.

A bind mount is the simpler one to understand. It takes a host path `/data` and mounts it inside your container eg. `/opt/app/data`. The positive about bind is that it is easy and connects directly to the host filesystem. The downside is that you need to specify it at runtime with no indication on what mounts a given container has, and that you need to deal with backup,migration etc. in an tool outside the Docker ecosystem.

A docker Volume is where you can use a named or unnamed volume to store the external data. You would normally use a volume driver for this, but you can get a host mounted path using the default local volume driver.

In the next section, you will get to try both of them.

Bind mounts

So let's look at the Nginx container. The server itself is of little use, if it cannot access our web content on the host.

We need to create a mapping between the host system, and the container with the `-v` command:

```
docker container run --name some-nginx -v /some/content:/usr/share/nginx/html:ro -d
nginx
```

That will map whatever files are in the `/some/content` folder on the host to `/usr/share/nginx/html` in the container.

> The `:ro` attribute is making the host volume read-only, making sure the container can not edit the files on the host.

Try to do the following:

- `git clone` this repository down and navigate to the `labs/volumes/` folder.
- Try to run the above command with the right folder instead of `/some/content`. You can use the command `pwd` (Print working directory) to display where you are.

This will give you a nginx server running, serving your static files... *But on which port?*

- Run a `docker container ls` command to find out if it has any ports forwarded from the host.

Remember the past exercise on port forwarding in Docker.

- Make it host the site on port 8000
- Check that it is running by navigating to the hostname or IP with your browser, and on port 8000.

Volumes

Volumes are entities inside docker, and can be created in three different ways.

- By explicitly creating it with the `docker volume create <volume_name>` command
- By creating a named volume at container creation time with `docker container run -d -v DataVolume:/opt/app/data nginx`
- By creating an anonymous volume at container creation time with `docker container run -d -v /opt/app/data nginx`

First off, lets try to make a data volume called `data`:

```
docker volume create data
```

Docker creates the volume and outputs the name of the volume created.

If you run `docker volume ls` you will have a list of all the volumes created and their driver:

```
DRIVER              VOLUME NAME
local               data
```

Unlike the bindmount, you do not specify where the data is stored on the host.

In the volume API, like for almost all other of Docker's APIs, there is an `inspect` command giving you low level details. Let's use it against the html volume.

```
docker volume inspect data
[
    {
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/data/_data",
        "Name": "data",
        "Options": {},
        "Scope": "local"
    }
]
```

You can see that the `data` volumes is mounted at `/var/lib/docker/volumes/data/_data` on the

host.

> **Note** we will not go through the different drivers. For more info look at Dockers own example.

You can now use this data volume in all containers. Try to mount it to an nginx server with the `docker container run --rm --name www -d -p 8080:80 -v data:/usr/share/nginx/html nginx` command.

> **Note:** If the volume refer to is empty and we provide the path to a directory that contains data in the base image, that data will be copied into the volume.

Try now to look at the data stored in `/var/lib/docker/volumes/data/_data` on the host:

```
ls /var/lib/docker/volumes/data/_data/
50x.html  index.html
```

Those two files comes from the Nginx image and is the standard files the webserver has.

Attaching multiple containers to a volume

Multiple containers can attach to the same volume with data. Docker doesn't handle any file locking, so applications must account for the file locking themselves.

Let's try to go in and make a new html page for nginx to serve. We do this by making a new ubuntu container that has the `data` volume attached to `/tmp`, and thereafter create a new html file with the `echo` command:

```
docker container run -ti --rm -v data:/tmp ubuntu bash
root@9c36fcfcc048:# echo "<html><h1>hello world</h1></html>" > /tmp/hello.html
root@9c36fcfcc048:# ls /tmp
hello.html  50x.html  index.html
```

Head over to your newly created webpage at: `http://<IP>:8080/hello.html`

cleanup

Exit out of your ubuntu server and execute a `docker container stop www` to stop the nginx container.

Run a `docker container ls` to make sure that no other containers are running.

```
docker container ls
```

```
CONTAINER ID        IMAGE                  COMMAND                  CREATED
STATUS              PORTS
NAMES
```

The data volume is still present, and will be there until you remove it with a `docker volume rm data` or make a general cleanup of all the unused volumes by running `docker volume prune`.

Tips and tricks

As you have seen, the `-v` flag can both create a bindmount or name a volume depending on the syntax. If the first argument begins with a / or ~/ you're creating a bindmount. Remove that, and you're naming the volume. For example:

- `-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.

Sharing data

If you want to share volumes or bindmounts between two containers, then use the `--volumes-from` option for the second container. The parameter maps the mapped volumes from the source container to the container being launched.

More advanced docker commands

Before you go on, use the Docker command line interface documentation to try a few more commands:

- While your detached container is running, use the `docker container ls` command to see what silly name Docker gave your container. **This is one command you're going to use often!**
- While your detached container is still running, look at its logs. Try following its logs and refreshing your browser.
- Stop your detached container, and confirm that it is stopped with the `ls` command.
- Start it again, wait 10 seconds for it to fire up, and stop it again.
- Then delete that container from your system.

> **NOTE:** When running most docker commands, you only need to specify the first few characters of a container's ID. For example, if a container has the ID `df4fd19392ba`, you can stop it with `docker container stop df4`. You can also use the silly names Docker provides containers by default, such as `boring_bardeen`.

If you want to read more, I recommend Digital Oceans guides to sharing data through containers, as well as Dockers own article about volumes.

summary

Now you have tried to bind volumes to a container to connect the host to the container.