Agent orchestrator

Python service providing AI agent chat, MCP tools integration, and RAG on CRM documents, based on FastAPI and LangChain. Called by the Worker's Consumer processes via HTTP/SSE.

System Requirements

`llama-cpp-python` is installed as a **pre-built wheel** (not compiled from source). `requirements.txt` specifies the **Vulkan** variant via `--extra-index-url https://abetlen.github.io/llama-cpp-python/whl/vulkan`. Other backends are available by changing the index URL:

Backend	Index URL suffix
cpu	`.../whl/cpu`
vulkan **(default)**	`.../whl/vulkan`
cuda	`.../whl/cuda`
rocm	`.../whl/rocm`
metal	`.../whl/metal`
sycl	`.../whl/sycl`

Docker image installs `libvulkan1`. GPU access (`/dev/dri`) is commented out in `compose.yaml` by default — uncomment for hardware acceleration. Falls back to CPU without GPU.

The x86-64-v2 baseline or equivalent is required by NumPy's pre-built wheels (see NumPy SIMD build options). CPUs without these instructions can still run the orchestrator by **recompiling NumPy from source** with reduced SIMD flags (`NPY_DISABLE_CPU_FEATURES`), or by using a distro that ships a compatible build.

Architecture Summary

Three layers:

1. **PHP CRM** (Worker Consumer), calls orchestrator via HTTP

2. **Python orchestrator** (FastAPI, Docker)
3. **LLM / MCP servers / Chroma**

Relevant Consumer methods: `llmChat()` (chat relay), `elaborateRag()` (document upload + vector build).

RAG Document Building

Indexing (Elaboration)

Triggered on agent save with the Documents feature enabled. The Worker Consumer:

1. Reads selected CRM Documents
2. `POST /rag/keep` — prunes stale docs from orchestrator
3. `POST /rag/upload` — uploads new/changed files (multipart), stored as `docs/shared/<md5>.<ext>`, symlinked into `docs/<agent_id>/`
4. `POST /rag/build` — indexes all docs for the agent into Chroma at `vectors/<agent_id>/`.

Querying (Runtime)

With `rag: true` in `/agent/run`, Python injects a `query_documents` tool. The LLM decides when to call it. The orchestrator decomposes the question into ≤3 sub-questions (`needs_retrieval` flag), queries Chroma per sub-question, deduplicates by `doc_id`, re-ranks with FlashRank, and returns context. The LLM answer is grounded strictly in retrieved context.

Python Orchestrator Endpoints

The Python service exposes the following REST endpoints. All are mounted on the FastAPI app at port 8120.

Endpoint	Description
`POST /agent/run`	Agent loop: LLM + MCP tools + guardrails + optional RAG. SSE or JSON.
`POST /tools/inspect`	Introspect MCP server tools.
`POST /rag/build`	Index documents for an `agent_id` into Chroma.
`POST /rag/run`	Query vector store with question decomposition.
`POST /rag/keep`	Prune agent's doc symlinks to match `{filename: md5}`.
`POST /rag/upload?agent_id=`	Upload file to shared pool + symlink into agent's dir.

Installation & Configuration

Docker Setup

The Python orchestrator runs in Docker. Quick start:

```
cd plugins/agent
docker compose up -d --build
```

Verify: `curl http://localhost:8120/docs` should show the Swagger UI.

Port `127.0.0.1:8120:8120` — bound to localhost only, **MUST NOT** be publicly exposed.

Environment Variables

Variable	Default
`HOST` (inside the container)	`0.0.0.0`
`PORT`	`8120`
`EMBED_MODEL`	`nomic-ai/nomic-embed-text-v2-moe-GGUF:Q8_0`
`RERANK_MODEL`	`ms-marco-MiniLM-L-12-v2`
`HF_CACHE_DIR`	`/app/hf_cache`
`DOCUMENTS_DIR`	`/app/docs`
`VECTORS_DIR`	`/app/vectors`

Troubleshooting

- **Startup slow:** Embedding model downloads from HuggingFace on first container start — cached in `HF_CACHE_DIR` afterwards.
- **Empty docs:** `/rag/build` raises `RuntimeError` if `docs/<agent_id>/` is empty.
- **GPU not used:** Uncomment `/dev/dri` in `compose.yaml`. Falls back to CPU otherwise.
- **CPU compat:** Verify with `/lib64/ld-linux-x86-64.so.2 --help`

File Reference

```
plugins/agent/
├── compose.yaml
├── app.Dockerfile
```

```
├── requirements.txt
├── src/vte_agent/
│   ├── __main__.py
│   ├── config.py
│   ├── schemas.py
│   ├── agent.py              # /agent/run, /tools/inspect, calculator + rag tools
│   ├── rag.py                # /rag/* endpoints
│   ├── docs.py               # doc loaders
│   ├── models.py             # GGUFEmbeddings
│   ├── user_manual.py        # builtin vtenext user manual search tool
│   └── utils.py
├── docs/
│   ├── shared/        # <md5>.<ext> — deduplicated by content hash
│   └── <agent_id>/    # symlinks → ../shared/<md5>.<ext>
└── vectors/
    └── <agent_id>/    # chroma.sqlite3, parent_docs.json, description.txt

cache_local/
└── huggingface/       # local models cache (embedding, rerank)
```

Revision #2
Created 2026-06-25 16:20:58 UTC by Diego
Updated 2026-06-25 16:37:22 UTC by Diego