AI + DOCUMENTS

Docu-Genius: AI-Powered Document System

Demo Overview

An end-to-end, scalable, AI-driven document insight platform

Presenter: [Your Name] | GitHub: lihuiniu/docu-genius

Demo Agenda

1. What is Docu-Genius?
2. Key Features
3. Architecture Overview
4. Model Choices: OpenAI + Seq2Seq + LangChain
5. Workflow & Controller: LangGraph
6. Technical Details & API Usage
7. Demo Highlights
8. Q&A

What is Docu-Genius?

AI-powered system for document chunking, semantic search, summarization, and evaluation

FastAPI-powered async backend

High-performance, scalable server with asynchronous processing

LLM-powered summarization & evaluation

Advanced AI for content generation and quality control

Semantic retrieval & caching

Milvus vector search with Redis caching for fast access

Flexible storage & interfaces

Local & cloud storage support with CLI & API access

Key Features

Flexible Storage
Local, S3, Azure, Delta Lake

Semantic Search
Milvus vector database

Fast Cache
Redis 8.0 for quick retrieval

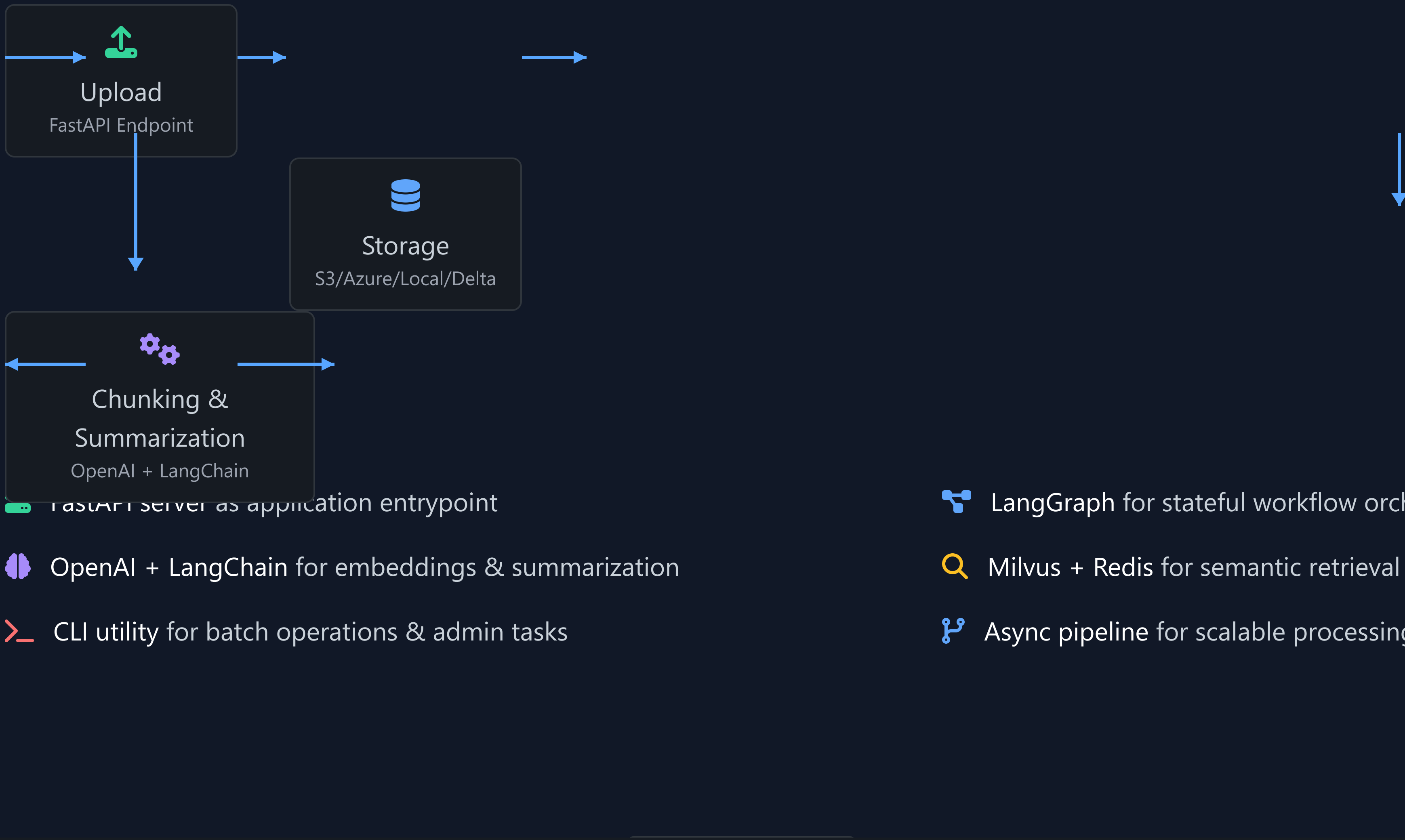
AI Evaluation
OpenAI LLMs (v1.x), LangChain LLMChain

Workflow Orchestration
LangGraph StateGraph workflows

Async & Scalable
FastAPI + Hypercorn server

CLI Utility
Batch/parallel reindexing with resume support

System Architecture



- FastAPI server as application entrypoint
- OpenAI + LangChain for embeddings & summarization
- CLI utility for batch operations & admin tasks
- LangGraph for stateful workflow orchestration
- Milvus + Redis for semantic retrieval & caching
- Async pipeline for scalable processing

Model Choices & Pipeline

Integrated AI components for document understanding and evaluation

OpenAI

- Embeddings for semantic search
- Document summarization
- OpenAI API v1.x integration

Seq2Seq

- Text transformation approach
- Via OpenAI's API
- Abstracted through LangChain

LangChain

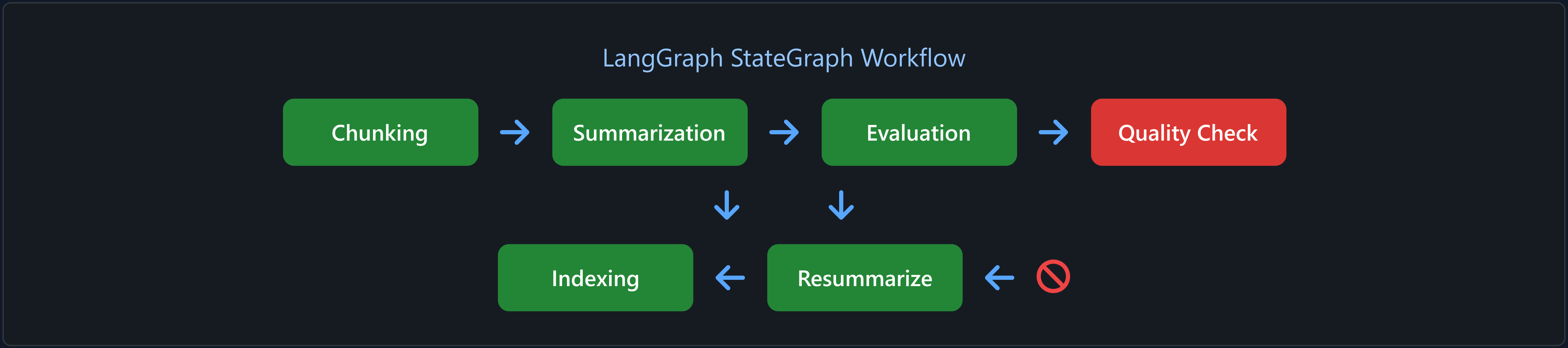
- Workflow integration
- LLMChain for evaluation
- Modular component chaining

Quality Control Pipeline

Document → Summarization → LLM Evaluation → Quality Check → Auto Resummarize (if low quality)

Workflow Orchestration: LangGraph

StateGraph-based workflow for complex document processing pipelines



Modular & Conditional Flows

Dynamic routing based on quality evaluations

Unified Orchestration

Same workflow powers CLI batch jobs & API requests

Async Execution

Non-blocking processing for high throughput

Extensible Architecture

Easy to add new states or decision logic

Technical Implementation & Demo

Setup

- Python virtual environment + requirements.txt
- FastAPI + Hypercorn for async API
- Milvus 2.6 for vector storage
- Redis 8.0 for caching
- OpenAI API key for LLM functionality

Testing

- Unit & integration tests with pytest
- Async testing with pytest-asyncio
- Continuous integration via GitHub Actions
- Test coverage reporting

API Usage Examples

Upload document:

```
curl -F "file=@./example.txt" http://localhost:8000/upload/local
```

Summarize document:

```
curl -X POST http://localhost:8000/summarize \
  -H "Content-Type: application/json" \
  -d '{"doc_id": "your_doc_id", "storage": "local"}'
```

Query similar chunks:

```
curl -X POST http://localhost:8000/query \
  -H "Content-Type: application/json" \
  -d '{"keyword": "example", "top_k": 5}'
```

Batch reindex via CLI:

```
python cli/reindex.py --doc-ids doc1 doc2 --storage local --concurrency
```

Perfect for real-time document intelligence in modern applications