Docker

- How to use Docker in Linux
- Install Docker on local drive
- Troubleshooting Odoo Docker and Docker Compose
- Move / Transfer file from host do docker container
- Some Docker commands

How to use Docker in Linux

Install Docker on local drive

Do you have docker?
`docker -v`
Install Docker
`apt install docker`
Do you have docker-compose?

```
docker-compose -v
```

- Make a "project" directory somewhere and place the compose.yml file in it.

Docker volumes
- Find a place to store the cache, media and config files (with main storage?)
docker-compose.yml file

```
ersion: '3'
services:
  mariadb:
    image: mariadb
    container_name: mariadb
    restart: always
    command: --transaction-isolation=READ-COMMITTED --binlog-format=ROW
    volumes:
      - /local/storage/location:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=Mariaisabitch
      - MYSQL_PASSWORD=Mariaisabitch
      - MYSQL_DATABASE=nextcloud
      - MYSQL_USER=nextcloud


  nextcloud:
    image: nextcloud
    container_name: nextcloud
    restart: always
    links:
      - mariadb
    volumes:
      - /local/storage/location/config:/config
      - /local/storage/location/data:/data
```

```
    - /local/storage/location/apps:/apps
  ports:
    - 5000:80
  environment:
    - MYSQL_PASSWORD=Mariaisabitch
    - MYSQL_DATABASE=nextcloud
    - MYSQL_USER=nextcloud
    - MYSQL_HOST=mariadb
```

Copy files from Host to Container
docker cp file.txt container-name:/path/**to**/**copy**/file.txt

Troubleshooting Odoo Docker and Docker Compose

I figured out some good debugging methods that both solved this problem and seem generally useful for figuring out Docker persistent data volume issues.

Test 1: can the container work with an empty Docker volume?

This is a really easy test: just create a new Docker volume and pass that in your -v argument (instead of a host directory absolute path):

```
sudo docker volume create hello
sudo docker run -v hello:/var/lib/odoo -p 8069:8069 --name odoo --link db:db -t odoo
```

The odoo container immediately worked successfully this way (i.e. my web browswer was able to connect to the Odoo server). This showed that it could work fine with an (initially) empty data directory. The obvious question then is why it didn't work with an empty host-directory volume. I had read that Docker containers can be persnickety about UID/GID ownership, so my next question was how do I figure out what it expects.

Test 2: inspect the running container's file system

I used docker exec to get an interactive bash shell in the running container:

```
sudo docker exec -ti odoo bash
```

Inside this shell I then looked at the data directory ownership, to get numeric UID and GID values:

```
ls -dn /var/lib/odoo
```

This showed me the UID/GID values were 101:101. (You can exit from this shell by just typing Control-D)

Test 3: re-run container with matching host-directory UID:GID

I then changed the ownership of my host directory to 101:101 and re-ran the odoo container with my host-directory mount:

```
sudo chown 101:101 /tmp/odoo
sudo docker stop odoo
```

```
sudo docker rm odoo
sudo docker run -v /tmp/odoo:/var/lib/odoo -p 8069:8069 --name odoo --link db:db -t odoo
```

Success! Finally the odoo container worked properly with a host-directory mount. While it's annoying the Odoo docker docs don't mention anything about this, it's easy to debug if you know how to use these basic tests.

https://stackoverflow.com/questions/54187785/how-to-debug-persistent-data-volume-mount-for-docker-odoo-container

Move / Transfer file from host do docker container

Using terminal

From container to host:

```
sudo docker cp container-id:/path/filename.txt ~/Desktop/filename.txt
```

From host to container:

```
sudo docker cp ~/Desktop/filename.txt container-id:/path/filename.txt
```

Some Docker commands

This will take all running containers, and show you the ports and container names.

```
docker ps --format "{{.Names}} {{.Ports}}" | awk '{name=$1; for(i=2;i<=NF;i++)
if($i~/0\.0\.0\.0:/) {split($i,a,":"); split(a[2],b,"->"); print b[1], name}}' | sort -n
```

Inside containers:

The quickest way — run a curl command through the container and ask it what its public IP is:

- You can use this to see if it is using a VPN or not

```
docker exec container-name curl -s https://ipinfo.io
```