Serverless Java and LLMs

on AWS

airhacks.live

airhacks.industries

airhacks.live

"It's not work if you like it"
…so I never worked. #java

airhacks.fm

#185 A Cloud Migration Story: From J2EE to Serverless Java

[episode link] Listen on Apple Podcasts LISTEN ON Spotify Listen on Google Podcasts [RSS]

An airhacks.fm

ZX Spectru
CPC 64, De
in 1993, usi
Lambda, Cl
clouds ther
services, n
quarkus in
the cloud h

#219 Java, CraC and Reducing Cold Start Duration with AWS Lambda SnapStart

[episode link] Listen on Apple Podcasts LISTEN ON Spotify Listen on Google Podcasts [RSS]

An airl

CR
sn
wit
be
Zo
Sn
La
me

#288 Integrating AI with Java: Quarkus and Langchain4j

[episode link] Listen on Apple Podcasts LISTEN ON Spotify Listen on Google Podcasts [RSS]

An airhacks.fm conversation with Dimitris Andreadis (@dandreadis) about:

Dimitris appeared previously on "#64 Quarkus 1.0 and SpringBoot", discussion abou
integrating AI language models (LLMs) with Java applications using quarkus and lar

airhacks.live

airhacks.TV

with the time machine, “100 episodes ago segment”

…any questions left?

youtube.com/
@bienadam/shorts


500
youtube.com/bienadam/shorts

airhacks.live

NEW https://discord.gg/airhacks

airhacks.live

NEW online, live virtual workshops

Continuous coding, explaining, interacting and sharing with Adam Bien

Live, Virtual Online Workshops, Summer 2024:

- Persistence Patterns for Serverless Java on AWS, July, 11th, 2024
- Serverless Generative AI with Java on AWS, July, 25th, 2024

Tickets are also available from: airhacks.eventbrite.com and meetup.com/airhacks

by Adam Bien

You don't like live, interactive virtual workshops? Checkout video courses: airhacks.io

…I started with DevOps in 1995

then continued with serverless computing in 2001…

Java && Clouds?

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

GraalVM™

reduces RAM footprint

https://sites.google.com/view/energy-efficiency-languages/results?authuser=0

Cost Driven Architectures

Configure AWS Lambda Info

Architecture

Arm

Number of requests

2

Unit

per second

Duration of each request (in ms)

Duration is calculated from the time your code begins executing until it returns or otherwise terminates.

100

Amount of memory allocated

Enter the amount between 128 MB and 10 GB

Value

2

Unit

GB

Amount of ephemeral storage allocated

Enter the amount between 512 MB and 10,240 MB. The first 512 MB are at no additional charge, you only pay for any additional storage that you configure for the function.

Value

512

Unit

MB

▶ Show calculations

Provisioned Concurrency Info

Total Upfront cost: 0.00 USD
Total Monthly cost: 15.07 USD

Show Details

Save and view summary

Save and add service

Java && GenAI?

The Perfect (Java) Storm

- https://www.tornadovm.org/
- https://openjdk.org/projects/babylon/
- https://inside.java/2023/08/28/code-reflection/
- https://openjdk.org/projects/panama/ (and jextract)
- https://openjdk.org/projects/valhalla/
- https://openjdk.org/projects/babylon/articles/triton

```
import jdk.incubator.vector.FloatVector;
import jdk.incubator.vector.VectorOperators;
import jdk.incubator.vector.VectorSpecies;

    // build the Transformer via the model .bin file
    Transformer transformer = new Transformer(checkpoint_path);
    if (steps == 0 || steps > transformer.config.seq_len) {
        steps = transformer.config.seq_len; // ovrerride to ~max length
    }

    // build the Tokenizer via the tokenizer .bin file
    Tokenizer tokenizer = new Tokenizer(tokenizer_path, transformer.config.vocab_size);

    // build the Sampler
    Sampler sampler = new Sampler(transformer.config.vocab_size, temperature, topp, rng_seed);

    // run!
    switch (mode) {
        case "generate" -> generate(transformer, tokenizer, sampler, prompt, steps);
        case "chat" -> chat(transformer, tokenizer, sampler, prompt, system_prompt, steps);
        default -> {
            System.err.println("unknown mode: " + mode);
            error_usage();
        }
    }
}
```

https://github.com/mukel/llama2.java/blob/main/Llama2.java

Generative AI

Generative Artificial Intelligence (GAI)

From Wikipedia, the free encyclopedia

> *Not to be confused with Artificial general intelligence.*

Generative artificial intelligence (**generative AI**, **GenAI**,[1] or **GAI**) is artificial intelligence capable of generating text, images, videos, or other data using generative models,[2] often in response to prompts.[3][4] Generative AI models learn the patterns and structure of their input training data and then generate new data that has similar characteristics.[5][6]

Improvements in transformer-based deep neural networks, particularly large language models (LLMs), enabled an AI boom of generative AI systems in the early 2020s. These include chatbots such as ChatGPT, Copilot, Gemini and LLaMA, text-to-image artificial intelligence image generation systems such as Stable Diffusion, Midjourney and DALL-E, and text-to-video AI generators such as Sora.[7][8][9][10] Companies such as OpenAI, Anthropic, Microsoft, Google, and Baidu as well as numerous smaller firms have developed generative AI models.[3][11][12]

LLMs?

Large Language Model LLM

From Wikipedia, the free encyclopedia

In natural language processing (NLP), a **word embedding** is a representation of a word. The embedding is used in text analysis. Typically, the representation is a real-valued vector that encodes the meaning of the word in such a way that the words that are closer in the vector space are expected to be similar in meaning.[1] Word embeddings can be obtained using language modeling and feature learning techniques, where words or phrases from the vocabulary are mapped to vectors of real numbers.

Methods to generate this mapping include neural networks,[2] dimensionality reduction on the word co-occurrence matrix,[3][4][5] probabilistic models,[6] explainable knowledge base method,[7] and explicit representation in terms of the context in which words appear.[8]

Word and phrase embeddings, when used as the underlying input representation, have been shown to boost the performance in NLP tasks such as syntactic parsing[9] and sentiment analysis.[10]

https://en.wikipedia.org/wiki/Large_language_model

Quantization

Quantization

Quantization is a technique to reduce the computational and memory costs of running inference by representing the weights and activations with low-precision data types like 8-bit integer (`int8`) instead of the usual 32-bit floating point (`float32`).

Reducing the number of bits means the resulting model requires less memory storage, consumes less energy (in theory), and operations like matrix multiplication can be performed much faster with integer arithmetic. It also allows to run models on embedded devices, which sometimes only support integer data types.

https://huggingface.co/docs/optimum/en/concept_guides/quantization

LLMs 👉 interaction

LLM

- stateless
- non-idempotent?
- unstable
- concurrent
- fuzzy
- “liquid logic”

LLMs and Microservices

MicroProfile

Telemetry 1.1

Open API 3.1

Rest Client 3.0

Config 3.1

Fault Tolerance 4.0

Metrics 5.1

JWT Authentication 2.1

Health 4.0

Jakarta EE 10 Core Profile

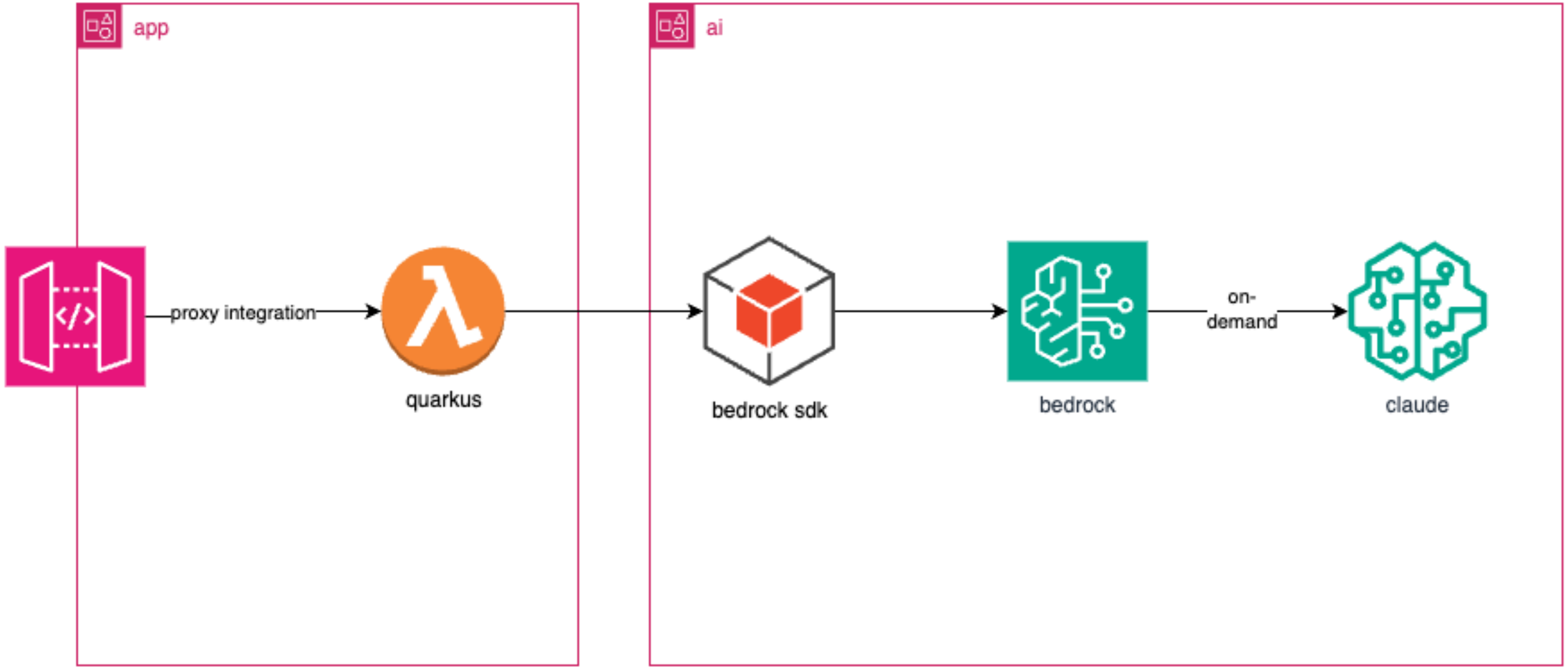
MP Fault Tolerance

- @Asynchronous
- @Bulkhead
- @CircuitBreaker
- @Fallback
- @Retry
- @Timeout
- …and `Thread.sleep` 😀

Amazon Bedrock

Amazon Bedrock

- serverless, fully managed, IAM integration
- models (FMs): AI21 Labs, Anthropic, Cohere, Meta, Mistral AI, Stability AI, and Amazon
- SDK available
- fully encrypted
- also available in e.g. eu-central-1
- on-demand or provisioned
- VPC / private link integration


app
ai
proxy integration
quarkus
bedrock sdk
bedrock
on-
demand
claude

Embedding Models

From Wikipedia, the free encyclopedia

In natural language processing (NLP), a **word embedding** is a representation of a word. The embedding is used in text analysis. Typically, the representation is a real-valued vector that encodes the meaning of the word in such a way that the words that are closer in the vector space are expected to be similar in meaning.[1] Word embeddings can be obtained using language modeling and feature learning techniques, where words or phrases from the vocabulary are mapped to vectors of real numbers.

Methods to generate this mapping include neural networks,[2] dimensionality reduction on the word co-occurrence matrix,[3][4][5] probabilistic models,[6] explainable knowledge base method,[7] and explicit representation in terms of the context in which words appear.[8]

Word and phrase embeddings, when used as the underlying input representation, have been shown to boost the performance in NLP tasks such as syntactic parsing[9] and sentiment analysis.[10]

https://en.wikipedia.org/wiki/Word_embedding

Vector Database

From Wikipedia, the free encyclopedia

A **vector database management system** (**VDBMS**) or simply **vector database** or **vector store** is a database that can store vectors (fixed-length lists of numbers) along with other data items. Vector databases typically implement one or more Approximate Nearest Neighbor (ANN) algorithms,[1][2] so that one can search the database with a query vector to retrieve the closest matching database records.

Vectors are mathematical representations of data in a high-dimensional space. In this space, each dimension corresponds to a feature of the data, with the number of dimensions ranging from few hundreds to tens of thousands, depending on the complexity of the data being represented. A vector's position in this space represents its characteristics. Words, phrases, or entire documents, and images, audio, and other types of data can all be vectorized.[3]

https://en.wikipedia.org/wiki/Vector_database

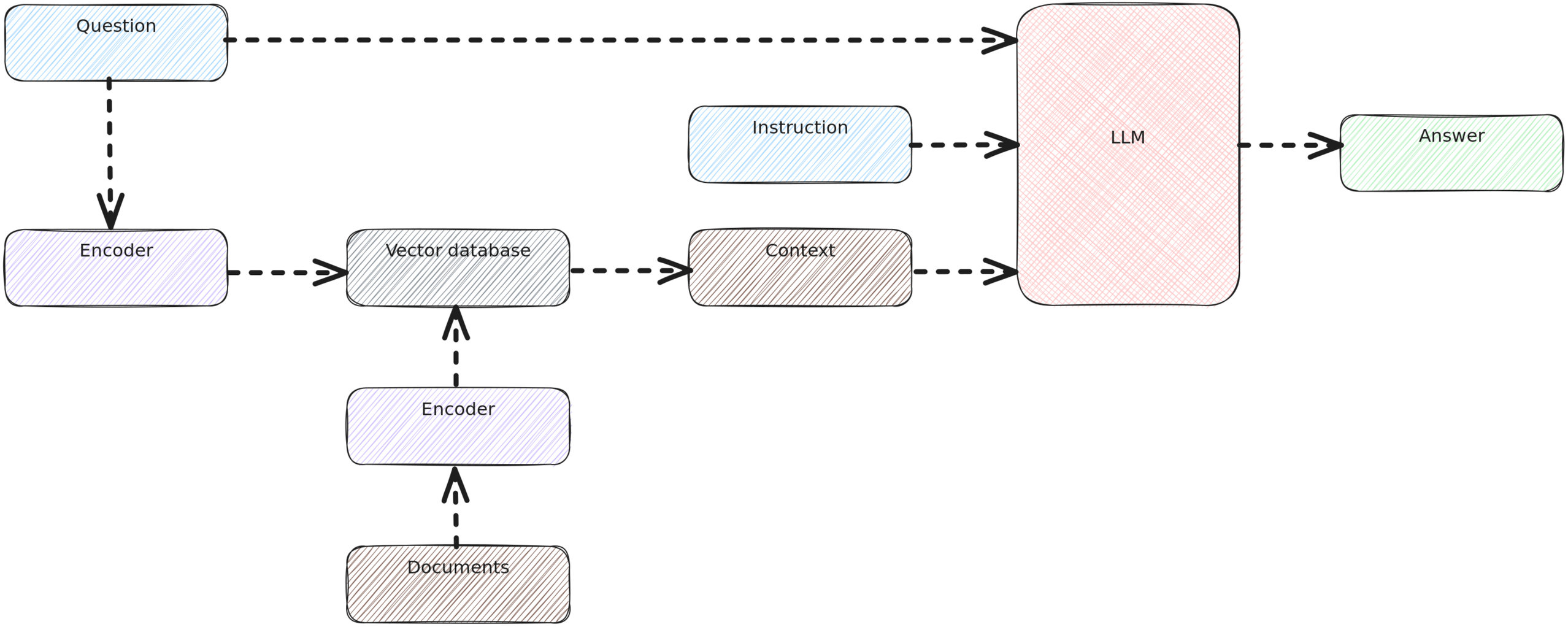
Vector Databases

- jvector: https://github.com/jbellis/jvector
- Cassandra: https://cassandra.apache.org/doc/latest/cassandra/vector-search/concepts.html
- AstraDB: https://www.datastax.com/products/datastax-astra
- pgvector: https://github.com/pgvector/pgvector

Retrieval Augmented Generation (RAG)

RAG

Prompts often contain a few examples (thus "few-shot"). Examples can be automatically retrieved from a database with document retrieval. These databases have multiple formats depending on the use case, including vector database, summary index, tree index, and key-word table index.[48] Given a query, a document retriever is called to retrieve the most relevant (usually measured by first encoding the query and the documents into vectors, then finding the documents with vectors closest in Euclidean norm to the query vector). The LLM then generates an output based on both the query and the retrieved documents,[49] this can be a useful technique for proprietary or dynamic information that was not included in the training or fine-tuning of the model.

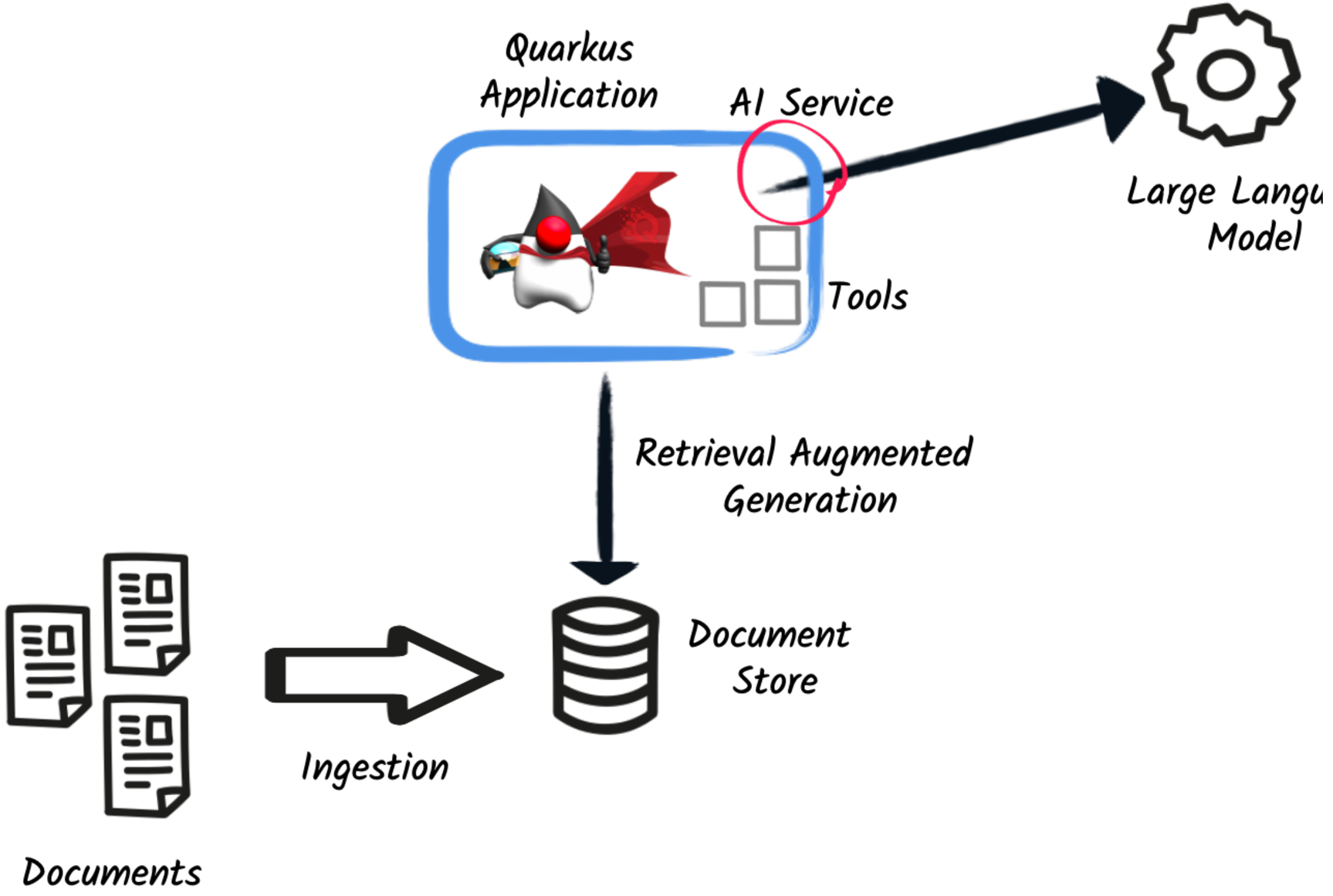


https://en.wikipedia.org/wiki/Prompt_engineering#Retrieval-augmented_generation

LLMs 👉 EAI

RAG 👉 No Hallucinations

langchain4j


Quarkus
Application
AI Service
Large Langu
Model
Tools
Retrieval Augmented
Generation
Document
Store
Ingestion
Documents

Tools

Model Tuning vs. RAG

Local Models

Trends / Future

airhacks.live

NEW online, live virtual workshops

Continuous coding, explaining, interacting and sharing with Adam Bien

Live, Virtual Online Workshops, Summer 2024:

- Persistence Patterns for Serverless Java on AWS, July, 11th, 2024
- Serverless Generative AI with Java on AWS, July, 25th, 2024

Tickets are also available from: airhacks.eventbrite.com and meetup.com/airhacks

by Adam Bien

You don't like live, interactive virtual workshops? Checkout video courses: airhacks.io

Thank YOU!

airhacks.industries

airhacks.live