n8n Troubleshooting Manual

Table of contents

- Purpose & scope
- How to use this guide
- Quick pre-checks
- Full troubleshooting sections
    - n8n local instance down
    - Container mismatch / stale image
    - Duplicate containers
    - Webhook URL mismatch (local vs tunnel vs prod)
    - pdf_purge & file-storage issues (path, volumes, permissions)
    - `.env` & `vite.config.ts` gotchas (repo-specific)
    - Path & permission checks
- Validation tests and quick checks
- Recommended `docker-compose.yml` example (dev)
- Moving n8n from local → cloud (checklist)
- Repo-specific findings & references
- Quick reference commands
- PDF export instructions

Purpose & scope

- A compact, actionable troubleshooting manual you can keep and reference when any n8n-related failure occurs while running the end-to-end agent. Keep this as the canonical checklist and reference.

How to use this guide

- Start at "Quick pre-checks" and work down the sections matching the symptom you observe.
- Use the repo-specific references to confirm what your app expects (webhook route, env var names, Vite proxy).
- Convert to PDF using the instructions at the end for sharing or future offline use.

Quick pre-checks (run first)

- docker: `docker ps` — is an `n8n` container running?
- compose: `docker-compose ps` in the project folder used to launch n8n
- logs: `docker logs -f <n8n-container>` or `docker-compose logs -f n8n`
- webhook: check the webhook address the app is calling (see `VITE_N8N_AI_SUMMARY_WEBHOOK` usage)
- pdf files: verify host folder exists with `ls -la ./storage/pdfs` and is mounted in container
- Vite dev proxy: check `vite.config.ts` proxy mapping to `http://localhost:5678`

Detailed troubleshooting & fixes

1. n8n local instance down

- Symptoms: UI unreachable (e.g., `http://localhost:5678`), container status `Exited`.
- Quick checks:
    - `docker ps -a | grep n8n` — any `Exited` state?
    - `docker logs <container>` — inspect errors (DB, permission, port).
    - Check host port conflicts: `netstat -ano | findstr 5678` (Windows) or `ss -ltnp | grep 5678` (Unix).
- Typical root causes & fixes:
    - DB connection issues → verify DB env vars and that Postgres is running.
    - Port conflict → stop conflicting service or remap ports in compose.
    - Missing persistent storage or permission errors → check volume mounts and container user ownership.
- Recovery:
    - `docker-compose down && docker-compose up -d --build`
    - If persistent data corrupt, restore `.n8n` folder from backup.

2. Container mismatch (stale/wrong image)

- Symptoms: behavior differs from expected (features missing, different API).
- Checks:
    - `docker inspect <container>` for image ID.

    - `docker images | grep n8n` to see available tags.

- Fix:
    - Pull intended tag: `docker pull n8nio/n8n:<tag>`
    - Recreate: `docker rm -f <container> && docker-compose up -d --force-recreate --build`
- Prevention:
    - Pin image tag in `docker-compose.yml` (avoid `:latest` in production).

3. Multiple container instances (duplicates)

- Symptoms: two containers listening or conflicting ports; agent calls the wrong instance.
- Checks:
    - `docker ps` — look for multiple n8n containers or duplicate port mapping.
    - `docker-compose -f <path> ps` — ensure correct compose project.
- Fix:
    - Stop duplicates: `docker rm -f <container-id>`
    - Consolidate the compose files and avoid launching the same service from multiple folders.
- Prevention:
    - Use explicit `container_name` or unique compose `project_name`.

4. Webhook URL not matching running n8n instance

- Why it fails: the app posts to a webhook URL that's not handled by the live n8n; n8n advertises a webhook base URL that must match the public URL or tunnel.
- How to check:
    - In your front-end, check env var usage: `VITE_N8N_AI_SUMMARY_WEBHOOK` (fallbacks in code exist).
    - Confirm Vite proxy rewrites map the front-end path to the correct internal n8n path.
- Fixes:
    - Local dev: use `http://localhost:5678/webhook/...` or configure `vite.config.ts` to proxy client requests to n8n (see repo proxy).
    - When using ngrok/localtunnel: set `WEBHOOK_URL` (n8n env) to the public tunnel URL so n8n advertises that to external services.
    - Production: set `WEBHOOK_URL` to `https://n8n.yourdomain.com` and ensure reverse proxy routes `/webhook/*` to n8n.
- Validate:
    - `curl -I <full-webhook-url>` should return 200 or the header expected.

5. pdf_purge & file-storage issues (path, volumes, purge job)

- Typical problems:
    - Files missing after restart → storage directory not mounted or container writes to ephemeral location.
    - Purge script cannot find files → incorrect path or missing `.env` var.
    - Vite dev server denies file access to host path.
- Checklist to fix:
    - Mount host folder as a volume: host `./storage/pdfs` → container `/data/pdfs`.
    - Use consistent absolute paths in workflows and scripts: e.g., `PDF_DIR=/data/pdfs`.
    - Check ownership & permissions inside container:
        - Host: `ls -la ./storage/pdfs`
        - Container: `docker exec -it <n8n> ls -la /data/pdfs`
        - Adjust `chown`/`chmod` or run container with proper UID if needed.
    - If purge is external script, ensure it runs as a user that can access the mounted folder.
- Vite dev server adjustments:
    - Vite may block dev server from accessing files outside project root. Add `server.fs.allow` entries in `vite.config.ts` to permit access to `./storage` if the dev UI reads files.
    - If front-end uses a proxy to send PDFs to n8n, ensure proxy rules include that path.

6. `.env` and `vite.config.ts` gotchas (repo-specific)

- Repo references (what was found):
    - `vite.config.ts` contains an explicit proxy mapping from `/api/n8n/ai-summary` to `http://localhost:5678` and rewrites to `/webhook/ai-summary`.
    - The front-end calls a webhook URL from env or fallback. See usage in `src/components/PatientTab/index.tsx` where the code reads `VITE_N8N_AI_SUMMARY_WEBHOOK` or falls back to `'https://your-n8n-instance.com/webhook/ai-summary'`.
- Implications:
    - If `vite.config.ts` proxy is used, the front-end should POST to `/api/n8n/ai-summary` in dev; that will be proxied to local n8n `/webhook/ai-summary`.
    - If front-end posts directly to the public webhook env `VITE_N8N_AI_SUMMARY_WEBHOOK`, ensure that env points to reachable URL in the current environment.
- Fixes & recommended `.env` entries:
    - For dev:
        - `VITE_N8N_AI_SUMMARY_WEBHOOK=/api/n8n/ai-summary` (so fetch uses local proxy)
        - `N8N_HOST=0.0.0.0`
        - `N8N_PORT=5678`
        - `WEBHOOK_URL=http://localhost:5678` (or public tunnel if needed)
    - For prod:
        - `VITE_N8N_AI_SUMMARY_WEBHOOK=https://n8n.yourdomain.com/webhook/ai-summary`
        - Ensure `WEBHOOK_URL` on the n8n service equals `https://n8n.yourdomain.com`.

7. Path & permission checks

- Always use explicit volume mapping, e.g. host `./storage/pdfs` → container `/data/pdfs`.
- Check file existences both on host and in container.
- If a background job runs as different user, ensure that user's UID/GID can access the mounted files.

Validation tests / quick checks

- Container health: `docker ps --filter name=n8n`
- Tail logs while triggering: `docker-compose logs -f n8n`
- Test webhook (dev proxy): `curl -X POST http://localhost:5173/api/n8n/ai-summary -d '{}' -H 'Content-Type: application/json' -v`
- Test webhook (direct): `curl -X POST http://localhost:5678/webhook/ai-summary -d '{}' -H 'Content-Type: application/json' -v`
- PDF existence: `ls -la ./storage/pdfs && docker exec -it <n8n> ls -la /data/pdfs`

Recommended `docker-compose.yml` (dev example)

- Use pinned tags in prod; dev example:

```yaml
version: '3.8'
services:
  n8n:
    image: n8nio/n8n:latest
    container_name: n8n
    restart: unless-stopped
    ports:
      - "5678:5678"
    environment:
      - N8N_HOST=0.0.0.0
      - N8N_PORT=5678
      - WEBHOOK_URL=${WEBHOOK_URL:-http://localhost:5678}
      - GENERIC_TIMEZONE=UTC
      - DB_TYPE=postgresdb
      - DB_POSTGRESDB_HOST=postgres
      - DB_POSTGRESDB_DATABASE=n8n
      - DB_POSTGRESDB_USER=n8n
      - DB_POSTGRESDB_PASSWORD=n8n
    volumes:
      - ./n8n-data:/home/node/.n8n
      - ./storage/pdfs:/data/pdfs
  postgres:
    image: postgres:15
    environment:
      - POSTGRES_USER=n8n
      - POSTGRES_PASSWORD=n8n
      - POSTGRES_DB=n8n
    volumes:
      - ./postgres-data:/var/lib/postgresql/data
```

Moving n8n from local → cloud (what to change)

- Public endpoint:
  - Set `WEBHOOK_URL` to `https://n8n.yourdomain.com` in n8n env.
  - Ensure DNS + TLS (Let's Encrypt) are configured on reverse proxy (Traefik/nginx).
- DB:
  - Move to managed Postgres; update DB env vars to managed DB host/credentials.
- Persistence:
  - Use persistent storage (EBS, PVC) for `~/.n8n` and any file storage (PDFs).
- Security:
  - Enable UI auth (`N8N_BASIC_AUTH_ACTIVE`) and secure credentials.
  - Use TLS and firewall rules to restrict access.
- Scaling:
  - For heavy execution, switch to queue mode: use Redis & `EXECUTIONS_PROCESS=queue`.
- Webhook reliability:
  - Use public domain + TLS; avoid exposing using ephemeral ngrok in production.
- Networking best practice:
  - If your application and n8n are in the same cloud, prefer internal service calls (private network) and secure with tokens rather than hitting public URLs unnecessarily.

Repo-specific findings & notes

- `vite.config.ts`: repo proxies `/api/n8n/ai-summary` → `http://localhost:5678/webhook/ai-summary`. This is the dev shortcut used by the front-end to call the n8n webhook without needing a public `WEBHOOK_URL`.
- `src/components/PatientTab/index.tsx`: front-end uses `VITE_N8N_AI_SUMMARY_WEBHOOK` env or fallback `https://your-n8n-instance.com/webhook/ai-summary`. Make sure in dev you set `VITE_N8N_AI_SUMMARY_WEBHOOK=/api/n8n/ai-summary` or the fetch will try to call external URL and fail.
- Many repository docs (e.g., `QUICK_FIX_GUIDE.md`, `STORAGE_SETUP_MANUAL.md`, `FIX_UPLOADED_REPORTS_GUIDE.md`) already document storage and migrations — ensure you run migrations and create Supabase buckets when relevant.

Quick reference commands

- List running containers: `docker ps`
- Show all containers: `docker ps -a`
- Inspect logs: `docker logs -f <container>` or `docker-compose logs -f n8n`
- Stop/remove container: `docker rm -f <container>`
- Recreate compose: `docker-compose down && docker-compose up -d --build`
- Test webhook (dev proxy): `curl -X POST http://localhost:5173/api/n8n/ai-summary -d '{}' -H 'Content-Type: application/json'`

PDF export instructions (pick one)

- Quick (pandoc):

    1. Save this Markdown as `n8n-troubleshooting-manual.md`.
    2. `pandoc n8n-troubleshooting-manual.md -s -o n8n-troubleshooting-manual.pdf`

- VS Code:

    1. Open `n8n-troubleshooting-manual.md` in VS Code.
    2. Open Markdown Preview (Ctrl+Shift+V) → Print → Save as PDF.

- Node tool:

    1. `npm i -g markdown-pdf`
    2. `markdown-pdf n8n-troubleshooting-manual.md -o n8n-troubleshooting-manual.pdf`

- If you want, I can generate the PDF for you and provide a download — I'll need permission to create files in the workspace (or you can run one of the commands above).

Next actions I can take (choose one)

- Produce the PDF now (I will create `n8n-troubleshooting-manual.pdf` in the repo) — I need your approval to write files.
- Tweak the manual to include any specific log excerpts or incidents you want called out (paste them here).
- Propose specific `.env` / `vite.config.ts` edits (I will follow your pre-change protocol and get explicit approval before applying).

End of manual.