Docker

https://docs.docker.com/storage/volumes/

Build image

To build the docker image based on a Dockerfile in the current directory, use:

```
docker build .
```

Update container

Update docker container to latest version via docker compose:

```
#Single container:
docker compose pull calibre-web
docker compose up -d calibre-web
docker image prune

#Update all containers:
docker compose pull
docker compose up -d
docker image prune
```

Health Checks

Docker compose health checks:
https://medium.com/geekculture/how-to-successfully-implement-a-healthcheck-in-docker-compose-efced60bc08e

https://stefanjarina.gitbooks.io/docker/content/swarm-mode/healthchecks.html

Copying files to/from containers

Copying files from/to containers

```
docker cp dockerimage:/bla ./bla
docker cp ./bla dockerimage:/bla
```

Flatten docker image layers

Flatten docker image layers:

```
FROM yourbuildimage as build
# your existing build steps here
FROM scratch
COPY --from=build / /
CMD ["/your/start/script"]
```

Change mount points of containers

Change mount points of existing docker containers;

e.g. mount /home/<user-name> folder of host to the /mnt folder of the existing (not running) container.

1. stop docker container or whole docker engine

   ```
   systemctl stop docker.service
   ```

2. Open configuration file corresponding to the stopped container, which can be found at /var/lib/docker/containers/99d...1fb/config.v2.json (may be config.json for older versions of docker). For pretty print use

   ```
   vi <(jq . /var/lib/docker/containers/<container-ID>/config.v2.json)
   Save updates to a file: :w config.v2.json
   Exit vim: :q!
   Update existing file: jq -c . config.v2.json >
   /var/lib/docker/containers/<container-ID>/config.v2.json
   ```

3. Find MountPoints section: “MountPoints”:{}.
4. Replace the contents with something like this (you can copy proper contents from another container with proper settings):

   ```
   "MountPoints":{"/mnt":{"Source":"/home/<user-
   name>","Destination":"/mnt","RW":true,"Name":"","Driver":"","Type":"bin
   d","Propagation":"rprivate","Spec":{"Type":"bind","Source":"/home/<user
   -name>","Target":"/mnt"},"SkipMountpointCreation":false}}
   ```

   or the same (formatted):

   ```
     "MountPoints": {
       "/mnt": {
         "Source": "/home/<user-name>",
         "Destination": "/mnt",
         "RW": true,
         "Name": "",
         "Driver": "",
   ```

```
      "Type": "bind",
      "Propagation": "rprivate",
      "Spec": {
        "Type": "bind",
        "Source": "/home/<user-name>",
        "Target": "/mnt"
      },
      "SkipMountpointCreation": false
    }
  }
```

5. Start or restart the docker service:

```
systemctl start docker.service
service docker restart
```

6. Start the container if necessary:

```
docker start <container-name/ID>
```

PHP&NGINX image

Containerised PHP & NGINX on Alpine, image size ~60MB:
https://levelup.gitconnected.com/containerizing-nginx-php-fpm-on-alpine-linux-953430ea6dbc
https://github.com/johnathanesanders/docker-nginx-fpm

Log nginx to stdout in Dockerfile:

```
# forward request and error logs to docker log collector
RUN ln -sf /dev/stdout /var/log/nginx/access.log \
    && ln -sf /dev/stderr /var/log/nginx/error.log
```

Rename/Retag image files

```
docker tag current/image:tag new/image:tag
```

cmd shell of container

```
docker exec -it <mycontainer> bash
docker exec -it <mycontainer> /bin/sh
```

GPU hardware acceleration

Make sure the relevant GPU drivers are installed on the base system, then pass through the device via docker compose. Check devices are available, there should be a device per GPU starting at renderD128 for the first GPU:

```
ls -la /dev/dri
```

```
  devices:
   - /dev/dri:/dev/dri
```