Docker (rootless) + Portainer

Docker is a powerful solution for setting up Services. This short Introducion will give you hints how to setup Docker in a good way in userspace, so no root-access is needed for Docker.

Currently i am experimenting on that topic, so maybe this documentation will be ready to use, maybe not.

Docker itself is nice, but it will run as root per default, which is a no-go at all. This will setup Docker in rootless- mode on OpenSuSE (currently Leap 15.5).

Warning: This is a very strong advise NOT to use docker default in rootmode at all! The reason is, that any service is able to talk to the Docker Daemon if there is a connection to the Docker socket in the Volumes (which some services require) - or simply if thers a bug somewhere. By that way, the Docker Container will be able to set up ANY service and bind ANY location on the Host, that the docker user may be able to see. So if the Service gets taken over and the service ist runnig as root... you know where you are.

So just: Don't set up Docker rootfull at all if possible - its even not neeeded nowadays.

Filesystem Layout

Mind, that at the time writing, `overlay2` is the way to go as storage driver in docker, but it only supports xfs as backing filesystems (with `d_type=true` which means `ftype=1`) for full support.

I personally dislike xfs, especially while its not rubust and won't shrink. I use it anyway, because of its srong advise to do so - with kernel 5.19+ it should be possible to get overlay2 working on btrfs, but still there are things that may not work even with that kernel - in the worst case, docker is unable to unlink files, so there will be huge Containers and Volumes and maybe Services will break.

So make sure, that the Home-Directory of you docker user is on XFS. The ftype is already ok on SuSE 15.5, check output of `xfs_info <volumename>`.

Warning: you may have umask set your way - i prefer 007 as written before. But if you change umask and permissions be very cautious, as docker uses userid-mapping and may change the permissions and ownersets of files in its directory to the subuserid.

That may change the ownership in a way, that even the docker user on the host cannot access the Files, which is OK !

STRONG WARNING: Don't change permissions or ownership of docker- directories on the Host directly as this will change them in the container, making them unavaiable and break your Services !!!

The only way to manage Volume- File- Permissions is to bash inside the running container itself and to change them there (to the right values of course)!

A short hint: Docker rootless uses Sub(g)uids, which is a feature of Linux. That means each user has a range of userids (quite a huge range) and groupid which the user may use. Those will be exclusive reserved for that user. But it does not mean, that the User can acces the Files created by those Subuids! Also the UIDs are onyl a number - not a real user in Linux having a username- They cannot be used to logon or to work with. Docker manages internally which Host-Subuserid is assigned to which caontainer and to which userid inside the running container/service. Inside the Container, you may become that user having a real username and a (different) uid.

Which Devices / Raid- Level

I strongly do not advise to use Raid5 with classical harddrives to run docker on it. If you plan to have large Data in /home/docker and you want to use xfs + Raid5 to better use your drive- space, than you should use another disc for the overlays of docker.

For me, i switched to RAID1 on SSD an mounted that to /home/docker/.local - where all Docker files will be stored. Large Data is than stored somewhere else if needed.

Packages NOT to install

I had really a lot of troubles with the package Docker-Rootless in the AddOn- Repository: `https://download.opensuse.org/repositories/Virtualization:/containers/${releasever}/` while they are not installing docker the same way, that docker would do. For example they will not be installed in User-Subspace only, but will use systems Docker executables installed in global paths. This is a problem when using btrfs - as btrfs is not fully compatible with docker. So i won't use this any more.

So i disabled the following packages and locked them to never install:

- docker
- docker-compose
- containerd

Check out beneath for install the docker way.

Docker- User

Create a new **group** called **docker** and a new **user** called **docker**. Make the user is in the **default group docker**.

Attention: The Home Directory should be on a volume having XFS as btrfs or others are not fully supported right now (20.04.2024 - patches in new Kernel 5.19 are incoming, but this Kernel is not released until now and still there are some problems open in developement there).

cGroups v2

OpenSuSE Leap 15.5 does not have cGroups v2 enabled, which are needed by docker.

You may see a warning (later) when running `docker info`:

```
WARNING: Running in rootless-mode without cgroups. To enable cgroups in rootless-mode, you need to boot the system in cgroup v2 mode.
```

According to this documentation https://rootlesscontaine.rs/getting-started/common/cgroup2/ it needs to be enabled by appending the yast/bootloader command line with:

```
systemd.unified_cgroup_hierarchy=1
```

and also the delegation for the user of cpu is needed:

```
$ sudo mkdir -p /etc/systemd/system/user@.service.d
$ cat <<EOF | sudo tee /etc/systemd/system/user@.service.d/delegate.conf
[Service]
Delegate=cpu cpuset io memory pids
EOF
$ sudo systemctl daemon-reload
```

after this, reboot and check if `/sys/fs/cgroup/cgroup.controllers` is present

After installing docker (see beneath), check if `docker info` says:

```
Cgroup Driver: systemd
Cgroup Version: 2
```

Than, its fine.

Install rootless Docker

Warning: You CANNOT sudo to the user and install docker, while logon via pam is needed, which is not when you sudo. You need to ssh into your machine, or yust logon in a usual way:

```
If you login in the system using either of
- graphical session
- login on terminal (username and password)
- ssh
then the PAM machinery will call pam_systemd, and this will setup all needed hooks to use systemctl;
if you switch user using sudo or su, this will not happen.
```

I chose to ssh into my machine directly, than check your umask to be secure and install docker like this:

```
# ~> ssh localhost -l docker
Password:
Have a lot of fun...

docker@pcserver2023:~> umask
0007

docker@pcserver2023:~> curl -fsSL https://get.docker.com/rootless | FORCE_ROOTLESS_INSTALL=1 sh
# Installing stable version 25.0.2
# Executing docker rootless install script, commit: 3b2a83b
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 68.2M  100 68.2M    0     0  10.0M      0  0:00:06  0:00:06 --:--:-- 10.3M
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 19.7M  100 19.7M    0     0   9.7M      0  0:00:02  0:00:02 --:--:--  9.7M
```

```
+ PATH=/home/docker/bin:/home/docker/bin:/usr/local/bin:/usr/bin:/bin
+ /home/docker/bin/dockerd-rootless-setuptool.sh install --force
[INFO] Creating /home/docker/.config/systemd/user/docker.service
[INFO] starting systemd service docker.service
+ systemctl --user start docker.service
+ sleep 3
+ systemctl --user --no-pager --full status docker.service
● docker.service - Docker Application Container Engine (Rootless)
     Loaded: loaded (/home/docker/.config/systemd/user/docker.service; disabled; vendor preset: disabled)
     Active: active (running) since Sat 2024-04-20 15:25:04 CEST; 3s ago
       Docs: https://docs.docker.com/go/rootless/
   Main PID: 3270 (rootlesskit)
      Tasks: 49
     Memory: 60.3M
        CPU: 224ms
     CGroup: /user.slice/user-1001.slice/user@1001.service/app.slice/docker.service
             ├─ 3270 rootlesskit --state-dir=/run/user/1001/dockerd-rootless --net=vpnkit --mtu=1500 --slirp4netns-
sandbox=auto --slirp4netns-seccomp=auto --disable-host-loopback --port-driver=builtin --copy-up=/etc --copy-up=/run --
propagation=rslave /home/docker/bin/dockerd-rootless.sh
             ├─ 3277 /proc/self/exe --state-dir=/run/user/1001/dockerd-rootless --net=vpnkit --mtu=1500 --slirp4netns-
sandbox=auto --slirp4netns-seccomp=auto --disable-host-loopback --port-driver=builtin --copy-up=/etc --copy-up=/run --
propagation=rslave /home/docker/bin/dockerd-rootless.sh
             ├─ 3290 vpnkit --ethernet /run/user/1001/dockerd-rootless/vpnkit-ethernet.sock --mtu 1500 --host-ip 0.0.0.0
             ├─ 3306 dockerd
             └─ 3327 containerd --config /run/user/1001/docker/containerd/containerd.toml
+ DOCKER_HOST=unix:///run/user/1001/docker.sock
+ /home/docker/bin/docker version
Client:
 Version:           25.0.2
 API version:       1.44
 Go version:        go1.21.6
 Git commit:        29cf629
 Built:             Thu Feb  1 00:22:06 2024
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
 Engine:
```

```
  Version:          25.0.2
  API version:      1.44 (minimum version 1.24)
  Go version:       go1.21.6
  Git commit:       fce6e0c
  Built:            Thu Feb  1 00:23:45 2024
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          v1.7.13
  GitCommit:        7c3aca7a610df76212171d200ca3811ff6096eb8
 runc:
  Version:          1.1.12
  GitCommit:        v1.1.12-0-g51d5e94
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
 rootlesskit:
  Version:          2.0.0
  ApiVersion:       1.1.1
  NetworkDriver:    vpnkit
  PortDriver:       builtin
  StateDir:         /run/user/1001/dockerd-rootless
 vpnkit:
  Version:          7f0eff0dd99b576c5474de53b4454a157c642834
+ systemctl --user enable docker.service
Created symlink /home/docker/.config/systemd/user/default.target.wants/docker.service →
/home/docker/.config/systemd/user/docker.service.
[INFO] Installed docker.service successfully.
[INFO] To control docker.service, run: `systemctl --user (start|stop|restart) docker.service`
[INFO] To run docker.service on system startup, run: `sudo loginctl enable-linger docker`

[INFO] Creating CLI context "rootless"
Successfully created context "rootless"
[INFO] Using CLI context "rootless"
Current context is now "rootless"

[INFO] Make sure the following environment variable(s) are set (or add them to ~/.bashrc):
export PATH=/home/docker/bin:$PATH
```

```
[INFO] Some applications may require the following environment variable too:
export DOCKER_HOST=unix:///run/user/1001/docker.sock
```

So, this looks very nice. **Important:** Do what the Installation says with the file `~/.bashrc`

Check Docker install

Log out of docker user if you are still in from install. Then, log back in to apply the bashrc- settings.

Ceck the Environement to have the settings:

```
docker@pcserver2023:~> Abgemeldet
Connection to localhost closed.
obel1x@pcserver2023:~> ssh localhost -l docker
Password:
Last login: Sat Apr 20 15:18:56 2024 from ::1
Have a lot of fun...
docker@pcserver2023:~> echo $DOCKER_HOST
unix:///run/user/1001/docker.sock
```

Now check `docker info`:

```
docker@pcserver2023:~> docker info
Client:
 Version:    25.0.2
 Context:    default
 Debug Mode: false

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 25.0.2
```

```
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Using metacopy: false
  Native Overlay Diff: false
  userxattr: true
Logging Driver: json-file
Cgroup Driver: systemd
Cgroup Version: 2
Plugins:
 Volume: local
 Network: bridge host ipvlan macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file local splunk syslog
Swarm: inactive
Runtimes: io.containerd.runc.v2 runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 7c3aca7a610df76212171d200ca3811ff6096eb8
runc version: v1.1.12-0-g51d5e94
init version: de40ad0
Security Options:
 seccomp
  Profile: builtin
 rootless
 cgroupns
Kernel Version: 5.14.21-150500.55.52-default
Operating System: openSUSE Leap 15.5
OSType: linux
Architecture: x86_64
CPUs: 8
Total Memory: 30.79GiB
Name: pcserver2023
ID: 45699224-ea9c-4865-8dea-a53bb20b788c
Docker Root Dir: /home/docker/.local/share/docker
Debug Mode: false
Experimental: false
Insecure Registries:
 127.0.0.0/8
```

```
 Live Restore Enabled: false
 Product License: Community Engine
```

Additional knowledge

- Storage driver and FS-Type : overlay2 should always be used, btrfs is outdated! XFS and d_type are important!
- CGroup Version needs to be 2 or better
- If you see Docker complaining about Module aufs at start: do not care about - that module is obsolete

IP-Filter

When starting Docker, an the log says:

```
level=warning msg="Running modprobe bridge br_netfilter failed with message: modprobe: ERROR: could not insert
'br_netfilter': Operation not permitted\ninsmod /lib/modul>
level=info msg="skipping firewalld management for rootless mode"
```

You first need to load the module with modprobe.

For system startup, use `/etc/modules-load.d` and creat e a file `docker-rootless.conf` in it, containing that module.

IPTables

If you see `docker info` saying:

```
WARNING: bridge-nf-call-iptables is disabled
WARNING: bridge-nf-call-ip6tables is disabled
```

This should be fixed by:

```
# sudo echo "net.bridge.bridge-nf-call-iptables = 1">> /etc/sysctl.conf
# sudo echo "net.bridge.bridge-nf-call-ip6tables = 1">> /etc/sysctl.conf
# sudo modprobe br_netfilter
```

```
# sudo sysctl --system
```

Configuring Docker Daemon

in rootless-mode, the file to configure docker is here:

`~/.config/docker/daemon.json`

by default, the path and the file is not existent, create it new within the docker user.

For example, enable IPv6. See https://docs.docker.com/config/daemon/ipv6/ for details.

```
{
 "ipv6": true,
 "ip6tables": true,
 "fixed-cidr-v6": "fd12:3456:1:::/48",
 "iptables": true,
 "fixed-cidr": "172.1.0.0/16",
 "log-opts": {
   "max-size": "10m",
   "max-file": "5"
 }
}
```

Notice: Don't use `userns-remap` - this won't work and makes no sense in rootless!

Edit: „experimental“: true has been removed for ipv6 with docker v27.

You need to adjust cidr to some unique ULA. ULAs are non internet routable adresses (like 192.X.X.X in ipv4). Select an unique adress only for that internal Docker network - you can choose anything that is not assigned anywhere else on your network to not cause trouble.

Maybe use this tool to generate: https://www.unique-local-ipv6.com

The default Network is not IPV6- enabled by default

If you specify no network, or use the network: default - than as the time of writing, IPV6 will not be enabled by default.

So, in your docker-compose.yml you need the lines:

```
networks:
# Still needs to be defined while without it won't enable ipv6
  default:
    driver: bridge
    enable_ipv6: true
```

Networking in Docker rootless

If you read docs in the net about networking with Docker you may see docker0 as bridge network. While this network is also there in docker rootless, you will not find that network as interface on your host like you would on a rootful docker.

Instead the network is encapsulated in the environement of rootlesskit and not visible to the host. From the Hosts view Docker is just another Application running on your Host talking to the internet like some app would do.

Performance

when you install pasta- networking driver, you can edit your docker systemd and use a much improved networking-driver.

Also you can make Docker a higher Priority and Nice-Value:

```
docker@server:~> systemctl --user edit docker.service

[Service]
# Higher Prio for Docker
Nice=10
IOSchedulingClass=best-effort
IOSchedulingPriority=7
```

```
# Use Pasta- network Driver (of your Host - Pasta needs to be installed)
Environment="DOCKERD_ROOTLESS_ROOTLESSKIT_NET=pasta"
Environment="DOCKERD_ROOTLESS_ROOTLESSKIT_PORT_DRIVER=implicit"
Environment="DOCKERD_ROOTLESS_ROOTLESSKIT_FLAGS=--ipv6"
```

Install docker compose

This Chapter may be obsolete as since docker v27 the compose plugin is part of installation script - check your output of `docker info` for the installed Plugins and if `docker compose version` already has a version. If so, skip this.

The command `docker-compose` has been obsoleted and been replaced by a plugin `compose` for docker (see https://docs.docker.com/compose/install/).

Installing it the manual way:

Edit the File `~/.bashrc` and add:

```
export DOCKER_CONFIG=${DOCKER_CONFIG:-$HOME/.docker}
```

Then relog to the docker user and do as the doc says to install and check you install:

```
docker@pcserver2023:~> mkdir -p $DOCKER_CONFIG/cli-plugins
docker@pcserver2023:~> curl -SL https://github.com/docker/compose/releases/download/v2.26.1/docker-compose-linux-x86_64 -o
$DOCKER_CONFIG/cli-plugins/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 59.8M  100 59.8M    0     0  9951k      0  0:00:06  0:00:06 --:--:-- 11.4M
docker@pcserver2023:~> chmod +x $DOCKER_CONFIG/cli-plugins/docker-compose
docker@pcserver2023:~> docker compose version
Docker Compose version v2.26.1
docker@pcserver2023:~>
```

Your done with the compose plugin

Update

If you want to update your docker- installation, there is not update- process but to use the same script again:

```
#!/bin/bash
#Upgrade docker rootless and plugin
./docker_stop_all.sh
sleep 5
systemctl --user stop docker
sleep 5
#uninstall
rm -f ~/bin/dockerd
rm ~/.config/systemd/user/docker.service.bak
mv ~/.config/systemd/user/docker.service ~/.config/systemd/user/docker.service.bak
#reinstall docker compose
COMPOSE_VER='2.32.4'
rm $DOCKER_CONFIG/cli-plugins/docker-compose
echo "Download Docker Compose Release ${COMPOSE_VER} - please check at https://github.com/docker/compose/releases for the
newes Version and change this File"
curl -SL https://github.com/docker/compose/releases/download/v${COMPOSE_VER}/docker-compose-linux-x86_64 -o
$DOCKER_CONFIG/cli-plugins/docker-compose
chmod +x $DOCKER_CONFIG/cli-plugins/docker-compose
#install docker
curl -fsSL https://get.docker.com/rootless | sh
#need to give the new binary permissions to acces privileged network ports (beneath 1024)
sudo setcap 'cap_net_bind_service=+ep' ~/bin/rootlesskit
#this should be everything
docker info
```

Create a place for Yamls

Now, that you have compose, you can use it to setup your services with YAML- Files. Each service should have a directory for its own.

Make a directory with `mkdir ~/docker_compose` and change to it.

First Docker App: Portainer

Now - finally its time for our first running Container. As the Portainer- App is an important Management- Software in Docker for inexperienced users, let's run it in a safe userspaced way now.

As always, SSH into your docker- user and than create the folders and yml-files for docker compose and portainer.

```
obel1x@server:~> ssh localhost -l docker
Password:
docker@pcserver2023:~> cd ~/docker_compose
docker@pcserver2023:~/docker_compose> mkdir portainer
docker@pcserver2023:~/docker_compose> cd portainer
docker@pcserver2023:~/docker_compose> touch docker-compose.yml
docker@pcserver2023:~/docker_compose>
```

Put the following into that file:

```
services:
 portainer:
   restart: always
   image: portainer/portainer-ce
   ports:
     - 9000:9000
     - 9433:9433
   volumes:
     - portainer_data:/data
     - /run/user/1001/docker.sock:/var/run/docker.sock

volumes:
 portainer_data:
```

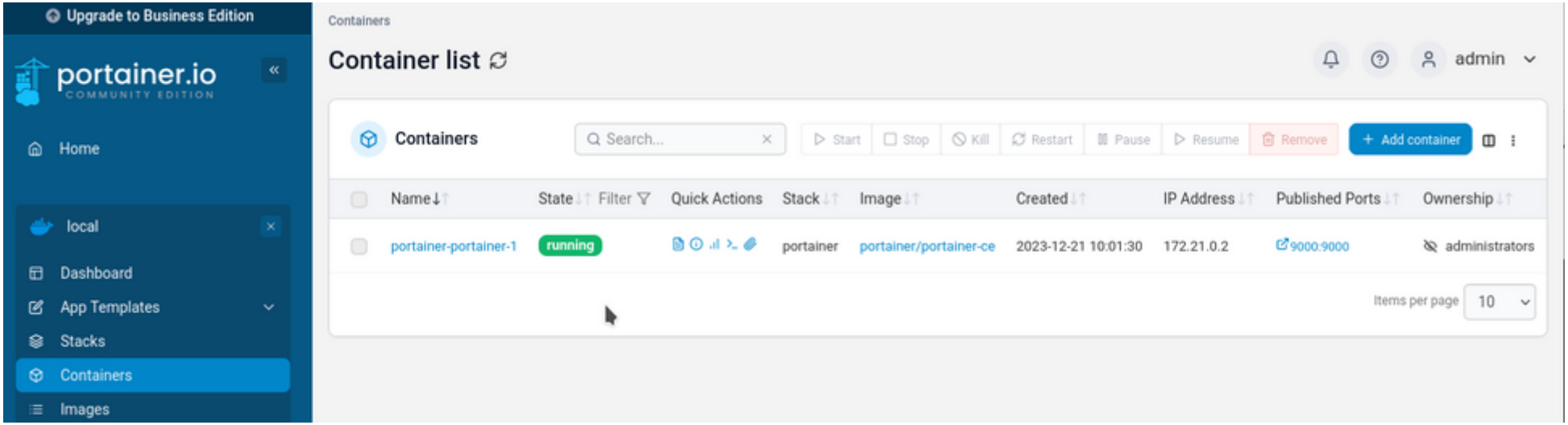
Check, that the Socket- Path is the correct one.

Now start your app and look the magic:

```
docker@pcserver2023:~/docker_compose/portainer> docker compose up -d
```

```
[+] Running 12/12
 ✔ portainer Pulled
17.6s
   ✔ 379538b6d68e Pull complete
0.5s
   ✔ 4ea3e2c3a39b Pull complete
0.5s
   ✔ 5171176db7f2 Pull complete
3.8s
   ✔ 52e9438966a5 Pull complete
6.5s
   ✔ 43d4775415ac Pull complete
6.7s
   ✔ c1cad9f5200f Pull complete
9.6s
   ✔ 22eab514564f Pull complete
7.1s
   ✔ 962b9fa821a2 Pull complete
10.0s
   ✔ c153fefda5ce Pull complete
10.9s
   ✔ bed990c4615b Pull complete
10.2s
   ✔ 4f4fb700ef54 Pull complete
10.5s
[+] Running 3/3
 ✔ Network portainer_default          Created
0.2s
 ✔ Volume "portainer_portainer_data"  Created
0.1s
 ✔ Container portainer-portainer-1    Started
0.3s
docker@pcserver2023:~/docker_compose/portainer>
```

Now you can go to http://localhost:9000 and pick a password to finish the setup of Portainer using the local Environment and enjoy the docker-party:



Thats all: Docker is running and serving your services, cheers!

Fast Stop of all Containers

This makes life easy `docker_stop_all.sh`:

```
#!/bin/bash
docker stop $(docker ps -a -q)
docker rm $(docker ps -a -q)
```

Find the latest Commit in git

Sometimes the Repository does not offer a latest Tatg, so its hard to find the right Tag to use. Maybe this coding helps (not testet wheter to get the right commitid for the Image):

```
if [ -z ${TVHEADEND_COMMIT+x} ]; then \
   TVHEADEND_COMMIT=$(curl -sX GET https://api.github.com/repos/tvheadend/tvheadend/commits/master \
   | jq -r '. | .sha'); \
```

```
  fi && \
```