THE ULTIMATE GUIDE TO DEPLOYING n8n COMMUNITY EDITION

HOUSE OF LOOPS COMMUNITY

Table of Contents

14. **Deploying with CapRover or Dokku**

Part III: Advanced Container & Kubernetes Deployments

15. **Docker Compose with PostgreSQL Database**
16. **Docker Compose with Postgres and Redis (Queue Mode)**
17. **Docker Swarm Cluster Deployment**
18. **Kubernetes Deployment (Single Pod)**
19. **Kubernetes with External Database**
20. **Kubernetes with Ingress and Let's Encrypt**
21. **Scaled Kubernetes (Queue Mode)**
22. **Deploying n8n via Helm Chart**
23. **AWS ECS Fargate Deployment**
24. **Google Cloud Run Serverless Deployment**
25. **Azure Container Instances Deployment**
26. **HashiCorp Nomad Deployment**

Part IV: Enterprise & Creative Architectures

27. **High Availability Setup (Horizontal Scaling)**
28. **Multi-Region Deployment & Failover**
29. **Backup, Restore, and Disaster Recovery**
30. **CI/CD and GitOps for n8n Deployments**
31. **Centralized Logging and Monitoring**

32. **Security Hardening and Best Practices**
33. **Workflow Version Control Strategies**

Preface

Welcome to **The Ultimate Guide to Deploying n8n Community Edition**—your one-stop, hands-on companion to mastering n8n from local experiments to global, production-grade orchestrations. Born from our collective experience building, securing, and scaling automation platforms, this book is crafted for:

- **Developers** eager to connect disparate services and eliminate manual toil.
- **System Administrators & DevOps Engineers** seeking reproducible, secure, and scalable deployments.
- **Advanced Hobbyists & Innovators** curious about harnessing self-hosted, open-source automation.

In these pages, you'll discover not only *how* to install and run n8n, but *why* design decisions matter—whether choosing Node.js for a lightweight local setup, Kubernetes for resilient container orchestration, or serverless options like Cloud Run and Fargate for hands-off infrastructure management.

Each chapter combines:

- **Step-by-Step Instructions:** Clear commands, configuration files, and troubleshooting tips.
- **Scenarios & Variations:** Examples spanning local VMs, cloud VMs, containers, orchestrators, and serverless platforms.

- **Security & Best Practices:** From basic auth and TLS to NetworkPolicies, IAM, and secrets management.
- **Observability & Recovery:** Logging, metrics, tracing, backups, and disaster recovery runbooks.
- **CI/CD & GitOps:** Treat your workflows as code, automate tests, and deploy via Argo CD or Flux.
- **Enterprise Architectures:** High availability, multi-region failover, and governance strategies for large teams.

No matter your background—whether you're spinning up your first workflow locally or architecting a global automation fabric—this guide will equip you with the concepts, commands, and configurations you need to succeed.

Introduction to n8n Community Edition

What is n8n?

n8n is a free, open-source workflow automation tool designed for both non-developers and seasoned engineers. At its core, n8n offers:

- A **visual, node-based editor** where each node represents a discrete task—sending emails, querying databases, invoking APIs, or processing data.
- **Over 200 built-in integrations** for popular services (HTTP, databases, messaging platforms, cloud providers) and the flexibility to craft custom nodes.
- **Self-hosting capability**, giving you full control over infrastructure, data sovereignty, and customizations without vendor lock-in.

Whether you need to orchestrate complex ETL pipelines, automate routine administrative chores, or build event-driven microservices, n8n's intuitive interface and extensible architecture put the power of automation in your hands.

Key Features of Community Edition

- **Persistent Workflows & Credentials:** Safeguard your workflows, credentials, and settings in a local database or external PostgreSQL with robust backup options.
- **Queue Mode & Scalability:** Offload execution to worker processes via Redis, enabling high throughput and horizontal scaling.
- **Open API & Webhooks:** Expose endpoints easily, integrate with webhook-driven architectures, and embed custom logic through the REST API.
- **Self-Hosted Flexibility:** Deploy on anything from your laptop to Docker Swarm, Kubernetes, HashiCorp Nomad, or serverless platforms—allowing bespoke security and network topologies.
- **Vibrant Community:** Benefit from community-contributed nodes, templates, and forum support to accelerate your automation journey.

Why Deploy n8n Yourself?

Self-hosting n8n Community Edition grants you:

1. **Data Ownership & Privacy:** Keep sensitive information within your environment, under your security controls.
2. **Customization & Extensibility:** Tailor node behavior, UI themes, and execution environments to your requirements.
3. **Cost Efficiency:** Avoid recurring per-user or per-workflow licensing fees; leverage open-source power.

4. **Security & Compliance:** Apply enterprise-grade measures—TLS, network segmentation, IAM, audit logging, and secrets management—aligned with your policies.
5. **Performance & Scale:** Optimize resource allocation—local development, cloud VMs, containers, or serverless—based on workload patterns.

Book Structure and Approach

We've organized this book into **four progressive parts**, each building on the last:

1. **Basic Local Deployments:** Get started quickly on your machine with Node.js, Docker Compose, and PM2, even on ARM devices like Raspberry Pi.
2. **Intermediate VPS/Cloud VM Deployments:** Migrate to production-like environments on Ubuntu VMs, secure your service with Nginx and Let's Encrypt, or use managed VPS offerings.
3. **Advanced Container & Kubernetes Deployments:** Leverage container orchestration—Docker Swarm, Helm, and native Kubernetes—to achieve resilience, scaling, and automated rollouts.
4. **Enterprise & Creative Architectures:** Architect globally distributed, fault-tolerant systems with high availability, multi-region failover, DR, CI/CD, GitOps, observability, security hardening, and collaborative version control.

Each chapter is self-contained yet designed to interlock, so you can jump to topics most relevant to your stage or follow straight through for a comprehensive mastery of n8n deployments. Along the way, you'll collect a toolkit of code snippets, configuration templates, and operational insights you can adapt to your organization's unique needs.

Let's begin our journey—starting with the simplest installation method—so you can see your first n8n workflow come to life in minutes. Welcome to the world of open-source automation!

Chapter 1: Local Installation via Node.js (Windows/Mac/Linux)

Overview

This chapter walks you through installing n8n Community Edition locally using Node.js across multiple operating systems and scenarios. You'll see:

1. **Official Installers** for Windows and macOS
2. **Package-Manager Installs** on popular Linux distributions (Ubuntu, Debian, CentOS/RHEL, Arch)
3. **Node Version Manager (nvm)** for side-by-side Node.js versions
4. **Alternative Package Tools** (Yarn, pnpm)
5. **Corporate/Offline Environments** with proxies or no internet
6. **Advanced Flags** for `n8n` CLI (custom ports, tunnels, database options)

By the end of this chapter, you'll have a rich understanding of how to get n8n running on any desktop or server for development and testing.

flowchart TD A[Choose Install Method] --> B[Official Installer] A --> C[Package Manager] A --> D[nvm] B --> E[Verify node/npm] C --> E

D --> E E --> F[Install n8n Globally] F --> G[Configure ENV Variables] G --> H[Run n8n CLI] H --> I[Test Workflows]

Prerequisites

- **Administrative Rights** on your machine (install software globally)
- **Internet Access**, or prepared **offline installers**
- **Basic CLI Proficiency** (terminal, PowerShell, or CMD)
- Recommended: **Node.js ≥ 22.15.0 LTS**

Step 1: Installing Node.js

1.1 Official Installer (Windows & macOS)

1. **Download** the LTS installer from https://nodejs.org
2. **Run** the installer and follow prompts:
    - Accept license
    - Select “Add to PATH”
    - Install npm by default
3. **Verify**:

```
node -v
npm -v
```

4. **Troubleshoot**:

- On Windows, if `node` isn't recognized, ensure **"Add to PATH"** was checked.
- On macOS, if version is old, consider using Homebrew or nvm (below).

1.2 Package Manager (Linux)

Ubuntu / Debian

```
# Using NodeSource for current LTS
curl -fsSL https://deb.nodesource.com/setup_22.x | sudo -E bash
-
sudo apt update && sudo apt install -y nodejs build-essential
node -v && npm -v
```

CentOS / RHEL

```
curl -fsSL https://rpm.nodesource.com/setup_lts.x | sudo bash -
sudo yum install -y nodejs gcc-c++ make
node -v && npm -v
```

Arch Linux

```
sudo pacman -Syu nodejs npm
node -v && npm -v
```

Alternative via Homebrew (macOS & Linux)

```
brew update
brew install node
```

1.3 Using Node Version Manager (nvm)

Allows side-by-side Node.js versions, ideal for switching.

```
# Install nvm
curl -o- v0.40.2/install.sh | bash
# Restart shell or source nvm
source ~/.nvm/nvm.sh

# Install and use LTS
nvm install --lts
nvm use --lts
# Optional: set default
nvm alias default lts/*
```

Check:

```
node -v
npm -v
```

1.4 Offline / Corporate Proxy Scenarios

- **Offline installer**: Download `.msi` / `.pkg` or Linux tarball on a machine with internet, transfer via USB, and install manually.
- **Behind a proxy**:

```
npm config set proxy http://proxy.company.local:3128
npm config set https-proxy http://proxy.company.local:3128
export HTTP_PROXY=http://proxy.company.local:3128
export HTTPS_PROXY=http://proxy.company.local:3128
```

- **Cert Issues**:

```
npm config set strict-ssl false
```

Step 2: Installing n8n Globally

You can use npm, Yarn, or pnpm depending on your preference.

2.1 npm (Standard)

```
npm install -g n8n
n8n --version   # should print installed version
```

- **Flags**:
  - `--unsafe-perm=true` on Linux if permission errors
  - `--registry=https://registry.npmjs.org/` for custom registries

2.2 Yarn

```
npm install -g yarn       # if you don't have Yarn
yarn global add n8n
```

```
n8n --version
```

2.3 pnpm

```
npm install -g pnpm
pnpm add -g n8n
n8n --version
```

Step 3: Configuring Environment Variables & Basic Authentication

n8n reads settings via environment variables. For local dev, basic auth is recommended.

3.1 Linux/macOS (bash / zsh)

Temporarily in session:

```
export N8N_BASIC_AUTH_USER="admin"
export N8N_BASIC_AUTH_PASSWORD="SuperSecret123"
export N8N_PORT="5678"           # optional
export N8N_PROTOCOL="https"      # if using self-signed
```

Persist via `~/.bashrc` or `~/.zshrc`.

3.2 Windows

- **PowerShell**:

```
$env:N8N_BASIC_AUTH_USER="admin"
$env:N8N_BASIC_AUTH_PASSWORD="SuperSecret123"
$env:N8N_PORT="5678"
```

- **CMD**:

```
set N8N_BASIC_AUTH_USER=admin
set N8N_BASIC_AUTH_PASSWORD=SuperSecret123
set N8N_PORT=5678
```

3.3 .env File & dotenv

Create a `.env` in your project folder:

```
N8N_BASIC_AUTH_USER=admin
N8N_BASIC_AUTH_PASSWORD=SuperSecret123
N8N_PORT=5678
```

Then start via:

```
npx dotenv n8n
```

3.4 PM2 Ecosystem Injection

```
// ecosystem.config.js
module.exports = {
  apps: [{
    name: 'n8n',
    script: 'n8n',
```

```
    env: {
      N8N_BASIC_AUTH_USER: 'admin',
      N8N_BASIC_AUTH_PASSWORD: 'SuperSecret123',
      N8N_PORT: 5678,
    }
  }]
};
```

Step 4: Running n8n

4.1 Basic Invocation

```
n8n
```

- **Default URL**: http://localhost:5678

4.2 Custom Port / Host

```
n8n start --tunnel --port 8080 --host 0.0.0.0
```

Flags:

- `--tunnel` – temporary public URL for webhooks
- `--db` – specify a database (e.g. SQLite vs Postgres)
- `--docs` – serve local docs

4.3 Using Yarn / pnpm

```
yarn n8n start --host=0.0.0.0
pnpm exec n8n start --tunnel
```

Step 5: Advanced Local Scenarios

5.1 Multiple Node.js Versions

With **nvm**, run:

```
nvm install 23
nvm install 22
nvm use 23 && n8n --version
nvm use 22 && n8n --version
```

5.2 Installing Specific n8n Versions

List of versions: https://www.npmjs.com/package/n8n?activeTab=versions

```
npm install -g n8n@1.89.2
```

5.3 From Source (GitHub)

```
git clone https://github.com/n8n-io/n8n.git
cd n8n
npm install
```

```
npm run build
npm run start
```

5.4 CI / Headless Environments

```
docker run --rm -e N8N_BASIC_AUTH_USER=... -e
N8N_BASIC_AUTH_PASSWORD=... n8nio/n8n:latest n8n start
```

5.5 Integrating with Local Reverse Proxy

- Use **Caddy**, **Nginx**, or **Traefik** locally to handle HTTPS and domain routing.

Troubleshooting Common Issues

Problem	Possible Cause	Solution
`n8n: command not found`	Global bin not in PATH	Add `$(npm root -g)/.bin` to PATH or reinstall Node.js with “Add to PATH”
Permission denied on Linux	Global install directory owned by root	Use `sudo npm install -g n8n` or fix npm prefix: `npm config set prefix ~/npm-global`

Problem	Possible Cause	Solution
Port 5678 already in use	Another app on same port	Run `n8n start --port 5679` or kill the other process
Environment variables not loading	Shell startup file not sourced	Ensure you added exports to `~/.bashrc` / `~/.zshrc` and ran `source ~/.bashrc`
Proxy / SSL errors	Corporate proxy blocking npm	Configure `npm config set proxy` and `strict-ssl false`, or install certs
Antivirus blocks Node.js on Windows	Security software interfering	Whitelist `node.exe` and the global npm directory
Native module build failures	Missing build tools on Linux	Install `build-essential` (Ubuntu/Debian) or `gcc-c++ make` (CentOS/RHEL)

Summary & Next Steps

You now have n8n running locally in multiple ways:

- **Installed via official installers** (Windows/macOS)
- **Package managers** on Linux
- **nvm** for version flexibility

- **Various package tools** (npm, Yarn, pnpm)
- **Advanced CLI flags** and authentication

In the next chapter, we'll containerize this setup using **Docker Compose** for reproducible, isolated deployments with persistent data.

Chapter 2: Running n8n in Local Docker & Docker Compose with Persistent Data

---

Overview

In this chapter, you'll learn how to run n8n using Docker:

1. **Direct** `docker run` invocations—ideal for quick tests and ephemeral use
2. **Docker Compose** setups—enabling data persistence across restarts
3. **Advanced scenarios** including multi-container stacks (PostgreSQL, Redis), custom networks, offline (air-gapped) environments, and Docker secrets

You'll see how to handle host-mounted volumes versus named volumes, customize environment variables, and back up your data. By the end, you'll have a robust, reproducible local Docker setup suitable for both development and staging.

flowchart TD A[Local Machine] --> B[Docker Engine] B --> C[docker run n8n] B --> D[Docker Compose Stack] D --> E[n8n Container] D --> F[Postgres Container] D --> G[Redis Container] E --> H[Persistent Volume]

Prerequisites

- **Docker Engine** installed (see Docker docs)
- **Docker Compose** (v2.35.1)
- **Terminal / PowerShell** access with permissions to run Docker
- Basic familiarity with Docker concepts (images, containers, volumes, networks)

1. Running n8n with `docker run`

For simple testing or one-off executions, the `docker run` command requires no extra files.

1.1 Quick Start (Ephemeral)

```
docker run --rm -it \
  -p 5678:5678 \
  -e N8N_BASIC_AUTH_USER=admin \
  -e N8N_BASIC_AUTH_PASSWORD=SuperSecret \
  n8nio/n8n:latest
```

- `--rm` cleans up the container after exit
- `-it` allocates a pseudo-TTY for interactive logs
- `-p 5678:5678` maps port 5678
- `-e` sets basic auth credentials

1.2 Adding Persistence with a Host-Mounted Volume

Store workflows and credentials on your host:

```
mkdir -p ~/n8n_data
docker run -d \
  --name n8n-docker \
  -p 5678:5678 \
  -e N8N_BASIC_AUTH_USER=admin \
```

```
  -e N8N_BASIC_AUTH_PASSWORD=SuperSecret \
  -v ~/n8n_data:/home/node/.n8n \
  --restart unless-stopped \
  n8nio/n8n:latest
```

- `-v ~/n8n_data:/home/node/.n8n` persists data in your home directory
- `--restart unless-stopped` ensures the container restarts on crashes or reboots

1.3 Named Volume (Docker-Managed)

```
docker volume create n8n_data
docker run -d \
  --name n8n-docker \
  -p 5678:5678 \
  -e N8N_BASIC_AUTH_USER=admin \
  -e N8N_BASIC_AUTH_PASSWORD=SuperSecret \
  -v n8n_data:/home/node/.n8n \
  --restart unless-stopped \
  n8nio/n8n:latest
```

- Pros: Docker manages storage location
- Cons: Harder to access raw files on the host

---

2. Docker Compose for Reproducible Stacks

When you require multiple services or more complex configuration, Docker Compose excels.

2.1 Basic `docker-compose.yml`

Create a file `docker-compose.yml`:

```
services:
  n8n:
    image: n8nio/n8n:latest
    container_name: n8n
    ports:
      - '5678:5678'
    environment:
      N8N_BASIC_AUTH_USER: admin
      N8N_BASIC_AUTH_PASSWORD: SuperSecret
      N8N_PORT: 5678
    volumes:
      - n8n_data:/home/node/.n8n
    restart: unless-stopped

volumes:
  n8n_data:
```

Bring it up:

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

Verify:

```
docker-compose ps
docker volume ls
```

2.2 Windows PowerShell Volume Paths

```
volumes:
  - C:\Users\YourUser\n8n_data:/home/node/.n8n
```

---

3. Advanced Docker Compose Scenarios

3.1 Adding a PostgreSQL Backend

Replace SQLite with Postgres for reliability:

```
services:
  postgres:
    image: postgres:17
    container_name: n8n-postgres
    environment:
      POSTGRES_USER: n8n
      POSTGRES_PASSWORD: secret
      POSTGRES_DB: n8n
    volumes:
      - postgres_data:/var/lib/postgresql/data
    restart: unless-stopped

  n8n:
    image: n8nio/n8n:latest
    container_name: n8n
    ports:
      - '5678:5678'
    environment:
      DB_TYPE: postgresdb
      DB_POSTGRESDB_HOST: postgres
      DB_POSTGRESDB_PORT: 5432
```

```
      DB_POSTGRESDB_DATABASE: n8n
      DB_POSTGRESDB_USER: n8n
      DB_POSTGRESDB_PASSWORD: secret
      N8N_BASIC_AUTH_USER: admin
      N8N_BASIC_AUTH_PASSWORD: SuperSecret
    volumes:
      - n8n_data:/home/node/.n8n
    depends_on:
      - postgres
    restart: unless-stopped

volumes:
  n8n_data:
  postgres_data:
```

- `depends_on` ensures Postgres starts first
- Enables horizontal scaling by separating DB

3.2 Enabling Redis Queue Mode

When you need high throughput, enable Redis for workflow queuing:

```
services:
  redis:
    image: redis:7
    container_name: n8n-redis
    restart: unless-stopped

  n8n:
```

```
    image: n8nio/n8n:latest
    environment:
      QUEUE_MODE: redis
      REDIS_HOST: redis
      REDIS_PORT: 6379
      # (other env like DB or basic auth)
    depends_on:
      - redis
```

3.3 Custom Docker Networks

Isolate services:

```
networks:
  n8n-net:
    driver: bridge

services:
  n8n:
    networks:
      - n8n-net
  postgres:
    networks:
      - n8n-net
```

3.4 Reverse Proxy with Traefik

Integrate Traefik for HTTPS and routing:

```yaml
services:
  traefik:
    image: traefik:v3.3.6
    ports:
      - '80:80'
      - '443:443'
    volumes:
      - ./traefik.toml:/etc/traefik/traefik.toml
      - ./acme.json:/acme.json
      - /var/run/docker.sock:/var/run/docker.sock

  n8n:
    labels:
      - 'traefik.enable=true'
      - 'traefik.http.routers.n8n.rule=Host(`n8n.local`)'
      - 'traefik.http.routers.n8n.entrypoints=websecure'
      - 'traefik.http.routers.n8n.tls.certresolver=myresolver'
    # (other n8n config)
```

4. Offline / Air-Gapped Environments

If Docker Hub isn’t reachable:

1. **Pull images ahead of time** on a machine with access:

```
docker pull n8nio/n8n:latest
docker save n8nio/n8n:latest -o n8n.tar
```

2. **Transfer** `n8n.tar` to air-gapped host, then:

```
docker load -i n8n.tar
```

3. **Run** as usual—no additional internet required.

---

5. Managing Data Persistence & Backups

5.1 Inspecting Volumes

```
docker volume ls
docker volume inspect n8n_data
```

5.2 Backing Up Your Workflows

```
docker run --rm \
  -v n8n_data:/data \
  -v $(pwd):/backup \
  alpine \
  sh -c "tar czf /backup/n8n_backup_$(date +%F).tar.gz -C /data
."
```

5.3 Restoring Data

```
docker run --rm \
  -v n8n_data:/data \
  -v $(pwd):/backup \
  alpine \
  sh -c "tar xzf /backup/n8n_backup_2025-04-21.tar.gz -C /data"
```

---

6. Troubleshooting Tips

Issue	Cause	Remedy
Container won't start	Typo in `docker-compose.yml`	Run `docker-compose config` to validate syntax
Volume permission denied	Host folder owned by root	`sudo chown -R $USER:$USER ~/n8n_data`
Port conflict	Another service bound to 5678	Change mapping: `- '5679:5678'`
Environment variables not applied	Syntax error or missing `environment:` block	Verify indentation and key/value format in YAML
`depends_on` isn't waiting for readiness	Postgres not ready when n8n starts	Add healthchecks or use `wait-for-it.sh`
Data not persisting after `down -v`	Named volume removed	Avoid `-v` when tearing down or back up before `docker-compose down`

Summary & Next Steps

You've learned to:

- Run n8n with a single `docker run` command for quick tests
- Persist data via host-mounts and named volumes
- Define reproducible multi-container stacks with Docker Compose
- Incorporate external services (Postgres, Redis), custom networks, and reverse proxies
- Handle offline, air-gapped scenarios and data backup/restore

In the next chapter, we'll explore **running n8n as a continuous service** using PM2 across Windows, macOS, and Linux environments.

Join the House of Loops Skool Community

Thank you for reading this excerpt of The Ultimate Guide to Deploying n8n Community Edition!

To unlock the complete book, exclusive resources, and connect with a thriving automation community, join us at the House of Loops Skool:

- **Website:** https://www.houseofloops.com
- **Skool Community:** https://skool.com/houseofloops

Why Join?

- **Access the Full Book:** Sign up for our newsletter and get the complete guide, including advanced cloud, Kubernetes, and enterprise chapters.
- **Community Support:** Collaborate with automation experts and peers in the House of Loops Skool community.
- **Startup Credits:** Get access to over **$100,000 in startup credits** to help launch and scale your business—available exclusively to Skool community members.
- **Early Access & Updates:** Receive updates, workflow templates, and deployment recipes before anyone else.

Ready to take your automation journey further?

👉 Join the House of Loops Skool Community

👉 Visit the House of Loops Website

Sign up for the newsletter to receive your full book download and unlock all community benefits!