# Docker

- Docker Commands
- Jitsi-Meet
- How to Install Docker and Docker Compose on Debian 11
- Installing Docker and Docker Compose on Debian 12
- □□Docker / Docker Compose + Dockge Setup Guide (Debian 12 Bookworm)
- UrBackup Server & Client Installation Guide
- □□ Mounting an External USB Hard Drive on Debian

# Docker Commands

## General CMD

## Show list of dockers:

```
docker ps
```

## copy CMD

```
cp <source file.ext> <new file.ext>
```

## Host Web Server

```
python3 -m http.server 1337
```

## Bash into Docker container

```
docker exec -it <container name/id> bash
```

accesed via SSH allows to run cmds as the container

## Make Dockerfile

https://stackoverflow.com/questions/45152426/create-docker-compose-file-from-dockerfile

Enable SSH and Root

cd sr

# Jitsi-Meet

Prerequisites

- This image assumes that you are using a reverse proxy such as jwilder/nginx-proxy and optionally the Let's Encrypt Proxy Companion @ https://github.com/JrCs/docker-letsencrypt-nginx-proxy-companion in order to serve your pages.

This is a complex series of images and relies on all packages to be working together. You will also need to open ports on your firewall (See below).

Installation

Automated builds of the image are available on Docker Hub and is the recommended method of installation.

docker pull tiredofit/jitsi-meet docker pull tiredofit/jitsi-prosody docker pull tiredofit/jitsi-videobridge docker pull tiredofit/jitsi-jicofo

Quick Start

The quickest way to get started is using docker-compose. See the examples folder for a working docker-compose.yml that can be modified for development or production use. All you will need to do is change the `HOST` and `VIRTUAL_HOST,LETSENCRYPT_HOST` variables and the system will automatically generate certificates for you and the system will function.

Set various environment variables to understand the capabilities of this image.

Map persistent storage for access to configuration and data files for backup.

Configuration

# Data-Volumes

The following directories are used for configuration and can be mapped for persistent storage.

## jitsi-prosody

| Directory | Description |
| --- | --- |

| | |
|---|---|
| `/certs` | Needed to Automatically Generate Certificates for other containers |

## jitsi-videobridge

| Directory | Description |
|---|---|
| `/certs` | Needed to share certificates between containers for Self Signed variants |

## jitsi-jicofo

| Directory | Description |
|---|---|
| `/certs` | Needed to share certificates between containers for Self Signed variants |

## jitsi-meet

| Directory | Description |
|---|---|
| `/assets/jitsi-meet` | Put your custom config.js/interfaceConfig.js in here and it will be added on bootup |

# Environment Variables

Below is the complete list of available options that can be used to customize your installation.

## jitsi-prosody

| Parameter | Description |
|---|---|
| `HOST` | Hostname of your server e.g. `meet.example.com` Should be same as all other hostnames |
| `JITSI_VIDEO_PASS` | Jitsi Video Bridge Secret e.g. `secret3` |
| `JICOFO_PASS` | Jitsi Conference Focus Secret e.g. `secret2` |

| Parameter | Description |
| --- | --- |
| `JICOFO_USER_PASS` | Jitsi Conference Focus User Secret e.g. `secret1` |

## jitsi-videobridge

| Parameter | Description |
| --- | --- |
| `HOST` | Hostname of your server e.g. `meet.example.com` Should be same as all other hostnames |
| `PROSODY_HOST` | Container Name of your prosody server e.g. `prosody` |
| `JITSI_VIDEO_PASS` | Jitsi Video Bridge Secret e.g. `secret3` |
| `NETWORK_MODE` | Network Mode `NAT` or `HOST` - Defaults to `NAT` |

## jitsi-jicofo

| Parameter | Description |
| --- | --- |
| `HOST` | Hostname of your server e.g. `meet.example.com` Should be same as all other hostnames |
| `PROSODY_HOST` | Container Name of your prosody server e.g. `prosody` |
| `JICOFO_PASS` | Jitsi Conference Focus Secret e.g. `secret2` |
| `JICOFO_USER_PASS` | Jitsi Conference Focus User Secret e.g. `secret1` |

## jitsi-meet

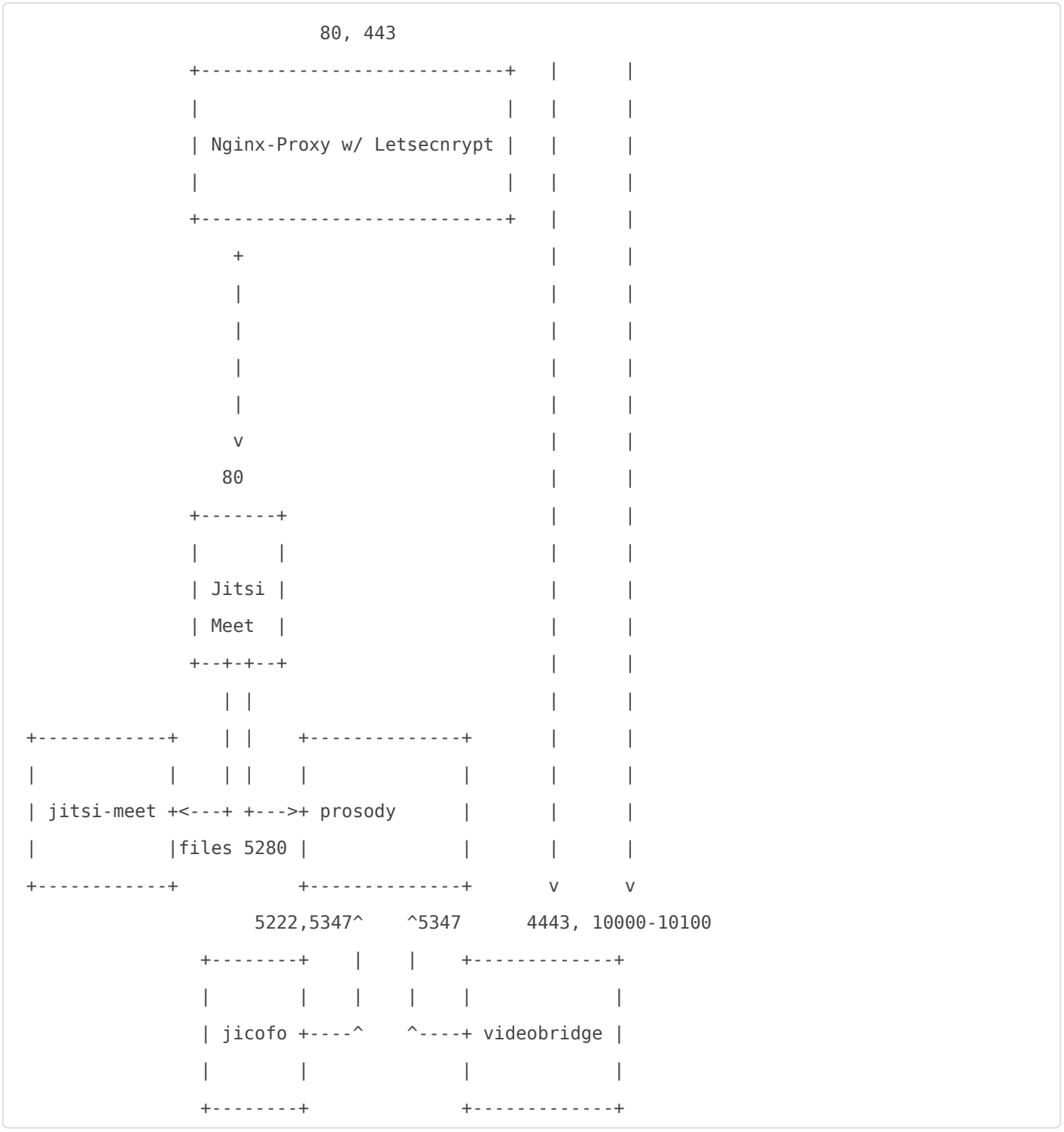
| Parameter | Description |
| --- | --- |
| `PROSODY_HOST` | Container Name of your prosody server e.g. `prosody` |

# Networking

This set of images relies on network ports being exposed to the outside world. 80, 443 for the initial web proxy (which should already be handled by the `jwilder/nginx-proxy` image) and then you must open port `4443` and `10000-10100/udp` to the outside world otherwise you *will* have issues with video

or audio.

See below diagram:




The following ports are exposed.

## jitsi-prosody

| Port | Description |
| --- | --- |
| `5222` | Prosody Clent Listening Port |
| `5280` | Prosody Server Listening Port |
| `5347` | Prosody Components |

## jitsi-videobridge

| Port | Description |
| --- | --- |
| `443` | Jitsi Video Bridge Harvester Port |
| `5347` | Prosody Components |
| `4443` | Jitsi Video Bridge Harvester Port |
| `10000-20000/udp` | Web RTC / ICE |

## jitsi-jicofo

| Port | Description |
| --- | --- |
| `5222` | Prosody Client Port |
| `5347` | Prosody Components |

## jitsi-meet

| Port | Description |
| --- | --- |
| `80` | Nginx Listening Port |
| `5280` | Prosody Server Listening Port |

Maintenance

# Shell Access

For debugging and maintenance purposes you may want access the containers shell.

docker exec -it (whatever your container name is e.g. jitsi-meet) bash

References

- https://github.com/jitsi/jitsi-meet/blob/master/doc/manual-install.md

# How to Install Docker and Docker Compose on Debian 11

## Prerequisites

- A server running Debian 11.
- A root password is configured on the server.

## Getting Started

First, it is recommended to update your system package cache to the latest version. You can update them using the following command:

```
apt-get update -y
```

Once you are done, install other required dependencies using the following command:

```
apt-get install apt-transport-https software-properties-common ca-certificates curl gnupg lsb-release -y
```

## Install Docker

By default, the latest version of Docker is not included in the Debian 11 official repository. So you will need to add the Docker CE repository to the APT. You can add it using the following command:

```
curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add -
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs)
```

Once the Docker repository is added, update the repository and install the Docker CE with the following command:

```
apt-get update -y
apt-get install docker-ce docker-ce-cli -y
```

After the installation, verify the Docker CE version using the following command:

```
docker version
```

You should get the following output:

```
Client: Docker Engine - Community
 Version:           20.10.8
 API version:       1.41
 Go version:        go1.16.6
 Git commit:        3967b7d
 Built:             Fri Jul 30 19:54:22 2021
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server: Docker Engine - Community
 Engine:
  Version:          20.10.8
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.16.6
  Git commit:       75249d8
  Built:            Fri Jul 30 19:52:31 2021
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.4.9
  GitCommit:        e25210fe30a0a703442421b0f60afac609f950a3
 runc:
  Version:          1.0.1
  GitCommit:        v1.0.1-0-g4144b63
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

# Manage Docker Services

You can manage the Docker service easily using the systemd utility.

To start a Docker service, run the following command:

```
systemctl start docker
```

To restart a Docker service, run the following command:

```
systemctl restart docker
```

To stop a Docker service, run the following command:

```
systemctl stop docker
```

To enable the Docker service to start at system reboot, run the following command:

```
systemctl enable docker
```

To check the Docker status, run the following command:

```
systemctl status docker
```

You should see the status of Docker in the following output:

```
? docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2021-09-10 07:19:35 UTC; 27s ago
TriggeredBy: ? docker.socket
       Docs: https://docs.docker.com
   Main PID: 29018 (dockerd)
      Tasks: 7
     Memory: 32.6M
        CPU: 407ms
     CGroup: /system.slice/docker.service
             ??29018 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Sep 10 07:19:34 debian11 dockerd[29018]: time="2021-09-10T07:19:34.809035575Z" level=info msg="s
Sep 10 07:19:34 debian11 dockerd[29018]: time="2021-09-10T07:19:34.809219999Z" level=info msg="c
Sep 10 07:19:34 debian11 dockerd[29018]: time="2021-09-10T07:19:34.809410545Z" level=info msg="C
Sep 10 07:19:34 debian11 dockerd[29018]: time="2021-09-10T07:19:34.897972507Z" level=info msg="L
Sep 10 07:19:35 debian11 dockerd[29018]: time="2021-09-10T07:19:35.186940748Z" level=info msg="D
Sep 10 07:19:35 debian11 dockerd[29018]: time="2021-09-10T07:19:35.298681937Z" level=info msg="L
Sep 10 07:19:35 debian11 dockerd[29018]: time="2021-09-10T07:19:35.356364773Z" level=info msg="D
Sep 10 07:19:35 debian11 dockerd[29018]: time="2021-09-10T07:19:35.357524464Z" level=info msg="D
Sep 10 07:19:35 debian11 systemd[1]: Started Docker Application Container Engine.
Sep 10 07:19:35 debian11 dockerd[29018]: time="2021-09-10T07:19:35.401626151Z" level=info msg="A
```

# Run a Container Using Docker

You can use the **docker run** command to download any image and run it inside the container.

For example, run the following command to download Debian image and run a container:

```
docker run --rm -it --name test debian:latest /bin/sh
```

You should get the following output:

```
Unable to find image 'debian:latest' locally
latest: Pulling from library/debian
955615a668ce: Pull complete
Digest: sha256:08db48d59c0a91afb802ebafc921be3154e200c452e4d0b19634b426b03e0e25
Status: Downloaded newer image for debian:latest
```

```
#
```

Run the following command to exit from the Debian container

```
#exit
```

# Install Docker Compose

Although you can install Docker Compose from the official Debian repositories, it is several minor versions behind the latest release, so in this tutorial you'll install it from Docker's GitHub repository. The command that follows is slightly different than the one you'll find on the Releases page. By using the `-o` flag to specify the output file first rather than redirecting the output, this syntax avoids running into a "permission denied" error caused when using `sudo`.

Check the current release and, if necessary, update it in the command that follows:

```
sudo curl -L https://github.com/docker/compose/releases/download/1.25.3/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
```

Next we'll set the permissions:

```
sudo chmod +x /usr/local/bin/docker-compose
```

Then we'll verify that the installation was successful by checking the version:

```
docker-compose --version
```

This will print out the version we installed:

You should see the following output:

```
docker-compose version 1.29.2, build 5becea4c
docker-py version: 5.0.0
CPython version: 3.7.10
OpenSSL version: OpenSSL 1.1.0l  10 Sep 2019
```

# Docker Compose Commands

To run a Docker compose file, run the following command:

```
docker-compose up -d
```

To stop all running containers, run the following command:

```
docker-compose down
```

To pause and unpause the running container, run the following command:

```
docker-compose pause
docker-compose unpause
```

To list all running containers, run the following command:

```
docker-compose ps
```

To check the logs of running services, run the following command:

```
docker-compose logs
```

# Installing Docker and Docker Compose on Debian 12

Before installing Docker, ensure your Debian system is up-to-date with the following command:

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

Shell

Once your system is updated, install the necessary packages to allow apt to use a repository over HTTPS:

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common gnupg2 -y
```

Shell

Next, add the official GPG key of Docker:

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

Shell

Add the Docker repository to APT sources:

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

Shell

Update your package index and install Docker CE (Community Edition):

```
sudo apt update && sudo apt install docker-ce -y
```

Shell

To ensure Docker starts on boot, use the following command:

```
sudo systemctl enable docker
```

Shell

Verify the Docker installation by running the hello-world image:

```
sudo docker run hello-world
```

Shell

# Installing Docker Compose on Debian 12

Docker Compose is a tool for defining and running multi-container Docker applications. To install Docker Compose, first, download the latest version from the official GitHub repository:

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

Shell

Next, set the appropriate permissions to make the binary executable:

```
sudo chmod +x /usr/local/bin/docker-compose
```

Shell

Verify the installation by checking the version of Docker Compose:

```
docker-compose --version
```

Shell

At this point, Docker and Docker Compose are installed and ready for use on your Debian 12 system.

# ?Docker / Docker Compose + Dockge Setup Guide (Debian 12 Bookworm)

## Prerequisites

- Debian Bookworm
- Root or sudo privileges
- Internet access

---

## 1. ? Update the System

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

---

## 2. ? Install Dependencies

```
apt install apt-transport-https ca-certificates curl software-properties-common gnupg2 -y
```

---

## 3. ? Add Docker GPG Key and Repository

> **Note:** `apt-key` is deprecated. A better method is to store the key in `/etc/apt/keyrings/`, but this follows the legacy method used in the original steps.

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

---

## 4. ? Install Docker Engine + Compose Plugin

```
apt update && apt install docker-ce -y
```

This installs:

- Docker Engine
- Docker CLI
- Docker Compose plugin (`docker compose`)
- Supporting packages like `containerd.io`, `iptables`, etc.

---

# 5. ? Enable Docker on Boot

```
systemctl enable docker
```

---

# 6. ? (Optional) Install Legacy `docker-compose` Binary

```
curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
docker-compose --version
```

---

# 7. ? Set Up Dockge (Docker Container Manager)

```
mkdir -p /opt/stacks /opt/dockge
cd /opt/dockge
curl https://raw.githubusercontent.com/louislam/dockge/master/compose.yaml --output compose.yaml
docker compose up -d
```

This:

- Creates project directories
- Downloads the `compose.yaml` for Dockge
- Deploys Dockge using Docker Compose

Once running, Dockge will be available via Docker (check `docker ps` for ports).

# ? Result

- Docker installed and running
- Docker Compose (v1 and plugin) available
- Dockge deployed and managing stacks under `/opt/stacks`

---

# ? Notes

- If `apt-key` shows a deprecation warning, consider migrating the key to `/etc/apt/keyrings/docker.gpg` and referencing that path in the `.list` file.
- Dockge stores stacks in `/opt/stacks`. You can start new apps here using Dockge's web UI.
- Port for Dockge is defined in the `compose.yaml`.

# UrBackup Server & Client Installation Guide

## 1. **Prerequisites**

- You're operating as **root**—no need for `sudo` (and it may not even be installed).
- On Linux, use filesystems **EXT4** or **XFS** (necessary for image downloads)
- Have at least **1–2 GB free** in `/tmp` (default unpacking site).

---

## 2. **Install UrBackup Server (Debian/Ubuntu-based)**

```
# Update base packages
apt update
apt install curl gnupg2 software-properties-common
```

### Option A: PPA method

```
add-apt-repository ppa:uroni/urbackup
apt update
apt install urbackup-server
```

### Option B: Direct .deb download

```
wget https://hndl.urbackup.org/Server/2.5.33/urbackup-server_2.5.33_amd64.deb
dpkg -i urbackup-server_2.5.33_amd64.deb
apt install -f   # fix missing dependencies
```

> You'll be prompted to choose a backup directory—e.g., `/media/backup/urbackup`

---

## 3. **Enable & Verify Service**

```
systemctl enable urbackupsrv
systemctl start urbackupsrv
systemctl status urbackupsrv
ss -antpl | grep -E "5541[34]"
```

- Ports: **55413** (FastCGI), **55414** (HTTP)

---

# 4. Access Web Interface

Open your browser to `http://<server-ip>:55414`.

1. Go to **Settings → Users**, create an **admin account** to secure the web UI
2. Tweak general settings like backup retention, temporary path, etc., via `/etc/default/urbackupsrv` or the UI

---

# 5. Install UrBackup Client (Linux)

On the client machine (as root):

```
TF=$(mktemp)
wget "https://hndl.urbackup.org/Client/2.5.25/UrBackup%20Client%20Linux%202.5.25.sh" -O "$TF"
sh "$TF"
rm -f "$TF"
```

- You'll be asked to choose a **snapshot mechanism**:
  - **LVM**, **datto**, or **none** (bare-metal backup needs LVM or root snapshot).

---

# 6. Register & Configure Client

Back on the **server web UI**:

1. Click **"Add new client"**, choose network or internet type.
2. Copy the install command and run it on the client (as above).

On the client, add directories to back up:

```
urbackupclientctl add-backupdir -d /path/to/important
```

Repeat for all directories you want to include

# 7. **Monitor & Manage Backups**

- Use **Status**, **Backups**, and **Activities** tabs to check client status and jobs
- **Delete old backups** manually in the UI or run `urbackupsrv cleanup` on the server for automated cleanup

# 8. **Uninstall Client** (if needed)

```
uninstall_urbackupclient
```

---

# Why UrBackup?

- Simple web interface
- Supports **file-level + image backups** while live
- Efficient **deduplication** – minimizes storage use during repeated backups
- Works across Linux, Windows, FreeBSD

---

# Quick Tips

- As root, **drop `sudo`.**
- Choose **EXT4** or **XFS** on servers for image backup support.
- Pick the right **snapshot mode** on clients: LVM or none.
- Always protect your **web UI** with an admin account.
- Monitor **ports 55413/55414** and ensure they're reachable.

# ? Mounting an External USB Hard Drive on Debian

# ? Mounting an External USB Hard Drive on Debian

## ? Overview

This guide outlines how to properly identify, mount, and persistently configure an external USB hard drive on a Debian-based system. It is particularly useful for setups involving backup containers like `urbackup` or media/file storage solutions.

---

## ? Step 1: Identify the USB Drive

```
lsblk
```

Look for a device (e.g., `/dev/sdb1`) with the expected size and no mount point.

```
sudo blkid /dev/sdb1
```

This confirms the filesystem type and gets the UUID (used for persistent mounting).

---

## ? Step 2: Install Required NTFS Support

If the drive is formatted as NTFS (common for Windows drives), install the NTFS driver:

```
sudo apt update
sudo apt install ntfs-3g
```

**Note:** Do not install `fuse` on Debian 12 (Bookworm); it conflicts with `fuse3`.

---

# ? Step 3: Create a Mount Point

```
sudo mkdir -p /mnt/usbbackup
```

---

# ? Step 4: Configure /etc/fstab

Edit the fstab file to auto-mount the drive at boot:

```
sudo nano /etc/fstab
```

Add this line (replace UUID with yours from `blkid`):

```
UUID=5E74F4D874F4B43D  /mnt/usbbackup  ntfs-3g
defaults,noatime,nofail,uid=1001,gid=1001,umask=0022,allow_other  0  2
```

| Option | Description |
|---|---|
| `ntfs-3g` | NTFS filesystem driver with write support |
| `noatime` | Improves performance by disabling access-time updates |
| `nofail` | Allows boot to continue if the drive is missing |
| `uid/gid` | Sets ownership for consistent Docker access |
| `umask=0022` | Applies `rwxr-xr-x` permissions |
| `allow_other` | Permits non-root users to access the mount |

---

# ? Step 5: Mount and Verify

```
sudo systemctl daemon-reexec
sudo mount -a
```

Check if it mounted successfully:

```
df -h | grep usbbackup
```

Confirm correct permissions:

```
ls -ld /mnt/usbbackup
```

---

# ? Example Output

```
/dev/sdb1 on /mnt/usbbackup type fuseblk (rw,nosuid,nodev,noatime,allow_other,blksize=4096)
```

---

# ?? Troubleshooting

- **Wrong filesystem type?**
  Make sure you used `ntfs-3g` and not `ext4` in your fstab line.
- **"Bad superblock" or mount error?**
  Use `dmesg | tail -30` to check for detailed kernel messages.
- **Permissions issue in Docker?**
  Ensure the Docker container's UID/GID matches the mount options.

---

# ? Related

- To mount an `ext4` drive, change `ntfs-3g` to `ext4` and remove `uid/gid/umask` options.
- To temporarily mount a USB without fstab:

  ```
  sudo mount -t ntfs-3g /dev/sdb1 /mnt/usbbackup
  ```