Instalação e configurações N8N

Instalação e configurações N8N

- Instalação N8N docker (1)
- Instalação N8N docker (2)
- Instalação, configuração e atualização N8N Docker

Instalação N8N docker (1)

Link: https://docs.n8n.io/hosting/installation/server-setups/docker-compose/#5-create-docker-compose-file

If you have already installed Docker and Docker-Compose, then you can start with step 4.

Self-hosting knowledge prerequisites

Self-hosting n8n requires technical knowledge, including:

- Setting up and configuring servers and containers
- Managing application resources and scaling
- Securing servers and applications
- Configuring n8n

n8n recommends self-hosting for expert users. Mistakes can lead to data loss, security issues, and downtime. If you aren't experienced at managing servers, n8n recommends n8n Cloud.

Latest and Next versions

n8n releases a new minor version most weeks. The `latest` version is for production use. `next` is the most recent release. You should treat `next` as a beta: it may be unstable. To report issues, use the forum.

Current `latest`: 1.58.2
Current `next`: 1.59.0

1. Install Docker#

This can vary depending on the Linux distribution used. You can find detailed instructions in the Docker documentation. The following example is for Ubuntu:

```
sudo apt-get remove docker docker-engine docker.io containerd runc
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg lsb-release
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

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

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

2. Optional: Non-root user access#

Run when logged in as the user that should also be allowed to run docker:

```
sudo usermod -aG docker ${USER}
su - ${USER}
```

3. Install Docker-Compose#

This can vary depending on the Linux distribution used. You can find detailed instructions in the Docker documentation.

The example below is for Ubuntu:

```
sudo apt-get install docker-compose-plugin
```

4. DNS setup#

Add A record to route the subdomain accordingly:

```
Type: A
Name: n8n (or the desired subdomain)
IP address: <IP_OF_YOUR_SERVER>
```

5. Create Docker Compose file#

Create a `docker-compose.yml` file. Paste the following in the file:

```
version: "3.7"

services:
  traefik:
    image: "traefik"
    restart: always
    command:
      - "--api=true"
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--providers.docker.exposedbydefault=false"
      - "--entrypoints.web.address=:80"
      - "--entrypoints.web.http.redirections.entryPoint.to=websecure"
      - "--entrypoints.web.http.redirections.entrypoint.scheme=https"
      - "--entrypoints.websecure.address=:443"
      - "--certificatesresolvers.mytlschallenge.acme.tlschallenge=true"
      - "--certificatesresolvers.mytlschallenge.acme.email=${SSL_EMAIL}"
      - "--certificatesresolvers.mytlschallenge.acme.storage=/letsencrypt/acme.json"
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - traefik_data:/letsencrypt
      - /var/run/docker.sock:/var/run/docker.sock:ro

  n8n:
    image: docker.n8n.io/n8nio/n8n
    restart: always
    ports:
      - "127.0.0.1:5678:5678"
    labels:
      - traefik.enable=true
      - traefik.http.routers.n8n.rule=Host(`${SUBDOMAIN}.${DOMAIN_NAME}`)
      - traefik.http.routers.n8n.tls=true
      - traefik.http.routers.n8n.entrypoints=web,websecure
      - traefik.http.routers.n8n.tls.certresolver=mytlschallenge
      - traefik.http.middlewares.n8n.headers.SSLRedirect=true
      - traefik.http.middlewares.n8n.headers.STSSeconds=315360000
```

```
      - traefik.http.middlewares.n8n.headers.browserXSSFilter=true
      - traefik.http.middlewares.n8n.headers.contentTypeNosniff=true
      - traefik.http.middlewares.n8n.headers.forceSTSHeader=true
      - traefik.http.middlewares.n8n.headers.SSLHost=${DOMAIN_NAME}
      - traefik.http.middlewares.n8n.headers.STSIncludeSubdomains=true
      - traefik.http.middlewares.n8n.headers.STSPreload=true
      - traefik.http.routers.n8n.middlewares=n8n@docker
    environment:
      - N8N_HOST=${SUBDOMAIN}.${DOMAIN_NAME}
      - N8N_PORT=5678
      - N8N_PROTOCOL=https
      - NODE_ENV=production
      - WEBHOOK_URL=https://${SUBDOMAIN}.${DOMAIN_NAME}/
      - GENERIC_TIMEZONE=${GENERIC_TIMEZONE}
    volumes:
      - n8n_data:/home/node/.n8n

volumes:
  traefik_data:
    external: true
  n8n_data:
    external: true
```

If you are planning on reading/writing local files with n8n (for example, by using the Read/Write Files from Disk node, you will need to configure a data directory for those files here. If you are running n8n as a root user, add this under `volumes` for the n8n service:

```
- /local-files:/files
```

If you are running n8n as a non-root user, add this under `volumes` for the n8n service:

```
- /home/<YOUR USERNAME>/n8n-local-files:/files
```

You will now be able to write files to the `/files` directory in n8n and they will appear on your server in either `/local-files` or `/home/<YOUR USERNAME>/n8n-local-files`, respectively.

6. Create `.env` file#

Create an `.env` file and change it accordingly.

```
# The top level domain to serve from
DOMAIN_NAME=example.com

# The subdomain to serve from
SUBDOMAIN=n8n

# DOMAIN_NAME and SUBDOMAIN combined decide where n8n will be reachable from
# above example would result in: https://n8n.example.com

# Optional timezone to set which gets used by Cron-Node by default
# If not set New York time will be used
GENERIC_TIMEZONE=Europe/Berlin

# The email address to use for the SSL certificate creation
SSL_EMAIL=user@example.com
```

7. Create data folder#

Create the Docker volume that's defined as `n8n_data`. n8n will save the database file from SQLite and the encryption key in this volume.

```
sudo docker volume create n8n_data
```

Create a volume for the Traefik data, This is defined as `traefik_data`.

```
sudo docker volume create traefik_data
```

8. Start Docker Compose#

n8n can now be started via:

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

To stop the container:

```
sudo docker compose stop
```

9. Done#

n8n will now be reachable using the above defined subdomain + domain combination. The above example would result in: `https://n8n.example.com`

n8n will only be reachable using `https` and not using `http`.

Secure your n8n instance

Make sure that you set up authentication for your n8n instance.

Next steps#

- Learn more about configuring and scaling n8n.
- Or explore using n8n: try the Quickstarts.

Instalação N8N docker (2)

Link: https://github.com/n8n-io/n8n-hosting/tree/main/docker-compose

git clone https://github.com/n8n-io/n8n-hosting.git

Instalação, configuração e atualização N8N Docker

Link: https://docs.n8n.io/hosting/installation/docker/

Docker Installation#

Docker offers the following advantages:

- Install n8n in a clean environment.
- Easier setup for your preferred database.
- Can avoid issues due to different operating systems, as Docker provides a consistent system.

You can also use n8n in Docker with Docker Compose. You can find Docker Compose configurations for various architectures in the n8n-hosting repository.

Prerequisites#

Before proceeding, install Docker Desktop.

Linux Users

Docker Desktop is available for Mac and Windows. Linux users must install Docker Engine and Docker Compose individually for your distribution.

Self-hosting knowledge prerequisites

Self-hosting n8n requires technical knowledge, including:

- Setting up and configuring servers and containers

- Managing application resources and scaling
- Securing servers and applications
- Configuring n8n

n8n recommends self-hosting for expert users. Mistakes can lead to data loss, security issues, and downtime. If you aren't experienced at managing servers, n8n recommends n8n Cloud.

Latest and Next versions

n8n releases a new minor version most weeks. The `latest` version is for production use. `next` is the most recent release. You should treat `next` as a beta: it may be unstable. To report issues, use the forum.

Current `latest`: 1.72.1
Current `next`: 1.73.1

Starting n8n#

From your terminal, run:

```
1
2
3
```

```
docker volume create n8n_data

docker run -it --rm --name n8n -p 5678:5678 -v
n8n_data:/home/node/.n8n docker.n8n.io/n8nio/n8n
```

This command will download all required n8n images and start your container, exposed on port `5678`. To save your work between container restarts, it also mounts a docker volume, `n8n_data`, to persist your data locally.

You can then access n8n by opening: http://localhost:5678

Using alternate databases#

By default n8n uses SQLite to save credentials, past executions and workflows. n8n also supports PostgresDB configurable using environment variables as detailed below.

It's important to still persist data in the `/home/node/.n8n` folder as it contains n8n user data and even more importantly the encryption key for credentials. It's also the name of the webhook when the n8n tunnel is used.

If no directory is found, n8n creates automatically one on startup. In this case, existing credentials saved with a different encryption key can not be used anymore.

Keep in mind

Persisting the `/home/node/.n8n` directory even when using alternate databases is the recommended best practice, but not explicitly required. The encryption key can be provided using the `N8N_ENCRYPTION_KEY` environment variable.

PostgresDB#

To use n8n with Postgres, provide the corresponding:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

```
docker volume create n8n_data


docker run -it --rm \
 --name n8n \
 -p 5678:5678 \
 -e DB_TYPE=postgresdb \
 -e
DB_POSTGRESDB_DATABASE=<POSTGRES_DATAB
ASE> \
 -e DB_POSTGRESDB_HOST=<POSTGRES_HOST> \
 -e DB_POSTGRESDB_PORT=<POSTGRES_PORT> \
 -e DB_POSTGRESDB_USER=<POSTGRES_USER> \
 -e
DB_POSTGRESDB_SCHEMA=<POSTGRES_SCHEMA
> \
 -e
DB_POSTGRESDB_PASSWORD=<POSTGRES_PASS
WORD> \
 -v n8n_data:/home/node/.n8n \
 docker.n8n.io/n8nio/n8n
```

A complete `docker-compose` file for Postgres can be found here.

Setting timezone#

To define the timezone n8n should use, the environment variable `GENERIC_TIMEZONE` can be set. This gets used by schedule based nodes such as the Cron node.

The timezone of the system can also be set separately. This controls what some scripts and commands return like `$ date`. The system timezone can be set using the environment variable `TZ`.

Example using the same timezone for both:

```
1
2
3
4
5
6
7
8
9
```

```
docker volume create n8n_data


docker run -it --rm \
 --name n8n \
 -p 5678:5678 \
 -e GENERIC_TIMEZONE="Europe/Berlin" \
 -e TZ="Europe/Berlin" \
 -v n8n_data:/home/node/.n8n \
 docker.n8n.io/n8nio/n8n
```

Updating#

From your Docker Desktop, navigate to the **Images** tab and select **Pull** from the context menu to download the latest n8n image:

You can also use the command line to pull the latest, or a specific version:

```
1
2
3
4
5
6
7
8
```

```
# Pull latest (stable) version
docker pull docker.n8n.io/n8nio/n8n

# Pull specific version
docker pull docker.n8n.io/n8nio/n8n:0.220.1

# Pull next (unstable) version
docker pull docker.n8n.io/n8nio/n8n:next
```

Stop the container and start it again. You can also use the command line:

```
1
2
3
4
5
6
7
8
9
10
11
```

```
# Get the container ID
docker ps -a

# Stop the container with ID container_id
docker stop [container_id]

# Remove the container with ID container_id
docker rm [container_id]

# Start the container
docker run --name=[container_name] [options] -d
docker.n8n.io/n8nio/n8n
```

Docker Compose#

If you run n8n using a Docker Compose file, follow these steps to update n8n:

```
1
2
3
4
5
6
7
8
```

```
# Pull latest version
docker compose pull

# Stop and remove older version
docker compose down

# Start the container
docker compose up -d
```

Further reading#

More information about Docker setup can be found in the README file of the Docker Image.

n8n with tunnel#

Danger

Use this for local development and testing. It isn't safe to use it in production.

To be able to use webhooks for trigger nodes of external services like GitHub, n8n has to be reachable from the web. n8n has a tunnel service which redirects requests from n8n's servers to your local n8n instance.

Start n8n with `--tunnel` by running:

```
1
2
3
4
5
6
7
8
```

```
docker volume create n8n_data


docker run -it --rm \
 --name n8n \
 -p 5678:5678 \
 -v n8n_data:/home/node/.n8n \
 docker.n8n.io/n8nio/n8n \
 start --tunnel
```

Next steps#

- Learn more about configuring and scaling n8n.
- Or explore using n8n: try the Quickstarts.