OPEA を使用した階層型のマルチエージェント RAG の構築

エクセルソフト株式会社

2025年9月12日

はじめに

- 本資料は 2025/09/10 時点の情報をもとに作成しています

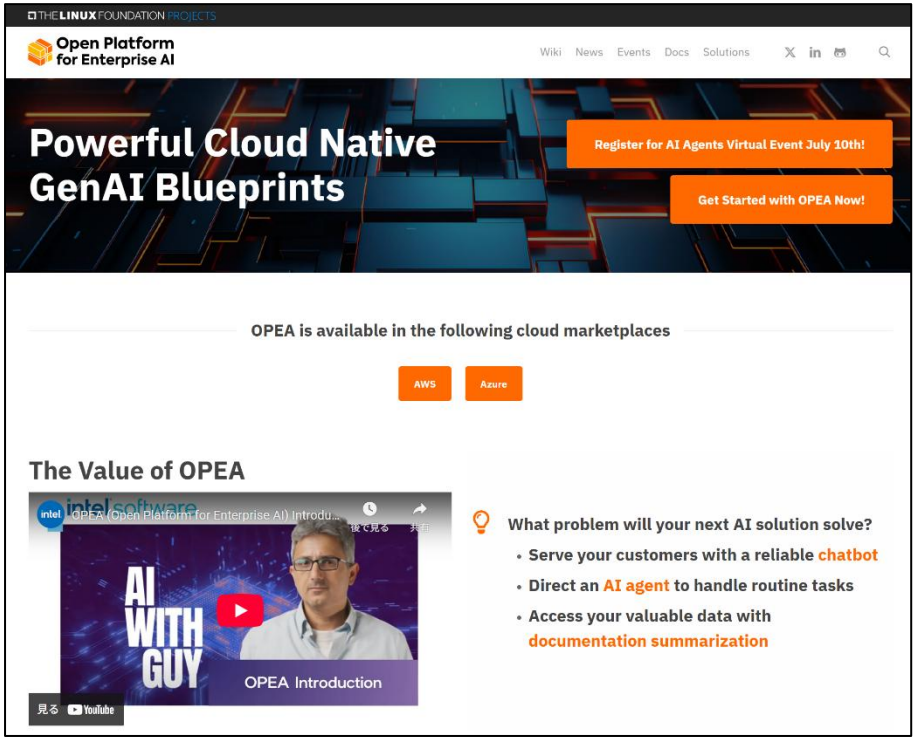
ご紹介したいこと

- OPEA を利用した生成 AI 利用基盤の構築
  - ✓ Open Platform for Enterprise AI (OPEA) とは
  - ✓ OPEA アーキテクチャーについて
  - ✓ OPEA v1.4 アップデート
    AI Agent を採用したチャットボットの導入

2025/08/25 に v1.4 がリリースされました

Open Platform for Enterprise AI

- Open Platform for Enterprise AI
  - ✓ OPEA (オペア) と呼ばれています
  - ✓ https://opea.dev (英語)
- LF AI & Data 傘下のプロジェクト
  - ✓ 2024年4月16日発表



OPEA が課題とする
企業における生成 AI 導入のハードル

- 生成 AI アプリケーションの開発/導入には、多くの課題に直面します
  - ✓ 例えば、新しいモデルやアルゴリズムの開発、ファインチューニングの技術、LLM が抱えるバイアスの検出とその解消、大規模なソリューションとしての展開方法
- 最大の課題として、開発に標準化されたソフトウェアやツールの選択肢が不足していることを挙げています
  - ✓ 迅速な開発や機能拡張を可能にしつつ安全性と信頼性を確保したい
  - ✓ 一方で独自のソリューションとオープンなソリューションの両方を包含できるフレームワークが存在しないため、開発の方向性を定めることが困難

OPEA の取り組み

- OPEA は企業や組織向けに生成 AI アプリケーションに求められる
  要件を実装するためのアーキテクチャーとフレームワークを整備/実装しています
  - ✓ フレームワークは OPEA のアーキテクチャーをベースに
    既存の技術を使用しつつ最新の技術を導入
  - ✓ また要件にあわせたカスタマイズができる拡張性
- あわせてフレームワークをもとに幅広いユースケースに対応した
  サンプル･アプリケーションを提供します
  - ✓ フレームワークが求められる要件を満たしているか
    確認するための検証ツールも提供

2025年9月時点

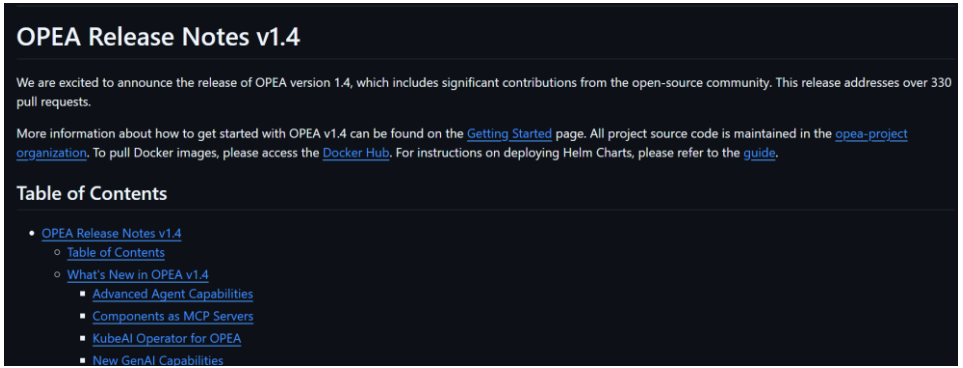
OPEA パートナー

- OPEA は、LF AI & Data のプロジェクト傘下となることで複数の
  プロバイダーによるエコシステムにより支援されたフレームワークを提供します
  - ✓ 企業や組織での運用にあたって信頼できるソリューションの
    評価・選定・カスタマイズ導入できる環境

出典: Open Platform For Enterprise AI (英語)

OPEA v1.4

- OPEA Release Notes v1.4 (英語)
- AI エージェント機能の拡張
  - ✓ ディープリサーチ機能の追加
  - ✓ MCP のサポート追加および OPEA コンポーネントを MCP サーバー対応
- 言語検出やプロンプト･テンプレートなどの機能を追加



OPEA Release Notes v1.4

We are excited to announce the release of OPEA version 1.4, which includes significant contributions from the open-source community. This release addresses over 330 pull requests.

More information about how to get started with OPEA v1.4 can be found on the Getting Started page. All project source code is maintained in the opea-project organization. To pull Docker images, please access the Docker Hub. For instructions on deploying Helm Charts, please refer to the guide.

Table of Contents

- OPEA Release Notes v1.4
  - Table of Contents
  - What's New in OPEA v1.4
    - Advanced Agent Capabilities
    - Components as MCP Servers
    - KubeAI Operator for OPEA
    - New GenAI Capabilities

OPEA アーキテクチャー

- OPEA が提供するフレームワークは OPEA のアーキテクチャーに基づき マイクロサービス主体の生成 AI アプリケーションを実装します
- 3 つの主要な構造を定義しています
  - ✓ Microservices
    生成 AI アプリケーションを構成するために基本的な要素を提供する小規模なサービス
  - ✓ Megaservices
    1 つ以上のマイクロサービスで構築された包括的なソリューション
  - ✓ Gateways
    ユーザーがメガサービスにアクセスするためのインターフェイス
    ユーザー要件によってカスタマイズされます

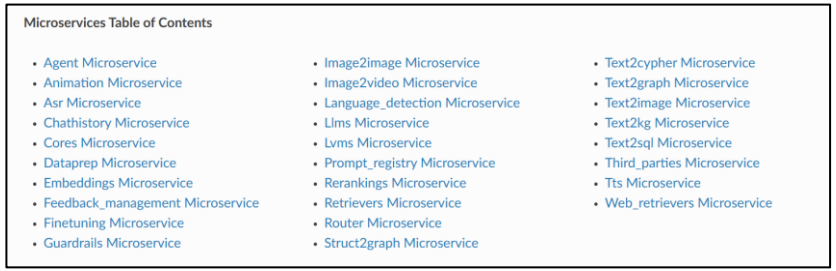
マイクロサービス

- マイクロサービスは特定の機能やタスクを実行します
  - ✓ モジュラー式の構造を採用しておりメンテナンス性の高さを確保しつつ本番環境へのデプロイやスケーリングを簡易にします
- マイクロサービスは独立しているため個々のコンポーネントを開発してデプロイできます
  - ✓ 独自の機能、サービスを開発してマイクロサービスと接続できます
  - ✓ すべてのマイクロサービスはコンテナ化されておりクラウドへの展開が可能

マイクロサービス

- ✓ Embeddings Microservice
  文字列をベクトルデータに変換
- ✓ Retrievers Microservice
  ベクトル･データベース内の検索
- ✓ LLMs Microservice
  LLM モデルを利用した推論
- ✓ Rerankings Microservice
  入力クエリーに対して関連性に基づいた優先付け

- ✓ Guardrails Microservice
  推論による出力に対して有害検出やバイアス検出を実施
- ✓ Dataprep Microservice
  画像、テキストなどのデータをベクトルデータへ変換してデータベースに保存



出典: GenAI Microservices (英語)

例: Dataprep マイクロサービス

- Dataprep マイクロサービスは
  さまざまなデータをベクトルデータへ変換して
  データベースに保存する機能を提供します
  1. 構造/非構造データの前処理
  2. テキストデータを埋め込みベクトルに変換
  3. データベースに保存
- 特定のデータ構造向けの設計や
  多様なデータベースへのサポートを含みます

Databases	Readme
Redis	Dataprep Microservice with Redis
Milvus	Dataprep Microservice with Milvus
Qdrant	Dataprep Microservice with Qdrant
Pinecone	Dataprep Microservice with Pinecone
PGVector	Dataprep Microservice with PGVector
VDMS	Dataprep Microservice with VDMS
Multimodal	Dataprep Microservice with Multimodal
ElasticSearch	Dataprep Microservice with ElasticSearch
OpenSearch	Dataprep Microservice with OpenSearch
neo4j	Dataprep Microservice with neo4j
financial domain data	Dataprep Microservice for financial domain data
MariaDB	Dataprep Microservice with MariaDB Vector
ArangoDB	Dataprep Microservice with ArangoDB Vector

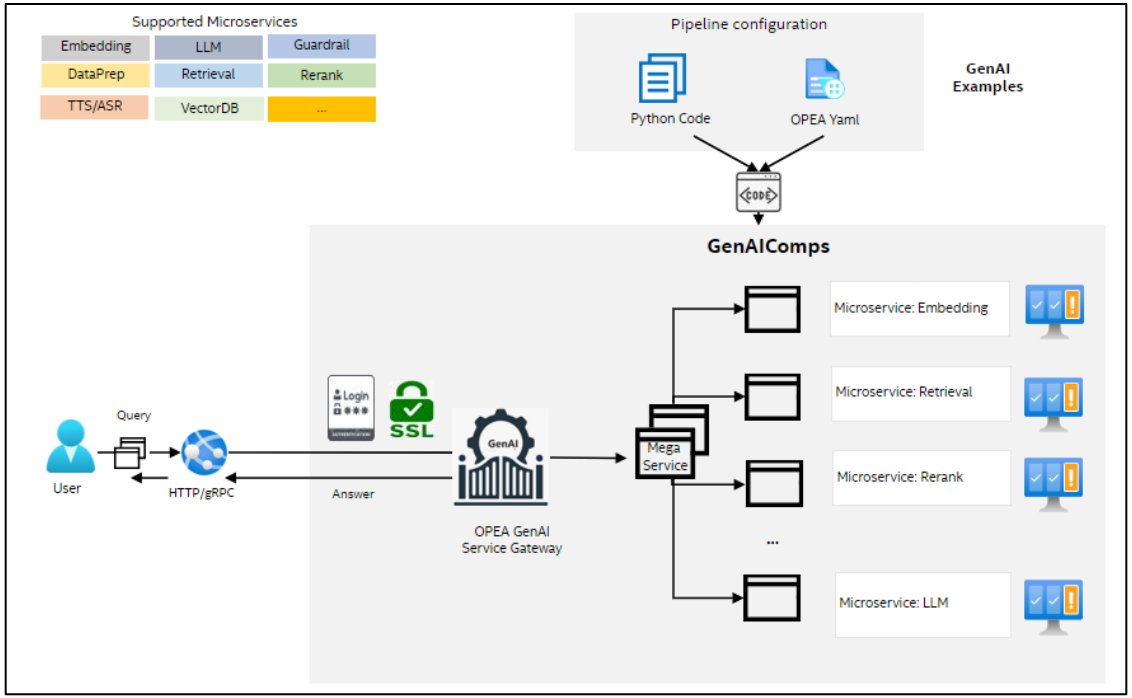
出典: Dataprep Microservice (英語)

例: Dataprep マイクロサービス

- ベクトル･データベースへの登録/参照/削除を操作するためには
  マイクロサービスに対してリクエストを送信します
  - ✓ 例: pdf ファイルの取り込み
    Dataprep マイクロサービスのエンドポイントを指定

```
curl -X POST ¥
     -H "Content-Type: multipart/form-data" ¥
     -F "files=@./your_file.pdf" ¥
     -F "process_table=true" ¥
     -F "table_strategy=hq" ¥
     http://localhost:6007/v1/dataprep/ingest
```

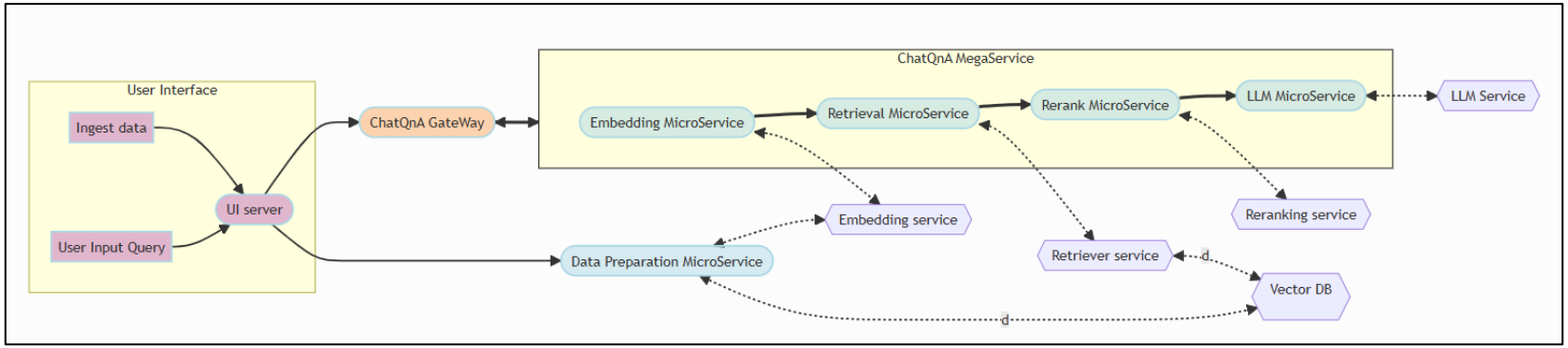
OPEA のアーキテクチャーを利用した生成 AI アプリケーション



出典: [opea-project/GenAIComps](#) (英語)

例: RAG チャットボットのワークフロー
ChatQnA サンプル

- Embedding、Retrieval、Rerank、LLM マイクロサービスを使用して ChatQnA メガサービスを構築
  - ✓ ユーザーは UI を通してメガサービスにアクセス
- RAG が参照するデータは Detaprep マイクロサービスによりベクトル DB に登録



出典: ChatQnA Application (英語)

OPEA Github

- OPEA [Open Platform for Enterprise AI] (英語)
- 主なリポジトリー
  - ✓ GenAIComps
    各種マイクロサービスの実装
    Docker Hub へ公開されています → OPEA | Docker Hub (英語)
    Helm Chart → Packages | OPEA (英語)
  - ✓ GenAIExamples
    GenAIComps を利用したソリューション・レベルのサンプル・アプリケーション

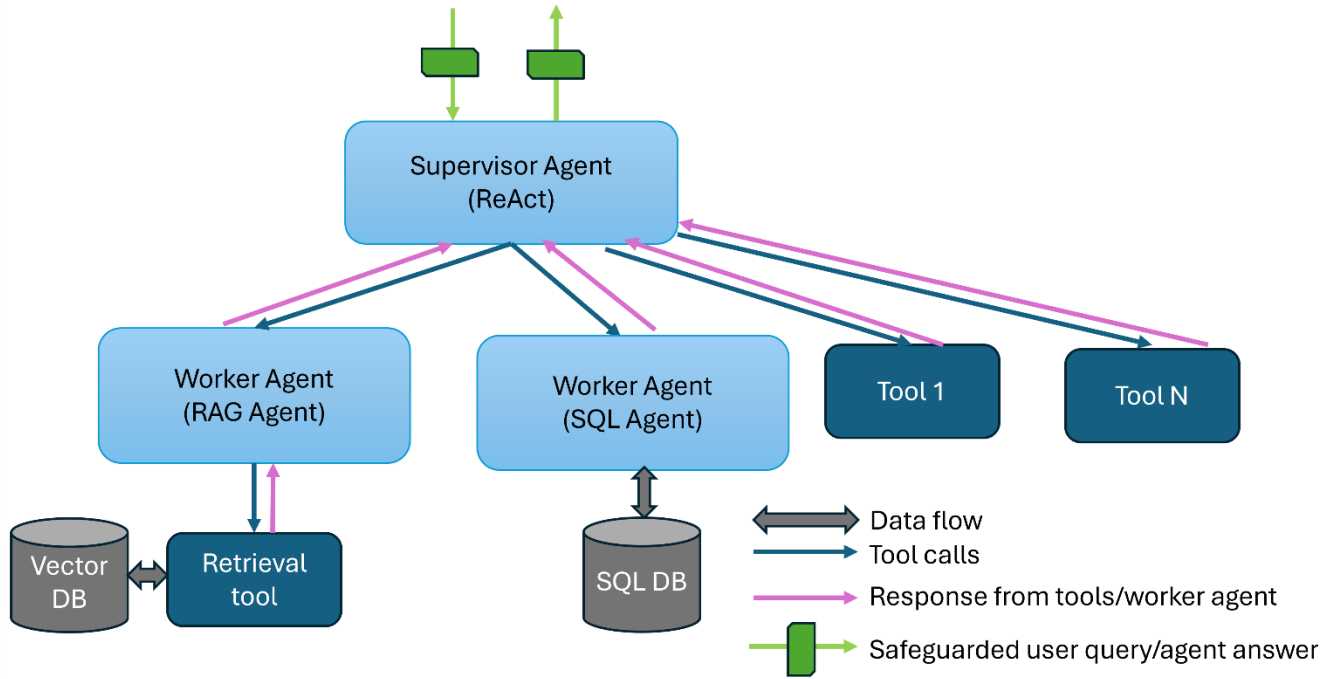
GenAIExamples リポジトリー

- OPEA のアーキテクチャーを採用した生成 AI アプリケーションのサンプル
  - ✓ サンプルは主要なハードウェア・メーカーが提供する CPU および GPU、アクセラレーター向けに動作します
- チャットボット、AI アバター、音声応答など 11 種類以上を公開
- サンプルのデプロイは Docker と Kubernetes を選択できます
- ここでは AgentQnA アプリケーションに注目します

GenAIExamples AgentQnA

- Github: AgentQnA (英語)
- 質問回答のチャットボットにマルチエージェントを採用したサンプル
- Agent/AI Agent は単一のタスクではなく、より複雑なタスクを行うために
  自律的にツールを使用したり、他のエージェントとコミュニケーションをとる AI
- AgentQnA ではいくつかの専門的な能力を持つエージェントと
  これらのエージェントとやり取りする管理エージェントを構築します
  - ✓ Worker Agent :
    Superviusor Agent から指示を受け回答する専門的な能力を持つエージェント
  - ✓ Supervisor Agent :
    Worker Agent やツールを使い情報を収集して高度な回答を生成するエージェント

AgentQnA



出典: Agents for Question and Answering Application (英語)

Agent マイクロサービス

- Agent マイクロサービスは Langchain/Langgraph フレームワークで実装されており、いくつかの種類のエージェントを提供します
  - ✓ ReAct エージェント

    例: react_llama yaml (英語)
  - ✓ RAG エージェント
  - ✓ SQL エージェント

```yaml
services:
  agent:
    image: ${agent_image}
    container_name: test-comps-agent-endpoint
    volumes:
      - ${TOOLSET_PATH}:/home/user/tools/
    ports:
      - "9095:9095"
    ipc: host
    environment:
      ip_address: ${ip_address}
      strategy: react_llama
      with_memory: true
      memory_type: checkpointer
      recursion_limit: ${recursion_limit}
      llm_engine: vllm
      HF_TOKEN: ${HF_TOKEN}
      llm_endpoint_url: ${LLM_ENDPOINT_URL}
      model: ${LLM_MODEL_ID}
      temperature: ${temperature}
      max_new_tokens: ${max_new_tokens}
      top_k: 10
      stream: true
      tools: /home/user/tools/custom_tools.yaml
      require_human_feedback: false
      no_proxy: ${no_proxy}
      http_proxy: ${http_proxy}
      https_proxy: ${https_proxy}
      port: 9095
```

AgentQnA の導入と実行

- AgentQnA を構成するマイクロサービスを
  Docker Compose を使ってデプロイします
  - ✓ agent、LLM、dataprep、embedding、retriever、reranking マイクロサービスを利用
  - ✓ RAG エージェントのため doc-index-retriever サンプルを併用します
    Github: GenAIExamples/DocIndexRetriever (英語)
- 個々のサービスの実装
  - ✓ embedding、rerankging マイクロサービス
    Text Embeddings Inference (TEI)
  - ✓ LLM マイクロサービス
    OpenAI API エンドポイント
- サービスの docker イメージは Docker Hub から取得

AgentQnA の導入と実行

- デプロイオプション
  - ✓ Docker Compose
  - ✓ Kubernetes (Helm Chart)
    現時点はインテル® Gaudi® AI アクセラレーター向けのみ
- インテル® Xeon® プロセッサー向けのデプロイオプションを実行します
  - ✓ 一部のハードウェア向け実装は作業中です
  - ✓ 現時点のデプロイオプション
    インテル® Gaudi® AI アクセラレーター
    AMD GPU サーバー

Category	Deployment Option	Description
On-premise Deployments	Docker compose	AgentQnA deployment on Xeon
		AgentQnA deployment on Gaudi
		AgentQnA deployment on AMD ROCm
	Kubernetes	Helm Charts
	Azure	Work-in-progress
	Intel Tiber AI Cloud	Work-in-progress

AgentQnA の導入と実行

- 作業ディレクトリーの指定

```
export WORKDIR=<your-work-directory>
cd $WORKDIR
git clone https://github.com/opea-project/GenAIExamples.git
cd GenAIExamples/AgentQnA
```

- 環境固有の環境変数を設定

```
export host_ip="External_Public_IP"           # ip address of the node
export HF_TOKEN="Your_HuggingFace_API_Token"  # the huggingface API token you applied
export http_proxy="Your_HTTP_Proxy"           # http proxy if any
export https_proxy="Your_HTTPs_Proxy"         # https proxy if any
export no_proxy=localhost,127.0.0.1,$host_ip  # additional no proxies if needed
```

AgentQnA の導入と実行

- OpenAI モデルを利用するため API キーを追加で指定します

```
export OPENAI_API_KEY=<your-openai-key>
```

- 実装先のディレクトリー移動と環境変数の設定

```
cd $WORKDIR/GenAIExamples/AgentQnA/docker_compose/intel/cpu/xeon
source ./set_env.sh
```

- docker コンテナの起動

```
docker compose ¥
  -f $WORKDIR/GenAIExamples/DocIndexRetriever/docker_compose/intel/cpu/xeon/compose.yaml ¥
  -f compose_openai.yaml ¥
  up -d
```

AgentQnA の導入と実行

- オプション: OpenAI API 以外の API エンドポイントを
  使用するには追加で環境変数を設定します

```
export REMOTE_ENDPOINT=<https-endpoint-of-remote-server>
export model=<model-card>
```

- オプション: docker コンテナの起動
  - ✓ compose_remote.yaml を追加します

```
docker compose ¥
  -f $WORKDIR/GenAIExamples/DocIndexRetriever/docker_compose/intel/cpu/xeon/compose.yaml ¥
  -f compose_openai.yaml ¥
  -f compose_remote.yaml ¥
  up -d
```

AgentQnA の導入と実行

- RAG Agent が参照するベクトル・データベースに
  サンプルのデータファイル test_docs_music.jsonl を入力します

```
cd  $WORKDIR/GenAIExamples/AgentQnA/retrieval_tool/
bash run_ingest_data.sh
```

- run_ingest_data.sh は index_data.py を実行します

```
FILEDIR=${WORKDIR}/GenAIExamples/AgentQnA/example_data/
FILENAME=test_docs_music.jsonl

…

python3 index_data.py --filedir ${FILEDIR} ¥
        --filename ${FILENAME} --host_ip $host_ip --port $port
```

AgentQnA の導入と実行

index_data.py

dataprep マイクロサービス・エンドポイント

test_docs_music.jsonl の取り込み

```
host_ip = args.host_ip
port = args.port
proxies = {"http": ""}
url = "http://{host_ip}:{port}/v1/dataprep/ingest".format(host_ip=host_ip, port=port)

…

files = split_jsonl_into_txts(os.path.join(args.filedir, args.filename))
file_list = write_docs_to_disk(files, args.filedir)

…

for file in tqdm.tqdm(file_list):
    print("Indexing file: ", file)
    files = [("files", (f, open(f, "rb"))) for f in [file]]
    payload = {"chunk_size": args.chunk_size, "chunk_overlap": args.chunk_overlap}
    resp = requests.request("POST", url=url, headers={}, files=files, data=payload,
proxies=proxies)
    print(resp.text)
```

AgentQnA の導入と実行

- test_docs_music.jsonl 内


```
{"query": "who sang the hit song \"thriller\"?", "domain": "music", "doc": "Thriller (song) - Wikipedia\nJump to content\nMain menu\r
{"query": "who sang the hit song \"thriller\"?", "domain": "music", "doc": "Recorded1982StudioWestlake (Los Angeles, California)Genre
{"query": "who sang the hit song \"thriller\"?", "domain": "music", "doc": "Jackson decided to release \"Thriller\" as a single after
{"query": "who sang the hit song \"thriller\"?", "domain": "music", "doc": "Problems playing this file? See media help.\n\"Thriller\"
{"query": "who sang the hit song \"thriller\"?", "domain": "music", "doc": "Recording[edit]\nQuincy Jones produced \"Thriller\".\nAlc
{"query": "who sang the hit song \"thriller\"?", "domain": "music", "doc": "Release[edit]\nThe album Thriller was released in Novembe
{"query": "who sang the hit song \"thriller\"?", "domain": "music", "doc": "It doubled sales of Thriller, and the documentary sold ov
{"query": "who sang the hit song \"thriller\"?", "domain": "music", "doc": "Following Jackson's death in 2009, his music surged in po
{"query": "who sang the hit song \"thriller\"?", "domain": "music", "doc": "\"Thriller\"\nhas returned to the Billboard Hot 100 chart
{"query": "who sang the hit song \"thriller\"?", "domain": "music", "doc": "Personnel[edit]\nWritten and composed by Rod Temperton\nP
{"query": "who sang the hit song \"thriller\"?", "domain": "music", "doc": "3\nChart (2007)\nPeakposition\nSpain (PROMUSICAE)[70]\n20
{"query": "who sang the hit song \"thriller\"?", "domain": "music", "doc": "34\nChart (2018)\nPeakposition\nCanada (Canadian Hot 100)
{"query": "who sang the hit song \"thriller\"?", "domain": "music", "doc": "Diamond\n10,000,000\u2021\nUnited States (RIAA)[121] Mast
{"query": "who sang the hit song \"thriller\"?", "domain": "music", "doc": "^ Glazer, Eliot (September 25, 2009). \"Top 1984 Songs\".
```

AgentQnA の導入と実行

- SQL Agent は AgentQnA サンプルに含まれる
  サンプルデータを参照します

compose_openai.yaml

```
  worker-sql-agent:
    image: opea/agent:latest
    container_name: sql-agent-endpoint
    volumes:
      - ${WORKDIR}/GenAIExamples/AgentQnA/tests:/home/user/chinook-db # SQL database
    ports:
      - "9096:9096"
    ipc: host
    environment:
      ip_address: ${ip_address}
      strategy: sql_agent
```

AgentQnA の導入と実行

- AgentQnA の WebUI (agent-ui) にアクセスします
  - ✓ http://<Host IP>:5173
    既定は 5173 ポートへアクセスします

compose_openai.yaml

```yaml
agent-ui:
  image: ghcr.io/open-webui/open-webui:main
  container_name: agent-ui
  ports:
    - "5173:8080"
  ipc: host
  environment:
    no_proxy: ${no_proxy}
    http_proxy: ${http_proxy}
    https_proxy: ${https_proxy}
    OPENAI_API_KEYS: "empty"
    OPENAI_API_BASE_URLS: ${SUPERVISOR_AGENT_ENDPOINT}
    ENABLE_OLLAMA_API: False
```

AgentQnA の導入と実行

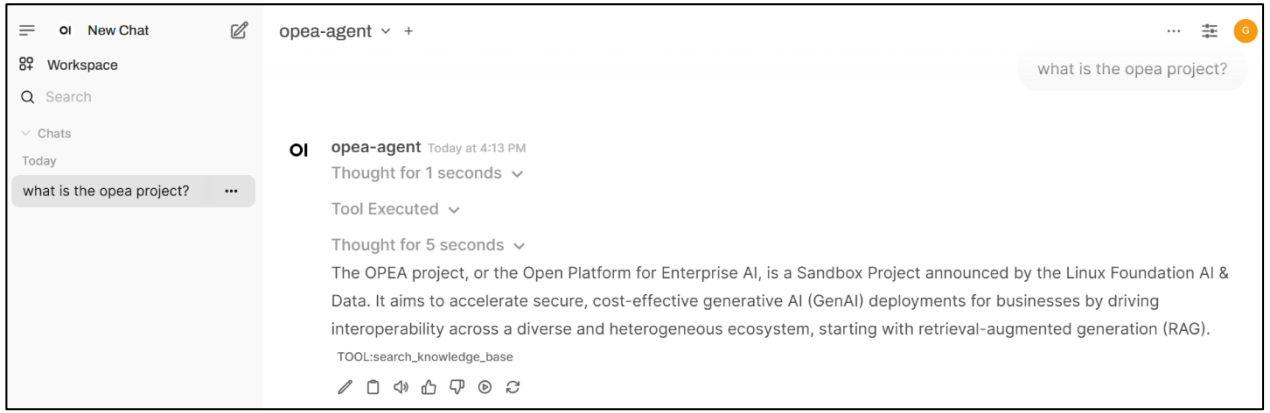
- 接続設定から AgentQnA の Supervisor Agent の接続先を指定
  - ✓ http://<Host IP>:9090/v1

compose_openai.yaml

```yaml
supervisor-react-agent:
  image: opea/agent:latest
  container_name: react-agent-endpoint
  depends_on:
    - worker-rag-agent
    - worker-sql-agent
  volumes:
    - ${TOOLSET_PATH:-../../../../tools}:/home/user/tools/
  ports:
    - "9090:9090"
  ipc: host
```

AgentQnA の導入と実行

- エージェントに質問します



まとめ

- OPEA は複雑化しやすい生成 AI アプリケーションの
  実装の標準化と整備を行うプロジェクトです
  - ✓ 2025/08/25 に v1.4 が公開されました
- OPEA はマイクロサービスを主体としたアーキテクチャーを
  採用することにより環境に依存しないアプリケーションを構築します
  - ✓ これらは任意にカスタマイズ可能です
- OPEA はマルチエージェントを利用した生成 AI アプリケーションなどの
  新しい技術やツールの導入、基盤の整備を積極的に行っています

お問い合わせはこちらまで

https://www.xlsoft.com/jp/qa