Installation Guides for Software and Services

Documentation on standing up homelab software's

- BookStack
  - BookStack Installation Guide
- Authentik
  - Authentik Docker Compose Install
  - Authentik Passwordless Login
  - Authentik OAuth/ OIDC Setup - Portainer
  - Authentik OAuth/ OIDC Setup - Home Assistnat
- Pangolin
  - Pangolin Install Guide
  - Post Install ACME Falining Fix
- Homarr Homepage
- Dockpeek
  - Installing Dockpeek
- Dockge
- Synology
  - NUT in Synology
- Komodo
  - Install Komodo Periphery Agents
- Proxmox

BookStack

Installation Guide for Internal Services

BookStack Installation Guide

This Guide goes trough steps necessary for creating a good well organized step by step process. This guide walks you through deploying BookStack using Docker and Docker Compose, with a working configuration that includes MariaDB, proper environment variables, and SSL disabled for local development.

BoookStack Installation Documentation

GitHub Repository

Prerequisites

- Docker and Docker Compose installed
- Portainer or terminal access

Steps

These steps are for terminal access

1. Create Project Directory

```
mkdir ~/bookstack-docker
cd ~/bookstack-docker
```

2. Generate Larvel app Key

```
docker run -it --rm --entrypoint /bin/bash lscr.io/linuxserver/bookstack:latest appkey
```

3. Create docker-compose.yml

- Change 'supersecurepassword' with unique password. Make sure they match for both environments

```
version: '3.8'

services:
  bookstack:
```

```
    image: lscr.io/linuxserver/bookstack:latest
    container_name: bookstack
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Etc/UTC
      - APP_URL=http://localhost:6875
      - DB_HOST=bookstack_db
      - DB_PORT=3306
      - DB_USERNAME=bookstack_user
      - DB_PASSWORD=supersecurepassword
      - DB_DATABASE=bookstack
      - APP_KEY=base64:YOUR_GENERATED_KEY_HERE
    volumes:
      - bookstack_config:/config
    ports:
      - 6875:80
    depends_on:
      - bookstack_db
    restart: unless-stopped

  bookstack_db:
    image: mariadb:10.5
    container_name: bookstack_db
    command: --ssl=OFF
    environment:
      - MYSQL_ROOT_PASSWORD=rootpassword
      - MYSQL_DATABASE=bookstack
      - MYSQL_USER=bookstack_user
      - MYSQL_PASSWORD=supersecurepassword
    volumes:
      - ./bookstack/db:/var/lib/mysql
    restart: unless-stopped

volumes:
  bookstack_config:
```

4. Start the Stack

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

- This will create containers, initialize the database, run Larvel migrations and serve BookStack on port 6875

5. Check Logs
    1. This will check if the bookstack started correctly. You should see Larvel migrations completing and no errors about SSL or DB access

```
docker logs bookstack
```

6. Inspect App Files
    1. You should see Laravel files like `artisan`, `routes/`, `app/`, etc.

```
docker exec -it bookstack /bin/bash
ls /app/www
```

7. Create Admin User
    1. Once completed and app is running properly, creating a local user with strong password is great way. Do this inside container

```
docker exec -it bookstack /bin/bash
cd /app/www
php artisan bookstack:create-admin
```

Optional Enhancements

- Presistent uploads/themes

```
volumes:
  - ./bookstack/uploads:/config/www/uploads
  - ./bookstack/themes:/config/www/themes
```

Authentik

Identity Provider self hosted on Internal Services VM

Authentik

Authentik Docker Compose Install

Authentik is an open-source Identity Provider (IdP) that helps you manage authentication and authorization across your apps and infrastructure. It supports:

- Single Sign-On (SSO) via OAuth2, OpenID Connect, SAML
- LDAP & SCIM integration
- Multi-factor authentication
- Reverse proxy for seamless app protection

Think of it as your self-hosted alternative to services like Okta or Auth0, but with full control and flexibility.

Prerequisites:

- Docker & Docker Compose

[Authentik Docker Compose Installation Guide]

Install Steps:

1. Open SSH and get to the device you want to run it on. (my case Overseer)
2. grab preconfigured yml

```
wget https://goauthentik.io/docker-compose.yml
```

If this is a fresh authentik installation, you need to generate a password and a secret key.

3. Run the following commands to generate a password and secret key and write them to your `.env` file:

```
echo "PG_PASS=$(openssl rand -base64 36 | tr -d '\n')" >> .env
echo "AUTHENTIK_SECRET_KEY=$(openssl rand -base64 60 | tr -d '\n')" >> .env
```

4. To enable error reporting, run the following command:

```
echo "AUTHENTIK_ERROR_REPORTING__ENABLED=true" >> .env
```

5. By default, authentik listens internally on port 9000 for HTTP and 9443 for HTTPS.

```
cd /docker/authentik/.env
```

6. To change the exposed ports to 80 and 443, you can set the following variables in `.env`:

```
COMPOSE_PORT_HTTP=80
COMPOSE_PORT_HTTPS=443
```

7. Startup docker compose

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

To start the initial setup, navigate to **http://<your server's IP or hostname>:9000/if/flow/initial-setup/**

Alternative Install Steps:

1. Open SSH and get to the device you want to run it on. (my case Overseer)
2. Create Directory

```
mkdir /docker/authentik
cd /docker/authentik
```

3. Create docker-compose.yml and edit it

```
nano docker-compose.yml #might need to use sudo if it doesn't give you access
```

```
version: '3.8'

services:
  postgresql:
    image: postgres:15
    environment:
      POSTGRES_DB: authentik
      POSTGRES_USER: authentik
      POSTGRES_PASSWORD: authentik
    volumes:
```

```
    - postgresql_data:/var/lib/postgresql/data

redis:
  image: redis:7
  volumes:
    - redis_data:/data

server:
  image: ghcr.io/goauthentik/server:latest
  depends_on:
    - postgresql
    - redis
  environment:
    AUTHENTIK_SECRET_KEY: "supersecretkey"
    AUTHENTIK_POSTGRESQL__HOST: postgresql
    AUTHENTIK_POSTGRESQL__USER: authentik
    AUTHENTIK_POSTGRESQL__PASSWORD: authentik
    AUTHENTIK_POSTGRESQL__NAME: authentik
    AUTHENTIK_REDIS__HOST: redis
  ports:
    - "8080:8000"  # Web UI
    - "9444:9443"  # Proxy port
  volumes:
    - authentik_media:/media
    - authentik_static:/static

worker:
  image: ghcr.io/goauthentik/worker:latest
  depends_on:
    - server
  environment:
    AUTHENTIK_SECRET_KEY: "supersecretkey"
    AUTHENTIK_POSTGRESQL__HOST: postgresql
    AUTHENTIK_POSTGRESQL__USER: authentik
    AUTHENTIK_POSTGRESQL__PASSWORD: authentik
    AUTHENTIK_POSTGRESQL__NAME: authentik
    AUTHENTIK_REDIS__HOST: redis
  volumes:
    - authentik_media:/media
    - /var/run/docker.sock:/var/run/docker.sock
```

```
volumes:
  postgresql_data:
  redis_data:
  authentik_media:
  authentik_static:
```

4. Create the .env file

```
nano .env #might need to run it with sudo
```

```
# Database credentials
PG_USER=authentik
PG_PASS=supersecurepassword123
PG_DB=authentik

# Authentik image tag
AUTHENTIK_IMAGE=ghcr.io/goauthentik/server
AUTHENTIK_TAG=2025.6

# Optional: HTTP/HTTPS ports (not forwarded externally)
COMPOSE_PORT_HTTP=9000
COMPOSE_PORT_HTTPS=9444

# Secret Key
AUTHENTIK_SECRET_KEY=your-super-secret-key
```

5. Start the stack

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

Once the stack is up, everything is finished installing you can check it with

```
docker-compose ps
```

To start the initial setup, navigate to http://<your server's IP or hostname>:9000/if/flow/initial-setup/.

Authentik Passwordless Login

Passwordless Login in Authentik allows us to login using passkey instead of password. This option provides higher security and faster authentication.

At the moment Passwordless Authentication only supposts WebAuth devicees (tokens, yubkey, 1password passkey).

Authentik Documentation on Passwordless Login

Steps to Set Up Passwordless Login Flow

1. Login to Authentik as Administrator
2. Click on F**lows and Stages** and click on **Flows**
3. Click **Create**
    1. Keep the name similar across the process for easier setup
    2. For Designation choose Authentication
4. Click on new created Flow
5. Click on **Stage Bindings** and choose **Create & Bind Stage**
    1. Choose Authenticator Validation Stage
    2. Click Next and add name similar to previous one
    3. Choose WebAuthn Authentication
    4. For not configured action choose **Force the user to configure an authenticator**
    5. For configuration stage find **default-autenticator-webauth-setup** and push over to the right
    6. Click Next and Finish
6. Click **Bind existing Stage**
    1. For Stage select **default-authentication-login** (or personal one)
    2. If you add Order number for previous part, add a higher number
    3. Click **Create**
7. Go back to Flows and select your **Welcome Page** or **defaulth-authentication-flow**
8. Go to **Stage Bindings** and for **Identification Stage** click **Edit Stage**
    1. Go to flow Settings
    2. Select **passwordless flow**

You should be ready to go

Authentik OAuth/ OIDC Setup - Portainer

Authentik uses many ways to connect to services, one being OAuth or Open ID Connect. This method is widely used on many services, such as Portainer.

Please follow Authentik and Portainer documentation

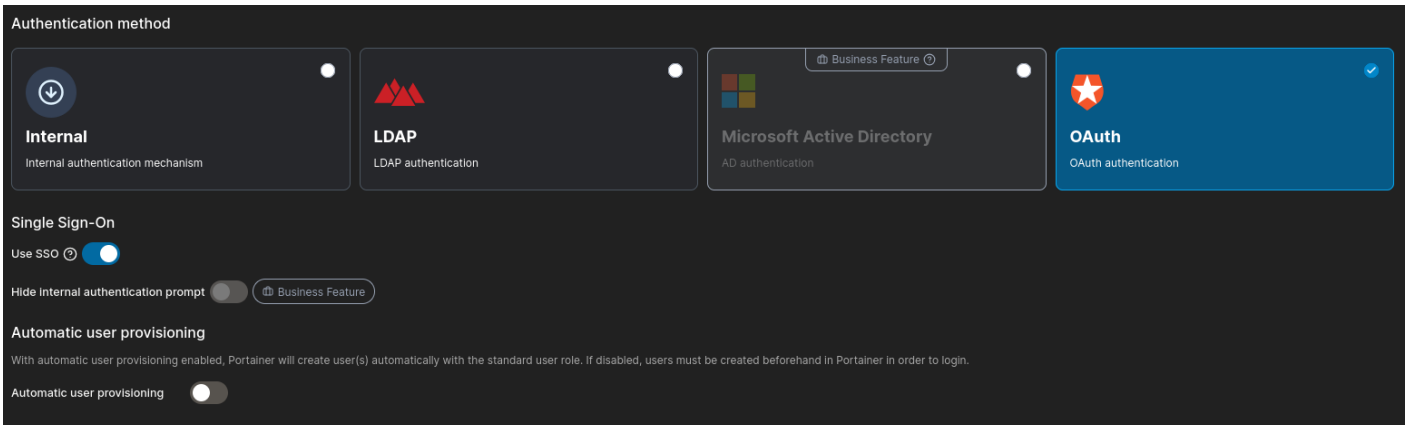
- Portainer OAuth Setup Documentation
- Authentik Portainer Integration Documentation

Authentik OAuth:

1. Login to Authentik **Admin Interface**
2. Go to **Applications** and select **Create with Provider**
    1. Choose a name and group
    2. Under **URI** in **Launch URL** enter **https://portainer.cyberpaw.org**
    3. Choose **Oauth2** Provider
    4. Name the provider same as application
    5. For Authorization Flow choose **Cyberpaw-authorization-flow** (or default one)
    6. Make sure **Confidential** is selected for Client Type
    7. Copy Client ID and Client Key
    8. In Redirect URIs enter **https://portainer.cyberpaw.org** (check portainer instructions for more detail)
    9. For Encryption key choose **default-authentic-self-signed-certificate**
    10. Under **Advanced flow** settings choose **Welcome to Authentick** (or default one)
    11. Under **Configure Bindings** click **Bind existing policy/group/users**
    12. Select **Group** and choose existing group that is authorized to use this service
    13. Review and Submit
3. The provider is created and should say it's connected to application

Portainer Steps:

1. Navigate to Portainer page and login
2. Under **Settings** go to **Authentication** and select **OAuth**
3. Enable **use SSO**



4. Choose **Automatic User Provisioning** allowing other Authentik users that don't have Portainer user can login
    1. If not selected you will need to create an account with same email as Authentik user
5. Scroll down to **OAuth Configuration**
    1. Copy and Paste all the field ID, secret and URLs from Provider information in Authentik
        1. Go back to Authentik **Admin Interface**
        2. Lower **Application** Section and click **Providers**
        3. Click on **Portainer** Provider and copy all the required information to Portainers OAuth Configuration
    2. For User Identification type "**email**"
    3. For Scope type "**email oauth provider**" **-**Portainer documentation says to use dashes but use space instead
    4. Save
6. Logout and you should see **Login with OAuth** button

Authentik OAuth/ OIDC Setup - Home Assistnat

Authentik uses many ways to connect to services, one being OAuth or Open ID Connect. This method is widely used on many services, such as Home Assistant. Home Assistant doesn't have native Open ID Connection, so we will need to use HACS for setup

Please follow Authentik and Portainer documentation

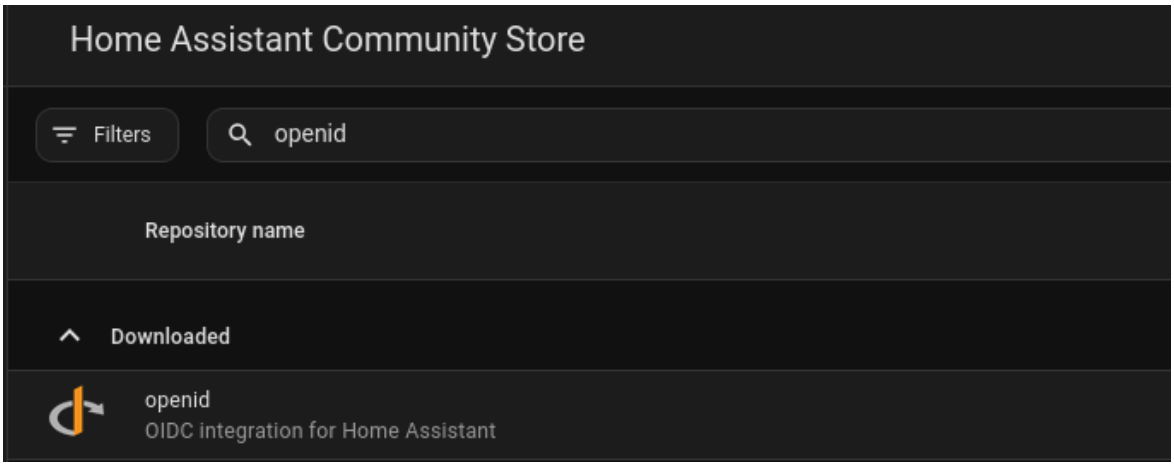
- Home Assistant OAuth Setup Documentation
- GitHub Hass-openid Setup with HACS

Authentik OAuth:

1. Login to Authentik **Admin Interface**
2. Go to **Applications** and select **Create with Provider**
    1. Choose a name and group
    2. Under **URI** in **Launch URL** enter **https://portainer.cyberpaw.org**
    3. Choose **Oauth2** Provider
    4. Name the provider same as application
    5. For Authorization Flow choose **Cyberpaw-authorization-flow** (or default one)
    6. Make sure **Confidential** is selected for Client Type
    7. Copy Client ID and Client Key
    8. In Redirect URIs enter http://overseer.cyberpaw.org:8123/auth/openid/callback
    9. For Encryption key choose **default-authentic-self-signed-certificate**
    10. Under **Advanced flow** settings choose **Welcome to Authentick** (or default one)
    11. Under **Configure Bindings** click **Bind existing policy/group/users**
    12. Select **Group** and choose existing group that is authorized to use this service
    13. Review and Submit
3. The provider is created and should say it's connected to application

Home Assistant Steps:

1. Login to Home Assistant with Admin
2. Open **HACS**
3. Search for hass-openid



4. Go to **Terminal** app on HA
5. Navigate to Your Home Assistant Config Directory

```
cd /config
```

6. Create custom_components Directory

```
mkdir -p /config/custom_components/openid
```

7. Download the Files from GitHub

```
git clone https://github.com/cavefire/hass-openid.git
cp -r hass-openid/custom_components/openid /config/custom_components/
```

8. Restart Home Assistant
9. Go back to Terminal and add following configuration to configuration.yaml file

```
  #OAuth with Authentik
openid:
  client_id: YOUR_CLIENT_ID
  client_secret: YOUR_CLIENT_SECRET
  configure_url: "https://auth.cyberpaw.org/application/o/home-assistant/.well-known/openid-
configuration"  # Replace with your Identity Provider's URL
  username_field: "email"  # Adjust based on your IdP's user info response
  scope: "openid profile email"
  block_login: false
  openid_text: "Login with Authentik"  # Text to display on the login page
```

10. Restart Home Assistant

If you want to disable the default Home Assistant login and only allow OpenID authentication, set `block_login` to `true` in your configuration

Pangolin

Pangolin Install and Setup Guide

Pangolin Install Guide

Pangolin is

Most of the guide is from their doc page, however, there is a part missing for making proxy redirect work properly.

Pangolin Quick Install Guide

VPS Hardening Security Guide

Pangolin Install Guide

This will resolve an issue of https://pangolin.cyberpaw.org/auth/initial-setup site not being reachable, or getting Invalid ssl cert error.

1. Login trough SSH to VPS server that is preset with necessary security steps
2. Download the installer

```
curl -fsSL https://digpangolin.com/get-installer.sh | bash
```

3. Run the installer

```
sudo ./installer
```

4. Once installer is finished Configure basic Settings from prompts. The installer will prompt you for essential configuration:
    1. Base Domain: Enter your root domain without subdomains (e.g., example.com)
    2. Dashboard Domain: Press Enter to accept the default pangolin.example.com or enter a custom domain
    3. Let’s Encrypt Email: Provide an email for SSL certificates and admin login
    4. Tunneling: Choose whether to install Gerbil for tunneled connections (default: yes). You can run Pangolin without tunneling. It will function as a standard reverse proxy.
    5. Email Configuration: Say no, if you don't have SMTP server set up
    6. CrowdSec: say Yes to install and self manager CrowdSec
5. Once installer is ready try to go to:

```
https://pangolin.example.com/auth/initial-setup
```

If you get Invalid SSL Certificate error or Site can't be reached continue with steps below

Traefik dynamic_config.yml Change

1. Navigate to Traefik Config Directory

```
cd /config/traefik
```

2. Backup existing file

```
cp dynamic_config.yml dynamic_config.yml.bak
```

3. Edit yml

```
nano dynamic_config.yml
```

4. Add new line in router part

```
    setup-router:
      rule: "Host(`pangolin.cyberpaw.org`) && PathPrefix(`/auth`)"
      service: api-service
      entryPoints:
        - websecure
      tls:
        certResolver: letsencrypt
```

5. Save and Exit
6. Restart traefik container

```
docker restart <traefik_container_name>
```

Now try to go to initial setup and follow initial steps.

Pangolin

Post Install ACME Falining Fix

This guide walks you through the exact steps to diagnose and fix ACME certificate issues during a Pangolin installation. These steps cover the most common real-world causes: DNS mismatches, blocked ports, Traefik misconfiguration, and redirect loops. Follow the checklist in order—each step rules out a specific failure point so you can quickly identify what’s wrong and get ACME issuing certificates again.

Troubleshoot Steps

Verify DNS is pointing to the correct server

ACME will always fail if DNS points to the wrong IP.

- `A yourdomain.com → <your VPS IP>`
- `A *.yourdomain.com → <your VPS IP>`

1. Check your server’s public IP and make sure it matches your DNS records

```
curl ifconfig.me
```

Test port 80 from outside the server

ACME HTTP-01 requires port 80 to be reachable publicly.

1. From your laptop or phone:

```
curl -I http://yourdomain.com
```

Interpret the result:

- **200 / 301 / 404** → Port 80 is open (good)
- **Timeout** → Firewall or provider is blocking port 80
- **Connection refused** → Traefik is not listening on port 80

Check VPS firewall (UFW)

```
sudo ufw status
```

You should see

```
80/tcp   ALLOW
443/tcp  ALLOW
```

If missing:

```
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
```

Check hosting provider firewall

For example Hetzner has an external firewall that overrides UFW

1. Go to your VPS dashboard
2. Server → Networking → Firewalls

```
TCP 80
TCP 443
```

If port 80 is missing → ACME will fail every time.

Confirm Traefik is listening on port 80

1. SSH into server and run following command

```
sudo ss -tulpn | grep :80
```

Expected:

```
docker-proxy ... LISTEN ... :80
```

If nothing is listening → Traefik didn’t bind to port 80.

Disable HTTP?HTTPS redirect during ACME

This is the most common Traefik issue.

If Traefik redirects ACME requests to HTTPS before a certificate exists, ACME fails.

1. SSH into the server, and go to dynamic-compose.yaml. Usually in config > traefik folder

```
main-app-router-redirect:
  entryPoints:
    - web
  middlewares:
    - redirect-to-https
```

2. Temporarily comment out the redirect:

```
# - redirect-to-https
```

3. Restart Traefik:

```
sudo docker compose restart traefik
```

Uncomment the redirect after successful redirect

Ensure ACME is using HTTP?01 on the correct entrypoint

In traefik onfig yaml

```
httpChallenge:
  entryPoint: web
```

Entrypoints must be:

```
entryPoints:
  web:
    address: ":80"
  websecure:
    address: ":443"
```

Homarr Homepage

Dockpeek

Docker Monitoring Tool

Installing Dockpeek

Here is the guide to install and standup Dockpeek tool that will monitor all docker and docker images on a server and remote servers with additional agent install

Github Repository

Install Dockpeek

1. SSH to Overseer machine, or get in trough Portainer/ Other GUI Compose Interface
2. Run following docker compose

```
services:
  dockpeek:
    container_name: dockpeek
    image: ghcr.io/dockpeek/dockpeek:latest
    environment:
      - SECRET_KEY=my_secret_key #change this
      - USERNAME=admin #change this
      - PASSWORD=admin #change this
    ports:
      - "3420:8000" #change 3420 as necessary for your setup. do not change 8000.
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock #this is how we connect to the docker-
socket.
    restart: unless-stopped #this is fine.
```

Adjust environment info for secure login

Install Dockpeek Agents for Remote Clients

If you have multiple server running dockers on a local network

1. Run following docker compose for installing agents

```
### Don't change anything here. Just deploy it as it is on the node/server you want to monitor
services:
  dockpeek-socket-proxy:
    image: lscr.io/linuxserver/socket-proxy:latest
    container_name: dockpeek-socket-proxy
    environment:
      - CONTAINERS=1
      - IMAGES=1
      - PING=1
      - VERSION=1
      - INFO=1
      - POST=1       # <-- This is needed for "Check for updates" operations
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
    read_only: true
    tmpfs:
      - /run
    ports:
      - "2375:2375"
    restart: unless-stopped
```

2. Once all agents are installed adjust main Dockpeek compose file by adding additional variables under environment

```
services:
  dockpeek:
    container_name: dockpeek
    image: ghcr.io/dockpeek/dockpeek:latest
    environment:
      - SECRET_KEY=my_secret_key #change this
      - USERNAME=admin #change this
      - PASSWORD=admin #change this

      # Docker Host 1 (This is our local server)
      - DOCKER_HOST_1_URL=unix:///var/run/docker.sock #this connects to the local docker socket.
      - DOCKER_HOST_1_NAME=LocalHost #this is the identifier that will show in the dashboard.
      - DOCKER_HOST_1_PUBLIC_HOSTNAME=192.168.0.18 #this is the IP address of the server (no http/https).
```

```
      # Docker Host 2 (This is a remote server on our local network)
      - DOCKER_HOST_2_URL=tcp://192.168.0.107:2375 #change the IP address to your remote server's IP address. Don't change anything else.
      - DOCKER_HOST_2_NAME=OpenCloud #this is the identifier that will show in the dashboard.
      - DOCKER_HOST_2_PUBLIC_HOSTNAME=192.168.0.107 #this is the IP address of the server (no http/https).

      # Docker Host 3 (This is a remote server on our local network)
      - DOCKER_HOST_3_URL=tcp://192.168.0.108:2375 #change the IP address to your remote server's IP address. Don't change anything else.
      - DOCKER_HOST_3_NAME=ClosedCloud #this is the identifier that will show in the dashboard.
      - DOCKER_HOST_3_PUBLIC_HOSTNAME=192.168.0.108 #this is the IP address of the server (no http/https).

      # Keep adding more hosts as necessary. Be sure to increase the number of the Docker Host.

    ports:
      - "3420:8000" #change 3420 as necessary for your setup. do not change 8000.
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock #this is how we connect to the docker-socket.
    restart: unless-stopped #this is fine.
```

Dockge

Docker Compose Management Tool

Synology

Installs in Synology NAS

Synology

NUT in Synology

This guide goes trough how to connect and enable NUT Server for UPS with Synology. Assuming that there is connected ups system to synology NAS

Synology Setup

1. Login to Synology DSM
2. Go to **Control Panel**
3. Under Hardware & Power go to UPS
    1. **Enable UPS Support**
    2. Check **Until Low Battery-** as time can vary
    3. **Enable network UPS server**
    4. Under Premitted DiskStation Devices enter IP address of devices that need access to NUT Server

Home Assistant Setup

1. Go to Home Assistant
2. Go to Settings
3. Click on Devices & Services
4. Click **Add Integration**
5. Search for NUT Network UPS Tools
    1. Add IP of Synology NAS
    2. leave rest blank

Komodo

Komodo

Install Komodo Periphery Agents

This document goes over setting up agents for connecting to Core Komodo. When making initial docker compose in the .env file we will define the password that will go into this docker-compose.yml file under line 18.

Komodo Documentation

1. SSH to server you want to establish connection with
2. In the desired folder path paste following docker-compose.yml

```
####################################
# □□KOMODO COMPOSE - PERIPHERY □□#
####################################

## This compose file will deploy:
##   1. Komodo Periphery

services:
  periphery:
    image: ghcr.io/moghtech/komodo-periphery:${COMPOSE_KOMODO_IMAGE_TAG:-latest}
    labels:
      komodo.skip: # Prevent Komodo from stopping with StopAllContainers
    restart: unless-stopped
    ## https://komo.do/docs/connect-servers#configuration
    environment:
      PERIPHERY_ROOT_DIRECTORY: ${PERIPHERY_ROOT_DIRECTORY:-/etc/komodo}
      ## Pass the same passkey as used by the Komodo Core connecting to this Periphery agent.
      PERIPHERY_PASSKEYS: XXXXXXXXXXXX
      ## Make server run over https
      PERIPHERY_SSL_ENABLED: true
      ## Specify whether to disable the terminals feature
      ## and disallow remote shell access (inside the Periphery container).
      PERIPHERY_DISABLE_TERMINALS: false
      ## If the disk size is overreporting, can use one of these to
      ## whitelist / blacklist the disks to filter them, whichever is easier.
      ## Accepts comma separated list of paths.
```

```
      ## Usually whitelisting just /etc/hostname gives correct size for single root disk.
      PERIPHERY_INCLUDE_DISK_MOUNTS: /etc/hostname
      # PERIPHERY_EXCLUDE_DISK_MOUNTS: /snap,/etc/repos
    volumes:
      ## Mount external docker socket
      - /var/run/docker.sock:/var/run/docker.sock
      ## Allow Periphery to see processes outside of container
      - /proc:/proc
      ## Specify the Periphery agent root directory.
      ## Must be the same inside and outside the container,
      ## or docker will get confused. See https://github.com/moghtech/komodo/discussions/180.
      ## Default: /etc/komodo.
      - ${PERIPHERY_ROOT_DIRECTORY:-/etc/komodo}:${PERIPHERY_ROOT_DIRECTORY:-/etc/komodo}
    ## If periphery is being run remote from the core server, ports need to be exposed
    ports:
     - 8120:8120
    ## If you want to use a custom periphery config file, use command to pass it to periphery.
    # command: periphery --config-path ${PERIPHERY_ROOT_DIRECTORY:-
/etc/komodo}/periphery.config.toml
```

3. Start the docker compose

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

Proxmox

Proxmox Installation guides

Proxmox

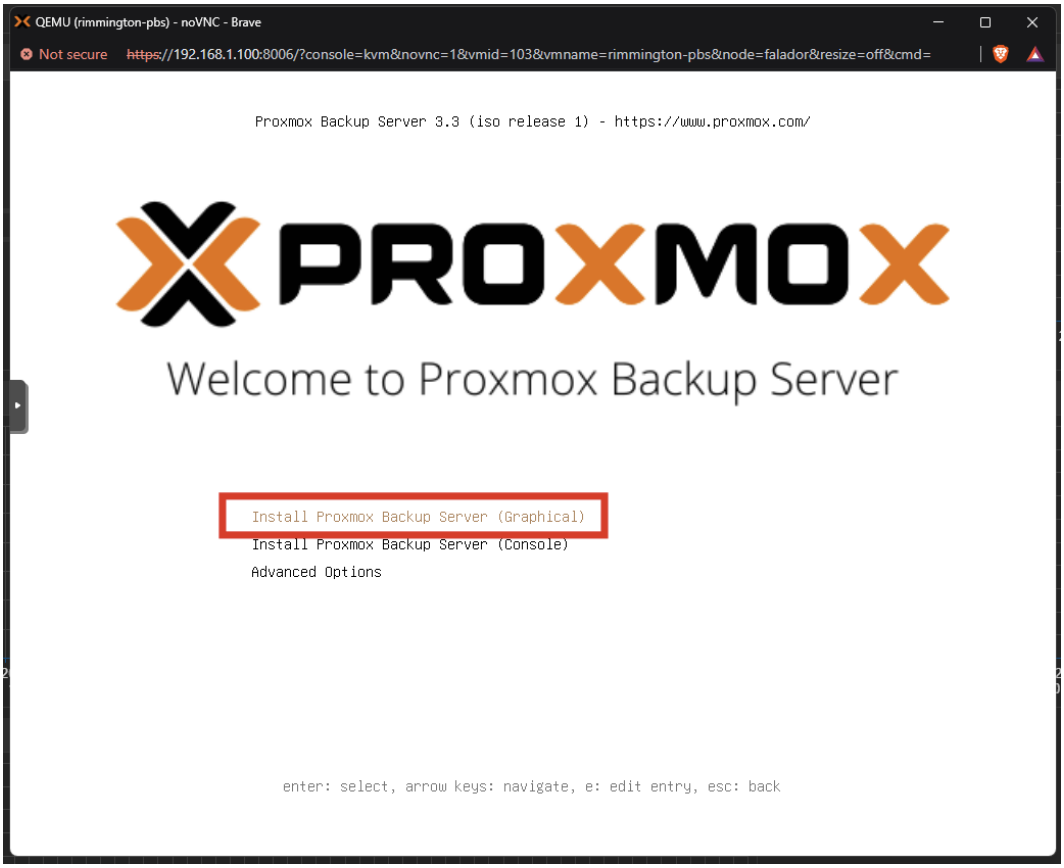
Installing Proxmox Backup Server

This article goes over steps for installing Proxmox Backup Server. There is multiple options on where to run PBS instance, however for best performance running it on seperate bare metal machine would be the best option. PBS can run in virtual machine if needed, just like any other OS with ISO image. If PBS runs in vm do not select that vm when scheduling backup job with PBS.

Prerequisite is machine ready and Proxmox Backup Server ISO flashed to flash drive

Installing PBS

1. Boot the machine and select flash drive with ISO image
2. Install PBS with a graphical user interface



3. **Accept** EULA
4. Select Target Disk to where OS will live (Hopefully separate from main storage if you have couple)
5. Fill in your country, timezone and keyboard layout

6. Choose a unique password and fill in your email address for your **root** access (you can disable root later)
7. Fill in your network details
    1. ID: A name for the connection, e.g., pbs-backup
    2. **Managment** Interface: unless you want/have a different interface, leave this as default
    3. **Hostname:** "NameOfYourPBS":local
    4. **IP Address:** Choose a static IP address
    5. **Gateway:** Fill in the gateway
    6. **DNS server:** Fill in a DNS server
8. Finish the installation by clicking continue
9. Access the PBS Dashboard by going to its [IP Address] and pot 8007 eg. https://192.168.1.101:8007

Gotify

Gotify

Installing Gotify with iGotify for iOS

Gotify is a self-hosted notification server that lets you send messages to devices and apps via a simple API. It’s great for server alerts, home automation, and custom scripts because you control delivery and history.

iGotify is the bridge that makes Gotify work with iOS push notifications. It listens to Gotify, translates messages, and forwards them to the SecNtfy app on your iPhone so you get real push alerts (iOS can’t poll in the background, so the bridge is required).

Once app is installed Set gotify and igotify domain in Nginx Reverse Proxy and Pangolin for domain with ssl login

Install Gotify

1. ssh to folder where gotify app will live or use Komodo and create Stack
2. Add docker-compose.yaml
    1. Add services: Gotify and iGotify, with ports exposed (e.g., Gotify 3030:80, iGotify 3031:8080)
    2. Persist data: Use a volume for Gotify (data:/app/data or ./gotify_data:/app/data)
    3. GOTIFY_DEFAULTUSER_PASS for the default admin
3. Start the stack
4. Check URLs: Gotify at http://<server-ip>:3030, iGotify at http://<server-ip>:3031/Version
5. Login: Open Gotify, sign in as admin

Create Tokens for iOS app

1. Login to Gotify in browser and got to Clients
2. Create a Gotify client token by clicking New client (e.g., “igotify”).
3. Copy token
4. Add to iGotify env: GOTIFY_CLIENT_TOKENS: "cXXXXXXXX" inside yml file
    1. Adjust the URL to the domain url of Gotify

iGotify iOS App Setup

1. Install iGotify app
2. Enable local instance: In the app’s settings
3. Connect to iGotify: Use http://<server-ip>:3031
4. Get igotify app token: In the app, Settings → Development → copy the notification token (NTFY-DEVICE-XXXXXX).
5. Add that token to the yml file
6. Update compose
7. Once everything is up and running, go to igotify app into settings
8. Select Instance and click Edit
9. Change the http://<server-ip>:3031 to https and point it to domain

Adding multiple devices Multiple tokens

Add each device’s SecNtfy token separated by semicolons.

Example:

```
Enviroment:
  SECNTFY_TOKENS: "NTFY-DEVICE-AAA;NTFY-DEVICE-BBB"
  GOTIFY_CLIENT_TOKENS: "cXXXX1;cXXXX2" (if using multiple Gotify clients)
  GOTIFY_URLS: "https://gotify1;https://gotify2" (if using multiple Gotify servers)
```

Gotify Compose File

```
services:
  gotify:
    container_name: gotify
    hostname: gotify
    image: gotify/server
    restart: unless-stopped
    security_opt:
      - no-new-privileges:true
    networks:
      - net
    ports:
      - "3030:80"
    volumes:
      - data:/app/data
    environment:
      GOTIFY_DEFAULTUSER_PASS:  'admin'   # Change me!!!!!

  igotify:
    container_name: igotify
    hostname: igotify
    image: ghcr.io/androidseb25/igotify-notification-assist:latest
    restart: unless-stopped
    security_opt:
      - no-new-privileges:true
    pull_policy: always
    healthcheck:
      test: [ "CMD", "curl", "-f", "http://localhost:8080/Version" ]
      interval: "3s"
      timeout: "3s"
      retries: 5
    networks:
      - net
    ports:
      - "3031:8080"
```

```
    volumes:
      - api-data:/app/data
    environment:                # option environment see above note
      GOTIFY_URLS:          'https://gotify.cyberpaw.org'
      GOTIFY_CLIENT_TOKENS: '' #create on the gotify browser client
      SECNTFY_TOKENS:       '' #after initial login get it from settings

networks:
  net:

volumes:
  data:
  api-data:
```

UxPlay

UxPlay

Installing UxPlay on Linux

This guide explains how to:

- **Install UxPlay** on Fedora
- **Create a launch script**
- **Create a .desktop launcher** so UxPlay appears in the **Apps** menu
- **Run UxPlay in the background** without freezing the launcher

Install UxPlay and create Launch Script

1. Open Terminal and run

```
sudo dnf install uxplay
```

2. Create a folder for custom scripts. Fedora doesn't always include a ~/bin folder by default, so create it:

```
mkdir -p ~/bin
```

3. Create UxPlay script

```
nano ~/bin/start-uxplay.sh
```

In nano add

```
#!/bin/bash
nohup uxplay >/dev/null 2>&1 &
```

4. Save and Exit
5. Make it executable

```
chmod +x ~/bin/start-uxplay.sh
```

Create desktop app launcher

1. Create Directory

```
mkdir -p ~/.local/share/applications
```

2. Create a launcher

```
nano ~/.local/share/applications/uxplay.desktop
```

In nano add

```
[Desktop Entry]
Type=Application
Name=UxPlay
Comment=Start the UxPlay AirPlay receiver
Exec=/home/YOURUSERNAME/bin/start-uxplay.sh
Icon=video-display
Terminal=false
Categories=AudioVideo;
```

3. Save and Exit
4. Make launcher executable

```
chmod +x ~/.local/share/applications/uxplay.desktop
```

5. Refresh application database

```
update-desktop-database ~/.local/share/applications/
```

To Stop UxPlay simply run command in terminal

```
pkill uxplay
```

Run UxPlay as a systemd service

This method gives you:

- **systemctl start uxplay**
- **systemctl stop uxplay**
- **systemctl restart uxplay**
- Optional **auto-start on boot**
- Clean background operation with no terminal needed

Create a systemd service file

1. Open Service file

```
sudo nano /etc/systemd/system/uxplay.service
```

In Nano Paste following command editing yoursuername

```
[Unit]
Description=UxPlay AirPlay Receiver
After=network.target

[Service]
Type=simple
User=YOURUSERNAME
ExecStart=/usr/bin/uxplay
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

2. Reload systemd

```
sudo systemctl daemon-reload
```

Optional: Enable auto start on boot

```
sudo systemctl enable uxplay
```

Systemd Commands for UxPlay

- Start

```
sudo systemctl start uxplay
```

- Stop

sudo systemctl stop uxplay

```
sudo systemctl stop uxplay
```

MeshCentral

MeshCentral Installation on Ubuntu Server

These Instructions follow the steps of starting personal MeshCentral remote support server. Prerequisites require you to have set up ubuntu server, and have DNS record set up to point to this ip:443. It's good idea to use cloudflare tunnel or other type of tunneling service so you don't need to do port forwarding.

Installing Steps for MeshCentral

1. Update Server

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

2. Install Node.js

```
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo bash -
sudo apt install -y nodejs
```

3. Create MesCentral Directory

```
sudo mkdir /opt/meshcentral
cd /opt/meshcentral
```

4. Install MeshCentral

```
sudo npm install meshcentral
```

5. Create Dedicated User

```
sudo useradd -r -d /opt/meshcentral -s /sbin/nologin meshcentral
sudo chown -R meshcentral:meshcentral /opt/meshcentral
```

6. Grant Port 443 Access

```
sudo setcap 'cap_net_bind_service=+ep' $(which node)
```

7. Start MeshCentral

```
sudo -u meshcentral node node_modules/meshcentral
```

8. Configure for Port 443

```
sudo nano /opt/meshcentral/meshcentral-data/config.json
```

Replace in json file:

```
{
  "settings": {
    "cert": "remote.cyberclaw.org",
    "port": 443,
    "redirPort": 80,
    "wanonly": true
  },
  "domains": {
    "": {
      "title": "My MeshCentral",
      "newAccounts": true
    }
  }
}
```

9. Create Systemd Service

```
sudo nano /etc/systemd/system/meshcentral.service
```

Add:

```
[Unit]
Description=MeshCentral Server
After=network.target

[Service]
Type=simple
User=meshcentral
WorkingDirectory=/opt/meshcentral
```

```
ExecStart=/usr/bin/node node_modules/meshcentral
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

10. Enable and Start Service

```
sudo systemctl daemon-reload
sudo systemctl enable meshcentral.service --now
sudo systemctl status meshcentral.service
```

Login via domain set up for meshcentral. Go to create account. First Time account creation will be admin account.

Docker

Installing Docker

Steps for Installing Docker

1. Add the Docker Repository

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
```

2. Update Apt Repository

```
sudo apt update
```

3. Install Docker

```
sudo apt install docker-ce docker-ce-cli containerd.io
```

4. Verify Docker is running

```
sudo docker --version
sudo systemctl status docker
```

Dockhand

Dockhand Hawser Docker Compose Agent

To install Docker Compose Agent for Dockhand

```
version: '3.8'

services:
  hawser:
    image: ghcr.io/finsys/hawser:latest
    container_name: hawser
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - hawser_stacks:/data/stacks
    ports:
      - "2376:2376"
    environment:
      - TOKEN=your-secret-token
    restart: unless-stopped

volumes:
  hawser_stacks:
```

Connecting Server Directory to Synology NAS (NFS Setup)

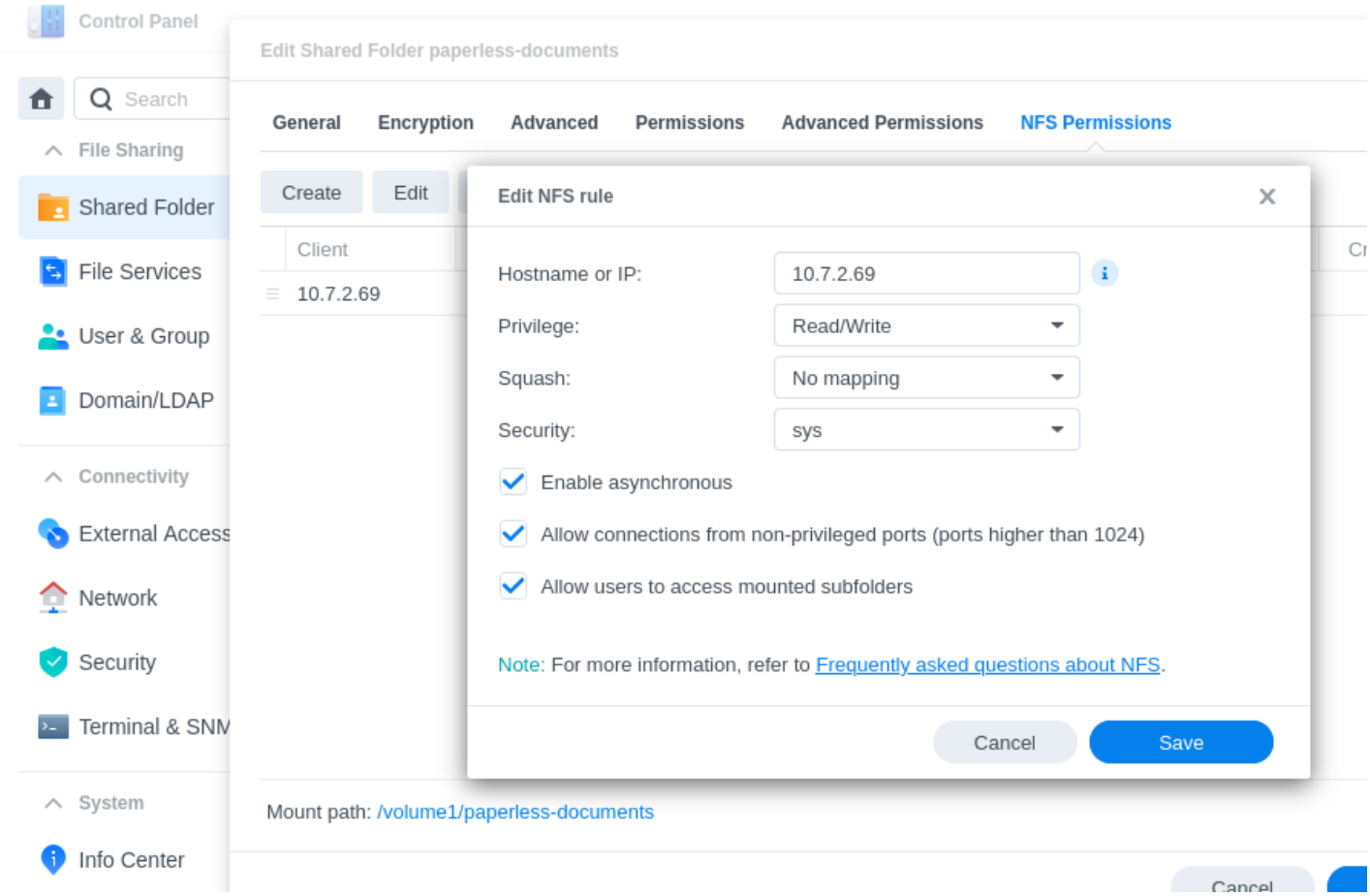
This is a step-by-step document on how to connect a server directory to a Synology NAS using NFS. This guide assumes you have access to both the server and the Synology NAS.

Prerequisites

Access to a server (Linux-based).
Access to a Synology NAS.
NFS service enabled on the Synology NAS.
Sufficient permissions to execute commands on both devices.

Configure NFS on Synology NAS

1. Access Synology NAS
2. Enable NFS Service by Going to **Control Panel > File Services**
3. Under the NFS tab, enable the NFS service
4. Configure NFS Permissions:
    1. Navigate to Shared Folder in Control Panel
    2. Select the folder you want to share (e.g., /volume1/paperless-documents)
    3. Click on Edit > NFS Permissions
    4. Click Create and set the following:
        1. Hostname or IP: Enter the IP address of your server
        2. Privilege: Set to Read/Write
        3. Squash: Select No mapping to allow direct access
        4. Asynchronous: Optional, you can enable this for better performance
        5. Cross-Mount: Enable if you intend to mount cross-shared folders
        6. Check Allow users to access mounted subfolders
5. Click **OK** to save the settings



Prepare Your Server

1. SSH into your server
2. Install NFS Client

```
sudo apt-get update
sudo apt-get install nfs-common
```

3. Create mount point- a directory where the NFS share will be mounted

```
sudo mkdir -p /mnt/nas/Import
```