Docker

Removal

Back up all volumes

```
for d in */; do tar -czf "${d%/}_$(date +%Y-%m-%dT%H-%M-%S).tgz" "$d"; done
```

Remove all of docker and start again

```
sudo systemctl stop docker
sudo apt-get purge -y docker-ce docker-ce-cli containerd.io docker-buildx-
plugin docker-compose-plugin
sudo rm -rf /var/lib/docker
sudo rm -rf /var/lib/containerd
sudo rm -rf /etc/docker
sudo apt-get autoremove -y
```

Fresh install

Then reinstall via Docker's official repo (not the Ubuntu snap/apt version):

```
curl -fsSL https://get.docker.com | sh
```

That script installs the latest stable Docker CE with all plugins. After that, add your user to the docker group if needed:

sudo usermod -aG docker sean

Then re-enable your services:

```
sudo systemctl enable --now docker
sudo systemctl enable --now hawser
```

Enable SSL/TLS connection

Generate a CA, server, and client certificate/key pair

1. Create a working directory

```
mkdir -p ~/.docker/tls && cd ~/.docker/tls
```

2. Generate the CA private key

```
openssl genrsa -aes256 -out ca-key.pem 4096
```

3. Generate the CA certificate

```
openssl req -new -x509 -days 365 -key ca-key.pem -sha256 -out ca.pem
```

You'll be prompted for a passphrase and subject info (Country, CN, etc.).

4. Generate the server private key

```
openssl genrsa -out server-key.pem 4096
```

5. Generate the server CSR (Certificate Signing Request)

```
openssl req -subj "/CN=<your-server-hostname-or-IP>" -sha256 -new -key
server-key.pem -out server.csr
```

Replace <your-server-hostname-or-IP> with your Docker host's hostname or IP (e.g., mydockerhost or 192.168.1.10).

6. Create a SANs (Subject Alternative Names) extension file

```
echo subjectAltName = DNS:<hostname>,IP:<IP>,IP:127.0.0.1 > extfile.cnf
echo extendedKeyUsage = serverAuth >> extfile.cnf
```

Add all hostnames/IPs clients will use to reach the server.

7. Sign the server certificate with the CA

```
openssl x509 -req -days 365 -sha256 \
  -in server.csr -CA ca.pem -CAkey ca-key.pem \
  -CAcreateserial -out server-cert.pem -extfile extfile.cnf
```

8. Generate the client private key

```
openssl genrsa -out key.pem 4096
```

9. Generate the client CSR

```
openssl req -subj '/CN=client' -new -key key.pem -out client.csr
```

10. Create a client extension file

```
echo extendedKeyUsage = clientAuth > extfile-client.cnf
```

11. Sign the client certificate with the CA

```
openssl x509 -req -days 365 -sha256 \
  -in client.csr -CA ca.pem -CAkey ca-key.pem \
```

```
  -CAcreateserial -out cert.pem -extfile extfile-client.cnf
```

12. Clean up CSRs and extension files, lock down permissions

```
rm -f client.csr server.csr extfile.cnf extfile-client.cnf
chmod 0400 ca-key.pem key.pem server-key.pem
chmod 0444 ca.pem server-cert.pem cert.pem
```

After this you'll have:

File	Purpose
ca.pem	CA certificate (needed by both server and client)
server-cert.pem / server-key.pem	Server certificate and key
cert.pem / key.pem	Client certificate and key

Configure /etc/docker/daemon.json with "tls": true, "tlscacert", "tlscert", and "tlskey"

1. Copy server certs to a permanent location

```
sudo mkdir -p /etc/docker/certs
sudo cp ~/.docker/tls/ca.pem /etc/docker/certs/
sudo cp ~/.docker/tls/server-cert.pem /etc/docker/certs/
sudo cp ~/.docker/tls/server-key.pem /etc/docker/certs/
sudo chmod 0444 /etc/docker/certs/ca.pem /etc/docker/certs/server-cert.pem
sudo chmod 0400 /etc/docker/certs/server-key.pem
```

2. Create or edit /etc/docker/daemon.json

```
sudo tee /etc/docker/daemon.json > /dev/null <<EOF
{
  "tls": true,
  "tlsverify": true,
  "tlscacert": "/etc/docker/certs/ca.pem",
  "tlscert": "/etc/docker/certs/server-cert.pem",
```

```
  "tlskey": "/etc/docker/certs/server-key.pem",
  "hosts": ["unix:///var/run/docker.sock", "tcp://0.0.0.0:2376"]
}
EOF
```

1. “tls”: true — enables TLS but allows unauthenticated clients
2. “tlsverify”: true — enforces mutual TLS (clients must present a valid cert)
3. Port 2376 is the standard TLS Docker port (2375 is unencrypted)

3. Handle the hosts conflict with systemd (WSL/Linux)

If Docker is managed by systemd, the -H flag in the service file conflicts with hosts in daemon.json. Override it:

```
sudo mkdir -p /etc/systemd/system/docker.service.d
sudo tee /etc/systemd/system/docker.service.d/override.conf > /dev/null
<<EOF
[Service]
ExecStart=
ExecStart=/usr/bin/dockerd
EOF
```

Enable systemd in WSL2 (recommended).

Edit /etc/wsl.conf

```
sudo tee /etc/wsl.conf > /dev/null <<EOF
[boot]
systemd=true
EOF
```

4. Reload and restart Docker

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

5. Verify Docker is listening on port 2376

```
sudo ss -tlnp | grep 2376
```

You should see dockerd bound to 0.0.0.0:2376. That confirms TLS is active and the daemon is accepting remote connections.

Portainer

Upgrade Portainer

Run:

```
cd /workspace/Portainer/
./portainer.sh
```

portainer.sh

```
#!/bin/sh

# From: https://rpi-wifi:8443/docker

docker ps |grep portainer|awk '{print $1}'|xargs docker stop
docker ps -a | grep portainer | awk '{print $1}' | xargs docker container rm
docker image ls|grep portainer|awk '{print $3}'|xargs docker rmi
docker run -d -p 8000:8000 -p 9000:9000 --name "Portainer" --restart always
-v "/var/run/docker.sock:/var/run/docker.sock" -v portainer_data:/data
portainer/portainer-ce:linux-arm
<code>

Start **Portainer** instance with ability to manage local **Docker Desktop
for Windows**.
<code>
C:\Users\Varimathras>docker run -d -p 8000:8000 -p 9000:9000 --name
"Portainer" --restart always -v "/var/run/docker.sock:/var/run/docker.sock"
portainer/portainer-ce
```

Start **Portainer** instance on **RPi**.

```
$ docker run -d -p 8000:8000 -p 9000:9000 --name "Portainer" --restart
always -v "/var/run/docker.sock:/var/run/docker.sock" -v
portainer_data:/data portainer/portainer-ce:linux-arm
```

DokuWiki

Upgrade DokuWiki

Run:

```
cd /workspace/Dokuwiki/
./dokuwiki.sh
```

dokuwiki.sh

```
#!/bin/sh

# From: https://rpi-wifi:8443/docker
```

```
docker ps |grep dokuwiki|awk '{print $1}'|xargs docker stop
docker ps -a | grep dokuwiki | awk '{print $1}' | xargs docker container rm
docker image ls|grep dokuwiki|awk '{print $3}'|xargs docker rmi
docker-compose build; docker-compose up -d
```

Start on boot

```
$ sudo nano /lib/systemd/system/docker.service
```

```
After=media-usb.mount media-ssd.mount
Requires=media-usb.mount media-ssd.mount
```

From “unit files”:

```
$ sudo systemctl list-unit-files|grep mount
```

```
media-ssd.mount
generated       -
media-usb.mount
generated       -
```

Matching mount points:

```
$ df -kh
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       361G   85G  258G  25% /media/ssd
/dev/sdb1       7.3G  1.3G  5.7G  18% /media/usb
```

Random bits

Access Docker volumes from WSL Linux instance:

```
# mount -t drvfs '\\wsl.localhost\docker-desktop\mnt\docker-desktop-
disk\data\docker\volumes' /mnt/volumes
```

Compose files

Cacti

Portainer Agent

References

How to change the timezone in XEAMS

Delay docker startup until all shared folders is mounted #458

How To Use Systemctl to Manage Systemd Services and Units

In a WSL2 distro - what determines the files that appear under: /wsl/{some-other-distro}

[Docker]