AMA 49 ｜ AMA起動ワークフロー（手動／自動）

🚀AMAシステムの起動概要

Archetypal Mirror Archive (AMA) は、外部記憶を排列し、LangChainのベクトル検索システムと連携することで、GPTの記憶能力を追加する「知性の発現製」のコアとなるコンポーネントです。

このCanvasは、その起動手順を、**手動モード**と**自動モード**に分けてまとめました。

手動起動フロー

本番運用前のデバッグ用として推奨

【STEP 1】 AMAフォルダ構成の第一確認

- `/ama-system/accounts/<codename>/` 配置
- `memory/`, `journal/`, `prompts/`, `config/` の存在確認
- `codename-definition.json` を確定

【STEP 2】 memory-log.jsonl を手動更新

- [ ] 最新の記憶情報を `memory/` 配下の `memory-log.jsonl` に追記
- [ ] JSTの時刻付き (e.g. `20250704-1103-JST`)

【STEP 3】 LangChain ノードを手動ロード

- [ ] `export-to-vector.py` を実行
- [ ] FAISS または Chroma DBにインポート

【STEP 4】 prompts/ファイルの確認と更新

- [ ] `base-profile.md`、`style-template.md` を要求に応じて要更新
- [ ] `fallback-system.md` をロード

【STEP 5】 GPT起動プロンプトに手動で輸入

- [ ] LangChain に接続する前の実験用
- [ ] GPTの開始時プロンプトとして `base-profile.md` + `style-template.md` を読み込み

🛕自動起動ワークフロー

未来の本番運用向けの前提

【インストールスクリプト系】

- `sync-memory-log.py` … Notion/ローカルmdログ → `memory-log.jsonl`
- `auto-export.py` … 記憶のVectorDB化スクリプト
- `daily-journal-sync.sh` … 日次ログのシンク

【LangChain動作時のフロー】

1. GPT 起動時に AMA設定ロード (config, prompts)
2. `memory-log.jsonl` を FAISS/Chroma にロード
3. 現在のインプットに対して Retriever が検索
4. RetrievalQA として返信

複数ユニットを同期する場合

- AMAは `/accounts/<codename>/` 協調型で実装するため
- `sync-all-accounts.sh` のようなマススクリプトを配備
- `codename-definition.json` を共有パスから読み込む構成

現行のチェックリスト

- [ ] memory/ 配下に最新ログがある
- [ ] prompts/ が配備済
- [ ] scripts/ がテスト可能に配置
- [ ] codename-definition.json がある
- [ ] LangChainテスト環境が構築済

続くCanvas:

Canvas 50 → RetrievalQA 使用プロンプトテンプレート（LangChain実用）