

Pydantic RAG

Agentic RAG chatbot built with Pydantic AI, Weaviate vector database, and local Ollama inference. Features hybrid search, conversation memory, and multiple RAG modes.

Features

- **Hybrid Search** - Combines BM25 keyword search with semantic vector search (configurable alpha)
- **3 RAG Modes** - Auto (agent decides), Force (always search), Disabled (plain chat)
- **Conversation Memory** - Multi-turn conversations with full context via `message_history`
- **Token Tracking** - Real-time token usage display with context limit warnings
- **Local Inference** - GPU-accelerated LLM and embedding generation via Ollama

Tech Stack

Component	Technology
Agent Framework	Pydantic AI
Vector Database	Weaviate (with text2vec-ollama)
LLM Inference	Ollama (llama3.2)
Embeddings	nomic-embed-text (768 dimensions)
Web UI	Gradio
Orchestration	Docker Compose

Prerequisites

- Docker with Compose v2
- NVIDIA GPU with CUDA drivers (for GPU inference)
- NVIDIA Container Toolkit ([nvidia-docker](#))

Quick Start

1. Clone and start services:

```
git clone <repo-url>
cd pydantic-rag
docker compose up -d
```

2. Wait for model downloads (first run only):

```
# Watch Ollama logs until models are ready
docker logs -f ollama
```

3. **Access the UI** at <http://localhost:7860>

Document Ingestion

Place documents in [data/documents/](#) and run the ingestion script:

```
# Create documents directory
mkdir -p data/documents

# Add your documents (supports .txt, .md, .pdf, .py, .js, .ts, .c, .cpp,
# .cu, .h)
cp /path/to/your/docs/* data/documents/

# Install dependencies and run ingestion
pip install weaviate-client pypdf
python scripts/ingest.py --name "my project"
```

Ingestion options:

```
python scripts/ingest.py --help
python scripts/ingest.py --reset #
Reset collection only (no ingestion)
python scripts/ingest.py --name "my project" --reset #
Delete and recreate collection, then ingest
python scripts/ingest.py --name "eu ai regulations" --extensions .pdf #
Only PDFs with label
python scripts/ingest.py --name "docs" --documents-dir ./my-docs #
Custom source folder
python scripts/ingest.py --name "code" --extensions .py,.md # Only
Python and Markdown files
python scripts/ingest.py --name "config" --extensions .yaml,Dockerfile #
Extensions and exact filenames
```

Note: The `--name` option is required when ingesting documents. Use `--reset` alone to recreate the schema without ingesting.

Documents are chunked (800 tokens, 200 overlap) and embedded automatically by Weaviate's `text2vec-ollama` module.

Usage

RAG Modes

Mode	Behavior
Auto	Agent decides when to search documents based on the question
Force	Always searches documents before answering
Disabled	Plain chat without document retrieval

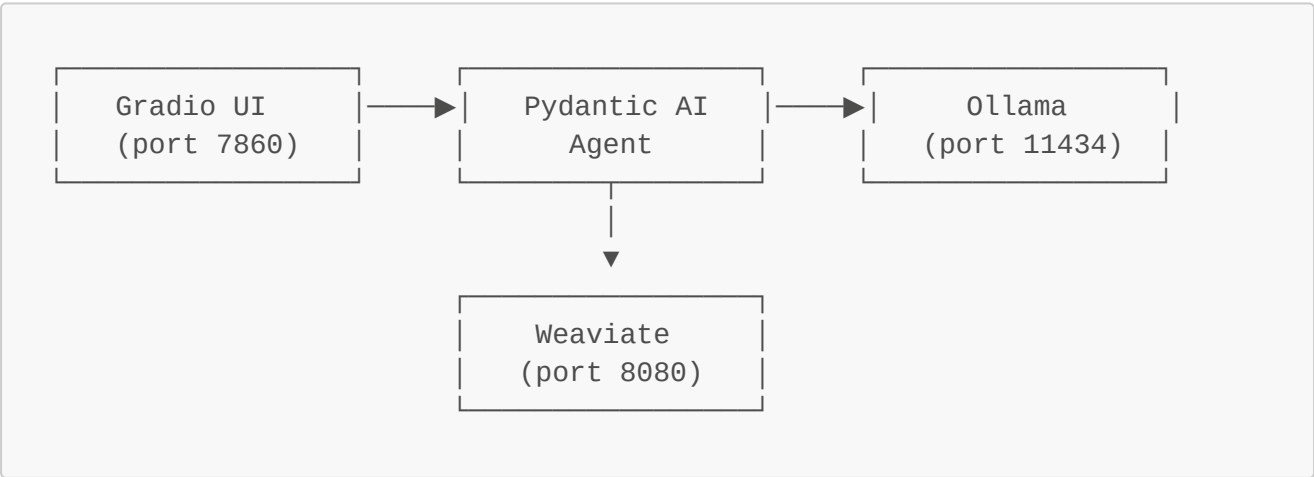
Chat Interface

- 1. Select a RAG mode
- 2. Type your question and press Enter
- 3. View token usage in the top-right display
- 4. Click "Reset Chat" to clear conversation history

Status Checks

Use the "Check Ollama" and "Check Weaviate" buttons to verify connections and see available models/collections.

Architecture



PROF

Configuration

Environment variables (set in `docker-compose.yml`):

Variable	Default	Description
OLLAMA_BASE_URL	<code>http://ollama:11434</code>	Ollama API endpoint
WEAVIATE_URL	<code>http://weaviate:8080</code>	Weaviate endpoint
CHAT_MODEL	<code>llama3.2</code>	LLM for chat
EMBED_MODEL	<code>nomic-embed-text</code>	Model for embeddings

Troubleshooting

Models not loading: Check Ollama logs with `docker logs ollama`. First startup downloads ~2GB of models.

Weaviate connection errors: Ensure Weaviate is healthy with `docker compose ps`. The app will show connection status.

Out of GPU memory: llama3.2 (3B) requires ~4GB VRAM. For larger models, adjust `CHAT_MODEL` or use CPU inference.