

Proxmox and Docker Bible

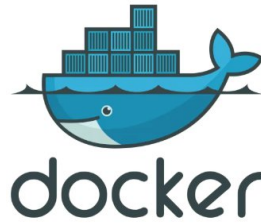


Table of Contents

1 Introduction.....	2
Chapter 1: What is Proxmox?	3
Chapter 2: Virtual Machines vs. Proxmox LXC Containers.....	3
Virtual Machines (VMs)	3
LXC Containers.....	3
Chapter 3: What is Docker?	4
Chapter 4: Installing the Proxmox server	5
Chapter 5: Creating an LXC container in Proxmox.....	7
Chapter 6: Install Docker in the LXC Container	13
Chapter 7: Docker Training	16
7.1 Creating a simple Ubuntu server using Docker.....	17
7.2 Using a docker compose file to create the same container.....	18
Chapter 8: Adding Dockge to Docker	20
8.1 Installing Dockge You can watch our video to see how we did this click here	20
Chapter 9: Our Best Docker Containers for 2025	22
9.1 NextCloud	22
9.2 Homarr.....	22
9.3 Dockge	22
9.4 Portainer	22
9.5 SmokePing	22
9.7 Droppy	23
Chapter 10: Docker Cheat Sheet	24

1 Introduction

If you are reading this document the you are either interested in setting up your own **Home-lab** or you are wanting to replace your **VMware ESXi** server with the best **Open-source Type 1 Hypervisor VM Server** featuring **LXC Containers** apart from Virtual Machines.

This document is going to explain some of the concepts regarding creating a Docker installation in Proxmox using “Best Practices”

Installation

We will also provide detailed procedures for doing the three installations needed to get Docker to work optimally in Proxmox.

Firstly we will Install Proxmox then create an LXC container and then we will show you how to install Docker there, and finally we will provide the Best Docker Applications to install.

Docker Tutorial

We will also provide a Docker Tutorial so that you can make use of the Docker installation

Docker Cheat-Sheet

Finally we will provide you with a Docker Cheat-Sheet

And lots of additional Chapters with Interesting things for you

Chapter 1: What is Proxmox?



Proxmox is an open-source server virtualization platform. It allows users to run and manage virtual machines (VMs) and containers from a central web-based interface. Proxmox combines two virtualization technologies: KVM (Kernel-based Virtual Machine) for full virtualization and LXC (Linux Containers) for lightweight container-based virtualization.

Proxmox VE (Virtual Environment) is popular in homelabs and enterprise environments due to its powerful features, user-friendly interface, and zero licensing cost. It includes built-in tools for backup, snapshots, live migration, clustering, and high availability.

Chapter 2: Virtual Machines vs. Proxmox LXC Containers

Virtual Machines (VMs)

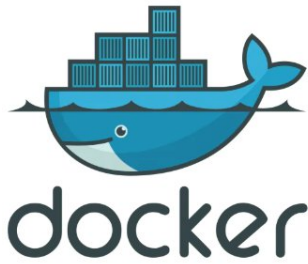
- VMs emulate entire hardware environments.
- Each VM runs a full operating system, including its own kernel.
- VMs are isolated and resource-intensive.
- Better for running different operating systems (e.g., Windows on Linux).

LXC Containers

- LXC containers share the host's Linux kernel.
- They are more lightweight and efficient.
- Faster to start and consume fewer resources.
- Suitable for running Linux-based applications with lower overhead.

In Proxmox, VMs are best for heavy or diverse OS workloads, while LXC containers are ideal for Linux services that benefit from lower resource use and faster performance.

Chapter 3: What is Docker?



Docker is a platform for developing, shipping, and running applications inside containers. Containers are lightweight, portable, and run the same regardless of the underlying infrastructure.

Unlike traditional virtual machines, Docker containers package only the application and its dependencies, sharing the host operating system's kernel. This makes them more efficient and faster to start.

Docker simplifies application deployment, scaling, and updates. It is widely used in modern DevOps workflows and microservices architecture. With Docker, developers can ensure their applications run the same in development, testing, and production.

Chapter 4: Installing the Proxmox server



You need to Download the latest ISO file here :

<https://www.proxmox.com/en/downloads>

We selected this file:

https://enterprise.proxmox.com/iso/proxmox-ve_8.4-1.iso

Step 1 Create a Bootable Thumbdrive

You would use balenaEtcher to create the bootable ThumbDrive: <https://etcher.balena.io/>

Step 2 you need to define the Network Parameters for your server:

```
# Hostname(FQDN) :
```

```
rhino.loseyourip.com
```

```
# IP Address (CIDR) :
```

```
10.154.2.188
```

```
# Gateway :
```

```
10.165.2.3
```

```
# DNS Server
```

```
8.8.8.8
```

If you need a DNS name for your Server you can use DynuDNS : <https://www.dynu.com/en-US>

Step 3 Boot the server from the installation Media

You have 2 choices depending on your hardware :

1. Install from a bootable USB Thumbdrive
2. Install from a DVD

Insert the installation media and start your server

Use the Installation media and retart the server hardware

wait for installation software to start

Choose : Install using Graphical Interface GUI

-> Install using Graphical Interface GUI

Accept the End User Licence Agreement (EULA)

-> Click Next Button

Select Country

-> United States

-> New York

keyboard

-> US

-> Click Next Button

Enter Password and Email

-> Click Next Button

Enter FQDN Details (See above) You need to type these values

rhino.loseyourip.com

10.154.2.188

10.154.2.3

8.8.8.8

-> Click Next Button

You will have a summary ensure this is correct

-> Click Next Button Now watch this video <https://youtu.be/ThyZhVWKiZ4>

Chapter 5: Creating an LXC container in Proxmox

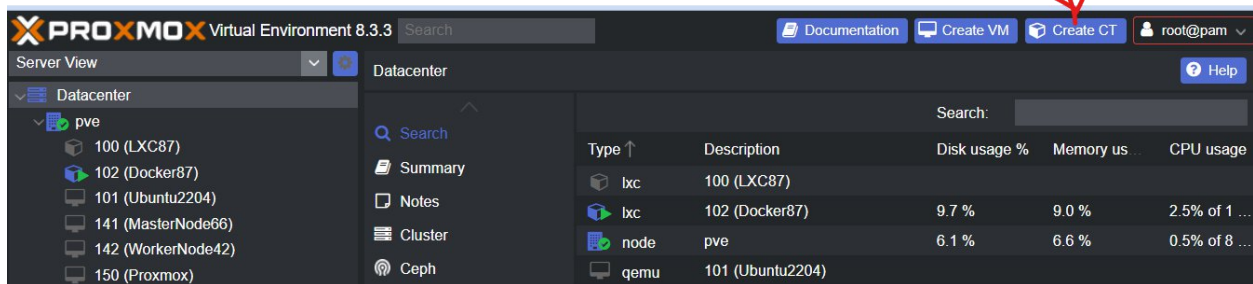


We now Install a LXC Container in Proxmox

We Open our Proxmox server in the browser:

<https://upupa.loseyourip.com:8006/>

Step 1: Click on Create Container Button



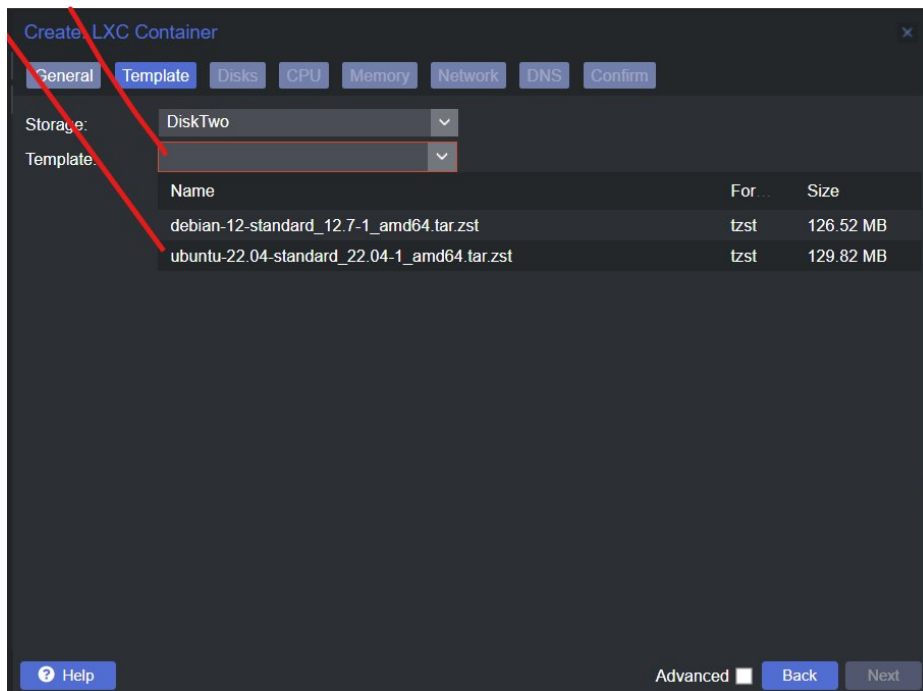
Step 2: Enter Server Name, Password and confirm password then press Next

The screenshot shows the 'Create LXC Container' window with the following fields and controls:

- General Tab:** Includes tabs for General, Template, Disks, CPU, Memory, Network, DNS, and Confirm.
- Node:** A dropdown menu with 'pve' selected.
- CT ID:** A dropdown menu with '103' selected.
- Hostname:** A text input field containing 'DockerServer'.
- Unprivileged container:** A checkbox that is checked.
- Nesting:** A checkbox that is checked.
- Resource Pool:** A dropdown menu.
- Password:** A text input field with masked characters (dots).
- Confirm password:** A text input field with masked characters (dots).
- SSH public key(s):** A large text area for pasting SSH keys.
- Load SSH Key File:** A button to load an SSH key file.
- Buttons:** 'Help' (with a question mark icon), 'Advanced' (with a checkbox), 'Back', and 'Next'.

Red arrows are drawn on the image to highlight the 'Hostname' field, the 'Password' and 'Confirm password' fields, and the 'Next' button.

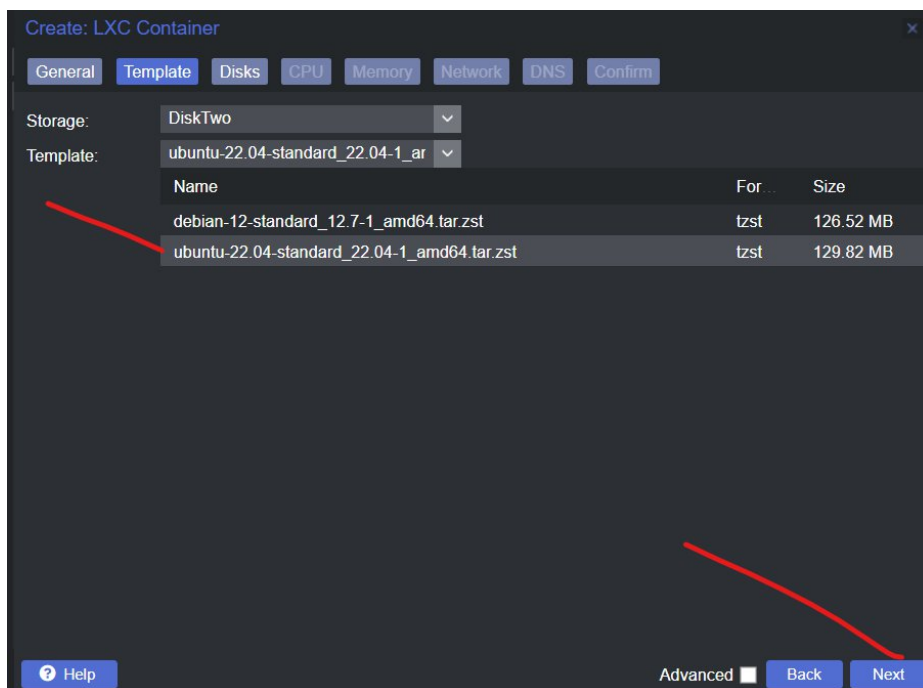
Step 3: Click Template (Dropdown) and select Ubuntu-22-04



The screenshot shows the 'Create LXC Container' dialog with the 'Template' tab selected. The 'Storage' dropdown is set to 'DiskTwo'. The 'Template' dropdown is open, showing a list of templates. A red arrow points to the dropdown menu.

Name	For ...	Size
debian-12-standard_12.7-1_amd64.tar.zst	tzst	126.52 MB
ubuntu-22.04-standard_22.04-1_amd64.tar.zst	tzst	129.82 MB

Now Click on Next



The screenshot shows the 'Create LXC Container' dialog with the 'Template' tab selected. The 'Storage' dropdown is set to 'DiskTwo'. The 'Template' dropdown is set to 'ubuntu-22.04-standard_22.04-1_ar'. A red arrow points to the 'Next' button.

Name	For ...	Size
debian-12-standard_12.7-1_amd64.tar.zst	tzst	126.52 MB
ubuntu-22.04-standard_22.04-1_amd64.tar.zst	tzst	129.82 MB

Step 4: Specify Storage 100GB and press Next

Create: LXC Container

General Template **Disk** CPU Memory Network DNS Confirm

rootfs Storage: DiskTwo

Disk size (GiB): 100

+ Add

? Help Advanced ☐ Back Next

Step 5: Unless you CPU has more then one core click Next

Create: LXC Container

General Template Disks **CPU** Memory Network DNS Confirm

Cores: 1

? Help Advanced ☐ Back Next

Step 6: Set memory to 8192GB also SWAP to 8192GB and Click Next

The screenshot shows the 'Create: LXC Container' dialog with the 'Memory' tab selected. The 'Memory (MiB)' field is set to 8192 and the 'Swap (MiB)' field is also set to 8192. Red arrows point to these fields. At the bottom right, a red arrow points to the 'Next' button. The 'Advanced' checkbox is unchecked.

Create: LXC Container

General Template Disks CPU Memory Network DNS Confirm

Memory (MiB): 8192

Swap (MiB): 8192

Help Advanced Back Next

Step 7: Enter IP Address followed by /12 and Gateway click on Next

The screenshot shows the 'Create: LXC Container' dialog with the 'Network' tab selected. The 'Name' is 'eth0', 'MAC address' is 'auto', 'Bridge' is 'vmbro', and 'VLAN Tag' is 'no VLAN'. The 'Firewall' checkbox is checked. The 'IPv4' section has 'Static' selected, 'IPv4/CIDR' is '10.154.2.33/12', and 'Gateway (IPv4)' is '10.154.2.3'. The 'IPv6' section has 'Static' selected, 'IPv6/CIDR' is 'None', and 'Gateway (IPv6)' is empty. Red arrows point to the 'IPv4/CIDR' and 'Gateway (IPv4)' fields. At the bottom right, a red arrow points to the 'Next' button. The 'Advanced' checkbox is unchecked.

Create: LXC Container

General Template Disks CPU Memory Network DNS Confirm

Name: eth0

MAC address: auto

Bridge: vmbro

VLAN Tag: no VLAN

Firewall: ☒

IPv4: ☐ Static ☐ DHCP

IPv4/CIDR: 10.154.2.33/12

Gateway (IPv4): 10.154.2.3

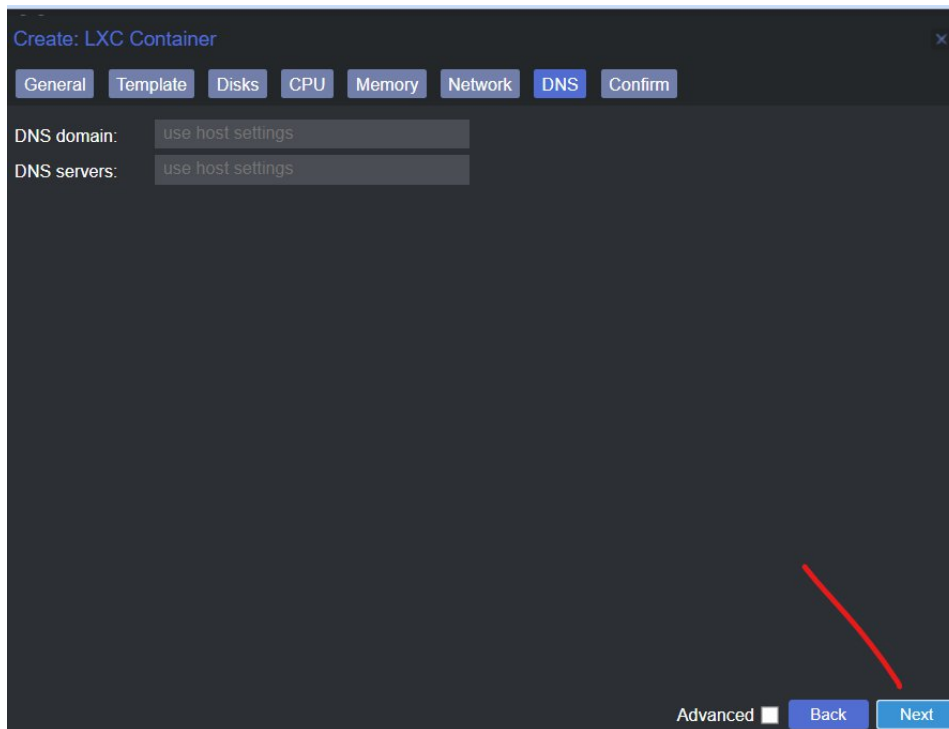
IPv6: ☐ Static ☐ DHCP ☐ SLAAC

IPv6/CIDR: None

Gateway (IPv6):

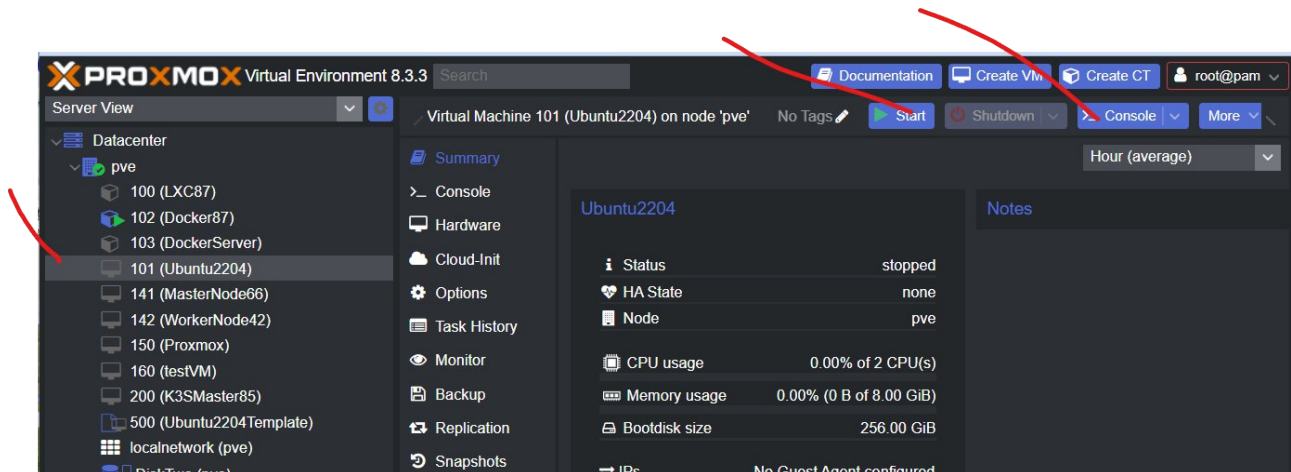
Help Advanced Back Next

Step 8: Click on Next



Step 9: If you are satisfied with the Server settings Click on Next

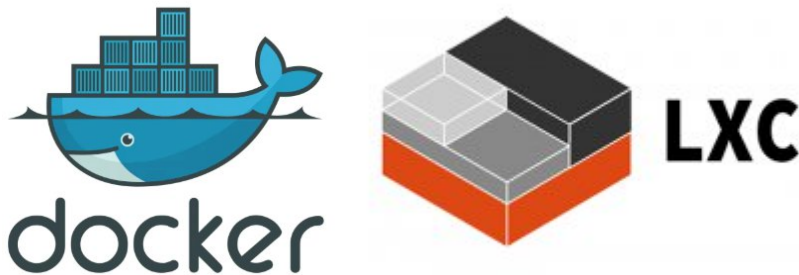
Step 10: Click on the LXC server Id 101 the click on Start then click on Console



Congratulations we now have created an LXC Container in Proxmox

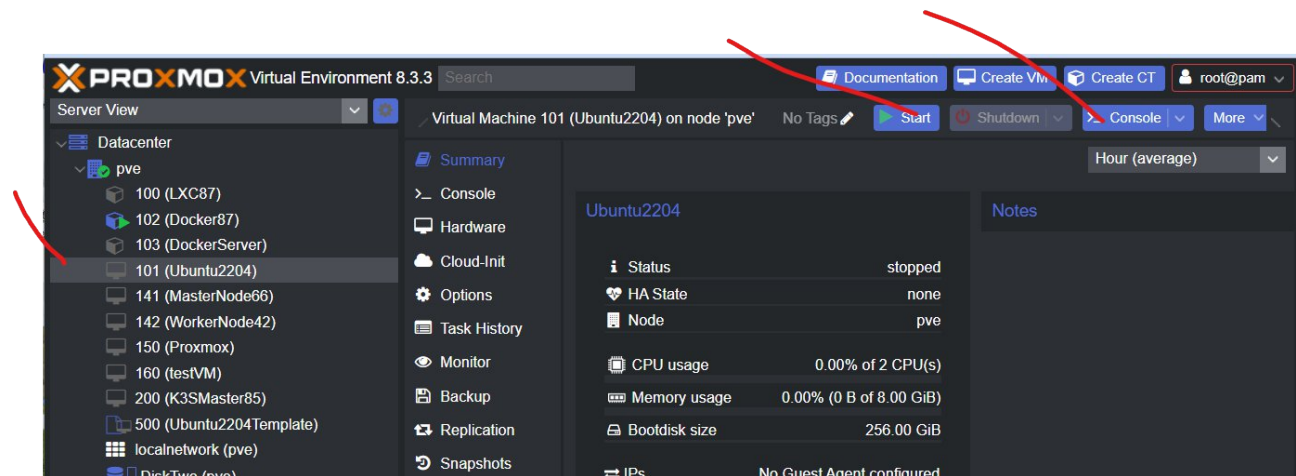
Chapter 6: Install Docker in the LXC Container

We now install Docker in our Proxmox LXC Container



Start the LXC Container

Step 1: Click on the LXC server Id 101 then click on Start then click on Console



We now have created the LXC Container

In the Console type these commands:

```
sudo apt update -y
```

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common -y
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```

```
apt-cache policy docker-ce
```

```
sudo apt update -y
```

```
sudo systemctl enable docker
```

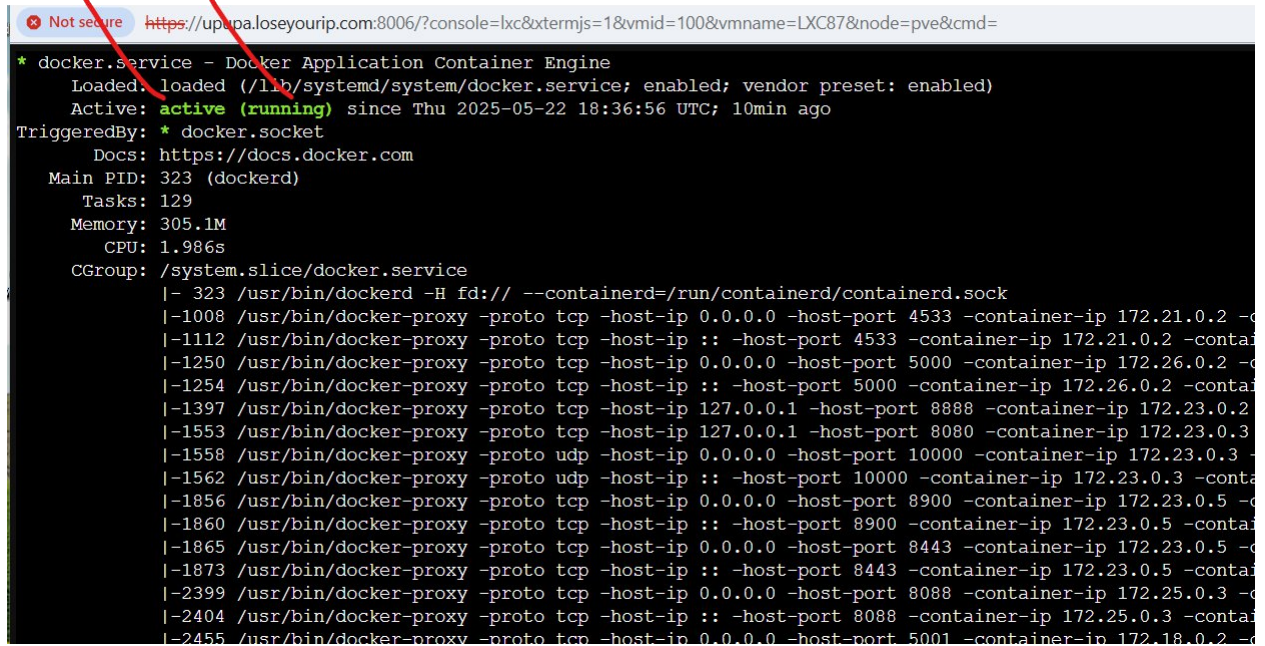
```
sudo systemctl start docker
```

Verify the Docker Service has started

Run this command:

```
sudo systemctl status docker
```

You should get:



```
* docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2025-05-22 18:36:56 UTC; 10min ago
 TriggeredBy: * docker.socket
     Docs: https://docs.docker.com
    Main PID: 323 (dockerd)
      Tasks: 129
     Memory: 305.1M
        CPU: 1.986s
    CGroup: /system.slice/docker.service
            └─ 323 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
               └─ 1008 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 4533 -container-ip 172.21.0.2 -c
                  └─ 1112 /usr/bin/docker-proxy -proto tcp -host-ip :: -host-port 4533 -container-ip 172.21.0.2 -c
                     └─ 1250 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 5000 -container-ip 172.26.0.2 -c
                        └─ 1254 /usr/bin/docker-proxy -proto tcp -host-ip :: -host-port 5000 -container-ip 172.26.0.2 -c
                           └─ 1397 /usr/bin/docker-proxy -proto tcp -host-ip 127.0.0.1 -host-port 8888 -container-ip 172.23.0.2 -c
                              └─ 1553 /usr/bin/docker-proxy -proto tcp -host-ip 127.0.0.1 -host-port 8080 -container-ip 172.23.0.3 -c
                                 └─ 1558 /usr/bin/docker-proxy -proto udp -host-ip 0.0.0.0 -host-port 10000 -container-ip 172.23.0.3 -c
                                    └─ 1562 /usr/bin/docker-proxy -proto udp -host-ip :: -host-port 10000 -container-ip 172.23.0.3 -c
                                       └─ 1856 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 8900 -container-ip 172.23.0.5 -c
                                          └─ 1860 /usr/bin/docker-proxy -proto tcp -host-ip :: -host-port 8900 -container-ip 172.23.0.5 -c
                                             └─ 1865 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 8443 -container-ip 172.23.0.5 -c
                                                └─ 1873 /usr/bin/docker-proxy -proto tcp -host-ip :: -host-port 8443 -container-ip 172.23.0.5 -c
                                                   └─ 2399 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 8088 -container-ip 172.25.0.3 -c
                                                      └─ 2404 /usr/bin/docker-proxy -proto tcp -host-ip :: -host-port 8088 -container-ip 172.25.0.3 -c
                                                         └─ 2455 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 5001 -container-ip 172.18.0.2 -c
```

Test docker is installed properly

Run this command:

docker version

You should get:

```
root@LXC87:~# docker version
Client: Docker Engine - Community
 Version:      27.5.1
 API version:  1.47
 Go version:   go1.22.11
 Git commit:   9f9e405
 Built:        Wed Jan 22 13:41:05 2025
 OS/Arch:      linux/amd64
 Context:      default

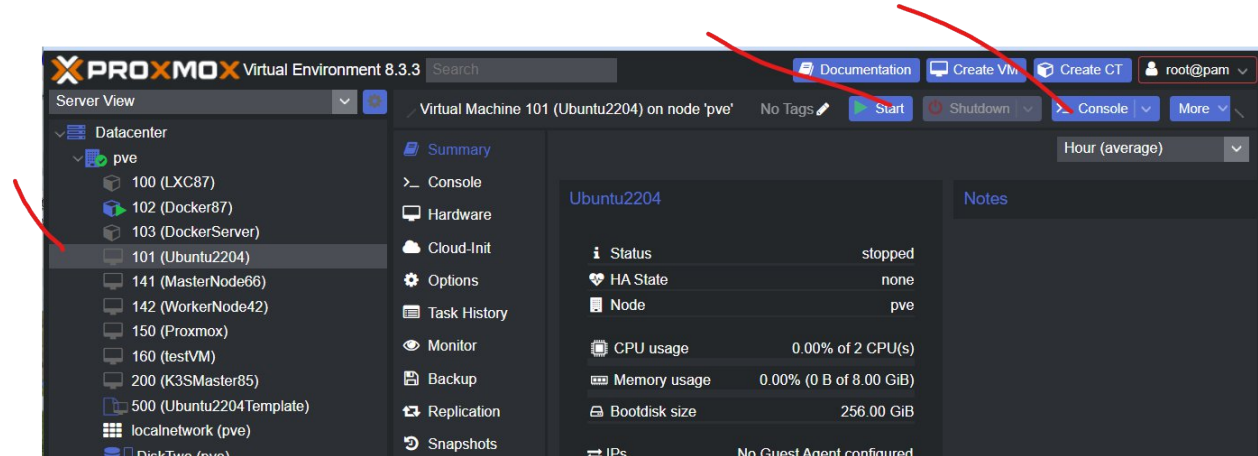
Server: Docker Engine - Community
 Engine:
  Version:      28.1.1
  API version:  1.49 (minimum version 1.24)
  Go version:   go1.23.8
  Git commit:   01f442b
  Built:        Fri Apr 18 09:52:18 2025
  OS/Arch:      linux/amd64
  Experimental: false
 containerd:
  Version:      1.7.25
  GitCommit:    bcc810d6b9066471b0b6fa75f557a15a1cbf31bb
 runc:
  Version:      1.2.4
  GitCommit:    v1.2.4-0-g6c52b3f
 docker-init:
  Version:      0.19.0
  GitCommit:    de40ad0
root@LXC87:~#
```

Congratulations you have now installed Docker.

Chapter 7: Docker Training

Open the LXC Container where you installed Docker:

Step 1 Select the LXC Container then click on Start then click on Console



Step 2: In the console run these commands after logging in

```
sudo su -  
  
mkdir DockerTraining -p  
  
cd DockerTraining  
  
cdDockerTraini
```


7.1 Creating a simple Ubuntu server using Docker

**** 1.1 Created from Docker Commands ****

1. Pull the Ubuntu Image:

```
docker pull ubuntu:latest  
docker image ls
```

2. Run a Container with Bash Terminal:

```
# Windows users run :  
winpty docker run -it ubuntu:latest bash  
  
# Mac and Linux users run this command:  
docker run -it ubuntu:latest bash
```

3. Running commands in the container :

```
ls -la /home  
apt update  
apt install curl
```

4. Deleting the image

```
docker ps -a  
docker container rm <CONTAINER ID>  
docker image ls  
docker image rm <IMAGE ID>  
docker image ls
```

7.2 Using a docker compose file to create the same container

Run these commands:

```
mkdir ubuntu -p
```

```
cd ubuntu
```

```
nano compose.yaml
```

Put this in the file: (dont add extra space this is a yaml file)

```
version: '3.9'
```

```
services:
```

```
  ubuntu-server:
```

```
    image: ubuntu:latest
```

```
    container_name: ubuntu_server
```

```
    volumes:
```

```
      - ./data:/opt/data
```

```
    stdin_open: true # Keeps the container interactive
```

```
    tty: true      # Allocates a pseudo-TTY
```

Run the following commands:

```
# pull the Docker images
docker compose pull

# Start the container in detached mode
docker compose up -d

# To access the container, Windows users run :
winpty docker exec -it ubuntu_server bash

# To access the container, Mac and Linux users run this command:
docker exec -it ubuntu_server bash

# Now Inside the container run these commands:
ls -la /home
ls -la /opt
cd /opt/data
touch test1.txt
docker compose down
```

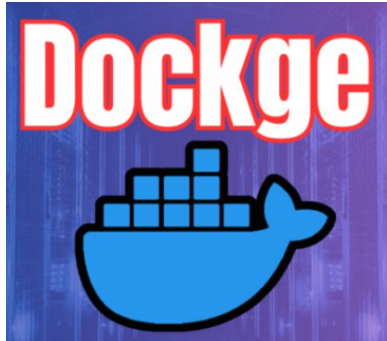
If you want to keep this server running then instead of running this command:

```
docker compose up
```

You run this in detached mode like this:

If you want to watch the video where we did this lesson here is the link:

Chapter 8: Adding Dockge to Docker



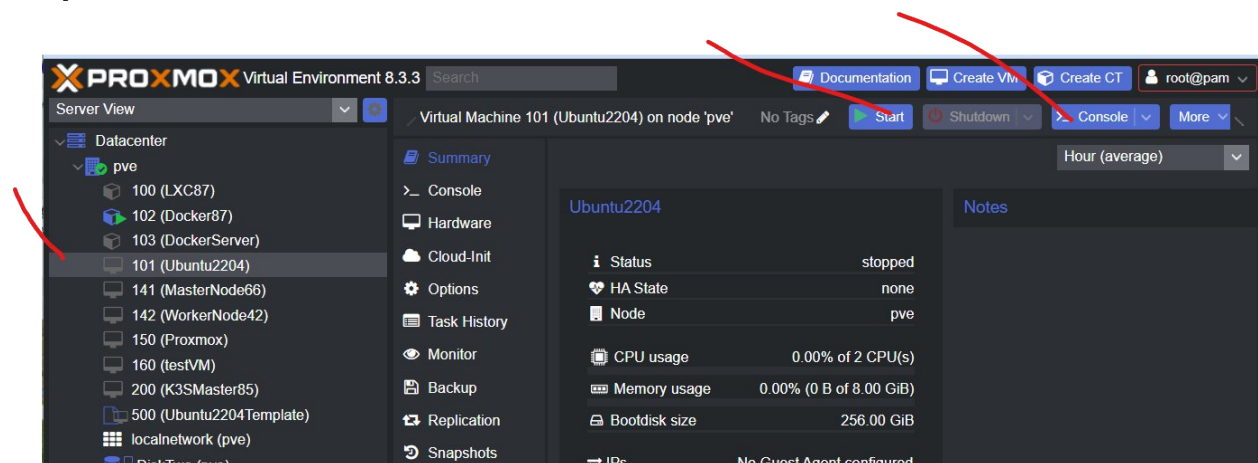
We provide instructions to install one of the finest Graphical User Interface for Docker

8.1 Installing Dockge

[You can watch our video to see how we did this click here](#)

Open the LXC Container where you installed Docker:

Step 1 Select the LXC Container then click on Start then click on Console



Step 2: Run these commands

```
# Create directories that store your stacks and stores Dockge's stack
mkdir -p /opt/stacks /opt/dockge
cd /opt/dockge
# Download the compose.yaml
curl https://raw.githubusercontent.com/louislam/dockge/master/compose.yaml --output
compose.yaml
# should you wish to customize this docker compose file
nano compose.yaml (however it is good)
# Start the server
docker compose up -d
```

Congratulations you have installed the most valuable program in Proxmox

Step 3: Open Dockge in the browser:

<http://your-lxc-containers-ip-address:5001/>

In our LXC server it is:

<http://10.154.2.87:5001/>

Chapter 9: Our Best Docker Containers for 2025

9.1 NextCloud

By self-hosting Nextcloud, ONLYOFFICE in Proxmox, say goodbye to Microsoft Office 365

(We used this to replace our Office 365 by self-hosting this)

We have two videos showing two ways to install this

(Watch both as we show different features there)

[Watch this video 1](#)

[Watch this video 2](#)

9.2 Homarr

Install Homarr one of the most popular Dashboards in Proxmox

(We use this to provide links and a dashboard to our infrastructure)

[Watch this video](#)

9.3 Dockge

One of the Best Graphical User Interfaces for Docker

(We use it for testing all the Docker containers before we publish videos)

[Watch this video](#)

9.4 Portainer

One of the Best Graphical User Interfaces for Docker

(We use this together with Dockge for managing our Docker infrastructure)

[Watch this video](#)

9.5 SmokePing

An Excellent Network Availability Tools test you local and International Bandwidth and Latency

[Watch this video](#)

9.6 Beszel

A lightweight, self-hosted server monitoring platform
(We use it to monitor our Docker containers)

[Watch this video](#)

9.7 Droppy

A self-hosted file storage server with a web interface to replace OneDrive.
(We use this to share files and collaborate)

[Watch this video](#)

Chapter 10: Docker Cheat Sheet

Manipulation Commands

`docker image ls` (This lists all the docker images)

`docker image rm IMAGE-ID` (this deletes a docker image)

`docker ps` (This lists all docker instances that are running)

`docker ps -a` (This lists all the docker images even those that are not running)

`docker container ls` (this lists all docker containers)

`docker container stop CONTAINER-ID`

`docker container rm CONTAINER-ID`

`docker container stop CONTAINER-NAME`

`docker container start CONTAINER-NAME`

Creation commands

`docker build -t IMAGE_NAME:version` (this builds a Docker image from a Dockerfile)

`docker build -t stubservice:latest .`

`docker compose up` (This starts a docker container)

`docker compose up -d` (This starts a docker container in DETACHED MODE)

`docker compose stop`

`docker compose start` (This starts a docker container after it was stopped)

`docker compose down` (this stops the container and deletes the image)

Connect to the Containing servers of Docker containers

`docker run -it ubuntu:latest bash` | create container from image and connect to its containing server

`docker exec -it ubuntu_server bash` | connect to containing server of existing and running container

Accessing the Logs

`docker logs CONTAINER-ID`

Final Thoughts

Thank you for your interest in our video and downloading this document.

Please visit our YouTube Channel and subscribe and watch our videos from start to end as we have not met our target for hours viewed.

Please leave us a comment on one of our YouTube videos after watching it.

Our YouTube Channel

<https://www.youtube.com/@dvp7388>