Docker

Useful staff for Docker and not only.

- Free Space from Docker container.
- Useful Docker commands
- Install Docker on Debian
- Install Docker ANY Linux Distro

Free Space from Docker container.

The below commands are useful to free up space from containers leave unused images, volumes, networks, logs.
From the cli where there is Docker installed run the below commands.

Commands

```
docker image prune -f -a
docker system prune --volumes -f -a
find /var/lib/docker/containers/ -type f -name "*.log" -delete
```

Download script

Or you can download the below shell script and add a Cron Job to run every 1 hour.

Download : **docker-freeup-space.sh**

Cron Job

Run crontab to add the job.

```
crontab -e
```

Then go to the end of the lines and add the below.

```
0 * * * * /path-of-the-script/docker-freeup-space.sh
```

Save and Exit

Done.

Ins0mniA

Useful Docker commands

Stop all Docker containers

To stop all Docker containers from the command line, you can use the following command:

```
docker stop $(docker ps -q)
```

Explanation:

- **docker ps -q**: This command lists the container IDs of all running containers.
- **docker stop**: This stops the containers provided as arguments.

So, **$(docker ps -q)** will pass all running container IDs to docker stop, effectively stopping all of them.

Remove ALL Docker containers Networks

WARNING (USED / UNUSED Networks) This will remove all Networks from ALL container even the one you use.

```
docker network rm $(docker network ls -q)
```

Explanation:

- **docker network ls -q**: Lists all network IDs (including those that are in use).
- **docker network rm**: Removes networks by their IDs.

This will attempt to remove all networks, including the default ones if they are not in use by containers. *However, if a network is in use by a container, Docker won't allow you to remove it* without first disconnecting the containers.

If you really want to remove all networks including the ones in use, you'd have to stop and remove the containers using those networks first.

Remove all Docker volumes (including unused ones)

To remove all Docker volumes (including unused ones), you can use the following command:

```
docker volume prune -f -a
```

Explanation:

- **docker volume prune**: Removes all unused volumes (volumes that are not currently associated with any container).
- **-f or --force**: This flag forces the operation without asking for confirmation.
- **-a or --all**: Remove all unused volumes, not just anonymous ones

If you want to remove all volumes — even the ones in use by containers (which is dangerous and typically not recommended) — you'd need to remove the containers and then delete the volumes. **Here's how you can do it:**

1. Stop and remove all containers:

```
docker stop $(docker ps -q)
docker rm $(docker ps -aq)
```

2. Remove all volumes:

```
docker volume rm $(docker volume ls -q)
```

Caution:

- Removing volumes that are in use by containers can result in data loss, as volumes often contain persistent data for applications running in containers.
- Always double-check what volumes are being used and whether you need to keep any before removing them.

Remove all Docker images (including images that are not being used by any containers)

To remove all unused images (not in use by any container), including dangling images, you can run:

```
docker image prune -f -a
```

Explanation:

- **docker image prune**: This will remove all dangling images.
- **-f or --force**: This flag forces the operation without asking for confirmation.
- **-a or --all**: Remove all unused images, not just anonymous ones

To remove all Docker images (including images that are not being used by any containers), you can use the following command:

```
docker rmi $(docker images -q)
```

Explanation:

- **docker images -q**: Lists the IDs of all images on your system.
- **docker rmi**: Removes the images by their IDs.

If you have containers using some of these images:
If there are containers still using these images, *Docker will not let you remove them. To force removal*, you'd need to stop and remove the containers first.
Here's how to do it:

1. Stop all running containers:

```
docker stop $(docker ps -q)
```

2. Remove all containers:

```
docker rm $(docker ps -aq)
```

3. Remove all images:

```
docker rmi $(docker images -q)
```

Caution:

- Removing all images will mean you lose access to those images unless you pull them again, and you may need to rebuild any images you were working on.
- Double-check before running these commands if you're in a production environment, as this can disrupt your workflows!

Remove all stacks of containers in Docker

To remove all stacks of containers in Docker, you'll typically be dealing with Docker Swarm. A stack in Docker refers to a collection of services deployed in a Docker Swarm, often using docker stack deploy.

If you want to remove all stacks (and the services, networks, and containers associated with them), you can do so with the following command:

```
docker stack rm $(docker stack ls -q)
```

Explanation:

- **docker stack ls -q**: Lists the names of all deployed stacks.
- **docker stack rm**: Removes the stack by name, along with the services and containers related to it.

This will remove all stacks deployed in your Docker Swarm environment.

Important Notes:

- This command removes the entire stack — containers, networks, services, and volumes that are part of the stack will be removed.
- If you only want to remove specific stacks, replace $(docker stack ls -q) with the name of the stack you want to remove.

Ins0mniA

Install Docker on Debian

? One-Line Download & Execute

```
apt install -y curl && curl -sSL https://docs.greenhome.stream/attachments/37 -o install_docker_portainer.sh && chmod +x install_docker_portainer.sh && ./install_docker_portainer.sh
```

Method A :

1. Install Docker on Debian 12

Before you install Docker Compose and Portainer, ensure that Docker is installed on your Debian 12 system.

Step 1: Update and Install Dependencies

```
apt update
apt install -y apt-transport-https ca-certificates curl
```

Step 2: Add Docker’s Official GPG Key

```
curl -fsSL https://download.docker.com/linux/debian/gpg | tee /etc/apt/trusted.gpg.d/docker.asc
```

Step 3: Add Docker's Official Repository

```
echo "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list
```

Step 4: Install Docker

```
apt update
apt install -y docker-ce docker-ce-cli containerd.io
```

Step 5: Verify Docker Installation

```
docker --version
```

Method B :

2. Install Docker Compose and Enable APT Updates

Step 1: Add Docker Compose APT Repository

Docker Compose is not included in the default Debian repositories, but you can use Docker’s official repository for Compose.

Create a Docker Compose APT repository source:

```
echo "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs) stable" |
tee /etc/apt/sources.list.d/docker.list
```

Step 2: Install Docker Compose

Now, install Docker Compose from Docker's APT repository:

```
apt update
apt install -y docker-compose-plugin
```

This version of Docker Compose (the plugin version) integrates directly into the Docker CLI and will update with `apt update` and `apt upgrade.`

Step 3: Verify Docker Compose Installation

```
docker compose version
```

Method C :

3. Install Portainer Using Docker Compose (Optional)

Portainer is a popular web UI to manage Docker environments. You can easily deploy it using Docker Compose.

Step 1: Create the docker-compose.yml File

Create a `docker-compose.yml` file in a directory where you want to manage Portainer. You can place it anywhere you like, but let’s say you create a directory called `portainer`:

```
mkdir -p ~/portainer
cd ~/portainer
```

Now create the docker-compose.yml file using your favorite editor:

```
nano docker-compose.yml
```

Paste the following content for the Portainer setup:

```
version: "3.9"
services:
  portainer:
    image: portainer/portainer-ce:latest
    container_name: portainer
    restart: always
    networks:
      - portainer_network
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - portainer_data:/data
    ports:
      - 9000:9000
    environment:
      - TZ=Europe/Athens  # Set your timezone

networks:
  portainer_network:
    driver: bridge

volumes:
  portainer_data:
    driver: local
```

Step 2: Start Portainer

Once the **`docker-compose.yml`** file is ready, run the following command to start Portainer :

```
docker compose up -d
```

This will pull the Portainer image, create the necessary containers, and start the Portainer web UI. You can now access it by going to **`http://<your-server-ip>:9000`** in your web browser.

Step 3: Access Portainer

When you visit Portainer for the first time, you'll be prompted to set up an admin account. After that, you can start managing your Docker environment through the web interface.

Method D :

1. Install Docker

Step 1:

Install docker using the official script, by executing the below command.

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

Check Update :

4. Ensure Automatic Updates for Docker and Docker Compose

Since you added Docker’s official repositories, Docker Compose will be updated via the same process as Docker itself when you run :

```
apt update && apt upgrade -y
```

Summary

1. **Docker** : Installed and running using the official Docker repositories.
2. **Docker Compose** : Installed via Docker's APT repository and integrated with Docker.
3. **Portainer** : Deployed using Docker Compose, with an example `docker-compose.yml` file to start the Portainer service.

You should now have Docker, Docker Compose, and Portainer running on Debian 12, with updates managed through `apt`.

Download the files

Script universal-health-monitor.sh

Attachment link : install_docker_portainer.sh

Ins0mniA

Install Docker ANY Linux Distro

Docker Installer & Verifier

A production-ready bash script that detects whether Docker is already installed and, if not, installs Docker CE + the Docker Compose plugin using the official repository for your Linux distribution.

? One-Line Download & Execute :

```
apt update && apt install -y curl && clear && curl -s
https://docs.greenhome.stream/attachments/69 | bash
```

?? English

Features

- **Smart detection** — if Docker and Compose are already installed and working, the script exits cleanly without touching anything.
- **Multi-distro support** — Debian, Ubuntu, Raspberry Pi OS, Fedora, RHEL, Rocky, AlmaLinux, CentOS, Arch, Manjaro, openSUSE, SLES, Alpine.
- **Real verification at every step** — binary presence, daemon reachability, Compose plugin, and a final `hello-world` container run.
- **Auto-installs missing dependencies** (`curl`, `gnupg`, `lsb-release`, etc.) before touching Docker.
- **Colored output**: orange `[PROC]`, green `[ OK ]`, red `[FAIL]`, cyan `[INFO]`.
- **Idempotent** — safe to run multiple times.
- **Sudo-aware** — works as root or as a regular user with sudo.
- **Adds your user to the `docker` group** so you don't need `sudo docker` after logout/login.

Requirements

- A supported Linux distribution (see above).
- Either root access or a user in the `sudo` group.
- Internet connection (to reach the Docker repository).

Usage

Download and run:

```
curl -fsSL -o install-docker.sh https://example.com/install-docker.sh
chmod +x install-docker.sh
./install-docker.sh
```

Or if you already have it locally:

```
chmod +x install-docker.sh
./install-docker.sh
```

What it does, step by step

1. **Clears the screen** and shows a banner.
2. **Sets up sudo** — uses root if already root, otherwise requires sudo.
3. **Detects OS family** via `/etc/os-release` and picks the right package manager.
4. **Checks if Docker is installed** — verifies the binary, the daemon, and the Compose plugin.
5. If **everything is OK** → exits with a green success message. Nothing is installed.
6. If **only Compose is missing** → installs only the Compose plugin.
7. If **Docker is missing** → adds the official Docker repository for your distro, installs `docker-ce`, `docker-ce-cli`, `containerd.io`, `docker-buildx-plugin`, `docker-compose-plugin`, enables and starts the service, runs `hello-world` as a final sanity check, and adds your user to the `docker` group.

Exit codes

- `0` — success (either already installed or freshly installed).
- Any non-zero — a step failed; a red `[FAIL]` message explains what.

Troubleshooting

- **"permission denied" after install** — log out and log back in so the new `docker` group membership takes effect, or run `newgrp docker`.
- **Behind a proxy** — export `HTTP_PROXY` / `HTTPS_PROXY` before running the script.
- **Old Docker from distro repo** — uninstall `docker.io` / `podman-docker` first with your package manager, then re-run this script to get the official Docker CE.

?? ????????

????????????

- **Έξυπνος εντοπισμός** — αν το Docker και το Compose είναι ήδη εγκατεστημένα και λειτουργούν, το script τερματίζει καθαρά χωρίς να αγγίξει τίποτα.
- **Υποστήριξη πολλών διανομών** — Debian, Ubuntu, Raspberry Pi OS, Fedora, RHEL, Rocky, AlmaLinux, CentOS, Arch, Manjaro, openSUSE, SLES, Alpine.
- **Πραγματική επαλήθευση σε κάθε βήμα** — ύπαρξη binary, προσβασιμότητα daemon, Compose plugin, και τελικό τρέξιμο του `hello-world`.
- **Αυτόματη εγκατάσταση εξαρτήσεων** (`curl`, `gnupg`, `lsb-release` κ.λπ.) πριν την εγκατάσταση του Docker.
- **Έγχρωμη έξοδος**: πορτοκαλί `[PROC]`, πράσινο `[ OK ]`, κόκκινο `[FAIL]`, κυανό `[INFO]`.
- **Idempotent** — ασφαλές να τρέξει πολλές φορές.
- **Υποστηρίζει sudo** — λειτουργεί ως root ή ως κανονικός χρήστης με sudo.
- **Προσθέτει τον χρήστη σας στην ομάδα** `docker` ώστε να μη χρειάζεται `sudo docker` μετά από logout/login.

????????????

- Υποστηριζόμενη διανομή Linux (βλ. παραπάνω).
- Πρόσβαση root ή χρήστης στην ομάδα `sudo`.
- Σύνδεση στο διαδίκτυο (για πρόσβαση στο αποθετήριο του Docker).

?????

Κατεβάστε και τρέξτε:

```
curl -fsSL -o install-docker.sh https://example.com/install-docker.sh
chmod +x install-docker.sh
./install-docker.sh
```

Ή, αν το έχετε ήδη τοπικά:

```
chmod +x install-docker.sh
./install-docker.sh
```

?? ?????, ???? ???? ????

1. **Καθαρίζει την οθόνη** και εμφανίζει banner.
2. **Ρυθμίζει sudo** — χρησιμοποιεί root αν είστε ήδη root, διαφορετικά απαιτεί sudo.
3. **Εντοπίζει την οικογένεια OS** μέσω του `/etc/os-release` και επιλέγει τον κατάλληλο package manager.

4. **Ελέγχει αν είναι εγκατεστημένο το Docker** — επαληθεύει binary, daemon, και το Compose plugin.
5. Αν **όλα είναι εντάξει** → τερματίζει με πράσινο μήνυμα επιτυχίας. Δεν εγκαθιστά τίποτα.
6. Αν **λείπει μόνο το Compose** → εγκαθιστά μόνο το Compose plugin.
7. Αν **λείπει το Docker** → προσθέτει το επίσημο αποθετήριο Docker για τη διανομή σας, εγκαθιστά `docker-ce`, `docker-ce-cli`, `containerd.io`, `docker-buildx-plugin`, `docker-compose-plugin`, ενεργοποιεί και ξεκινά την υπηρεσία, τρέχει το `hello-world` ως τελικό έλεγχο, και προσθέτει τον χρήστη στην ομάδα `docker`.

??????? ??????

- `0` — επιτυχία (είτε ήταν ήδη εγκατεστημένο, είτε εγκαταστάθηκε τώρα).
- Οτιδήποτε άλλο — κάποιο βήμα απέτυχε· ένα κόκκινο `[FAIL]` μήνυμα εξηγεί τι.

???????????? ????????????

- **"permission denied" μετά την εγκατάσταση** — κάντε logout/login ώστε να εφαρμοστεί η νέα συμμετοχή στην ομάδα `docker`, ή τρέξτε `newgrp docker`.
- **Πίσω από proxy** — ορίστε `HTTP_PROXY`/`HTTPS_PROXY` πριν τρέξετε το script.
- **Παλιό Docker από το αποθετήριο της διανομής** — απεγκαταστήστε πρώτα το `docker.io` / `podman-docker` με τον package manager σας, και μετά ξανατρέξτε αυτό το script για να πάρετε το επίσημο Docker CE.

License

MIT — use, modify, and redistribute freely.