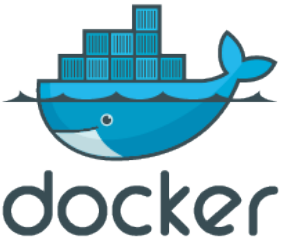
# Quick Paperless Stack Setup Guide


–





Paperless NGX is an open-source document management solution that allows you to digitize and efficiently manage your paperwork. In this guide, we will deploy Paperless NGX on a Docker Swarm cluster using a shared storage volume provided by GlusterFS (or a similar NAS-mounted setup) to ensure all nodes share the same data. If you intend to expose Paperless NGX to the internet, you can use Traefik as a reverse proxy for SSL termination.

## Prerequisites

- Docker Swarm
- GlusterFS (or a similar NAS mount) so that **all nodes** have access to the same directories
- (Optional) Traefik, if you plan to make Paperless NGX accessible externally (recommended for TLS/SSL)

## Step 1: Set Up Directory Structure

Create the directories for Paperless NGX data, ensuring they reside on your GlusterFS (or equivalent) mount so that data is shared among all Swarm nodes. For example:

```
mkdir -p /mnt/glustermount/data/paperless/
mkdir -p /mnt/glustermount/data/paperless/redisdata
mkdir -p /mnt/glustermount/data/paperless/data
mkdir -p /mnt/glustermount/data/paperless/media
mkdir -p /mnt/glustermount/data/paperless/export
mkdir -p /mnt/glustermount/data/paperless/consume
mkdir -p /mnt/glustermount/data/paperless/postgresqldata
```

# Step 2: Create Your Docker Compose File

> **Important:** In all configurations and code snippets below, replace `YOUR-DOMAIN.com` with your actual domain wherever applicable.

Below is an example `docker-compose.yml` that sets up Paperless NGX alongside Redis, PostgreSQL, Gotenberg, and Apache Tika. This file is intended for Docker Swarm with a GlusterFS-backed volume. You can adapt paths and replicas to your needs.

```
version: "3.7"

services:
  broker:
    image: docker.io/library/redis:7
    restart: unless-stopped
    volumes:
      - /mnt/glustermount/data/paperless/redisdata:/data
    deploy:
      mode: replicated
      replicas: 1
    networks:
      - internal

  webserver:
    image: ghcr.io/paperless-ngx/paperless-ngx:latest
    restart: unless-stopped
    depends_on:
      - broker
```

```
    - gotenberg
    - tika
  ports:
    - "8000:8000"
  volumes:
    - /mnt/glustermount/data/paperless/data:/usr/src/paperless/data
    - /mnt/glustermount/data/paperless/media:/usr/src/paperless/media
    - /mnt/glustermount/data/paperless/export:/usr/src/paperless/export
    - /mnt/glustermount/data/paperless/consume:/usr/src/paperless/consume
  environment:
    PAPERLESS_REDIS: "redis://broker:6379"
    PAPERLESS_TIKA_ENABLED: 1
    PAPERLESS_TIKA_GOTENBERG_ENDPOINT: "http://gotenberg:3000"
    PAPERLESS_TIKA_ENDPOINT: "http://tika:9998"
    PAPERLESS_URL: "https://paperless.YOUR-DOMAIN.com"
    PAPERLESS_OCR_LANGUAGE: "eng"
    PAPERLESS_TIME_ZONE: "Europe/Zurich"
    PAPERLESS_ADMIN_USER: "${PAPERLESS_ADMIN_USER}"
    PAPERLESS_ADMIN_PASSWORD: "${PAPERLESS_ADMIN_PW}"
    PAPERLESS_ADMIN_MAIL: "${PAPERLESS_ADMIN_EMAIL}"
    PAPERLESS_SECRET_KEY: "${PAPERLESS_SECRET_KEY}"
    PAPERLESS_DBHOST: "db"
    PAPERLESS_DBNAME: "${PAPERLESS_POSTGRES_DB}"
    PAPERLESS_DBUSER: "${PAPERLESS_POSTGRES_USER}"
    PAPERLESS_DBPASS: "${PAPERLESS_POSTGRES_PASSWORD}"
  deploy:
    mode: replicated
    replicas: 1
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.webserver.rule=Host(`paperless.YOUR-DOMAIN.com`)"
      - "traefik.http.routers.webserver.entrypoints=websecure"
      - "traefik.http.services.webserver.loadbalancer.server.port=8000"
      - "traefik.docker.network=management_net"
  networks:
    - management_net
    - internal

db:
  image: docker.io/library/postgres:16
```

```
    restart: unless-stopped
    volumes:
      - /mnt/glustermount/data/paperless/postgresqldata:/var/lib/postgresql/data
    environment:
      POSTGRES_DB: "${PAPERLESS_POSTGRES_DB}"
      POSTGRES_USER: "${PAPERLESS_POSTGRES_USER}"
      POSTGRES_PASSWORD: "${PAPERLESS_POSTGRES_PASSWORD}"
    deploy:
      mode: replicated
      replicas: 1
    networks:
      - internal

  gotenberg:
    image: docker.io/gotenberg/gotenberg:8.7
    restart: unless-stopped
    command:
      - "gotenberg"
      - "--chromium-disable-javascript=true"
      - "--chromium-allow-list=file:///tmp/.*"
    deploy:
      mode: replicated
      replicas: 1
    networks:
      - internal

  tika:
    image: docker.io/apache/tika:latest
    restart: unless-stopped
    deploy:
      mode: replicated
      replicas: 1
    networks:
      - internal

networks:
  management_net:
    external: true

  internal:
```

```
  driver: overlay
  ipam:
    config:
      - subnet: 172.16.58.0/24
```

## Why We Define a Custom Subnet for the `internal` Network

The `internal` network is an overlay network dedicated to internal communication between Paperless NGX services (like Redis, Gotenberg, Tika, and PostgreSQL). By assigning a specific subnet ( `172.16.58.0/24`), you ensure:

- **Isolation:** Only these containers communicate on this internal overlay, reducing exposure to the outside world.
- **Predictability:** Having a known subnet range helps avoid IP conflicts with other networks.

## Defining Environment Variables

If you are using Portainer, you can define environment variables such as `stack.env` directly in the Portainer Web-GUI when deploying the stack.

If you are **not** using Portainer, create a `.env` file in the same directory as your `docker-compose.yml` and specify:

```
services:
  webserver:
    ...
    env_file:
      - .env

  db:
    ...
    env_file:
      - .env
```

Then add the following variables in your `.env` file:

```
PAPERLESS_POSTGRES_DB=
PAPERLESS_POSTGRES_USER=
PAPERLESS_POSTGRES_PASSWORD=
PAPERLESS_ADMIN_USER=
PAPERLESS_ADMIN_PW=
PAPERLESS_ADMIN_EMAIL=
PAPERLESS_SECRET_KEY=
```

You can also check out the official sample environment file for Paperless NGX to see additional variables you may configure.

## Environment Variables Used in the Compose File

Below is a brief explanation of some key environment variables in the `docker-compose.yml` file. For a full list of available variables and their usage, refer to the Paperless NGX Configuration Documentation.

- **PAPERLESS_URL:** The base URL where Paperless NGX is accessible (e.g., `https://paperless.your-domain.com`).
- **PAPERLESS_REDIS:** The Redis connection string (host:port) for caching and background tasks.
- **PAPERLESS_TIKA_ENABLED:** Enables Apache Tika integration for advanced document parsing.
- **PAPERLESS_TIKA_GOTENBERG_ENDPOINT / PAPERLESS_TIKA_ENDPOINT:** Endpoints for Gotenberg and Tika services, respectively, used to convert and parse documents.
- **PAPERLESS_TIME_ZONE:** Sets the timezone inside the container (e.g., `Europe/Zurich`).
- **PAPERLESS_ADMIN_USER / PAPERLESS_ADMIN_PASSWORD / PAPERLESS_ADMIN_MAIL:** Credentials for the default Paperless NGX admin account.
- **PAPERLESS_SECRET_KEY:** A secret key used by the Django framework within Paperless NGX for cryptographic functions.
- **PAPERLESS_DBHOST / PAPERLESS_DBNAME / PAPERLESS_DBUSER / PAPERLESS_DBPASS:** Connection details for PostgreSQL. These point to the `db` service and use the credentials defined in the environment variables.

# Step 3: Deploy the Stack

Navigate to the directory containing your `docker-compose.yml` file and deploy the stack with Docker Swarm:

```
docker stack deploy -c docker-compose.yml paperless
```

Alternatively, you can deploy the stack via Portainer or any other Docker Swarm management tool.

# Step 4: Accessing Paperless NGX

- Once deployed, Paperless NGX will be available at the port you specified (`8000` by default) on the Swarm node.
- If you configured Traefik and DNS correctly, you should be able to access your Paperless NGX instance at `https://paperless.your-domain.com`.
- Log in with the admin credentials you set in the environment variables.

# Additional Notes

- **Multi-Node Persistence:** Because GlusterFS (or an equivalent) is used, your Paperless NGX data and database files are stored on shared volumes accessible by all nodes in the Swarm.
- **Security & SSL:** If you're exposing Paperless NGX externally, ensure you have valid SSL certificates set up (e.g., via Traefik with Let's Encrypt).
- **Scaling Services:** You can increase the `replicas` value for different services if you want to distribute the load across multiple nodes.

# Conclusion

By deploying Paperless NGX on a Docker Swarm, you gain the benefits of high availability and scalability, especially when backed by a distributed storage solution like GlusterFS. Whether you use Portainer for an easier management interface or rely on `.env` files for more traditional Docker workflows, the key is consistent environment configuration and ensuring all nodes share the necessary data volumes. With this setup, your document management solution is primed for production use—secure, resilient, and easy to extend.

---

Revision #7
Created 4 April 2025 11:46:08 by aeoneros

Updated 8 April 2025 05:51:33 by aeoneros