Paperless-NGX

Here we go. The granddaddy of containers. Let's kick this pig!

Here's a docker-compose.yml:

```
services:
  broker:
    image: docker.io/library/redis:7
    restart: unless-stopped
    volumes:
      - redisdata:/data
    # ADDED: Connect broker to the traefik_proxy network so it can be discovered
    networks:
      - default

  webserver:
    image: ghcr.io/paperless-ngx/paperless-ngx:latest # It's good practice to use 'latest' or
a specific version
    container_name: paperless
    hostname: paperless
    restart: unless-stopped
    depends_on:
      - broker
      - gotenberg
      - tika
    # The ports section is not needed when using Traefik, as it handles external access.
    # ports:
    #   - 8000:8000
    healthcheck:
      test: ["CMD", "curl", "-fs", "-S", "--max-time", "2", "http://localhost:8000"]
      interval: 30s
      timeout: 10s
      retries: 5
    volumes:
      - data:/usr/src/paperless/data
```

```
    - ./consume/media:/usr/src/paperless/media # Assuming this is your media directory
    - ./consume:/usr/src/paperless/consume # Assuming this is your consume directory
    - paperless:/usr/src/paperless/export # Using your defined NFS volume for export
  environment:
    PAPERLESS_URL: https://<yourdomain>
    PAPERLESS_CSRF_TRUSTED_ORIGINS: https://<yourdomain>
    PAPERLESS_REDIS: redis://broker:6379
    PAPERLESS_DBPASS: paperless
    PAPERLESS_DBPORT: 3306
    # ADDED: Tika and Gotenberg Configuration
    PAPERLESS_TIKA_ENABLED: 1
    PAPERLESS_TIKA_ENDPOINT: http://tika:9998
    PAPERLESS_TIKA_GOTENBERG_ENDPOINT: http://gotenberg:3000
    PAPERLESS_TIKA_EML_BODY_CONTENT_TYPES: text/html,text/plain
    PAPERLESS_FILENAME_FORMAT: '{{added_year}}/{{correspondent}}/{{title}}'

  labels:
    - "traefik.enable=true"
    - "traefik.http.routers.paperless-https.rule=Host(`<yourdomain>`)"
    - "traefik.http.routers.paperless-https.entrypoints=websecure"
    - "traefik.http.routers.paperless-https.tls.certresolver=letsencrypt"
    - "traefik.http.services.paperless-svc.loadbalancer.server.port=8000"
  networks:
    - default
    - traefik_proxy

# ADDED: Gotenberg Service
gotenberg:
  image: docker.io/gotenberg/gotenberg:latest
  restart: unless-stopped
  command:
    - "gotenberg"
    - "--chromium-disable-javascript=true"
    - "--chromium-allow-list=file:///tmp/.*"
  networks:
    - default

# ADDED: Tika Service
tika:
  image: ghcr.io/paperless-ngx/tika:latest
```

```
    restart: unless-stopped
    networks:
      - default

volumes:
  data:
  dbdata:
  redisdata:
  paperless:
    driver: local
    driver_opts:
      type: nfs
      o: "addr=192.168.0.3,rw"
      device: ":/mnt/NAS-7200/NFS/Paperless/processed"

networks:
  default:
  traefik_proxy:
    external: true
```

There's a metric fuckton going on here. Let me break it down.

There are 4 containers in this compose:

broker

webserver

gotenberg

tika

We only need to expose access to the webserver container. Everything else runs internally inside docker.

There's a labels section inside the stanza for the webserver, with the config for Traefik. The Traefik daemon will read it and configure itself accordingly.

> update this compose file to change every instance of <yourdomain> to the domain you're actually using.

Lastly, there's a docker volume definition for each container. The most consequential one is the 'paperless' volume. In my case, I am mapping the volume to an NFS share on my NAS (line 78-83). This is where paperless will store all it's documents.

If you don't specify the config of a volume, docker does the work for you and links the volume definition to directory on the localhost that Docker uses for volumes. But there are different ways of defining volumes to use network shares etc. It's worth reading the documentation about it.

Alternatively, if there is space on the localhost, you can decide not to use a volume and go with what's called a bind mount. You can just tell docker to use a directory inside the folder that has the docker-compose file and store the data there. That folder could be an NFS mount itself. It really depends on your use case. The syntax for a bind mount to replace the paperless volume would look like this:

```
volumes:
      - data:/usr/src/paperless/data
      - ./consume/media:/usr/src/paperless/media # Assuming this is your media directory
      - ./consume:/usr/src/paperless/consume # Assuming this is your consume directory
      - ./paperless:/usr/src/paperless/export # Using your defined NFS volume for export
```

Next up is the 'consume' folder. If you want to have paperless just eat whatever you copy there, then the 'consume' folder in the root directory of the app (/apps/paperless/consume/) would need to be mapped to a file share that's accessible from anywhere on the network.

In my case, my NAS has an NFS/CIFS file share. On the docker machine, I map a folder on that file share to /apps/paperless/consume on the docker host.

> The volumes part of the compose, mounts the folder into the container at the given path so /apps/paperless/consume will map to /usr/src/paperless/consume inside the container, which is where the paperless application is scanning for documents.

If any or all of that made sense, then you can do a

```
dc up -d
```

and see if it all works .

Created 2025-07-17 06:43:26 UTC by Alex
Updated 2025-07-17 07:02:27 UTC by Alex