Hetznet Setup

**1. Create Hetzner Cloud Instance**

1. Go to [Hetzner Cloud Console](https://console.hetzner.cloud/)
2. Create a new project if you don't have one
3. Click "Add Server"
4. Choose:
   * Location: Select nearest region
   * Image: Ubuntu 22.04
   * Type: CX41 (4 vCPU, 8GB RAM - adjust based on your needs)
   * Add SSH Key (recommended) or use password authentication
5. Enable "Volumes" if you need additional storage for models
6. Click "Create & Buy Now"

**2. Connect to Your Server**

1. Find your server IP in the Hetzner console
2. Connect using SSH:

bash

Copy

ssh root@your-server-ip

**3. Initial Server Setup**

bash

Copy

# Update system

apt update && apt upgrade -y

# Install Docker

curl -fsSL https://get.docker.com | sh

# Install Docker Compose

apt install docker-compose-plugin -y

# Verify installation

docker compose version

**4. Prepare Deployment Files**

1. Create project directory:

bash

Copy

mkdir my-stack && cd my-stack

1. Create docker-compose.yml:

yaml

Copy

version: '3'

volumes:

n8n\_storage:

postgres\_storage:

ollama\_storage:

qdrant\_storage:

open-webui:

flowise:

networks:

demo:

x-n8n: &service-n8n

image: n8nio/n8n:latest

networks: ['demo']

environment:

- DB\_TYPE=postgresdb

- DB\_POSTGRESDB\_HOST=postgres

- DB\_POSTGRESDB\_USER=${POSTGRES\_USER}

- DB\_POSTGRESDB\_PASSWORD=${POSTGRES\_PASSWORD}

- N8N\_DIAGNOSTICS\_ENABLED=false

- N8N\_PERSONALIZATION\_ENABLED=false

- N8N\_ENCRYPTION\_KEY=${N8N\_ENCRYPTION\_KEY}

- N8N\_USER\_MANAGEMENT\_JWT\_SECRET=${N8N\_JWT\_SECRET}

depends\_on:

- postgres

x-ollama: &service-ollama

image: ollama/ollama:latest

container\_name: ollama

networks: ['demo']

restart: unless-stopped

ports:

- 11434:11434

volumes:

- ollama\_storage:/root/.ollama

services:

flowise:

image: flowiseai/flowise

networks: ['demo']

restart: unless-stopped

environment:

- PORT=3001

ports:

- 3001:3001

volumes:

- flowise:/root/.flowise

open-webui:

image: ghcr.io/open-webui/open-webui:main

networks: ['demo']

restart: unless-stopped

ports:

- "3000:8080"

volumes:

- open-webui:/app/backend/data

postgres:

image: postgres:15-alpine

networks: ['demo']

restart: unless-stopped

environment:

- POSTGRES\_USER=${POSTGRES\_USER}

- POSTGRES\_PASSWORD=${POSTGRES\_PASSWORD}

- POSTGRES\_DB=${POSTGRES\_DB}

volumes:

- postgres\_storage:/var/lib/postgresql/data

healthcheck:

test: ['CMD-SHELL', 'pg\_isready -h localhost -U ${POSTGRES\_USER} -d ${POSTGRES\_DB}']

interval: 5s

timeout: 5s

retries: 10

n8n:

<<: \*service-n8n

restart: unless-stopped

ports:

- 5678:5678

volumes:

- n8n\_storage:/home/node/.n8n

qdrant:

image: qdrant/qdrant:latest

networks: ['demo']

restart: unless-stopped

ports:

- 6333:6333

volumes:

- qdrant\_storage:/qdrant/storage

ollama:

<<: \*service-ollama

1. Create .env file:

bash

Copy

# Database

POSTGRES\_USER=n8n-user

POSTGRES\_PASSWORD=strong-password-here

POSTGRES\_DB=n8n-db

# n8n Security

N8N\_ENCRYPTION\_KEY=your-encryption-key-here

N8N\_JWT\_SECRET=your-jwt-secret-here

**5. Start the Stack**

bash

Copy

docker compose up -d

**6. Install Reverse Proxy (Nginx)**

bash

Copy

apt install nginx -y

Create /etc/nginx/sites-available/my-stack:

nginx

Copy

server {

listen 80;

server\_name your-server-ip;

location /n8n/ {

proxy\_pass http://localhost:5678/;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

}

location /flowise/ {

proxy\_pass http://localhost:3001/;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

}

location /webui/ {

proxy\_pass http://localhost:3000/;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

}

location /ollama/ {

proxy\_pass http://localhost:11434/;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

}

}

Enable the configuration:

bash

Copy

ln -s /etc/nginx/sites-available/my-stack /etc/nginx/sites-enabled/

nginx -t && systemctl reload nginx

**7. Firewall Configuration**

bash

Copy

ufw allow 80

ufw allow 22

ufw enable

**8. Access Your Services**

Use these URLs from your Windows machine:

* n8n: http://your-server-ip/n8n
* Flowise: http://your-server-ip/flowise
* Open WebUI: http://your-server-ip/webui
* Ollama API: http://your-server-ip/ollama

**9. Post-Installation Steps**

1. Pull Ollama models:

bash

Copy

docker exec -it ollama ollama pull llama2

1. Initialize n8n:
   * Access n8n through the web interface
   * Create your first user account
2. Verify all services:

bash

Copy

docker ps -a

**Important Security Notes:**

1. Change all default passwords in the .env file
2. Consider adding HTTPS with Let's Encrypt
3. Restrict access to specific IPs if possible
4. Regularly update your containers:

bash

Copy

docker compose pull && docker compose up -d

**Troubleshooting:**

1. Check container logs:

bash

Copy

docker logs <container-name>

1. Verify network connections:

bash

Copy

docker network inspect demo

1. Check Nginx logs:

bash

Copy

tail -f /var/log/nginx/access.log

This setup gives you a working stack accessible via your Hetzner IP address. Remember that exposing services directly via IP without HTTPS is not recommended for production use, but suitable for personal/testing environments.

* - -

**Step-by-Step Instructions:**

1. **Save the new compose file**  
   Replace your existing docker-compose.yml with this version
2. **Clean existing containers**

powershell

Copy

docker compose down -v

1. **Start core services**

powershell

Copy

docker compose --profile cpu up -d postgres ollama

1. **Wait for Ollama to initialize** (2-3 minutes)  
   Verify with:

powershell

Copy

docker logs ollama -f

1. **Manually pull models** (Run these one by one):

powershell

Copy

docker exec ollama ollama pull qwen2-math:7b

docker exec ollama ollama pull llama3-chatqa:8b

docker exec ollama ollama pull wizard-math:7b

docker exec ollama ollama pull sqlcoder:7b

docker exec ollama ollama pull deepseek-coder:6.7b

docker exec ollama ollama pull codellama:7b

docker exec ollama ollama pull deepseek-r1:8b

docker exec ollama ollama pull llama2

docker exec ollama ollama pull llama3.2

docker exec ollama ollama pull llama3.1

docker exec ollama ollama pull nomic-embed-text

1. **Start remaining services**

powershell

Copy

docker compose --profile cpu up -d

1. **Verify all containers**

powershell

Copy

docker compose ps

On my local machine

#### **PostgreSQL**

* **What it is**: Open-source relational database
* **Role**: Stores n8n's workflow data and credentials
* **Access**: Port 5432 (but you'll mainly interact via n8n's UI)
* **In your setup**: Used as persistent storage for n8n

#### **n8n**

* **What it is**: Workflow automation tool (like visual programming)
* **Role**: Automate tasks between different services
* **Access**: http://localhost:5678
* **Example use**: Create workflows that trigger AI models in Ollama

#### **Ollama**

* **What it is**: Framework for running large language models (LLMs) locally
* **Role**: Hosts AI models like Llama 3
* **Access**: API at http://localhost:11434
* **In your setup**: Pre-downloads 3 models (llama3.1, llama3.2, nomic-embed-text)

#### **Qdrant**

* **What it is**: Vector similarity search engine
* **Role**: Stores and searches AI embeddings
* **Access**: API at http://localhost:6333
* **Example use**: Store document embeddings for semantic search

#### **Open WebUI**

* **What it is**: ChatGPT-style interface for local LLMs
* **Role**: User-friendly chat interface for Ollama
* **Access**: http://localhost:3000
* **Example use**: Chat with your local Llama 3 model

#### **Flowise**

* **What it is**: Visual LangChain builder
* **Role**: Create AI workflows/chains without coding
* **Access**: http://localhost:3001
* **Example use**: Build RAG (Retrieval-Augmented Generation) pipelines

### **2. How They Work Together**

mermaid

Copy

graph LR

A[Open WebUI] --> B(Ollama)

B --> C(Qdrant)

D[Flowise] --> B

D --> C

E[n8n] --> B

E --> C

E --> F[PostgreSQL]

1. **Open WebUI** provides a friendly chat interface using **Ollama**'s models
2. **Flowise** can create chains that use **Ollama** for processing and **Qdrant** for vector storage
3. **n8n** can orchestrate workflows between all components
4. **PostgreSQL** persists n8n's configuration and workflow data

### **3. Project Examples**

#### **Project 1: Simple Chatbot**

1. Access Open WebUI at http://localhost:3000
2. Create an account
3. Select "Llama3" from model dropdown
4. Start chatting with your local LLM

**Flow**:

Copy

User → Open WebUI → Ollama (Llama3) → Response

#### **Project 2: Document Q&A System**

1. Use Flowise (http://localhost:3001) to create:
   * Document loader (PDF/Text)
   * Text splitter
   * Ollama embeddings (nomic-embed-text)
   * Qdrant vector store
   * Llama3 for final answer
2. Upload documents through Flowise UI
3. Ask questions about your documents

**Flow**:

Copy

Document → Flowise → Ollama(embeddings) → Qdrant → User Question → Ollama(Llama3) → Answer

#### **Project 3: Automated Content Moderator**

Using n8n (http://localhost:5678):

1. Create workflow:
   * Trigger: New form submission
   * Ollama node: Analyze text for inappropriate content
   * Conditional branch: Flag or approve
   * Email/Slack notification
2. Deploy as webhook

**Flow**:

Copy

Web Form → n8n → Ollama(Llama3) Analysis → Conditional Logic → Notifications

### **4. Access Points**

| **Service** | **URL** | **Port** |
| --- | --- | --- |
| Open WebUI | [http://localhost:3000](http://localhost:3000/) | 3000 |
| Flowise | [http://localhost:3001](http://localhost:3001/) | 3001 |
| n8n | [http://localhost:5678](http://localhost:5678/) | 5678 |
| Qdrant | <http://localhost:6333/dashboard> | 6333 |
| Ollama API | [http://localhost:11434](http://localhost:11434/) | 11434 |

### **5. First-Time Setup Guide**

1. Create .env file with:

env

Copy

POSTGRES\_USER=youruser

POSTGRES\_PASSWORD=yourpassword

POSTGRES\_DB=n8n

1. Start services (CPU version):

bash

Copy

docker compose --profile cpu up -d

1. Wait 10-15 minutes for initial model downloads
2. Verify services:

bash

Copy

docker ps -a # Check all containers are running

### **6. Key Considerations**

1. **Storage**: All services use Docker volumes (check with docker volume ls)
2. **Models**: Initial pull of Llama3 models may require 10-20GB disk space
3. **Hardware**: Use --profile gpu-nvidia if you have compatible GPU
4. **Security**: Set N8N\_ENCRYPTION\_KEY and N8N\_USER\_MANAGEMENT\_JWT\_SECRET in production