

General Setup

- [Docker](#)
 - [Getting started](#)
 - [Organizing](#)
- [First steps](#)

Docker

Getting started

Before we continue, make sure you have a recent version of docker installed and setup correctly.

For a step-by-step guide see [here](#).

Changing the default ip range

Depending on your home network setup, the default ip range for docker networks might conflict with the ip range in your home network. To avoid conflicts we can change the default ip range that docker uses. Since docker uses the subnet `172.17.0.0/16` for their default bridge network, we need to avoid it. We can use a different one like `172.42.0.0/16` for example.

1. Create or edit the config file for the docker daemon:

```
sudo nano /etc/docker/daemon.json
```

2. Add the following using your

```
{
  "default-address-pools":
  [
    {"base":"172.42.0.0/16","size":24}
  ]
}
```

This will make docker use subnets like `172.42.1.0/24` or `172.42.2.0/24` whenever a new network is created.

3. Restart the docker service to apply the changes

```
service docker restart
```

4. When you already have some docker networks, simply `docker compose down` the corresponding containers and use `docker network rm <<networkname>>` to remove the network. The next time you start up the containers, docker will automatically recreate the network with the correct settings.

You can also manually define the subnet used for a given network from inside a compose file. It could look like this:

```
networks:
  default:
    name: "network_name"
    ipam:
      driver: default
      config:
        - subnet: "172.42.10.0/24"
```

See also

- <https://blog.uni-koeln.de/rrzk-knowhow/2020/09/23/privaten-ip-adressbereich-von-docker-anpassen/>

Organizing

There are different ways to organize a setup of a dozen different services. I've decided to split up the different app-stacks into different compose files. This way it is easier to change the apps seperately from each other and backup and versioning is also simplified.

The file structure looks something like this:

```
homeserver/  
├─ app1/  
  │─ backup/  
  │─ config/  
  │─ data/  
  │─ logs/  
  └─ docker-compose.yml  
├─ app2/  
  │─ backup/  
  │─ config/  
  │─ data/  
  │─ logs/  
  └─ docker-compose.yml  
├─ docker-compose.yml  
└─ .env
```

Every app gets its own folder with a `docker-compose.yml` and additional folders for stuff like configuration files or data stored by the app. In the compose files every service will be listed, that is needed for the app. This could be a combination of the main service, database and webserver for example. Configs for the services will be stores in the respective config folders. Things like databases will be stored in the data folder. Possible backups can be stored in the backup folder. App-specific logs (not those by `docker logs`) can be stored in the logs folder. Persistent data (e.g. userdata like documents or pictures) will be stored seperately.

At the root `homeserver` folder, is another `docker-compose.yml` file and a single `.env` file. From this folder we can start and stop individual or all containers with `dcup` or `dcdn` for example (see [Tips & Tricks](#) for an explanation on the aliases).

The `.env` file holds information like database users and passwords, hostname, email-adresses and so on.

First steps

Apache

By default apache2 might be running and will be using port 80. To disable it use

```
sudo systemctl disable apache2 && sudo systemctl stop apache2
```