n8n 셀프 호스팅 완전 가이드: 로컬 설정부터 클라우드까지

📋 목차

🎯 워크플로우 자동화 여정으로의 소개

오늘날의 디지털 환경에서 우리는 끊임없이 여러 애플리케이션 간의 작업을 처리합니다: 새 이메일이 도착하면 스프레드시트 업데이트, 새 고객에 대해 Slack 채널 알림, 데이터베이스에서 보고서 생성 등. 이러한 반복적인 작업들은 필요하지만 소중한 시간과 에너지를 소모합니다. 이것이 바로 워크플로우 자동화 플랫폼이 등장하는 이유이며, n8n은 독특하게 강력하고 유연한 옵션으로 두각을 나타냅니다.

n8n이란? 간단한 비유

디지털 레고 블록 모음이 있다고 상상해보세요. 각 블록은 Gmail, Google Sheets, Slack, Twitter 또는 OpenAI의 복잡한 AI 모델과 같이 매일 사용하는 특정 애플리케이션이나 서비스를 나타냅니다. n8n(엔-에잇-엔으로 발음)은 이러한 블록들을 연결하여 놀라운 것들을 만들 수 있게 해주는 보드이자 설명서입니다.

복잡한 코드를 작성하는 대신, n8n의 시각적이고 노드 기반의 인터페이스를 사용합니다. "노드"는 그런 레고 블록 중 하나입니다. 노드들을 캔버스에 드래그 앤 드롭하고 그들 사이에 선을 그어 "워크플로우"를 만듭니다. 이 워크플로우는 본질적으로 자동화된 작업의 순서도입니다.

예시 워크플로우:

1. 트리거 노드: "제목에 '송장'이 포함된 새 이메일이 Gmail에 도착하면..."
2. 액션 노드: "...첨부된 PDF를 가져와서..."
3. 액션 노드: "...텍스트를 추출하고..."
4. 액션 노드: "...발신자 이름과 송장 금액을 '송장' Google 시트에 새 행으로 추가한다."

왜 셀프 호스팅인가? 제어권과 비용 효율성의 힘

n8n은 설정과 유지보수를 모두 처리해주는 편리한 유료 클라우드 서비스를 제공합니다. 그렇다면 왜 직접 호스팅하려고 할까요? 답은 세 가지 핵심 혜택에 있습니다: **비용, 제어권, 유연성**.

비용 효율성: 가장 중요한 장점입니다. n8n을 셀프 호스팅하면 사실상 무료(기존 컴퓨터에서 실행하는 경우)이거나 공식 클라우드 플랜보다 훨씬 저렴할 수 있습니다. 실행하는 워크플로우 수가 아닌 소비하는 하드웨어나 클라우드 리소스에만 비용을 지불합니다.

완전한 제어권과 데이터 프라이버시: 셀프 호스팅할 때 데이터는 절대 인프라를 벗어나지 않습니다. 워크플로우, 자격 증명, 처리되는 데이터가 모두 본인의 머신이나 개인 클라우드 서버에 상주합니다.

무제한 유연성: 셀프 호스팅된 n8n에는 인위적인 제한이 없습니다. 원하는 만큼 워크플로우를 만들고 원하는 만큼 실행할 수 있습니다.

🐳 Part 1: Docker를 이용한 로컬 설치 (권장 시작점)

n8n이나 셀프 호스팅을 처음 접하는 분들에게는 로컬 설치로 시작하는 것이 최선의 접근법입니다. 재정적 부담이나 복잡한 서버 설정 없이 n8n의 모든 기능을 탐색할 수 있습니다.

왜 Docker인가? 상자 안의 앱

Docker는 소프트웨어 개발자들이 소프트웨어를 구축하고 공유하는 방식을 혁신한 도구입니다. Docker를 복잡한 모형 자동차를 만드는 것에 비유해봅시다. 특정 부품, 특정 도구, 특정 지침이 필요합니다.

Docker는 소프트웨어를 위한 표준화된 운송 컨테이너와 같습니다. 애플리케이션(n8n 같은)을 실행에 필요한 모든 것(코드, 라이브러리, 시스템 도구, 설정)과 함께 "이미지"라는 하나의 깔끔한 패키지로 묶습니다.

Docker 사용의 장점:

- **쉬운 설치 및 관리**: Docker가 설정되면 n8n 실행은 단일 명령으로 간단합니다
- **시스템 격리**: n8n 컨테이너는 메인 운영체제에서 격리됩니다
- **일관성**: Docker를 사용한 n8n 설정은 Windows, macOS, Linux에서 동일하게 작동합니다

로컬 Docker 설정의 단점:

- **컴퓨터가 켜져있어야 함**: 컴퓨터가 켜져있고 Docker 컨테이너가 활성화되어 있을 때만 자동화가 실행됩니다
- **웹훅에는 추가 설정 필요**: 기본적으로 로컬 n8n 인스턴스는 인터넷에서 접근할 수 없습니다

시스템 요구사항 (빠른 체크)

운영체제	최소 요구사항	권장 사양
Windows	Windows 10/11 (64비트), 4GB RAM, 2GB 여유 디스크 공간, PowerShell 5.1+	8GB+ RAM, 4코어+ CPU, SSD 저장공간
macOS	macOS 10.15 (Catalina) 이상, 4GB RAM, 2GB 여유 디스크 공간	8GB+ RAM (M1/M2 Mac 적극 권장), SSD 저장공간

단계별 가이드: Docker 설치

첫 번째 단계는 모든 것을 관리해주는 사용자 친화적인 애플리케이션인 Docker Desktop을 설치하는 것입니다.

Windows 사용자:

1단계: Docker Desktop 다운로드

```powershell
# PowerShell에서 다운로드 (선택사항)
Invoke-WebRequest -Uri "https://desktop.docker.com/win/main/amd64/Docker%20Desktop%20Installer.exe" -Ou
```

GUI 방법 (권장):

1. https://docs.docker.com/desktop/install/windows-install/ 접속
2. "Docker Desktop for Windows" 다운로드
3. 다운로드된 파일 실행 (관리자 권한 필요)
4. WSL 2 백엔드 사용 옵션 체크
5. 시스템 재시작

2단계: WSL 2 활성화 WSL 2가 설치되지 않은 경우 PowerShell을 관리자로 열고 다음 명령어 실행:

```powershell
wsl --install
wsl --set-default-version 2
```

Mac 사용자:

1단계: Docker Desktop 다운로드

1. https://docs.docker.com/desktop/install/mac-install/ 접속
2. Mac 프로세서에 맞는 올바른 버전 선택:
   - Intel Mac: "Mac with Intel chip"
   - Apple Silicon Mac (M1, M2): "Mac with Apple silicon"
3. 다운로드된 .dmg 파일 열기
4. Docker 아이콘을 Applications 폴더로 드래그

Homebrew 대안:

```bash
brew install --cask docker
```

단계별 가이드: Docker로 n8n 실행

Docker가 실행되면 이제 n8n을 시작할 수 있습니다. `docker-compose.yml`이라는 특별한 파일을 사용합니다.

1단계: 전용 폴더 생성

```bash
# Windows (PowerShell에서)
mkdir C:\n8n-docker
cd C:\n8n-docker

# Mac (Terminal에서)
mkdir ~/n8n-docker
cd ~/n8n-docker
```

2단계: docker-compose.yml 파일 생성 새 폴더 내에 `docker-compose.yml`이라는 파일을 생성하고 다음 내용을 복사해 넣습니다:

```yaml
```

```
version: '3.8'

services:
  n8n:
    image: docker.n8n.io/n8nio/n8n
    restart: always
    ports:
      - "5678:5678"
    volumes:
      - n8n_data:/home/node/.n8n
    environment:
      - GENERIC_TIMEZONE=Asia/Seoul  # 본인의 시간대로 변경!
      - N8N_BASIC_AUTH_ACTIVE=true
      - N8N_BASIC_AUTH_USER=admin
      - N8N_BASIC_AUTH_PASSWORD=changeme123

volumes:
  n8n_data: {}
```

3단계: "레시피" 이해하기

- `image`: Docker에게 공식 n8n "레고 세트"를 사용하라고 지시
- `restart: always`: n8n이 충돌하거나 Docker 재시작 시 자동으로 재시작되도록 보장
- `ports`: 컨테이너의 "문"(포트 5678)을 컴퓨터의 "문"(포트 5678)에 연결
- `volumes`: 모든 워크플로우, 자격 증명, 설정을 안전하게 저장하는 영구 저장 공간 생성
- `GENERIC_TIMEZONE`: 시간에 의존하는 워크플로우를 위해 중요

4단계: n8n 실행 터미널이나 PowerShell에서 `n8n-docker` 디렉터리에 있는지 확인하고 다음 명령어 실행:

```
bash

docker-compose up -d
```

5단계: n8n 접속 명령어가 완료되면 웹 브라우저를 열고 다음 주소로 이동:

```
http://localhost:5678
```

소유자 계정을 설정하라는 n8n 환영 화면이 나타날 것입니다. 축하합니다! n8n 셀프 호스팅에 성공했습니다!

🌐 Part 2: 웹훅으로 로컬 n8n을 세상과 연결하기

로컬 머신에서 n8n을 성공적으로 설정했습니다. 서비스에서 데이터를 가져와 처리하고 다른 곳으로 보내는 워크플로우를 구축할 수 있습니다. 하지만 서비스가 실시간으로 데이터를 *보내주기*를 원한다면? 이것이 바로 웹훅이 필요한 곳이

며, 로컬 설정에서는 작지만 중요한 추가 단계가 필요합니다.

웹훅이란 무엇이고 왜 필요한가?

n8n 인스턴스를 홈 오피스에 비유해봅시다. 누구에게든 전화를 *걸* 수 있습니다(API에서 데이터 가져오기처럼). 하지만 누군가가 *당신에게* 전화를 걸려면 전화번호가 필요합니다.

웹훅은 n8n 워크플로우를 위한 특별하고 고유한 "전화번호"(URL)입니다. 다른 애플리케이션들(Stripe, GitHub, 웹사이트의 커스텀 폼 등)이 이벤트가 발생하는 순간 이 "번호로 전화"하여 즉시 데이터를 보낼 수 있습니다.

문제: 로컬 n8n 인스턴스는 `http://localhost:5678`과 같은 주소에 있습니다. 이는 "내 책상에서"라고 말하는 것과 같은 개인적이고 내부적인 주소입니다. 공개 인터넷은 "당신의 책상"을 찾는 방법을 모릅니다.

해결책: Cloudflare Tunnel이라는 무료이고 강력한 도구를 사용합니다. 공개 URL에서 로컬 n8n 인스턴스로 직접 보안 연결("터널")을 만듭니다.

방법 1: Cloudflare로 임시 공개 URL (테스트용)

이 방법은 빠른 테스트에 완벽합니다. 터미널 창을 열어두는 동안 작동하는 랜덤한 공개 URL을 제공합니다.

1단계: cloudflared 명령줄 도구 설치

```bash
# Mac (Homebrew 사용)
brew install cloudflared

# Windows (Winget 사용)
winget install --id Cloudflare.cloudflared
```

2단계: 터널 생성 새 터미널/PowerShell 창을 열고(Docker 실행 중인 창은 닫지 말고) 다음 명령어 실행:

```bash
cloudflared tunnel --url http://localhost:5678
```

출력에서 `https://random-words-and-letters.trycloudflare.com`과 같은 줄을 찾을 수 있습니다. 이것이 임시 공개 URL입니다!

3단계: 새 URL로 n8n 구성 먼저 현재 n8n 컨테이너 중지:

```bash
docker-compose down
```

다음으로 `docker-compose.yml` 파일을 편집하여 `WEBHOOK_URL` 환경 변수 추가:

```yaml
environment:
  - GENERIC_TIMEZONE=Asia/Seoul
  - N8N_BASIC_AUTH_ACTIVE=true
  - N8N_BASIC_AUTH_USER=admin
  - N8N_BASIC_AUTH_PASSWORD=changeme123
  - WEBHOOK_URL=https://your-actual-url.trycloudflare.com  # 실제 URL로 교체
```

마지막으로 n8n 다시 시작:

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

방법 2: Cloudflare로 영구 커스텀 도메인 (프로덕션용)

임시 URL은 테스트에는 훌륭하지만, 진지한 자동화를 위해서는 안정적이고 영구적인 주소가 필요합니다.

1단계: 도메인 획득 및 Cloudflare 연결

1. 도메인 등록업체에서 도메인명 구매
2. Cloudflare에서 무료 계정 생성
3. Cloudflare 지침에 따라 도메인 추가

2단계: 영구 터널 생성

```bash
# Cloudflare 계정에 로그인
cloudflared login

# 명명된 영구 터널 생성
cloudflared tunnel create n8n-tunnel
```

3단계: 터널 구성 `~/.cloudflared/config.yml` (Mac/Linux) 또는 `C:\Users\YourUsername\.cloudflared\config.yml` (Windows) 파일 생성:

```yaml
tunnel: n8n-tunnel
credentials-file: /Users/your_username/.cloudflared/your_tunnel_id.json
ingress:
  - hostname: n8n.yourdomain.com
    service: http://localhost:5678
  - service: http_status:404
```

4단계: DNS 연결 및 터널 실행

```bash
# DNS 레코드 생성
cloudflared tunnel route dns n8n-tunnel n8n.yourdomain.com

# 터널 실행
cloudflared tunnel run n8n-tunnel
```

5단계: 프로덕션용 n8n 구성 업데이트 `docker-compose.yml`을 영구 URL로 최종 업데이트:

```yaml
environment:
  - GENERIC_TIMEZONE=Asia/Seoul
  - N8N_BASIC_AUTH_ACTIVE=true
  - N8N_BASIC_AUTH_USER=admin
  - N8N_BASIC_AUTH_PASSWORD=secure_password123
  - WEBHOOK_URL=https://n8n.yourdomain.com
```

☁️ Part 3: Railway를 이용한 클라우드 설치 (24/7 자동화 엔진)

로컬 설정은 학습과 개발에 환상적입니다. 하지만 자동화가 비즈니스에 중요해지면—주문 처리, 고객 문의 응답, 시스템 모니터링—24/7 안정적으로 실행되어야 합니다. 개인 컴퓨터가 항상 켜져있고 연결되어 있기를 의존하는 것은 견고한 솔루션이 아닙니다.

왜 클라우드인가? "항상 켜져있는" 자동화

클라우드에서 n8n을 호스팅한다는 것은 데이터 센터의 강력하고 관리되는 서버에서 실행한다는 의미입니다. 이 서버는 100% 가동시간을 위해 설계되었고, 고속 인터넷 연결을 가지고 있으며, 전문가들에 의해 유지관리됩니다.

Railway 사용의 장점:

- **24/7 가동시간**: 개인 컴퓨터 상태와 관계없이 워크플로우가 지속적으로 실행
- **간편한 설정**: Railway는 몇 분 만에 프로덕션급 n8n 설정을 배포하는 원클릭 템플릿 제공
- **관리되는 인프라**: 서버 유지보수, 보안 패치, 네트워크 구성을 걱정할 필요 없음
- **웹훅이 즉시 작동**: 이미 공개 인터넷에 있으므로 자동으로 공개 URL 제공

Railway 사용의 단점:

- **비용**: 매우 저렴하지만 무료는 아님. Railway 플랜은 일반적으로 월 $5부터 시작
- **제3자 의존성**: Railway 플랫폼이 작동하는 것에 의존

단계별 가이드: Railway에 n8n 배포

Railway의 템플릿 시스템 덕분에 Railway에 n8n을 배포하는 것은 놀랍도록 간단합니다.

1단계: 가입 및 플랜 선택

1. Railway 웹사이트로 가서 가입 (일반적으로 GitHub 계정으로)
2. Railway는 시작 크레딧이 있는 무료 체험 제공
3. n8n을 지속적으로 실행하려면 유료 플랜 구독 필요

2단계: n8n 템플릿 배포

1. 로그인 후 새 프로젝트 생성
2. "Deploy from Template" 선택
3. 템플릿 마켓플레이스에서 "n8n" 검색
4. 공식 n8n 템플릿 선택 (종종 "Webhook Processor" 등으로 표시)

3단계: 배포된 구성 요소 이해 ("큐 모드" 설정)

"Deploy"를 클릭하면 Railway가 자동으로 여러 서비스를 프로비저닝하고 연결합니다. 이는 단순한 n8n 인스턴스가 아니라 "큐 모드"라고 알려진 전문적이고 확장 가능한 아키텍처입니다.

- **PostgreSQL (데이터베이스)**: 장기 메모리. 저장된 워크플로우, 자격 증명, 모든 실행 기록을 영구적으로 저장
- **Redis (캐시/큐)**: 초고속 단기 메모리 또는 "할 일 목록"
- **n8n-main (기본)**: 뇌이자 얼굴. 워크플로우를 구축, 편집, 관리하기 위해 로그인하는 웹 인터페이스
- **n8n-worker**: 손. Redis 큐에서 새 작업을 지속적으로 확인하는 백그라운드 프로세스
- **n8n-webhook**: 전담 접수원. 들어오는 웹훅을 가능한 한 빨리 처리하여 Redis 큐로 전달하는 것이 유일한 임무

4단계: n8n 인스턴스 접속 및 구성

1. 몇 분 후 모든 서비스가 배포됩니다
2. Railway 프로젝트 대시보드에서 `n8n-main` 서비스 클릭
3. "Settings" 탭으로 이동하여 Railway가 생성한 공개 URL 찾기
4. 해당 링크 클릭하면 로컬 설정과 같은 n8n 설정 화면으로 이동
5. 소유자 계정을 생성하면 준비 완료!

5단계: 배포 관리

- **로그 보기**: 문제가 발생하면 Railway 대시보드에서 각 개별 서비스의 로그를 직접 볼 수 있음
- **n8n 업데이트**: 새 버전이 출시되면 Railway의 "Deployments" 탭에서 "Redeploy" 트리거하여 업데이트

📦 Part 4: npm을 이용한 고급 로컬 설치 (개발자용)

Docker가 단순함과 격리 때문에 대부분의 사용자에게 권장되는 방법이지만, n8n을 로컬에서 실행하는 다른 방법이 있습니다: Node.js JavaScript 런타임의 기본 패키지 매니저인 npm(Node Package Manager)을 사용하는 것입니다.

누구를 위한 것인가?

이 방법은 일반적으로 더 기술적인 대상을 위한 것입니다:

- **Node.js 개발자**: Node.js와 그 생태계로 작업하는 것이 이미 편하다면 더 자연스럽게 느껴질 수 있음
- **커스터마이저와 기여자**: n8n 소스 코드를 살펴보고, 커스텀 수정을 하거나, 프로젝트에 기여하려면 npm 패키지로 직접 작업해야 함

사전 요구사항: Node.js & npm 설치

n8n은 Node.js로 구축되어 있으므로 먼저 설치해야 합니다. npm은 Node.js와 함께 번들로 제공됩니다.

Windows의 경우:

1. 공식 Node.js 웹사이트로 이동
2. LTS(Long Term Support) 버전 다운로드 (가장 안정적인 릴리스)
3. .msi 설치 프로그램을 실행하고 설정 마법사의 단계를 따름

Mac의 경우:

몇 가지 옵션이 있지만 많은 일반적인 권한 문제를 방지하고 서로 다른 Node.js 버전 간 쉽게 전환할 수 있게 해주는 nvm(Node Version Manager) 사용을 권장합니다.

방법 1 (공식 설치 프로그램): Node.js 웹사이트에서 LTS 버전을 다운로드하고 .pkg 설치 프로그램 실행

방법 2 (Homebrew): Homebrew가 있다면 `brew install node` 실행

방법 3 (nvm - 권장):

```bash
# 터미널에서 이 명령어를 실행하여 nvm 설치
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash

# 터미널을 닫고 다시 열기. 그 다음 최신 Node.js LTS 설치 (예: 버전 20)
nvm install 20
nvm use 20
nvm alias default 20
```

설치 확인 새 터미널이나 PowerShell을 열고 다음 명령어를 실행하여 Node.js와 npm이 올바르게 설치되었는지 확인:

```bash
```

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

화면에 버전 번호가 출력되어야 합니다.

npm을 통한 n8n 설치 및 실행

1단계: n8n 전역 설치 npm이 준비되면 단일 명령어로 n8n을 설치할 수 있습니다. `-g` 플래그는 전역으로 설치하여 터미널 어디서든 `n8n` 명령어를 사용할 수 있게 합니다.

```bash
npm install -g n8n
```

2단계: n8n 시작 n8n을 실행하려면 간단히 다음을 실행:

```bash
n8n start
```

이렇게 하면 기본 포트 5678에서 n8n이 시작됩니다. 터미널 창에서 직접 로그 출력을 볼 수 있습니다. 중지하려면 `Ctrl + C`를 누르세요.

3단계: n8n 접속 브라우저를 열고 Docker 설정과 마찬가지로 `http://localhost:5678`로 이동합니다.

추가 명령어:

```bash
# 다른 포트에서 실행
n8n start --port 5679

# 백그라운드에서 실행 (Mac/Linux)
nohup n8n start > n8n.log 2>&1 &
```

🪲 Part 5: 일반적인 문제 해결

최고의 지침이 있어도 때때로 문제에 부딪힐 수 있습니다. 이 섹션에서는 가장 일반적인 문제들과 해결책을 다룹니다. 인내심과 체계적인 문제 해결이 개발자의 가장 좋은 친구라는 것을 기억하세요.

문제 해결의 일반적인 접근법

특정 해결책을 살펴보기 전에 항상 여기서 시작하세요:

1. 로그 확인: 로그는 오류 메시지를 찾아볼 첫 번째 장소입니다. 실패했을 때 애플리케이션이 정확히 무엇을 시도하고 있었는지 알려줍니다.

- **Docker용**: `n8n-docker` 디렉터리에서 `docker-compose logs n8n` 실행. 실시간으로 로그를 따라가려면 `docker-compose logs -f n8n` 사용
- **npm용**: 로그가 터미널 창에 직접 출력됩니다. 백그라운드에서 실행했다면 생성한 로그 파일 확인 (예: `n8n.log`)
- **Railway용**: Railway 대시보드에서 특정 서비스의 "Deploy Logs" 확인

2. 모든 것 재시작: 고전적인 "끄고 다시 켜기"가 놀랍도록 자주 작동합니다.

문제 1: 포트 충돌 ("Port is already allocated")

증상: `Error starting userland proxy: listen tcp4 0.0.0.0:5678: bind: address already in use` 또는 `Port 5678 is already in use`와 같은 텍스트가 포함된 오류 메시지

원인: 컴퓨터의 다른 애플리케이션이 이미 포트 5678을 사용하고 있습니다.

해결책 1: 충돌하는 프로세스 찾기 및 중지

Windows (PowerShell에서):

```
powershell

# 포트를 사용하는 프로세스 ID(PID) 찾기
netstat -ano | findstr :5678

# 해당 프로세스 종료 ([PID]를 번호로 교체)
taskkill /PID [PID] /F
```

Mac/Linux (Terminal에서):

```
bash

# 포트를 사용하는 PID 찾기
lsof -i :5678

# 프로세스 종료
kill -9 [PID]
```

해결책 2: n8n 포트 변경

Docker용: `docker-compose.yml` 파일에서 `ports` 라인을 `"5678:5678"`에서 `"5679:5678"`로 변경하고 `docker-compose up -d`로 재시작. `http://localhost:5679`에서 n8n에 접속.

npm용: 포트 플래그로 n8n 시작: `n8n start --port 5679`

문제 2: 권한 오류

증상: "Permission denied," "EACCES," 또는 "operation not permitted"와 같은 단어가 포함된 오류

Windows 원인: PowerShell에 스크립트 실행을 방지하는 보안 정책이 있습니다.

Windows 해결책: PowerShell을 관리자로 열고 다음 명령어를 실행하여 사용자 계정에 대해 서명된 스크립트 실행을 허용:

```
powershell

Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

Mac 원인: `sudo`로 npm 패키지를 전역 설치하면 파일 권한이 엉망이 되거나, 사용자 계정이 전역 `node_modules` 디렉터리에 대한 쓰기 권한이 없을 수 있습니다.

Mac 해결책: nvm(Part 4에서 설명)을 사용하여 문제를 완전히 피하는 것이 최선의 해결책입니다.

문제 3: 방화벽이 접속 차단

증상: n8n이 터미널/Docker에서 올바르게 실행되는 것처럼 보이지만 브라우저에서 `http://localhost:5678`에 접속할 수 없음(시간 초과)

원인: 컴퓨터의 내장 방화벽이나 안티바이러스 프로그램이 n8n이 사용하는 포트로의 들어오는 연결을 차단하고 있습니다.

Windows 해결책: PowerShell을 관리자로 열고 실행:

```
powershell

New-NetFirewallRule -DisplayName "n8n Allow HTTP" -Direction Inbound -Action Allow -Port 5678 -Protocol TCP
```

Mac 해결책: `시스템 설정 > 네트워크 > 방화벽`으로 이동. 켜져 있다면 "옵션"을 클릭하고 Docker나 Node에 대한 예외를 추가할 수 있는지 확인.

문제 4: Docker/npm 환경 문제

증상: `docker-compose`나 `npm`과 같은 명령어를 찾을 수 없거나 이상하고 비특정적인 오류가 발생

원인: 도구 자체가 잘못된 상태에 있거나 설치 경로가 올바르게 구성되지 않았을 수 있습니다.

Docker 해결책:

- Docker Desktop이 실행 중인지 확인. 시스템 트레이나 메뉴 바에서 고래 아이콘을 찾아보세요
- Docker Desktop 재시작을 시도. 일시적인 결함을 종종 해결합니다
- 디스크 공간이 부족하면 Docker가 불규칙하게 작동할 수 있습니다. `docker system prune` 명령어로 오래되고 사용하지 않는 컨테이너와 이미지를 정리해 보세요

npm 해결책:

- npm 캐시 정리: `npm cache clean --force`

- 최후의 수단으로 Node.js를 완전히 제거하고 재설치

🎯 결론: n8n 셀프 호스팅 모험 선택하기

노트북의 간단한 로컬 설정부터 클라우드의 견고하고 프로덕션 준비가 된 배포까지 n8n 셀프 호스팅의 전체 환경을 여행했습니다. 이제 목표, 기술적 편안함 수준, 예산에 가장 적합한 경로를 선택하는 지식을 갖추었습니다.

선택은 궁극적으로 비용, 편의성, 신뢰성 간의 절충안으로 귀결됩니다.

요약 표: 로컬 Docker vs 클라우드 Railway

기준	Docker 로컬 호스팅	Railway 클라우드 호스팅
비용	사실상 무료 (컴퓨터 리소스 사용)	낮은 월 요금(~$5+)부터 시작하여 사용량에 따라 확장
가동시간 및 신뢰성	PC에 의존. 컴퓨터가 켜져있을 때만 실행	24/7/365. 높은 신뢰성을 위해 설계
설정 복잡성	중간. Docker 설치 및 명령줄 사용 필요	쉬움. 원클릭 템플릿 배포
웹훅 설정	수동. 터널 설정 필요 (예: Cloudflare)	자동. 공개 URL로 즉시 작동
성능	컴퓨터의 CPU와 RAM에 제한	확장 가능한 클라우드 리소스. 더 높은 부하 처리 가능
적합한 용도	학습, 개발, 테스트, 비중요한 개인 자동화	프로덕션 워크플로우, 비즈니스 중요 작업, 대용량 웹훅

최종 권장사항

n8n 여정을 위한 권장 로드맵:

1. 모든 사람은 로컬 머신에서 Docker로 시작해야 합니다. 이것은 의심할 여지없이 n8n의 세부사항을 배우는 최고의 방법입니다. 시간 외에는 비용이 들지 않으며, 안전한 환경에서 모든 기능을 탐색하고 복잡한 워크플로우를 구축할 수 있습니다.

2. 자동화가 중요해지면 클라우드로 마이그레이션하세요. 워크플로우가 상당한 시간이나 돈을 절약하기 시작하거나 비즈니스 프로세스의 중요한 부분이 되는 순간, 졸업할 때입니다. Railway와 같은 서비스는 최소한의 번거로움으로 필요한 신뢰성을 제공하여 "설정하고 잊어버릴" 수 있게 해줍니다.

이제 상상할 수 있는 거의 모든 디지털 서비스를 연결할 수 있는 강력한 자동화를 구축할 준비가 되었습니다. 워크플로우 자동화의 세계는 광대하고 잠재력이 가득합니다. 더 고급 구성을 위해서는 우수한 공식 n8n 문서를 계속 탐색하고 실험하며 참조하기를 권장합니다.

행복한 자동화를 하세요!

📚 추가 학습 자료

공식 문서

- n8n 공식 문서
- Docker 설치 가이드
- npm 설치 가이드

- 보안 모범 사례

Docker 관련 자료

- Docker 공식 문서
- Docker Compose 가이드

Cloudflare 관련 자료

- Cloudflare Tunnel 문서
- Cloudflare Dashboard

Railway 관련 자료

- Railway 공식 문서
- Railway 템플릿 마켓플레이스

Node.js 관련 자료

- Node.js 공식 사이트
- nvm (Node Version Manager)
- npm 공식 문서

커뮤니티 자료

- n8n 커뮤니티 포럼
- n8n GitHub 저장소
- n8n YouTube 채널

이 포괄적인 가이드를 통해 Windows와 Mac 환경 모두에서 n8n을 성공적으로 셀프 호스팅하고, 로컬 개발부터 프로덕션 배포까지 전체 여정을 완주할 수 있습니다.