Title: PostgreSQL + PGVector Setup on a VPS for n8n and RAG Agents

---

Purpose: This guide provides a repeatable setup process for running a self-hosted PostgreSQL environment with both a standard relational database and a vector database (pgvector) on a VPS. It is optimized for use with n8n AI agents, LangChain integrations, and future RAG pipelines.

Step 1: Install PostgreSQL and Essentials on Hetzner VPS

```
sudo apt update
sudo apt install build-essential curl -y
sudo apt install postgresql postgresql-contrib -y
```

Create Databases and User

```
sudo -u postgres psql
```

```
CREATE USER n8nuser WITH PASSWORD 'securepass';
CREATE DATABASE n8n_data OWNER n8nuser;
CREATE DATABASE n8n_vectors OWNER n8nuser;
\q
```

Step 2: Install pgvector Extension

```
sudo apt install postgresql-server-dev-15 make gcc git -y

# Clone and compile
cd ~
git clone https://github.com/pgvector/pgvector.git
cd pgvector
make
sudo make install
```

Enable pgvector in the `n8n_vectors` database:

```
sudo -u postgres psql -d n8n_vectors -c "CREATE EXTENSION vector;"
```

Step 3: Create Table Schemas

🔹 **Standard Table in** `n8n_data`

```
CREATE TABLE users (
  id serial PRIMARY KEY,
  email text UNIQUE NOT NULL,
  created_at timestamp DEFAULT now()
);
```

🔹 **Vector Table in** `n8n_vectors`

```
CREATE TABLE documents (
  id serial PRIMARY KEY,
  content text,
  embedding vector(1536)
);

-- Optional index for fast ANN search:
CREATE INDEX ON documents USING ivfflat (embedding vector_cosine_ops) WITH
(lists = 100);
```

Step 4: Connect PostgreSQL to n8n

1. Go to **n8n Web UI** → Credentials → New **PostgreSQL** credential
2. Fill in:
3. **Host**: `localhost` (or your VPS IP/domain if outside the Docker container)
4. **Port**: `5432`
5. **User**: `n8nuser`
6. **Password**: `securepass`
7. **Database**: `n8n_data` for standard workflows, or `n8n_vectors` for vector embeddings and LangChain RAG support
8. Save the credential.
9. In your n8n workflows:
10. Use the **PostgreSQL node** to run queries on `n8n_data`

11. Use the **LangChain PGVector Vector Store node** to connect to `n8n_vectors` for RAG-based document search, vector insertions, or retrieval

    Note: If you're running n8n inside Docker, ensure Postgres is accessible from within the container — either via `host.docker.internal`, `172.17.0.1`, or by linking containers using Docker Compose networking. Save credentials and use them in:

- **PostgreSQL Node** for standard queries
- **LangChain PGVector Vector Store Node** for RAG / embeddings

---

Step 5: Example Vector Usage

Insert Example Embedding

```
INSERT INTO documents (content, embedding)
VALUES ('Sample text', '[0.12, 0.33, ..., 0.07]');
```

Query Top-K Similar Vectors

```
SELECT id, content
FROM documents
ORDER BY embedding <-> '[0.12, 0.33, ..., 0.07]'
LIMIT 5;
```

Final Step: Configure the Postgres PGVector Store Node in n8n

1. **Drag in a node**: Use the **Postgres PGVector Store** node in your workflow.
2. **Credential**: Select the PostgreSQL credential you saved (connected to the `n8n_vectors` database).
3. **Operation Mode**: Choose `Retrieve Documents (As Tool for AI Agent)` or any mode appropriate for your use case.
4. **Table Name**: Enter `documents`
5. **Embedding Input**: Pass in the 1536-dim embedding (from OpenAI or other source) into the **Embedding** field
6. **Include Metadata**: Enable if you'd like to return content alongside vector similarity
7. **Limit**: Set how many top matches to retrieve (e.g., 4 or 5)
8. **Rerank (optional)**: You may leave this off unless you're adding a reranking model downstream

Click **Execute step** to test the connection and return results.

Component Overview

Component	Role
`n8n_data`	Standard DB (users, logs, misc)
`n8n_vectors`	Vector store (RAG, embeddings)
pgvector	Extension for similarity search
PostgreSQL	Core database engine

PostgreSQL Navigation Cheat Sheet

Task	Command
Exit psql	`\q`
Connect to postgres (root)	`sudo -u postgres psql`
Connect to n8n_data	`psql -U n8nuser -d n8n_data`
Connect to n8n_vectors	`psql -U n8nuser -d n8n_vectors`
List all databases	`\l`
List tables in current DB	`\dt`
Show current DB connection	`\c`
List installed extensions	`\dx`
Switch to postgres user in shell	`sudo su - postgres`

End of Document