RAG with In-Memory Java Microservices

Phil Chung
Director of Product Management
Enterprise Cloud Native Java
May 13, 2025

Safe harbor statement

$ whoami _

#Director of Product Management – Enterprise Cloud Native Java

#Ex-Presales, Solution/Cloud architect at TimesTen IMDB & Oracle

#Help key customers implement/migrate large scale Oracle Middleware deployment & OCI

#Member of the Eclipse MicroProfile AI/LLM working group

#Oracle Cloud, Java, Open Source & CNCF evangelist

#https://www.linkedin.com/in/phchung/

#https://x.com/philchungny

Agenda

- Introduction
- Anatomy of RAG
- Cloud Native RAG with Helidon and Coherence

The Economic Potential of Generative AI

$2.6-4.4T

GDP Financial Impact of Generative AI

Source: Aggregated analyst findings from Accenture, E&Y, Goldman Sachs, McKinsey
https://futuretech.mit.edu/news/putting-the-economic-impact-of-genai-into-scale

What is the scale of your ambition? – Stephanie Zhan, Sequoia VC

- NVIDIA says robotics could become the world's biggest **multi-trillion dollar industry**
  - Open sourced world's first humanoid robot brain LLM - GR00T N1
  - CUDA DTX – CUDA Distributed Execution runtime announced, planned for 2027
- Edge, locally hosted AI positioned to benefit most from explosion of open source LLMs
  - Tariffs incentivize vendor/country neutral open source at Edge and locally hosted
- **Java and AI** – modular, portable, powerful together
  - Scale from single, small robotics, drones to complex edge interaction, AI data centers
  - Leverage Java for Inference, RAG, Agents – post training + Java cloud native apps/services
  - AI workloads that create economically valuable work

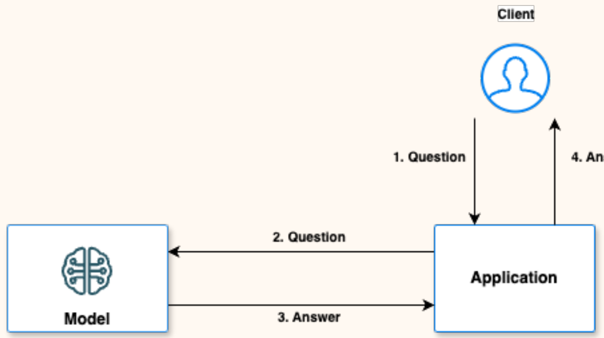
Enhancing existing systems with AI

- Anomaly and fraud detection
  - Cell Tower equipment monitoring
  - Financial transactions monitoring, insurance fraud
  - Hospitals, diagnostics testing in radiology, pathology, cardiology and dermatology
- Language translation and NLP
  - Customer support at retail, healthcare, restaurants
  - Interactive Voice Response
- Similarity Matching in Recommendation system – audio/video streaming, retail, restaurant
- Secure personalized content
  - Medicine, ads, sales reports, response generation
- Image recognition
  - Identify objects, classify images, and track motion
  - In-vehicle autonomous systems – aircraft, car, boat, rocket

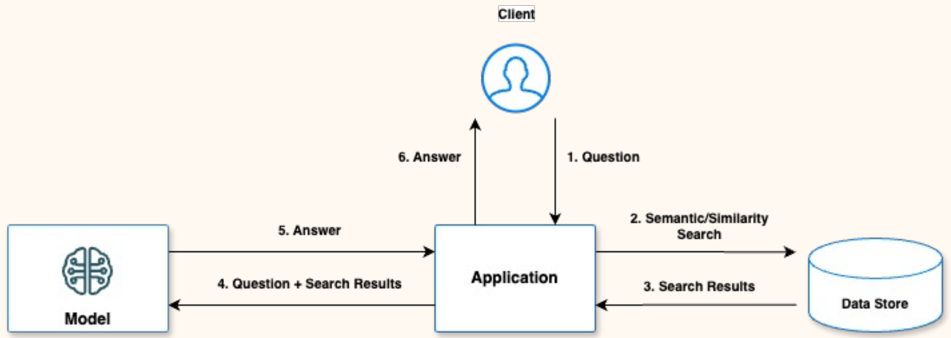
Generative AI challenges

- Improve accuracy
- Reduce hallucinations
- Keep up to date
- Security and Privacy
- Costs

Enter RAG



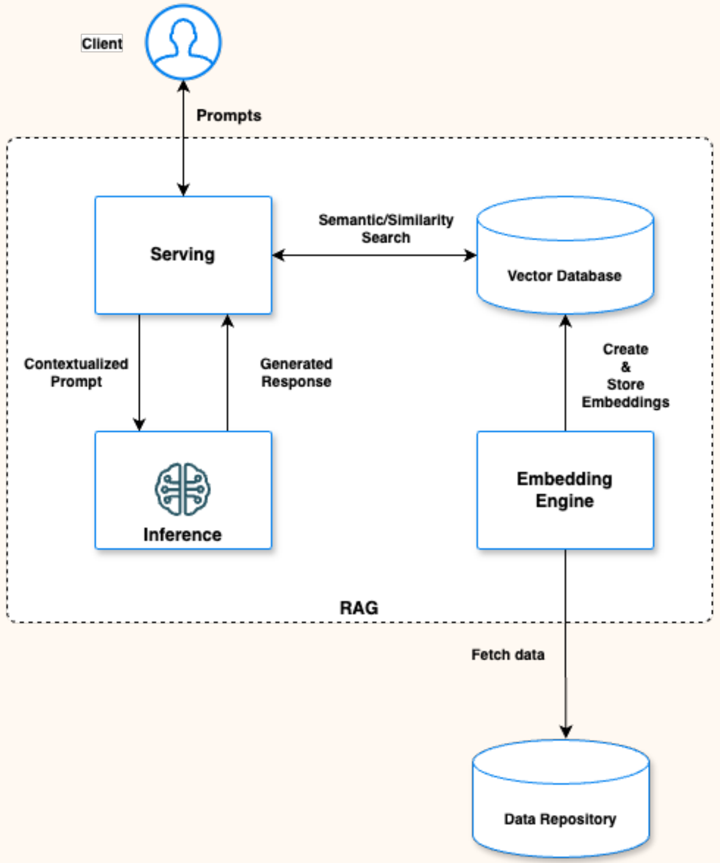
Without RAG



With RAG

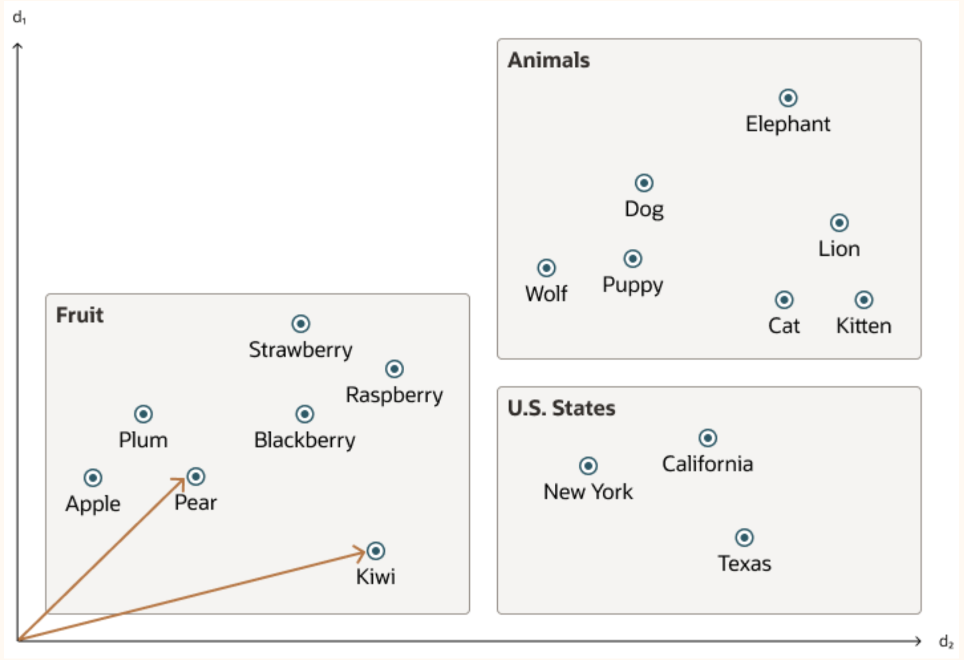
RAG: A technique for improving the output of pre-trained Large Language Models in Generative AI, by augmenting questions with retrieved contextual information for the model generating answers.

Anatomy of RAG

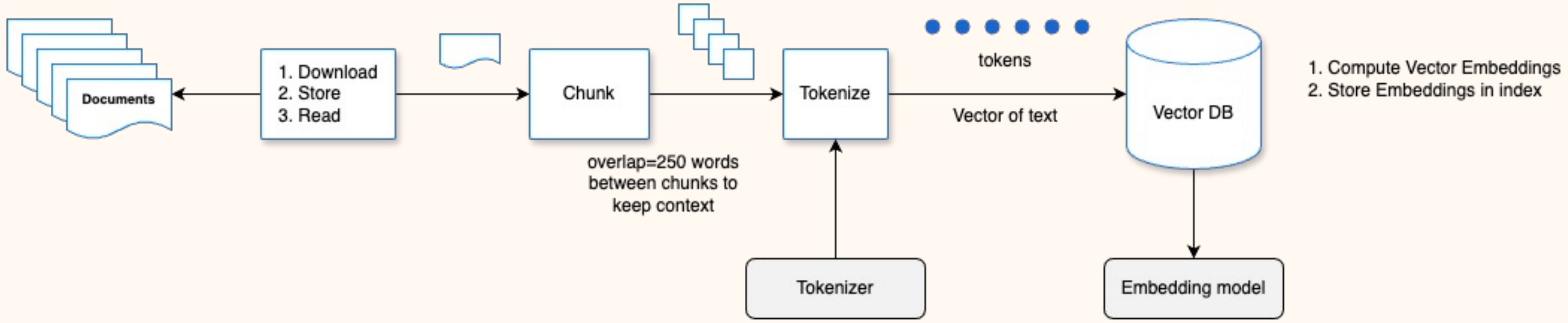


What is a Vector Database?

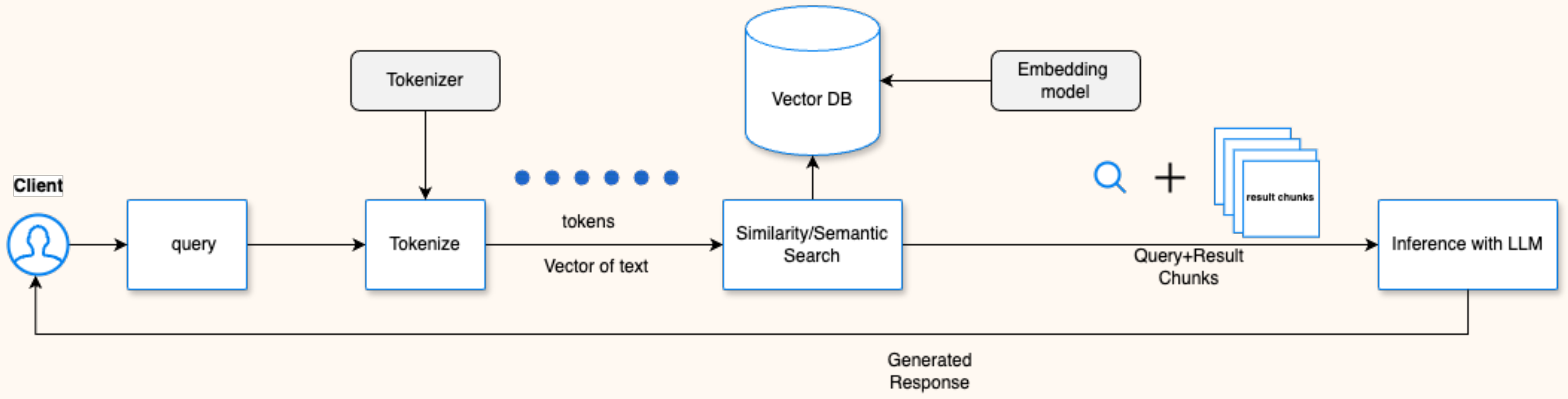
- A database that stores Vector embeddings
- A Vector embedding:
  - Is a vector of floating point numbers
  - Has magnitude and direction
- Embeddings are used for:
  - Search
  - Clustering
  - Recommendations
  - Anomaly detection
  - …
- Use distance metrics (e.g. cosine distance) to search for similar items



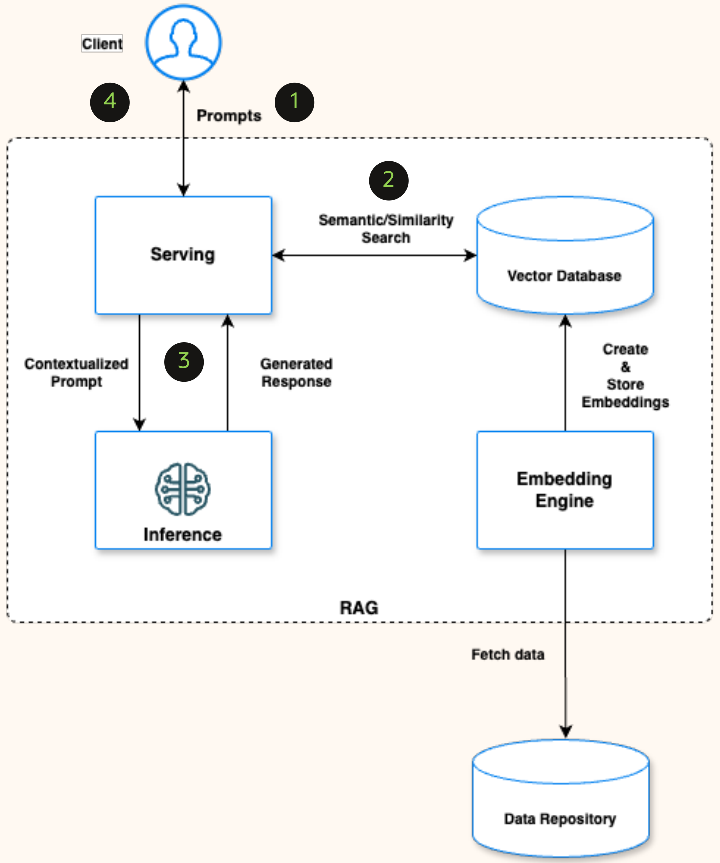
Embedding Process


Documents
1. Download
2. Store
3. Read
Chunk
overlap=250 words between chunks to keep context
Tokenize
Tokenizer
tokens
Vector of text
Vector DB
Embedding model
1. Compute Vector Embeddings
2. Store Embeddings in index

Query Process



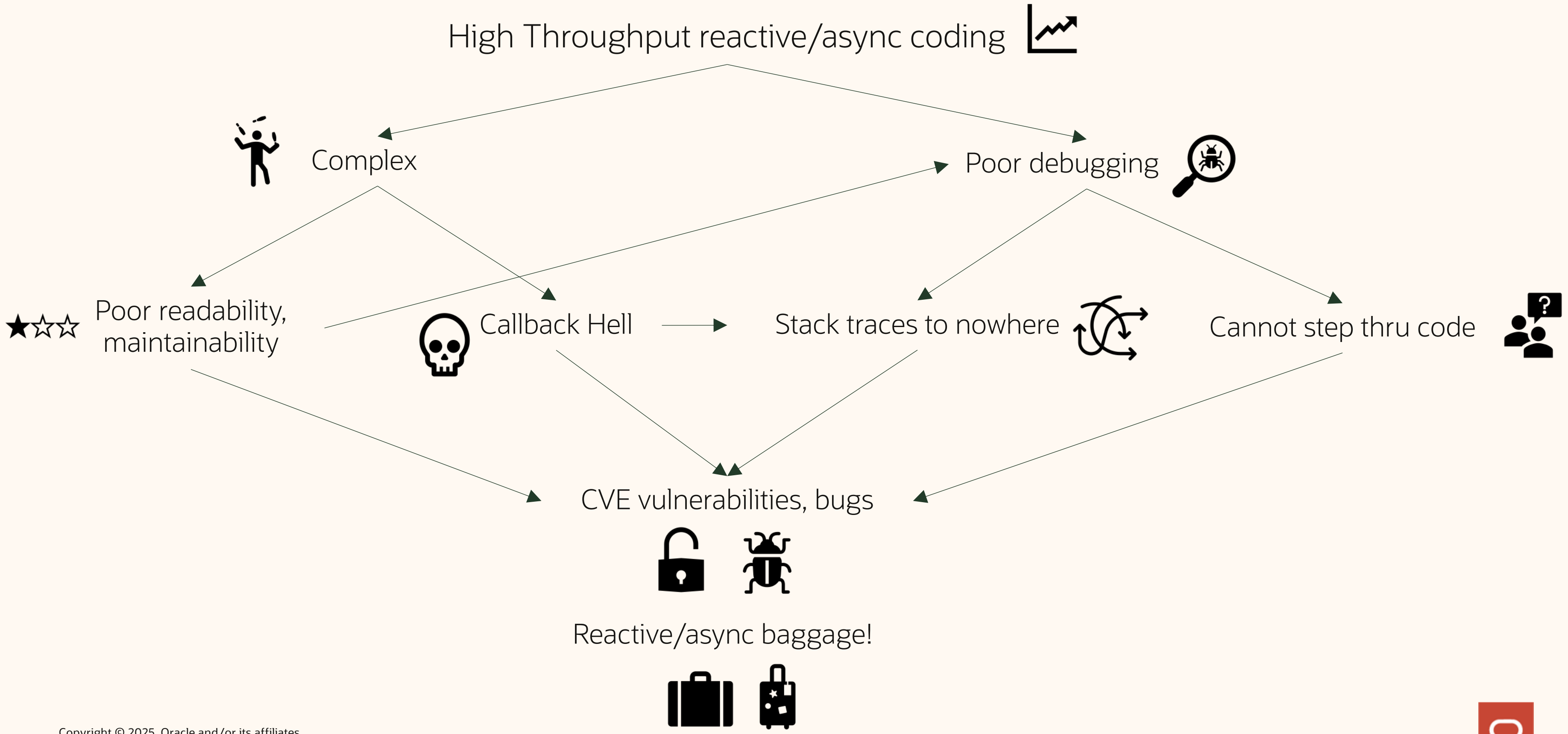
Anatomy of RAG (post embedding)



Cloud Native RAG

Build a cloud native RAG with Helidon and Coherence

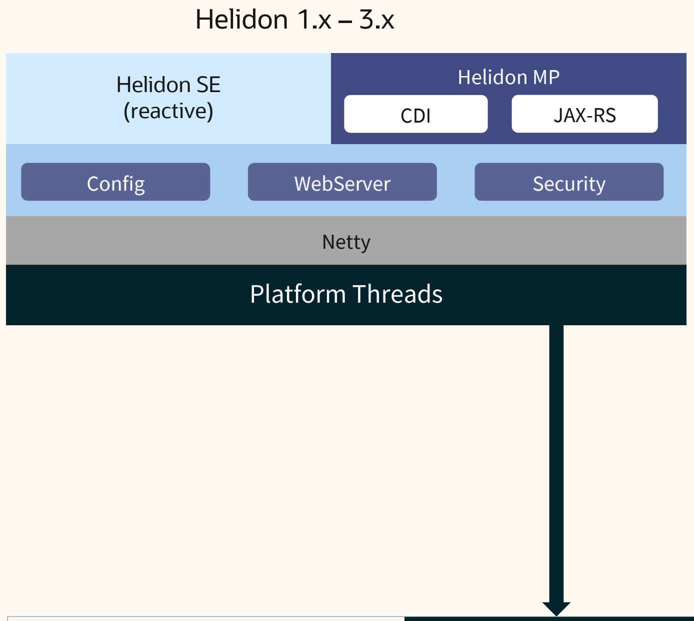
Rethinking microservices and AI Agents



Reactive Java is a trap

Helidon 1.x-3.x

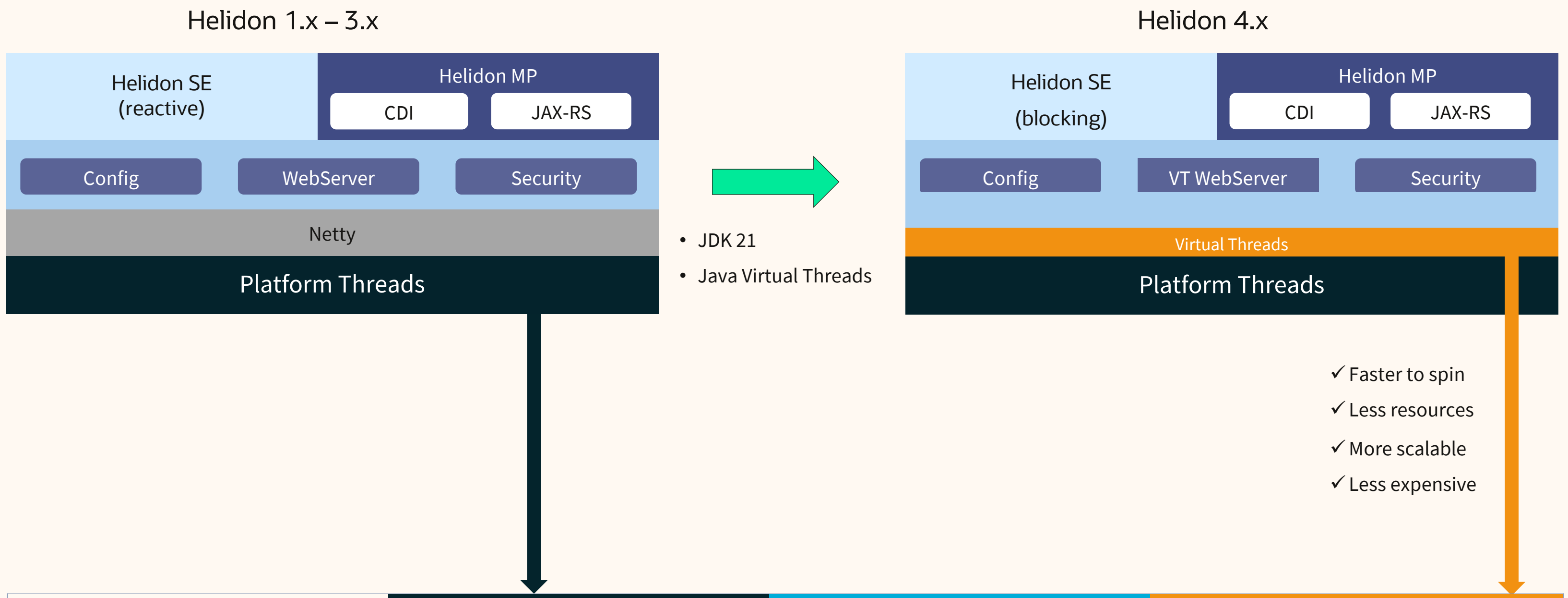
Architectural evolution



	Platform (OS) Threads	Golang Goroutines
Memory overhead	Hundreds of kilobytes to megabytes	A few kilobytes
Concurrent upper bound	Thousands	Hundreds of thousands

Helidon 4

Architectural evolution



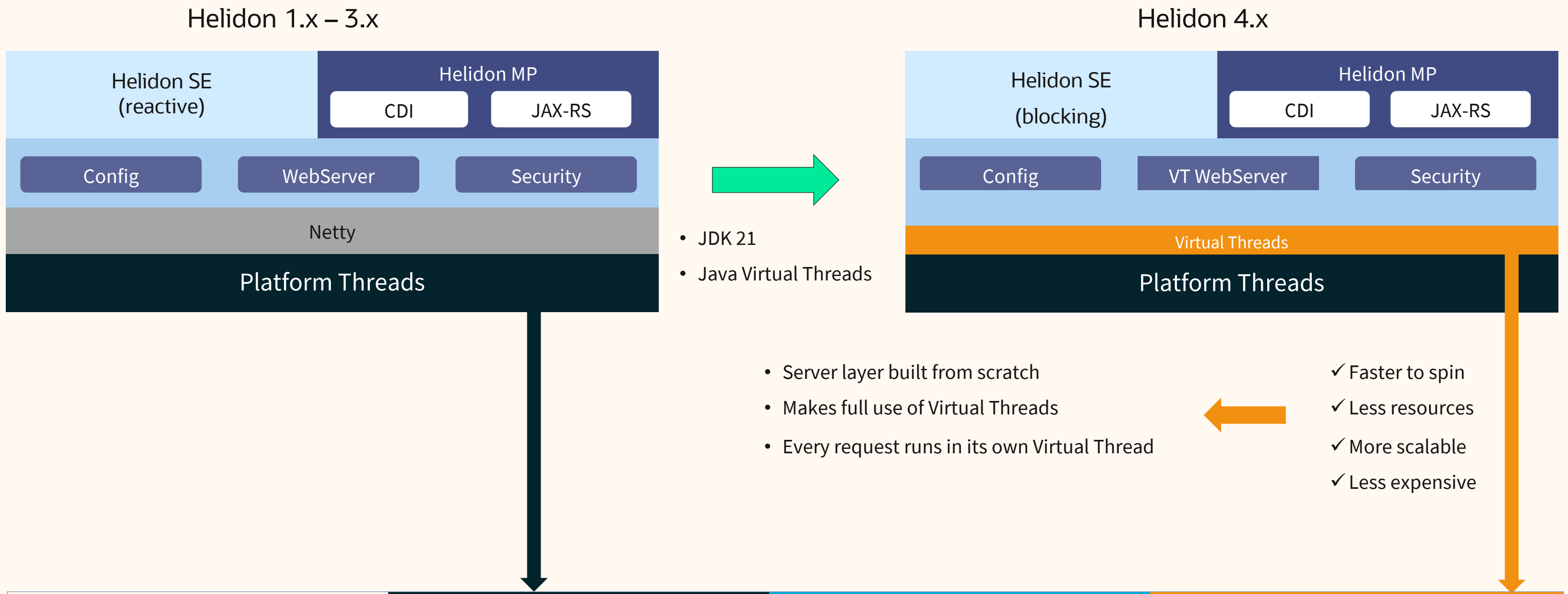
- JDK 21
- Java Virtual Threads

✓ Faster to spin
✓ Less resources
✓ More scalable
✓ Less expensive

	Platform (OS) Threads	Golang Goroutines	Java Virtual Threads
Memory overhead	Hundreds of kilobytes to megabytes	A few kilobytes	A few hundred bytes
Concurrent upper bound	Thousands	Hundreds of thousands	Millions

Helidon 4

Architectural evolution: scalability of reactive, simplicity of 1 thread per request



- JDK 21
- Java Virtual Threads

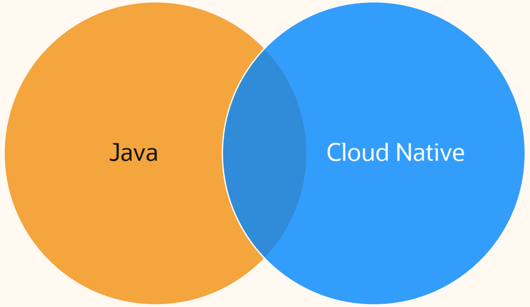
- Server layer built from scratch
- Makes full use of Virtual Threads
- Every request runs in its own Virtual Thread

✓ Faster to spin
✓ Less resources
✓ More scalable
✓ Less expensive

	Platform (OS) Threads	Golang Goroutines	Java Virtual Threads
Memory overhead	Hundreds of kilobytes to megabytes	A few kilobytes	A few hundred bytes
Concurrent upper bound	Thousands	Hundreds of thousands	Millions

Helidon: An introduction

- Modern, open-source Java framework for developing cloud native micro-services
  - Created, maintained, supported and fully owned by Oracle
  - Container, Kubernetes and CNCF-friendly
  - Only framework to implement Virtual Threads from the socket up



https://helidon.io/

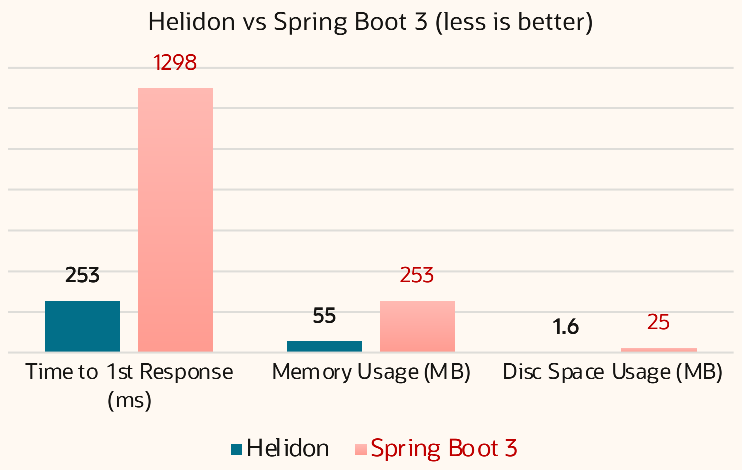
Helidon: An introduction

- Modern, open-source Java framework for developing cloud native micro-services
  - Created, maintained, supported and fully owned by Oracle
  - Container, Kubernetes and CNCF-friendly
  - Only framework to implement Virtual Threads from the socket up
- Developer-friendly, flexible, observable
  - Scalability of reactive, simplicity of thread per request
  - Easy to write *clean* code, understand, debug, maintain
  - Lower learning curve cloud native for Jakarta EE developers

https://helidon.io/

Helidon: An introduction

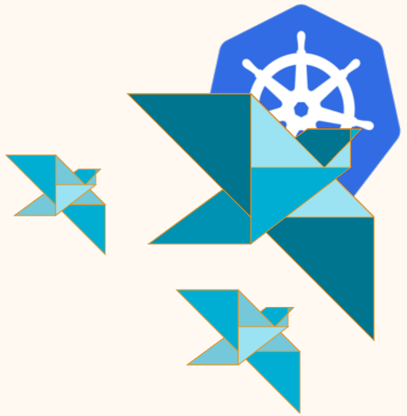
- Modern, open-source Java framework for developing cloud native micro-services
  - Created, maintained, supported and fully owned by Oracle
  - Container, Kubernetes and CNCF-friendly
  - Only framework to implement Virtual Threads from the socket up
- Developer-friendly, flexible, observable
  - Scalability of reactive, simplicity of thread per request
  - Easy to write *clean* code, understand, debug, maintain
  - Lower learning curve cloud native for Jakarta EE developers
- Light, fast and secure
  - No app server required
  - Adoption of cutting-edge Java features to improve performance & productivity
  - Judicious use of external dependencies to minimize CVEs



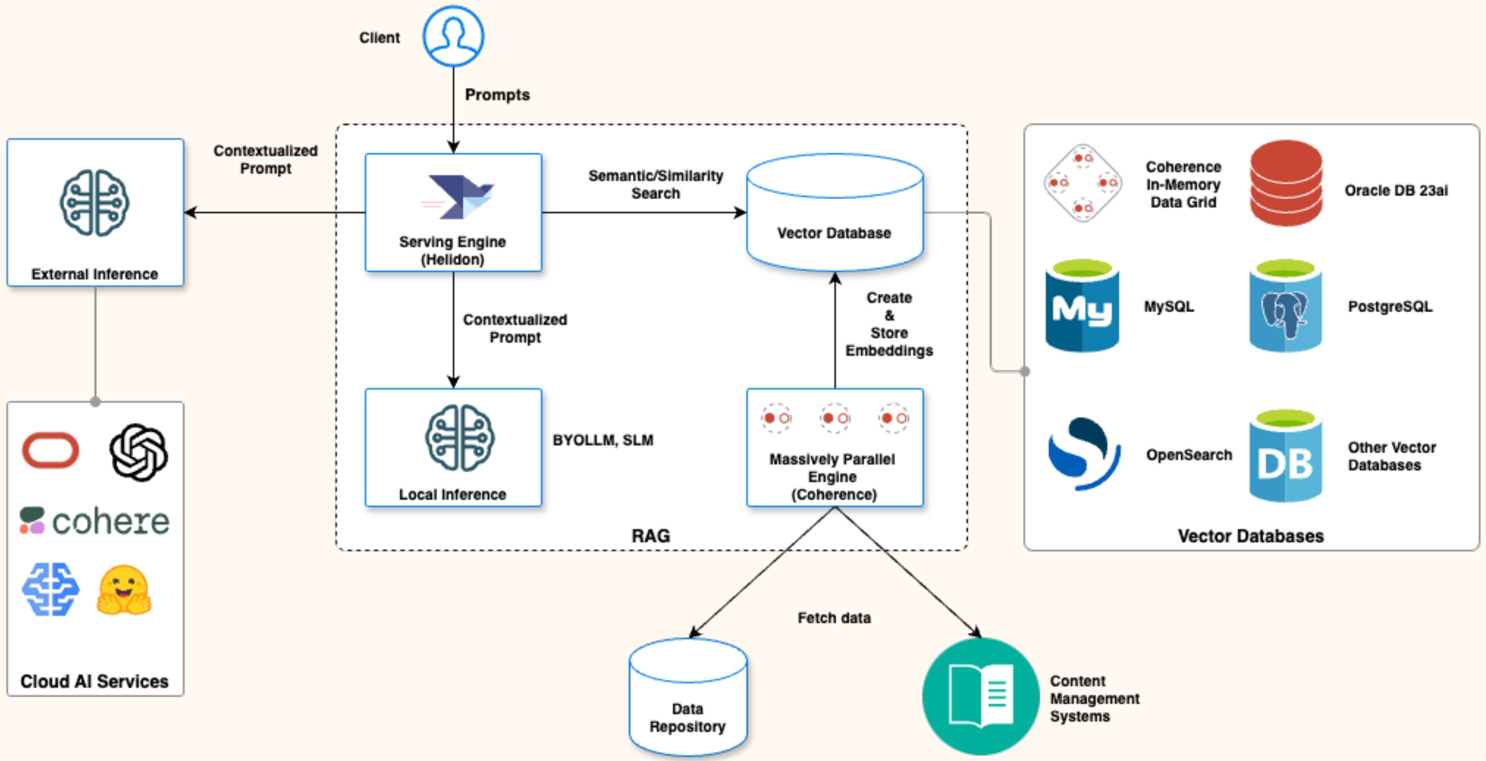
https://helidon.io/

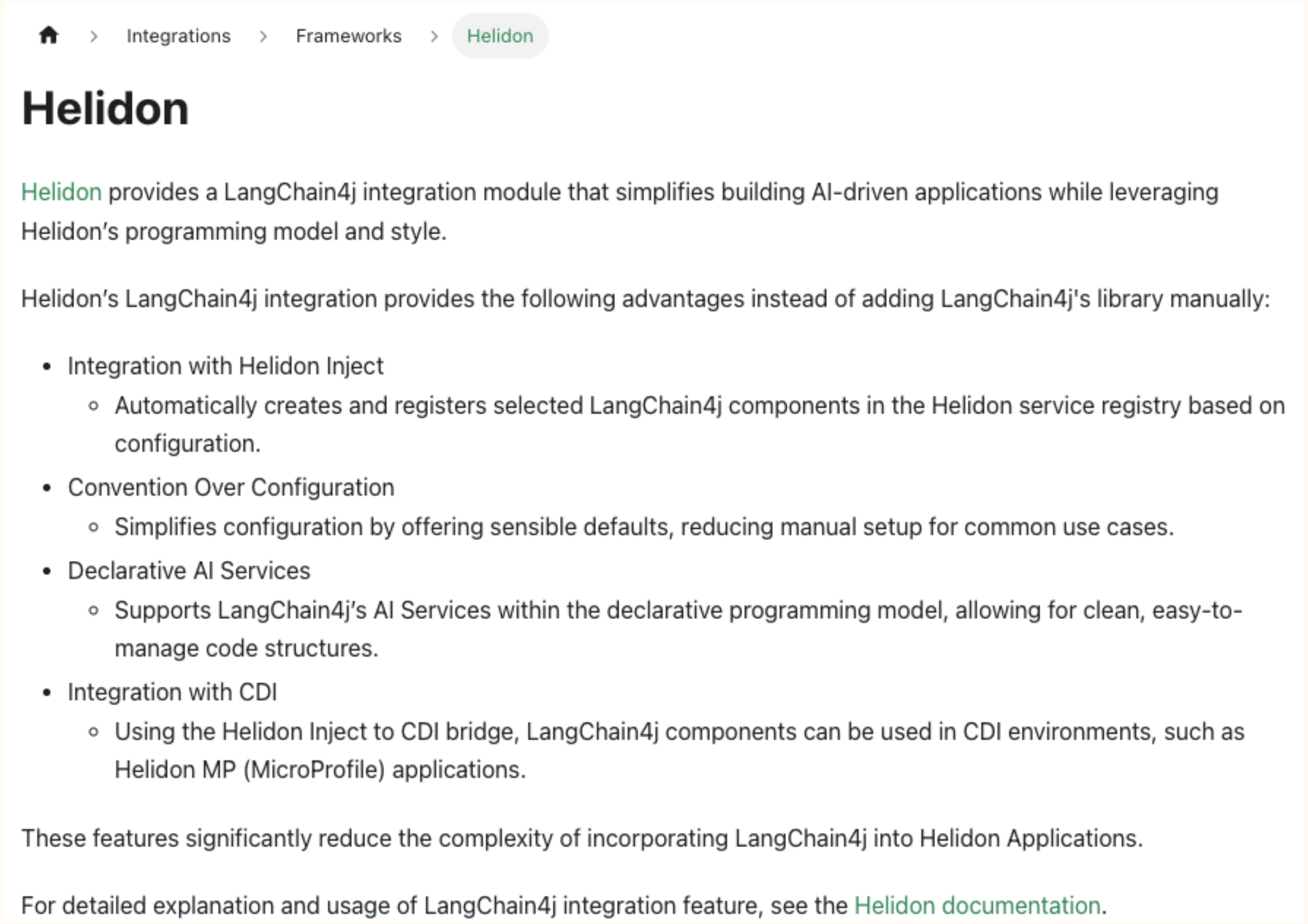
Helidon AI

- Designed for light, fast, scalable, cloud native AI agents as microservices
  - Easy to cleanly scale AI Agents to millions of requests w/o reactive code technical debt
- Support for LangChain4j
  - Best-in-class AI Integration framework for Java
  - Provides a unified API to experiment, develop and use AI with Java
  - Support for common abstractions, patterns and techniques e.g.
    - Prompt templates, parsers, document loaders, splitters
    - Support for various LLMs and Vector Stores, including Oracle DB 23ai
    - Chat memory management, Function calling
    - LLM Extension with RAG, Agent
- Simplifies building AI-enabled microservices in a cloud-native way
  - OpenTelemetry, Health, compile time Dependency Injection for Config



Building a RAG-based system with Helidon





Integrations > Frameworks > Helidon

Helidon

Helidon provides a LangChain4j integration module that simplifies building AI-driven applications while leveraging Helidon's programming model and style.

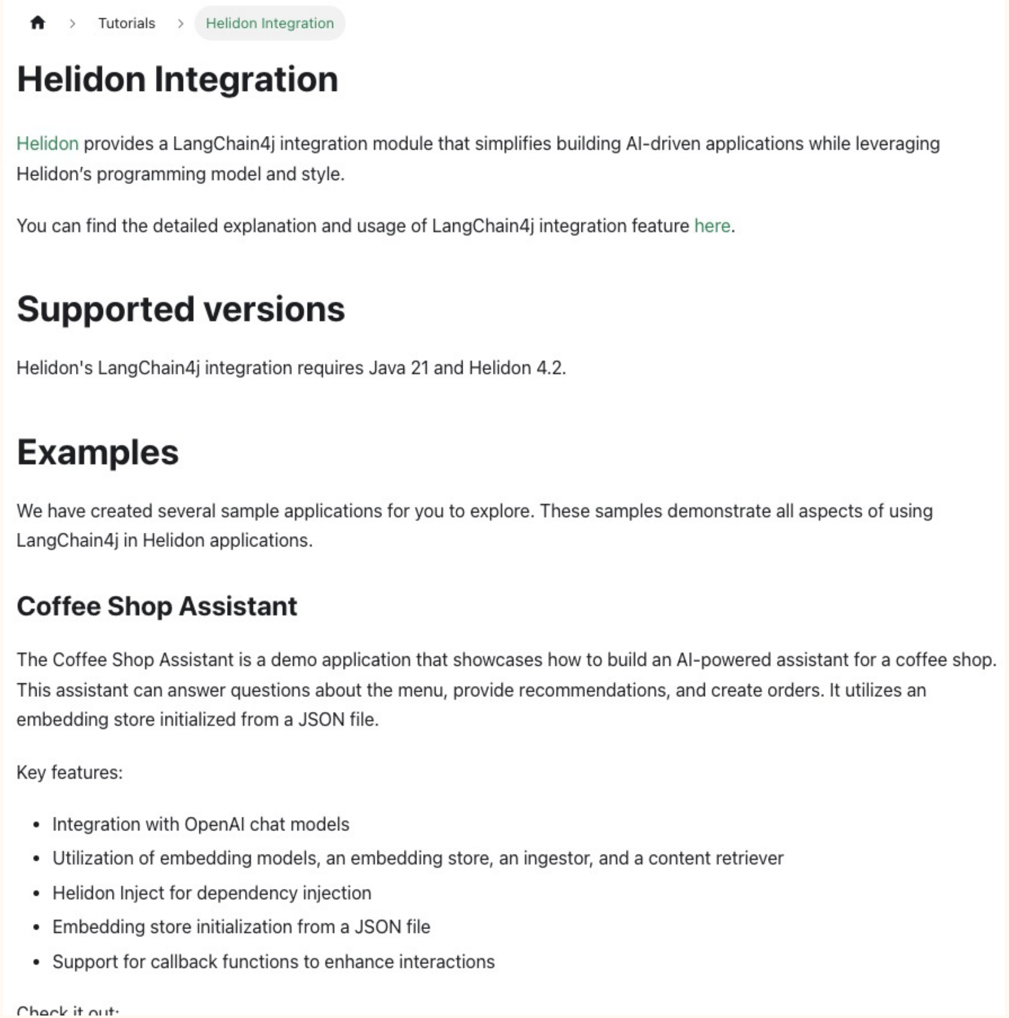
Helidon's LangChain4j integration provides the following advantages instead of adding LangChain4j's library manually:

- Integration with Helidon Inject
  - Automatically creates and registers selected LangChain4j components in the Helidon service registry based on configuration.
- Convention Over Configuration
  - Simplifies configuration by offering sensible defaults, reducing manual setup for common use cases.
- Declarative AI Services
  - Supports LangChain4j's AI Services within the declarative programming model, allowing for clean, easy-to-manage code structures.
- Integration with CDI
  - Using the Helidon Inject to CDI bridge, LangChain4j components can be used in CDI environments, such as Helidon MP (MicroProfile) applications.

These features significantly reduce the complexity of incorporating LangChain4j into Helidon Applications.

For detailed explanation and usage of LangChain4j integration feature, see the Helidon documentation.

https://docs.langchain4j.dev/integrations/frameworks/helidon


Tutorials > Helidon Integration

Helidon Integration

Helidon provides a LangChain4j integration module that simplifies building AI-driven applications while leveraging Helidon's programming model and style.

You can find the detailed explanation and usage of LangChain4j integration feature here.

Supported versions

Helidon's LangChain4j integration requires Java 21 and Helidon 4.2.

Examples

We have created several sample applications for you to explore. These samples demonstrate all aspects of using LangChain4j in Helidon applications.

Coffee Shop Assistant

The Coffee Shop Assistant is a demo application that showcases how to build an AI-powered assistant for a coffee shop. This assistant can answer questions about the menu, provide recommendations, and create orders. It utilizes an embedding store initialized from a JSON file.

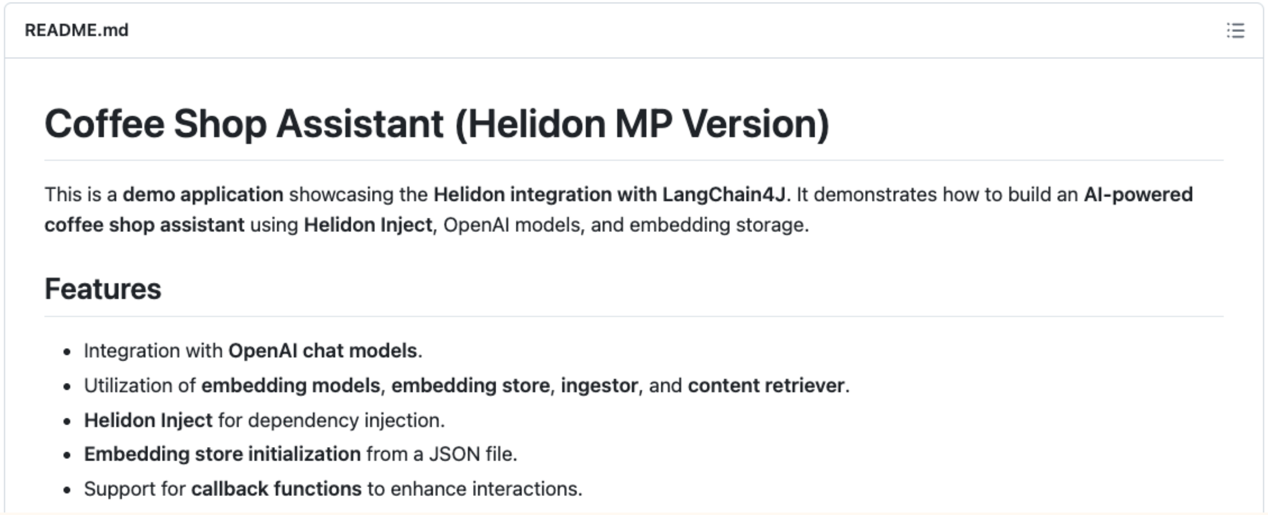
Key features:

- Integration with OpenAI chat models
- Utilization of embedding models, an embedding store, an ingestor, and a content retriever
- Helidon Inject for dependency injection
- Embedding store initialization from a JSON file
- Support for callback functions to enhance interactions

Check it out:

https://docs.langchain4j.dev/tutorials/helidon-integration

Demo


README.md

Coffee Shop Assistant (Helidon MP Version)

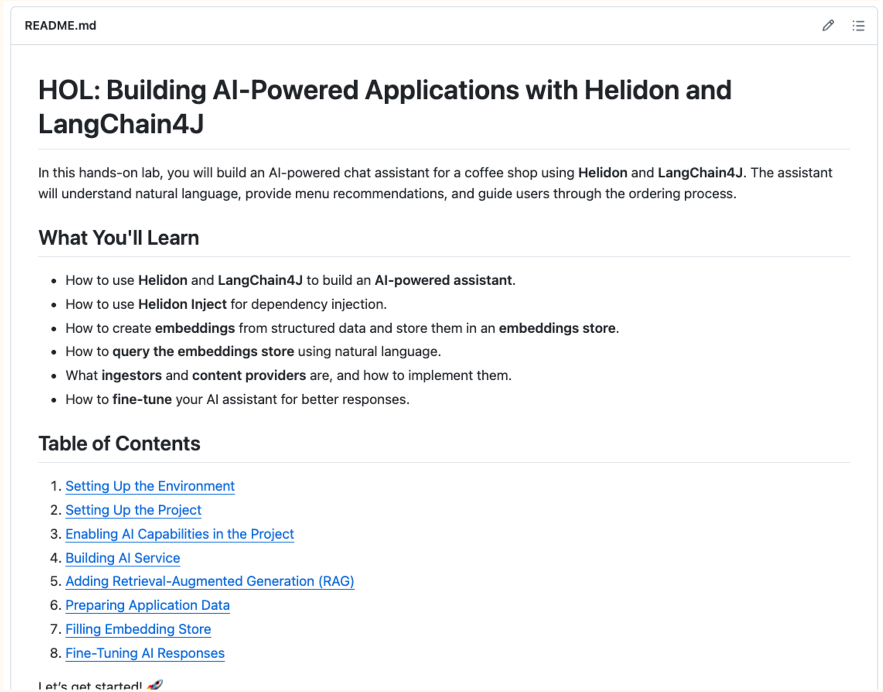
This is a **demo application** showcasing the **Helidon integration with LangChain4J**. It demonstrates how to build an **AI-powered coffee shop assistant** using **Helidon Inject**, OpenAI models, and embedding storage.

Features

- Integration with **OpenAI chat models**.
- Utilization of **embedding models**, **embedding store**, **ingestor**, and **content retriever**.
- **Helidon Inject** for dependency injection.
- **Embedding store initialization** from a JSON file.
- Support for **callback functions** to enhance interactions.

Demo: Helidon MP

Example Code: https://github.com/helidon-io/helidon-examples/tree/4.2.0/examples/integrations/langchain4j/coffee-shop-assistant-mp


README.md

HOL: Building AI-Powered Applications with Helidon and LangChain4J

In this hands-on lab, you will build an AI-powered chat assistant for a coffee shop using **Helidon** and **LangChain4J**. The assistant will understand natural language, provide menu recommendations, and guide users through the ordering process.

What You'll Learn

- How to use **Helidon** and **LangChain4J** to build an **AI-powered assistant**.
- How to use **Helidon Inject** for dependency injection.
- How to create **embeddings** from structured data and store them in an **embeddings store**.
- How to **query the embeddings store** using natural language.
- What **ingestors** and **content providers** are, and how to implement them.
- How to **fine-tune** your AI assistant for better responses.

Table of Contents

Let's get started! 🚀

Demo: Helidon SE

Example Code: https://github.com/helidon-io/helidon-examples/tree/4.2.0/examples/integrations/langchain4j/coffee-shop-assistant-se

Hands-on lab: https://livelabs.oracle.com/pls/apex/dbpm/r/livelabs/view-workshop?wid=4115

https://github.com/helidon-io/helidon-labs/tree/main/hols/langchain4j

LangChain4J

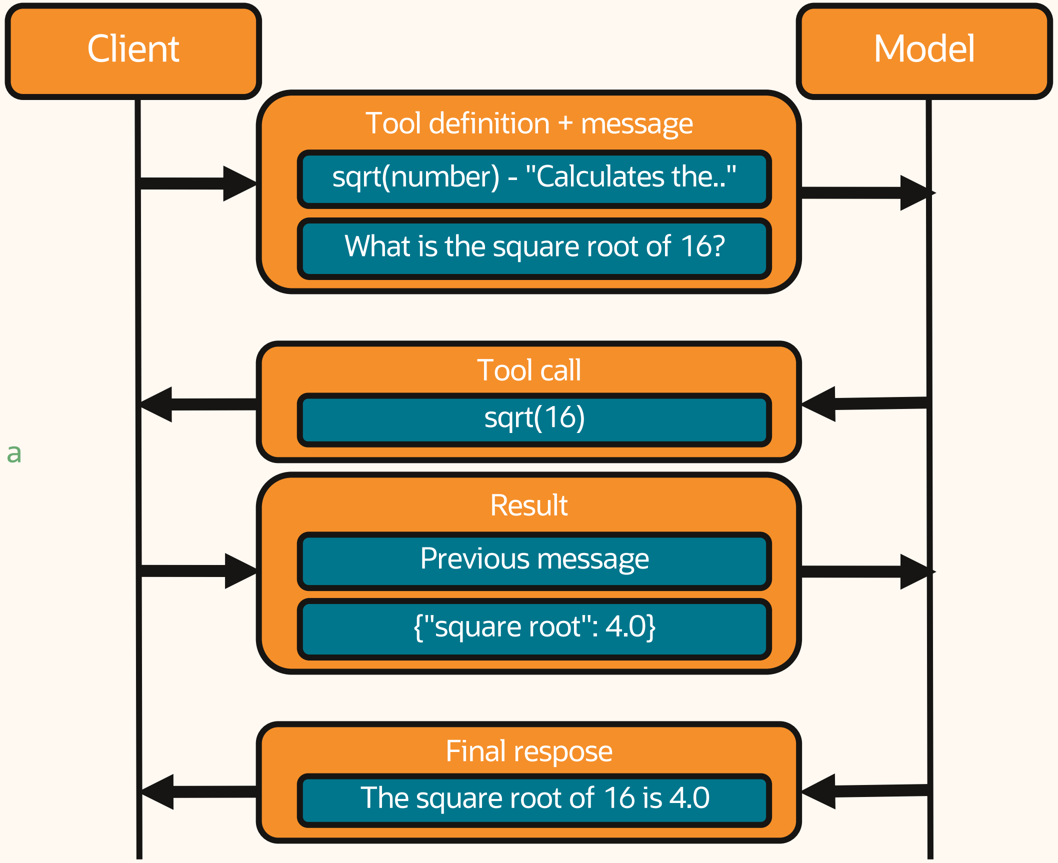
Basic Chat API

```java
ChatLanguageModel chatModel = OpenAiChatModel.builder()
        .apiKey("demo")
        .modelName(OpenAiChatModelName.GPT_4_O_MINI)
        .build();
String joke = chatModel.generate("Tell me a joke about Java");
System.out.println(joke);
```

LangChain4J

Tools: Function Calling

```java
class Calculator {
    @Tool("Calculates the square root of a
number")
    double sqrt(int x) {
        return Math.sqrt(x);
    }
}
```


Client
Model
Tool definition + message
sqrt(number) - "Calculates the.."
What is the square root of 16?
Tool call
sqrt(16)
Result
Previous message
{"square root": 4.0}
Final respose
The square root of 16 is 4.0

LangChain4J

Tools: Function Calling

```
class Calculator {
    @Tool("Calculates the square root of
a number")
    double sqrt(int x) {
        return Math.sqrt(x);
    }
}
```

```
ChatLanguageModel model = OpenAiChatModel.builder()
        .apiKey("demo")
        .modelName(OpenAiChatModelName.GPT_4_O_MINI)
        .strictTools(true)
        .build();

Assistant assistant = AiServices.builder(Assistant.class)
        .chatLanguageModel(model)
        .tools(new Calculator())
        .chatMemory(MessageWindowChatMemory.withMaxMessages(10))
        .build();

String answer = assistant.chat("What is the square root of 16?");
```

```
interface Assistant {
    String chat(String userMessage);
}
```

LangChain4J

Comparison with official OpenAI Java client

```java
var client = OpenAIOkHttpClient.builder().apiKey(TOKEN).build();
var createParams = ChatCompletionCreateParams.builder()
       .model(ChatModel.GPT_4O_MINI)
       .addTool(ChatCompletionTool.builder()
           .function(FunctionDefinition.builder()
               .name("sqrt")
               .description("Calculates the square root of a number")
               .strict(true)
               .parameters(FunctionParameters.builder()
                  .putAdditionalProperty("type", JsonValue.from("object"))
                  .putAdditionalProperty("properties", JsonValue.from(Map.of("number", Map.of("type", "number"))))
                  .putAdditionalProperty("required", JsonValue.from(List.of("number")))
                  .putAdditionalProperty("additionalProperties", JsonValue.from(false))
                  .build())
               .build())
           .build())
       .addUserMessage("What is a square root of 16?")
       .build();
var chatCompletion = client.chat().completions().create(createParams);
var toolCall = chatCompletion.choices().stream()
      .map(ChatCompletion.Choice::message)
      .map(ChatCompletionMessage::toolCalls)
      .flatMap(Optional::stream)
      .flatMap(Collection::stream)
      .filter(call -> call.function().name().equals("sqrt"))
      .findFirst()
      .orElseThrow();
int toolArgument = jsonMapper().readValue(toolCall.function().arguments(), JsonNode.class).get("number").asInt();
// Calling the tool function manually
double toolResult = new Calculator().sqrt(toolArgument);
var paramsWithToolResult = createParams.toBuilder()
       .addMessage(chatCompletion.choices().getFirst().message())
       .addMessage(ChatCompletionToolMessageParam.builder()
            .toolCallId(toolCall.id())
            .content(String.valueOf(toolResult))
            .build())
        .build();
String answer = client.chat().completions().create(paramsWithToolResult)
    .choices()
    .stream()
    .findFirst()
    .map(ChatCompletion.Choice::message)
    .flatMap(ChatCompletionMessage::content)
    .orElseThrow();
```

```java
class Calculator {
    double sqrt(int x) {
        return Math.sqrt(x);
    }
}
```

LangChain4J

Switch to Ollama model

```java
ChatLanguageModel model = OpenAiChatModel.builder()
            .apiKey("demo")
            .modelName(OpenAiChatModelName.GPT_4_O_MINI)
            .strictTools(true)
            .build();

Assistant assistant = AiServices.builder(Assistant.class)
            .chatLanguageModel(model)
            .tools(new Calculator()
            .chatMemory(
              MessageWindowChatMemory.withMaxMessages(10))
            .build();

String answer = assistant
            .chat("What is the square root of 16?");
```

```java
ChatLanguageModel model = OllamaChatModel.builder()
          .modelName("llama3-groq-tool-use:8b")
          .baseUrl("http://localhost:11434")
          .build();

Assistant assistant = AiServices.builder(Assistant.class)
          .chatLanguageModel(model)
          .tools(new Calculator())
          .chatMemory(
             MessageWindowChatMemory.withMaxMessages(10))
          .build();

String answer = assistant
          .chat("What is the square root of 16?");
```

LangChain4J

The most popular LLM Java library

- API abstraction
- Easy to use: Ai Services
- Easy to switch models!
- Lots of goodies: RAG, Tools etc.

- Feedback and community support.

LangChain4j Integration

Helidon & AI

LangChain4J Integration

Preview feature in Helidon 4.2

- Brings the power of AI to Helidon Applications
- Based on Helidon Inject
  - Build-time
  - No impact on performance
- Features
  - All features of LangChain4J
  - Convention over configuration (for selected components)
  - Declarative AI Services
- Bridge to CDI
  - Create once, use in Helidon SE or Helidon MP
  - Selective two-way injection

Helidon AI Integration

LangChain4j

```java
@Ai.Service
public interface ChatAiService {
    String chat(String question);
}
```

```java
var chatSvc = Services.get(ChatAiService.class);
String response = chatSvc.chat("Tell me a joke please.");
```

./application.yaml

```yaml
langchain4j:
  open-ai:
    chat-model:
      enabled: true
      api-key: "demo"
      model-name: "gpt-4o-mini"
```

Helidon AI Integration

LangChain4j - Tools

```java
@Ai.Service
public interface ChatAiService {
    String chat(String question);
}


@Service.Singleton
public class CalculatorService {
    @Tool("Calculates the square root of a number")
    double sqrt(int x) {
        return Math.sqrt(x);
    }
}
```

```java
var chatSvc = Services.get(ChatAiService.class);
String response = chatSvc.chat("What is the square root of 16?");
```

./application.yaml

```yaml
langchain4j:
  open-ai:
    chat-model:
      enabled: true
      api-key: "demo"
      model-name: "gpt-4o-mini"
```

Helidon AI Integration

LangChain4j – Switching models

```java
@Ai.Service
public interface ChatAiService {
    String chat(String question);
}


@Service.Singleton
public class CalculatorService {
    @Tool("Calculates the square root of a number")
    double sqrt(int x) {
        return Math.sqrt(x);
    }
}



var chatSvc = Services.get(ChatAiService.class);
String response = chatSvc.chat("What is the square root of 16?");
```

./application.yaml

```yaml
langchain4j:
  open-ai:
    chat-model:
      enabled: true
      api-key: "demo"
      model-name: "gpt-4o-mini"
```

Helidon AI Integration

LangChain4j – Switching models

```java
@Ai.Service
public interface ChatAiService {
    String chat(String question);
}


@Service.Singleton
public class CalculatorService {
    @Tool("Calculates the square root of a number")
    double sqrt(int x) {
        return Math.sqrt(x);
    }
}
```

```java
var chatSvc = Services.get(ChatAiService.class);
String response = chatSvc.chat("What is the square root of 16?");
```

./application.yaml

```yaml
langchain4j:
  ollama:
    chat-model:
      enabled: true
      model-name: llama3-groq-tool-use:8b
      base-url: http://localhost:11434
```

Helidon AI Integration

LangChain4j - Embedding

```java
@Ai.Service
@Ai.ChatMemoryWindow(10)
public interface ChatAiService {
    @SystemMessage("""
            You are Frank - a server in a coffee shop.
            You must not answer any questions not related
            to the menu or making orders.
            """)
    String chat(String question);
}
```

./application.yaml

```yaml
langchain4j:
  open-ai:
    chat-model:
      enabled: true
      api-key: "demo"
      model-name: "gpt-4o-mini"
  rag:
    embedding-store-content-retriever:
      enabled: true
      max-results: 10
      min-score: 0.6
      embedding-store: "EmbeddingStore"
```

Helidon AI Integration

LangChain4j - Embedding

```java
@Service.Singleton
public class EmbeddingModelFactory implements Supplier<EmbeddingModel>
{
    @Override
    public EmbeddingModel get() {
        return new AllMiniLmL6V2EmbeddingModel();
    }
}
```

```java
@Service.Singleton
public class EmbeddingStoreFactory
                  implements Supplier<EmbeddingStore<TextSegment>> {
    @Override
    public EmbeddingStore<TextSegment> get() {
        return new InMemoryEmbeddingStore<>();
    }
}
```

```yaml
langchain4j:
  open-ai:
    chat-model:
      enabled: true
      api-key: "demo"
      model-name: "gpt-4o-mini"
  rag:
    embedding-store-content-retriever:
      enabled: true
      max-results: 10
      min-score: 0.6
      embedding-store: "EmbeddingStore"
```

Helidon AI Integration

LangChain4j - Embedding

```java
@Service.Singleton
public class MenuItemsIngestor {
    private final EmbeddingStore<TextSegment> embeddingStore;
    private final EmbeddingModel embeddingModel;

    @Service.Inject
    MenuItemsIngestor(EmbeddingStore<TextSegment> embeddingStore,
                      EmbeddingModel embeddingModel) {
        this.embeddingStore = embeddingStore;
        this.embeddingModel = embeddingModel;
    }

    public void ingest(List<String> menuItems) {
        var ingestor = EmbeddingStoreIngestor.builder()
                .embeddingModel(embeddingModel)
                .embeddingStore(embeddingStore)
                .build();

        ingestor.ingest(menuItems.stream()
                                 .map(Document::from)
                                 .toList());
    }
```

./application.yaml

```yaml
langchain4j:
  open-ai:
    chat-model:
      enabled: true
      api-key: "demo"
      model-name: "gpt-4o-mini"
  rag:
    embedding-store-content-retriever:
      enabled: true
      max-results: 10
      min-score: 0.6
      embedding-store: "EmbeddingStore"
```

Helidon for Agentic AI Design Patterns

- Reflection
  - examine and review own response, iterative improvement
- Tool use
  - LLM decides to call API in Agent to access external system/tool
- Planning
  - decompose complex task to a sequence of smaller items and execute
- Multi-agent, multi-LLM collaboration
  - Specialized agents, LLMs collaborate
  - Philosophy similar to UNIX, microservices orchestration of subtasks

Coherence: A jewel in Oracle's middleware

Coherence CE

- In-memory data grid with persistence
- Out-of-the-box integration with Oracle Database, MySQL, PostgreSQL
- More than just another cache: distributed processing, eventing, continuous querying
- Fast, scalable, reliable, multi-site capable
- Successful adoption by large customers
- Used in mission critical systems, analytics
- {Kubernetes, cloud, polyglot}-ready
  - Open source Kubernetes operator
  - Native Java, C++, .Net, Go, Python, JavaScript libraries + REST, gRPC, GraphQL, OpenTelemetry

Coherence AI: A new jewel facet

- Two major things added to become a Vector Store/Database

1. Calculating and Storing Vector Embeddings

Massively parallel RAG engine

2. Semantic Similarity Search

Recommendations

Anomaly/Fraud Detection

Image/object detection/classification

Pattern Analysis

Innovating with Coherence CE - AI
Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation (RAG)

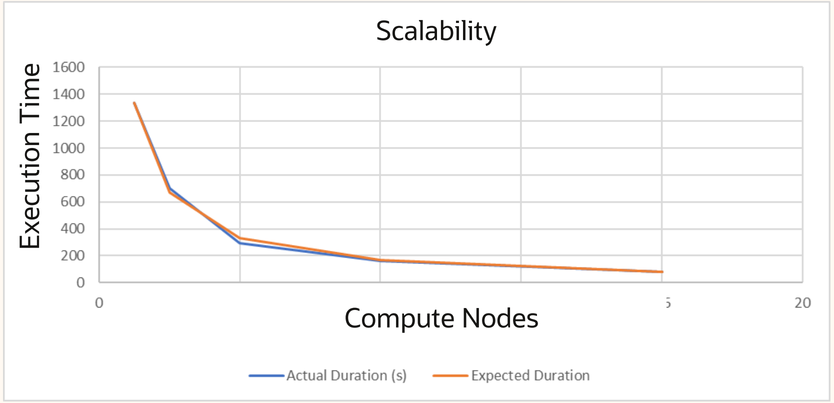
- Technique for improving output of LLMs in Generative AI
- Augment LLMs with specialized content
- Requires ingestion and vectorization of data

Coherence distributed computing uniquely suited to RAG

- Vector store **already in Coherence CE 24.09**

Developed RAG system on Coherence + Helidon

- Highly scalable data ingestion (millions of docs)
- Improves accuracy, reduces hallucinations
- Integration with popular AI libraries, vector DBs



How Can Coherence (and Helidon) Do This?

Using existing Coherence features that Redis and OCI Cache don't have:

- Cache Stores: read/write-behind cache to document sources and Oracle DB
- Server-Side Events: parallel content fetching, splitting and vectorization
- Queries & Aggregators: parallel vector-based proximity search
- Custom Indexes: HNSW, Binary Quantization, Cosine Distance, Hybrid semantic similarity + exact filter searches
- Elastic Data: to store documents, document chunks and vectors on disk (commercial feature)

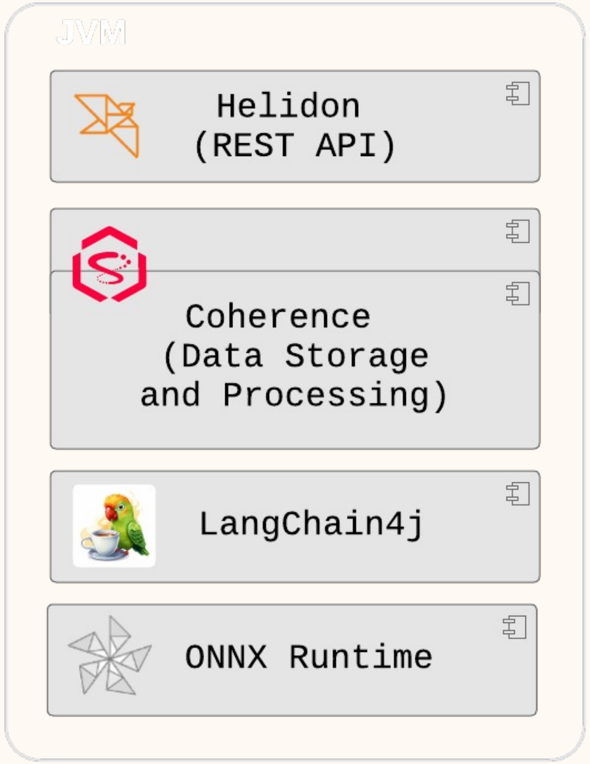
Using existing Helidon features:

- Virtual Threads to scale to millions of requests w/o reactive code technical debt
- Web Server: serve REST endpoints we want to expose to clients; static content (Chatbot UI)
- Web Client: make REST calls to fetch documents to index, create vector embeddings
- Configuration: fast, light, secure compile time Dependency Injection to configure various integration components (LLMs, DMS, DB, etc.)

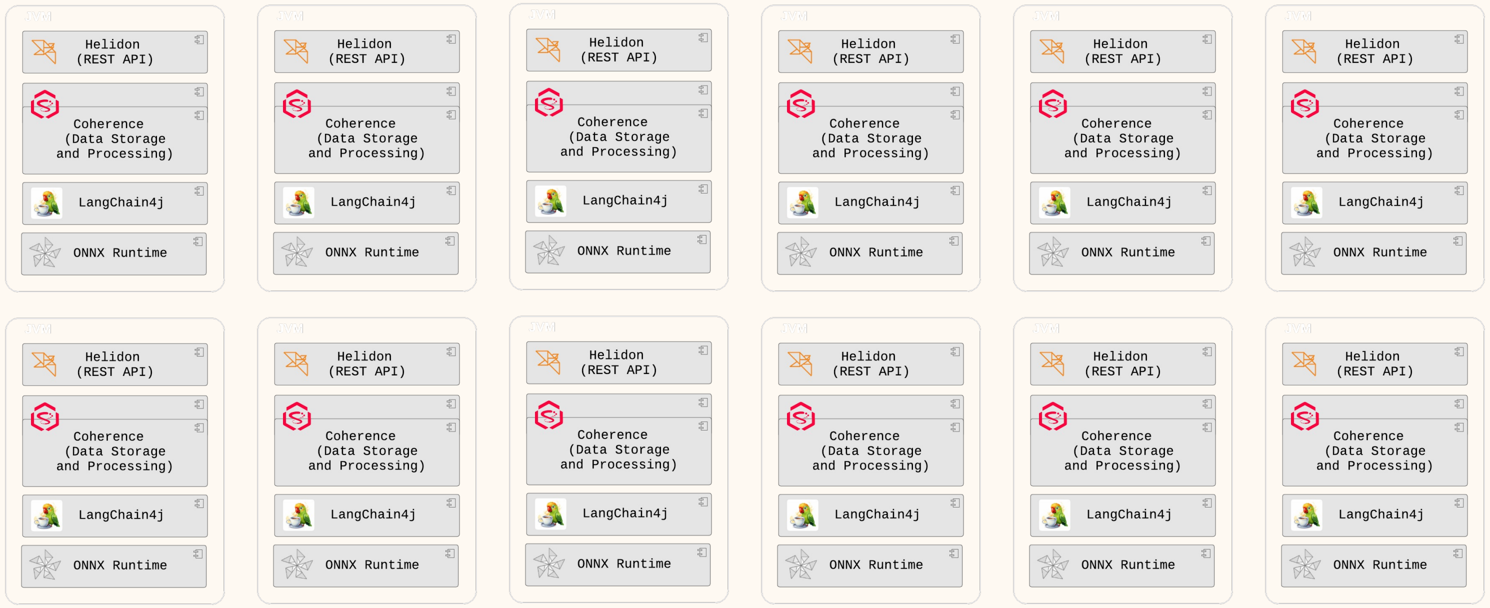
Using existing open source libraries:

- Langchain4j: content extraction and splitting, LLM integration, etc.
- ONNX Runtime: localized embeddings creation/content vectorization

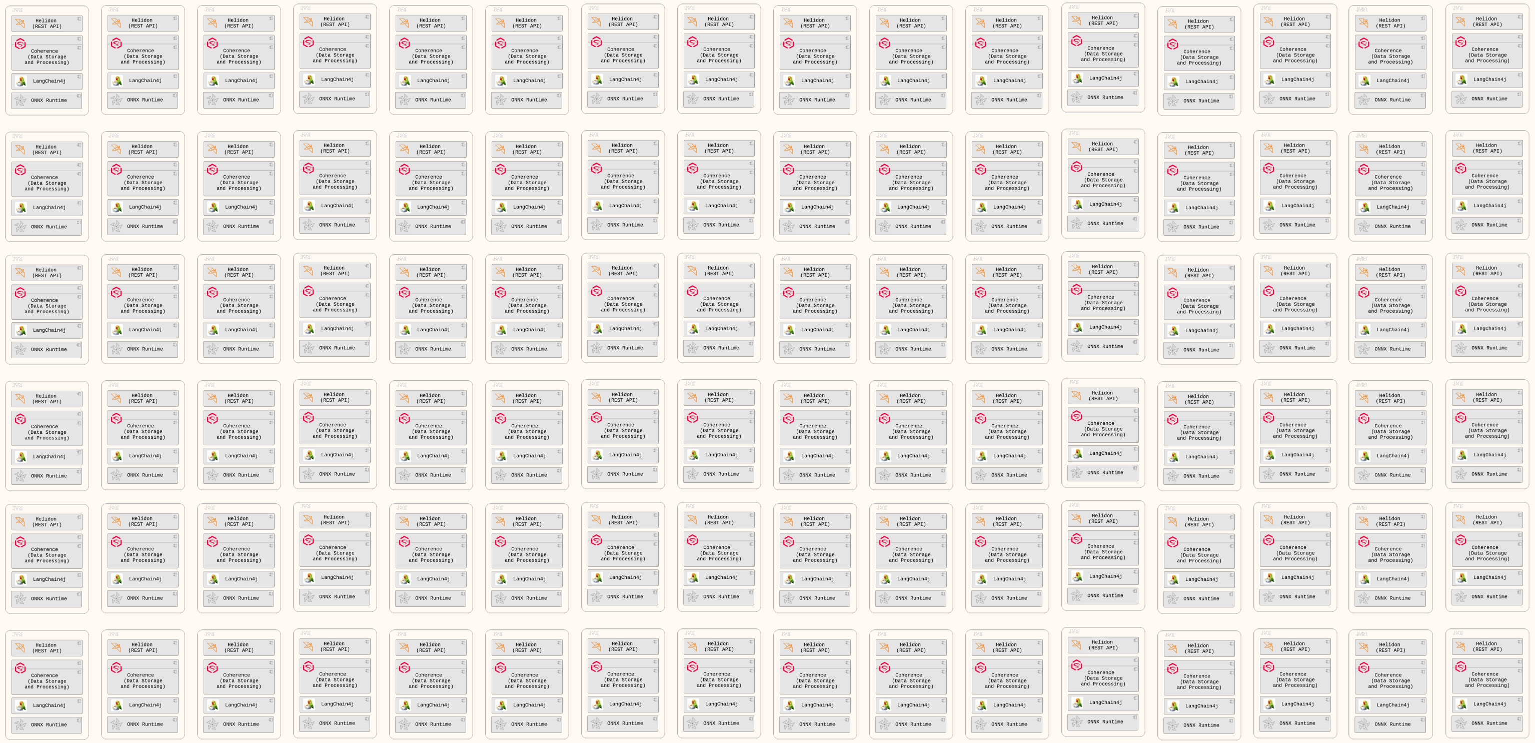
Architecture: Single Member



Architecture: Cluster



Architecture: Cluster

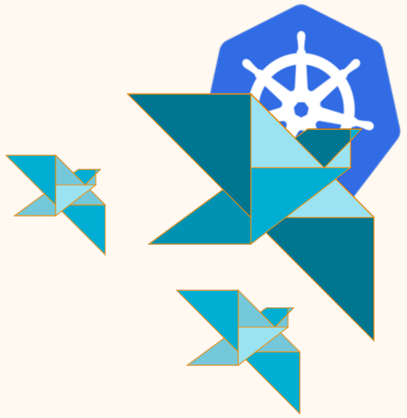


Demo

Summary

Benefits of Helidon and AI

- Easy way for building AI Agents/services and RAG in Java
  - Unified API for many LLMs, Vector Stores/DB
  - RAG pipeline, prompt templating, chat memory mgmt
- Helidon's Philosophy
  - "Simple things should be simple, complex things should be possible" - Alan Kay
  - Easy to cleanly scale AI Agents to millions of requests w/o reactive technical debt
  - Simplicity – Thin jars, fewer dependencies, fewer CVEs
  - Cloud native, k8s friendly
- Built in
  - OpenAPI, OpenTelemetry, MicroProfile Config, JAX-RS, Health Checks, FT, etc.
- Community driven, standards based open source



Unique to Coherence AI – Elegant Simplicity at Edge

- ✓ Multipurpose In-Memory Data Grid w/ Vector Store/DB capabilities
- ✓ Scale from 1 JVM on Raspberry Pi to thousands in Data Center
- ✓ Unlimited data types
- ✓ Distributed Cache w/ auto rebalancing, fault tolerance
- ✓ In-memory vector storage and similarity search + metadata filtering
- ✓ Continuous Query cache & pub/sub events
- ✓ Polyglot, GraphQL, CohQL
- ✓ Observability using Mbeans, OpenTelemetry
- ✓ Optional Integration w/ any document storage system
  - ✓ Oracle DB 23ai

Great time to be an Agentic AI Developer

Thank you

@helidon_project

helidon.io

medium.com/helidon

github.com/helidon-io/helidon

youtube.com/Helidon_Project

helidon.slack.com

helidon-io.bksy.social

Thank you

@OracleCoherence

coherence.community

medium.com/oracle-coherence

github.com/oracle/coherence

youtube.com/@OracleCoherence

oraclecoherence.slack.com

oracle-coherence.bksy.social