Containers

- Check used ports on Linux
- Docker
  - Portainer app templates
  - Move docker on another disk on OMV
  - Portainer on OMV
  - Backup containers
  - Learn Docker
- Kubernetes
  - Kubernetes cluster

Check used ports on Linux

```
sudo lsof -i -P -n | grep LISTEN
sudo netstat -tulpn | grep LISTEN
sudo ss -tulpn | grep LISTEN
sudo lsof -i:22 ## see a specific port such as 22 ##
sudo nmap -sTU -O IP-address-Here
```

Docker

Docker

Portainer app templates

- https://raw.githubusercontent.com/mycroftwilde/portainer_templates/master/Template/template.json
- https://raw.githubusercontent.com/ntv-one/portainer/main/template.json
- https://raw.githubusercontent.com/xneo1/portainer_templates/master/Template/template.json
- https://raw.githubusercontent.com/Qballjos/portainer_templates/master/Template/template.json
- https://raw.githubusercontent.com/technorabilia/portainer-templates/main/lsio/templates/templates-2.0.json
- https://raw.githubusercontent.com/dnburgess/dbtechtemplate/master/Template/v2/templates.json
- https://raw.githubusercontent.com/SelfhostedPro/selfhosted_templates/master/Template/omv-v2.json
- https://raw.githubusercontent.com/mycroftwilde/portainer_templates/master/Template/template.json
- https://raw.githubusercontent.com/TheLustriVA/portainer-templates-Nov-2022-collection/main/templates_2_2_rc_2_2.json
- https://raw.githubusercontent.com/Lissy93/portainer-templates/main/templates.json

Docker

Move docker on another disk on OMV

1. Stop the docker daemon (and all containers with it) `systemctl stop docker`
2. ```
   rsync -aHv /var/lib/docker /srv/dev-disk-by-id-usb-WDC_WD10_EARS-00Y5B1_152D00539000-0-0-part2/omv/docker
   ```
   Copy all old docker contents to new location keeping permissions and possible hard links and be verbose about what it is doing
3. Uninstall docker
4. change the docker location in the OMV UI
5. install docker with the path changed and saved.
6. SAVE
7. Restart the docker daemon in the OMV UI (will restart containers as well)
8. Check if all your containers are running as expected
9. remove /var/lib/docker
   ```

Docker

Portainer on OMV

(c) https://forum.openmediavault.org/index.php?thread/40489-how-to-prepare-omv-to-install-docker-applications/

Introduction and basic concepts

Docker is a technology that enables the creation and use of Linux containers. A container is a closed environment where one or more applications and their dependencies are installed, grouped and isolated from each other, running on the same operating system kernel.

Docker allows you to install, uninstall, modify, update applications as many times as you want without causing damage to the system.

Starting from a system with OMV installed and with a file system with shared folders, to use Docker we need::

- **A SPACE** in our file system to store the containers. The recommendation is to dedicate an SSD disk, independent of the operating system, with EXT4 format, of at least 60GB. Speed will be achieved in the applications, avoiding docker incompatibilities with special file systems and completely removing the applications from the system disk. In this space, PROGRAM FILES and CONFIGURATION FILES will be stored:
  - PROGRAM FILES:
    - They are **expendable** files.
    - We will store them in the *docker* folder of the SSD disk.
    - Docker takes care of downloading these files from the internet and configuring them automatically.
    - Docker containers are updated by downloading the latest version of these files and replacing the old one. There are some exceptions, consult the documentation of each application you install to know how it works.
  - CONFIGURATION FILES:
    - They are **irreplaceable** files.
    - We will store them in the *config* folder of the SSD disk.
    - They are created, modified and personalized during the operation of the application with passwords, personal settings, databases, etc. depending on the case.
    - We must keep them if we want to keep the application in the same state or restore it after an OMV reinstall. It is a good idea to **make regular backups** of this folder.

- **A USER** who will be in charge of executing the container.
  - Docker applications execute the actions on the system from the container by means of a user. So the permissions of this user will be the permissions that the application will have. This will be the control mechanism to prevent the application from doing anything on our system..
  - For security, it is convenient to create a user and grant only the necessary permissions for the application to work. In this guide this user is called *appuser*.
  - Applications need to write to the docker and config folders. The way to give permissions to the application is to give them to the user who executes it. Therefore we must give write permissions to *appuser* in these folders. Also if we want the app to use any data from our shared folders, *appuser* must also have access to those folders

Instructions for preparing and installing docker

1. Previous steps.

- Install an SSD on the server, format it to EXT4 and mount to the file system.
  - For help you can refer to the New User Guide
- Install openmediavault-omvextrasorg.

  - You can check here
- Install openmediavault-wetty and openmediavault-symlinks.
  - In the OMV GUI go to *System>Plugins>* find and select the plugin, then click Install on the top menu. Repeat the process for the next plugin.

2. Create symlinks /SSD and /DATA

- Symlink /SSD
  - In the OMV GUI go to *Storage>File Systems>* open the *Mount Point* column and press the button to copy the SSD disk mount point to the clipboard.
  - Go to *Services>Symlinks>* and click on + *Create*.
  - In the *Source* field, copy the SSD disk mounting path from the clipboard (or find the path in the tree). In the *Destination* field type /SSD and press save.
- Symlink /DATA
  - Following the same procedure, now create the symlink /DATA. The path is found in *Storage>Shared Folders>* in the *Absolute Path* column.

3. Create folders for Docker files

- Open a terminal and log in with root.

    - In the OMV GUI go to *Services>Wetty>* Click on *Certificate* and choose the existing Certificate, select the *Enabled* checkbox, click on *Save* and accept changes. If you don't have a certificate, create it in *System>Certificates>SSH*
    - Click the *Open UI* button.
    - Enter the username and password.
- Create the folders with the following commands:

```
mkdir /SSD/docker
mkdir -m 770 /SSD/config
chgrp users /SSD/config
```

4. Define the default location of the docker PROGRAM files. Install Docker and Portainer.

- In the OMV GUI, go to the *System>omv-extras>Docker>* and in the *Docker storage* field replace the value with the path of the *docker* folder created earlier, a symlink like /SSD/docker may not work here. Example: /srv/dev-disk-by-uuid-861acf8c-761a-4b60-9123-3aa98d445f72/docker
- Install Docker: In the bottom menu, click the *Save* button. Then click the *Install* button.
- Install Portainer: In *System>omv-extras>Portainer>* click the *install* button in the bottom menu.

5. Create the user "appuser".

- In the OMV GUI go to *Users>Users>* click on the + *Create* button, define *appuser* name and assign password, in the groups field we add it to the *docker* and *users* groups. Click on save.
- Open the UID and GID columns and make a note of the values that the *appuser* user has. Example UID=1002 GID=100 This may vary on your system.

Example of installing an application (Jellyfin):

6. Choose a stack

- On the dockerhub there are thousands of containers ready to configure. Try to choose containers from reputable publishers (linuxserver is very popular) or with many downloads and current ones. Check that the container is compatible with your server's architecture x86-64 arm64... When choosing one read the publisher's recommendations before installing it.

- As an example we are going to install Jellyfin. Many results appear in the search engine, we chose the one for linuxserver see here https://hub.docker.com/r/linuxserver/jellyfin
- We fetch the docker-compose stack and copy it to a text file. In this case:

(Note: Verify on the official page that this stack has not changed before installing it)

```
---
version: "2.1"
services:
  jellyfin:
    image: lscr.io/linuxserver/jellyfin:latest
    container_name: jellyfin
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Europe/London
      - JELLYFIN_PublishedServerUrl=192.168.0.5 #optional
    volumes:
      - /path/to/library:/config
      - /path/to/tvseries:/data/tvshows
      - /path/to/movies:/data/movies
    ports:
      - 8096:8096
      - 8920:8920 #optional
      - 7359:7359/udp #optional
      - 1900:1900/udp #optional
    restart: unless-stopped
```

7. Customize the stack.

- It's always a good idea to read the editor's comments to familiarize yourself with the options and to check that the container is valid for your architecture.
- We customize our stack to fit our system. In this example, we remove all the optional lines and keep the essentials.
- Let's assume as an example that inside /DATA we have the media folder that contains other subfolders with movies, photos, etc.
- We adjust the stack as follows:

```
---
version: "2.1"
services:
```

```
jellyfin:
  image: lscr.io/linuxserver/jellyfin:latest
  container_name: jellyfin
  environment:
    - PUID=1002  # See Comment 1
    - PGID=100   # See Comment 1
    - TZ=Europe/Madrid   # See Comment 2
  volumes:
    - /SSD/config/Jellyfin:/config   # See Comment 3
    - /DATA/media:/media   # See Comment 3
  ports:
    - 8888:8096   # See Comment 4
  restart: unless-stopped
```

- **Comment 1:** See point 5 of this guide and adjust the correct values.
- **Comment 2:** Adjust it to your location. You can see it by typing `cat /etc/timezone` in a terminal with the openmediavault-wetty plugin.
- **Comment 3:** In the *volumes* section we map folders from the container to the host. This means that when the container writes to an internal folder, it will actually be writing to the folder we want outside the container. The syntax means that to the left of the ":" we define the actual path of the folder in our file system and to the right will be the equivalent path within the container.
  - So in the first line we tell the app that the config files are written to and read from our SSD in the *config* folder plus another subfolder called *Jellyfin*. It is not necessary to create the *Jellyfin* folder beforehand, docker can create up to one level of subfolders when the container is started.
  - In the second line we tell the application where the data to use is, movies, etc. The container will access our /DATA/media shared folder and mount it inside the container by naming it /media.
    - For this to work we must **GIVE PERMISSIONS** to the *userapp* on our /DATA/media shared folder. Otherwise the container (using the *appuser* permissions) will not be able to access this folder and the application will not start, it will show an error. We can give *appuser* permissions from the GUI by editing the user's privileges.
    - Note: In Jellyfin it is enough to map the DATA/media folder. Within Jellyfin we can configure the libraries of movies, music... by choosing the folders /media/films or /media/music...
- **Comment 4:** In the "ports" section we define the ports for the application to communicate with the system.
  - The syntax is the same as in the previous point, to the left of ":" we define the port of our nas that we want to use to access the application; on the right we define the port that the application uses internally.

    - Docker will communicate both ports. In this case we decided to change the real port to 8888, so we change the one on the left and keep the one used internally by the 8096 container.
- It's a good idea to keep this text file with the docker stack for future reference or reinstallation. If we have this file and the config folder, reinstalling the application on another system takes two minutes.

8. Deploy the Docker stack and access the application

- In the OMV GUI go to *System>omv-extras>Portainer>* and click on *Open web*. If it is the first time we set a password for admin and enter.
- Click on the whale icon, then on the left menu click on *Stacks*, then click on the top button + *Add stack*
- Copy the custom stack into the window and fill in the name field above by typing *jellyfin* (or the name of your application). Scroll the window down and click the *Deploy the stack* button. Wait for the stack to be deployed.
- When the process is finished, we can go to a browser and access the application by typing the IP of our server followed by ":" and the access port defined in the point seven of this guide. For example if the IP of our server was 192.168.1.100 we would write http://192.168.1.100:8888

9. Stack Modification.

- To modify parameters in a stack:
    - Click on the stack name to modify.
    - Click on the Editor button at the top.
    - Modify stack parameters
    - Click the *Update the stack* button at the bottom.
- Restore the container to its initial state, this will delete any configuration that we have made in the container.
    - Stop the execution of the container.
    - Delete the config folder corresponding to the container. (in the example /SSD/config/Jellyfin)
    - Start the container.

10. More information.

- Ports available to configure in the applications, see here
- How to backup Portainer see here
- How to move the /var/lib/docker folder to another disk if it is already up and running see here

Backup containers

https://hub.docker.com/r/offen/docker-volume-backup

https://torsion.org/borgmatic/

https://www.modem7.com/books/docker-backup/page/backup-docker-using-borgmatic

https://github.com/borgbase/vorta

https://kopia.io/

Learn Docker

Getting Started with Docker

https://github.com/collabnix/dockerlabs

Kubernetes

Kubernetes cluster

https://www.linuxtechi.com/install-kubernetes-on-ubuntu-22-04/