Testing GenAI apps with Docker

Mes de QA ‘25

About me

You can find me as **@mdelapenya** everywhere

Manuel de la Peña

Staff Software Engineer @ Docker
Computer Science degree, Master in Software Engineering

- Testcontainers Go maintainer since 2020
- Engineering Productivity at Elastic Observability
- QA Tech lead at Liferay Cloud
- Core Engineer at Liferay
- In OSS since 2011
- Hitting keyboards since 1994

What we are going to see today:

01. GenAI in today's software
02. A mental model for testing: Cloud apps
03. Gen AI Tooling in Go
04. Let's talk about tests, b-AI-by!
05. Conclusions

1. GenAI in today's software

GenAI in Today's software

FOMO: Fear Of Missing Out!

Every day there is a new company offering AI services, exposing their models for you to consume them, and new papers are published every day.

- ➔ OpenAI
- ➔ Google
- ➔ Anthropic
- ➔ Meta
- ➔ Mistral
- ➔ DeepSeek
- ➔ ...

The M/L + AI + Data (MAD) landscape

Sources:

https://www.linkedin.com/pulse/ai-landscape-2024-trends-top-startups-leta-capital-orqpe

https://mad.firstmark.com

2011 logos

In 2024

1416

In 2023

578

Newcomers in 2024

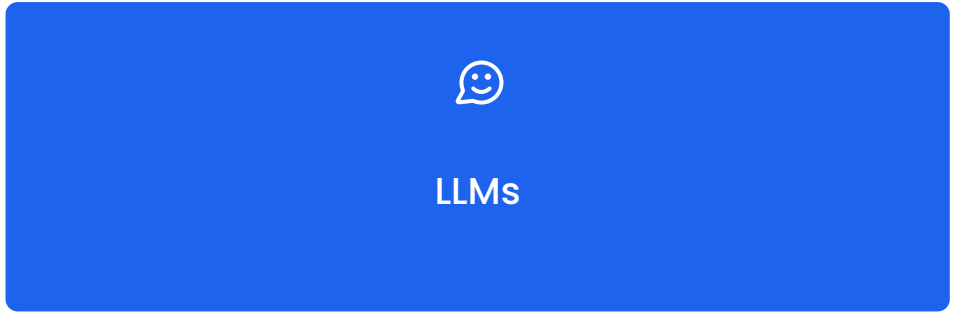
Develop with LLMs

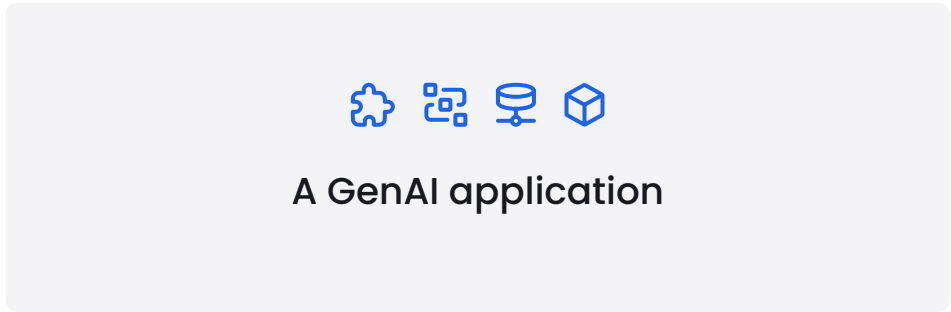
Langchain (Python, Node), **Langchain4j, SpringAI** (Java), **LlamaIndex**, and many more tools:

- Allow you to talk to LLMs
- Design prompts
- Create chats, tools and agents
- Talk to Vector databases

Depending on the model you talk to, you can use it for:

- Image recognition
- Text to text generation
- Text to image/video/audio
- Multimodal generation
- ...more in 3,2,1





2. A mental model for testing: Cloud apps

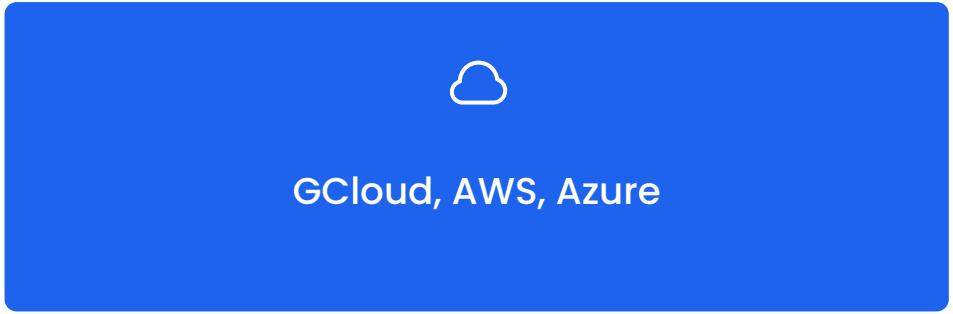
Develop with the Cloud

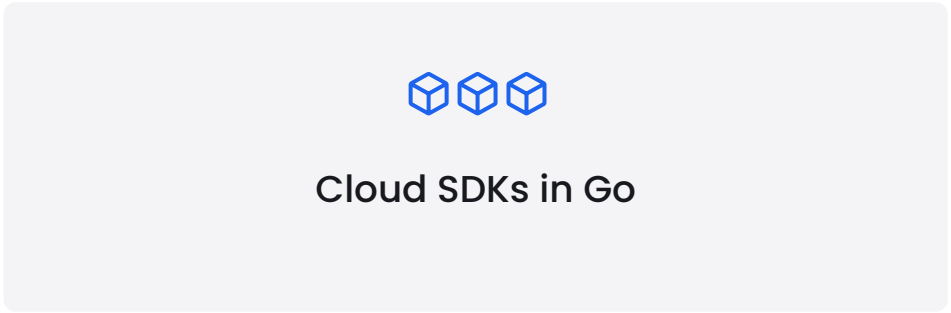
How it works

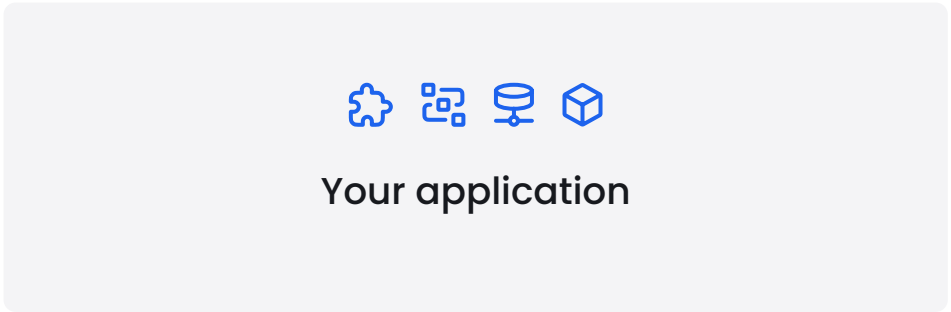
- Our company uses a given Cloud provider
- We setup that Cloud's SDKs into your project
- We configure the credentials
- We start coding...

Seems pretty similar to the LLM approach, doesn't it?

But how do you test these applications?



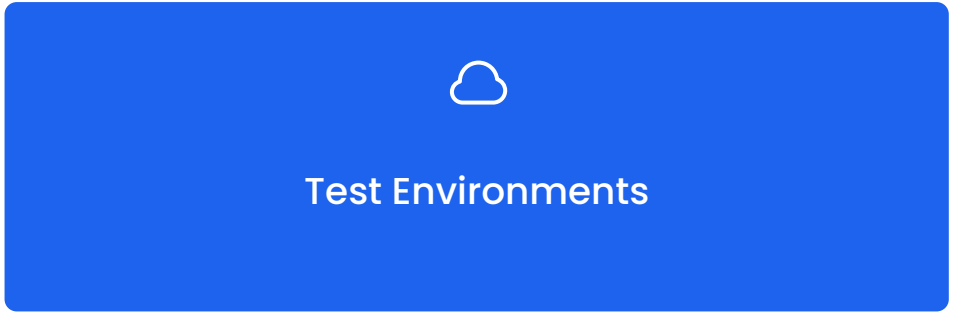


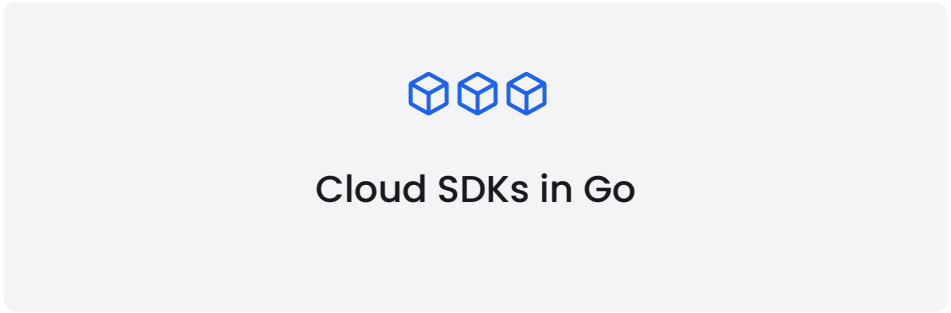


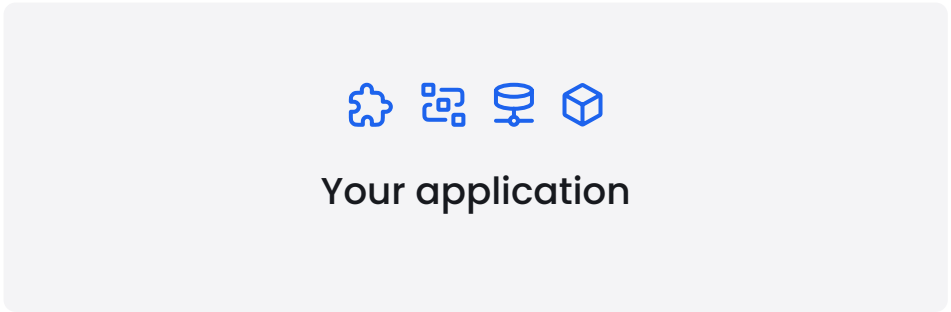
Testing Cloud applications

- ~~No tests, my code is perfect!~~
- Local Service emulating a given Cloud service
- Test environment in the Cloud provider
  - Per team?
  - Per developer?
  - Shared across the company?
  - How long does it take to have them?
  - Do you prune outdated resources?
  - Do you measure costs?

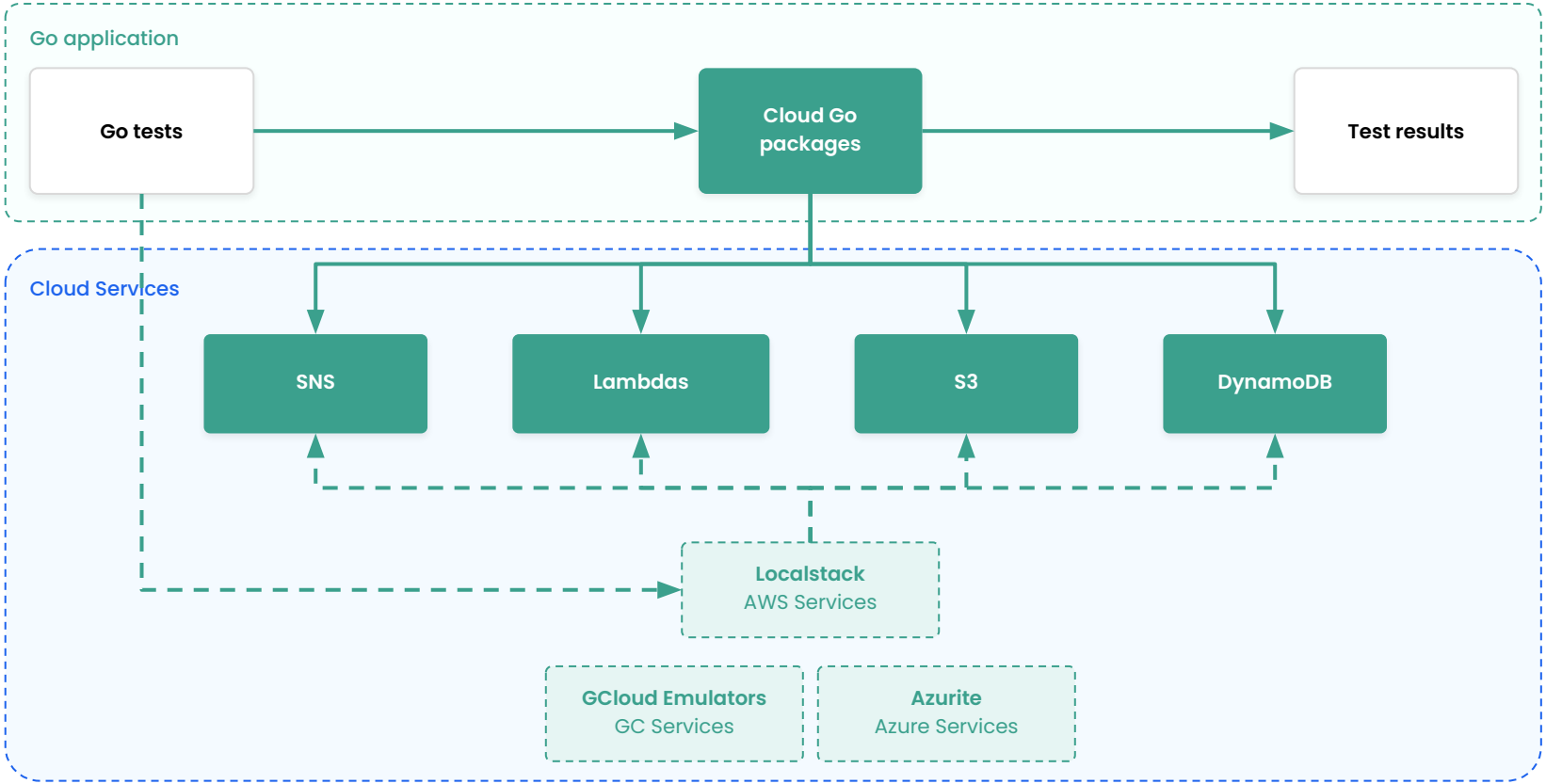
- Do you know Localstack, Google Cloud and Azurite emulators?
  - Standalone applications, or
  - Docker containers.







Testing Cloud applications with emulators



3. Gen AI Tooling in Go

Langchaingo

Go implementation for Langchain: https://github.com/tmc/langchaingo

Community driven project, led by Travis Cline.

- ➔ Generate completions from an LLM (OpenAI, Anthropic, Google...)
- ➔ Calculate embeddings for words, texts, images...
- ➔ Talk to Vector databases to look for similar/relevant documents to augment LLM responses (Retrieval Augmented Generation)
    - ◆ Chroma, Milvus, pgVector, Pinecone, Qdrant, Weaviate...

langchaingo: completions

Create a completion from an LLM, using a streaming function so that the answer is printed at the moment it's produced by the LLM.

It comes with APIs to abstract the LLM creation, obtaining it from multiple providers: Google, OpenAI, Mistral, LlamaFile...:

- The completion code would be exactly the same.

https://github.com/mdelapenya/generative-ai-with-testcontainers/tree/main/02-streamin/main.go

```
// llm is llama3.2:3b
ctx := context.Background()
completion, err := llms.GenerateFromSinglePrompt(
  ctx, llm, "Give me a detailed and long explanation of why
Testcontainers for Go is great",
  llms.WithTemperature(0.8),
  llms.WithStreamingFunc(func(ctx context.Context, chunk []byte)
error {
    fmt.Print(string(chunk))
    return nil
  }),
)
if err != nil {
  log.Fatal(err)
}
```

langchaingo: embeddings

Using the right model, you can generate the embeddings for a text.

Embeddings are dense numerical representation (n-dimensional vectors) of texts, that can be used to calculate similarity between them.

https://github.com/mdelapenya/generative-ai-with-testcontainers/tree/main/06-embeddings/main.go

```
// llm is all-minilm:22m
embedder, err := embeddings.NewEmbedder(llm)
if err != nil {
  return fmt.Errorf("embedder new: %w", err)
}
docs := []string{
  "Testcontainers is a Go package that provides lightweight, throwaway instances of common databases, web browsers, or anything else that can run in a Docker container",
  "Docker is a platform designed to help developers build, share, and run container applications.",
}
vecs, err := embedder.EmbedDocuments(context.Background(), docs)
if err != nil {
  log.Fatal("embed query", err)
}
```

langchaingo: RAG

Retrieval and Augmented Generation.

It's possible to pass a vector of embeddings to a vector database, and leverage the power of these systems to obtain relevant documents to enrich the response from the LLM.

https://github.com/mdelapenya/generative-ai-with-testcontainers/tree/main/07-rag/main.go

```
// llm is all-minilm:22m
embedder, err := embeddings.NewEmbedder(llm)
if err != nil {
  log.Fatalf("embedder new: %w", err)
}
store, err := weaviate.NewStore(context.Background(), embedder)
if err != nil {
  return fmt.Errorf("weaviate new store: %w", err)
}
// ingest relevant documents in the store
if err := ingestion(store); err != nil {
  log.Fatalf("ingestion: %w", err)
}
// similarity search
relevantDocs, err := store.SimilaritySearch(context.Background(),
"What is my favorite sport?", 1, optionsVector...)
if err != nil {
  log.Fatalf("similarity search: %w", err)
}
```

Docker Model Runner

Inference Engine: directly embedded into Docker Desktop

https://docs.docker.com/ai/model-runner/

➔ docker model pull $MODEL

➔ docker model run $MODEL

➔ Currently for Mac with Apple Silicon and Window with NVIDIA/Qualcomm GPUs

➔ Main features:

- OpenAI-compatible APIs
- Package GGUF file as OCI artifacts

Demo:
DMR in action

Testcontainers Go

An Open Source Go package (MIT license) providing developer-friendly API's on top of the Docker engine.

https://github.com/testcontainers/testcontainers-go

- Start, stop, terminate containers and networks
- Wait for containers on custom conditions
- Lifecycle hooks to inject custom code (Pre/Post)
- Copy files to/from containers
- Garbage collection of Docker resources

Go package: docker/docker

Go package: testcontainers-go

Your Go app

Testcontainers Go: modules

Go packages providing access to the most used technologies:

- Relational DBs: Mysql, Postgres, ...
- **Vector DBs: Weaviate, Chroma, Qdrant, Milvus...**
- Non Relational DBs: Elasticsearch, Redis, MongoDB, Neo4j, Opensearch...
- Cloud Emulators: Localstack, Google Cloud, Azurite
- **Inference Engines: Docker Model Runner, Ollama**
- Keycloak, OpenFGA, Vault...
- 60 different Go modules!
- Convenient API specific to each module.

https://www.testcontainers.com/modules

Go package: testcontainers-go

Go packages: testcontainers-go/modules

Your Go app

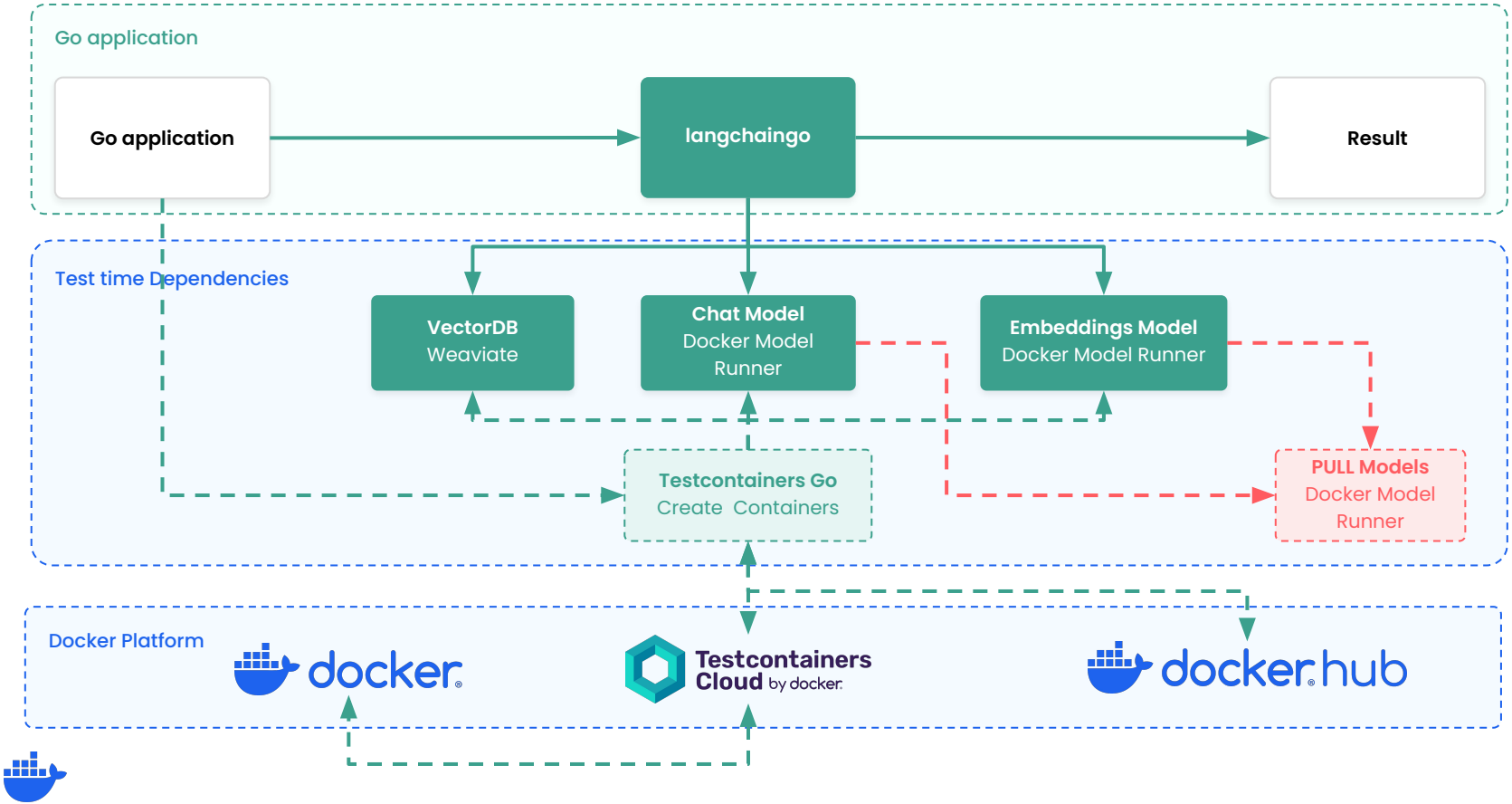
Remember the Docker Model Runner?

A module exists! https://testcontainers.com/modules/dockermodelrunner/?language=go

Since TC Go v0.37.0, it's possible to interact with the Docker Model Runner that is bundled into Docker Desktop (+4.41.0) as it was a container, proxying the requests using a socat container. It can automatically pull models from Docker Hub (and GGUF models from Huggingface!)

```
dmrContainer, err := dmr.Run(ctx, dmr.WithModel("ai/llama3.2:1B-Q4_0"))
if err != nil {
	log.Printf("failed to start container: %s", err)
	return
}
```

Dockerised workflow



4. Let’s talk about tests, b-AI-by!

The application

- An application talking to two models:
  - Raw calls to the model
  - Calls to the same model using RAG
- It uses langchaingo, Docker Model Runner and testcontainers-go
- How can we verify that non-deterministic LLM responses are correct?

https://github.com/mdelapenya/generative-ai-with-testcontainers/blob/main/08-testing

Round 1: strings comparison

https://github.com/mdelapenya/generative-ai-with-testcontainers/blob/main/08-testing

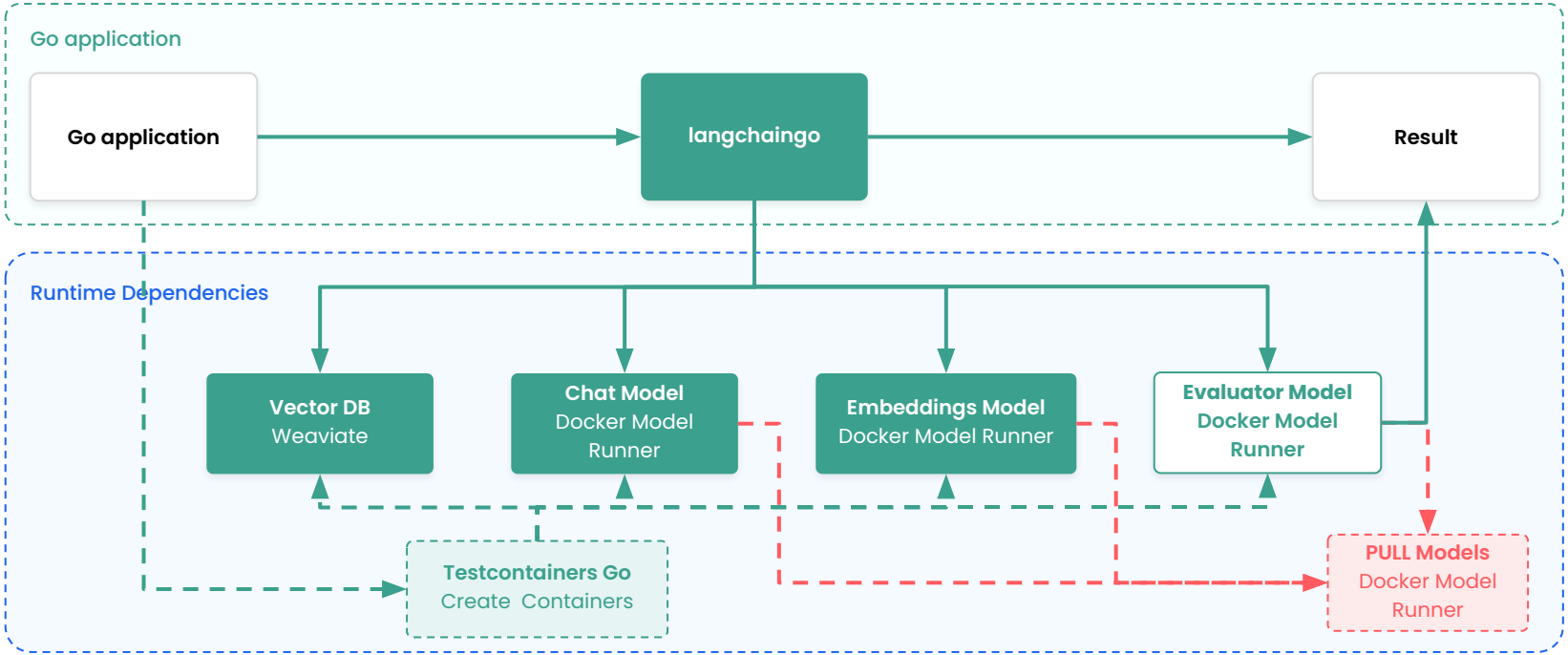
Round 2: cosine similarity

https://github.com/mdelapenya/generative-ai-with-testcontainers/blob/main/08-testing

Enter Evaluators

- AKA “LLM-as-a-Judge” (https://eugeneyan.com/writing/llm-evaluators/).
- Evaluate the quality of another LLM’s response to an instruction or query.
- Define a very strict **System** Prompt:
  - Provide Instructions: e.g. response format
  - Provide reference examples
- Define a very strict **User** Prompt:
  - Provide a detailed format: e.g. *### question ### answer ### reference ###*.
  - Provide a reference (e.g. in the test as an expectation)
  - Structured output, semantic/style constraints
    - Respond with “yes” or “no” including the reasoning.

Adding an Evaluator



Final Round: using an Evaluator

https://github.com/mdelapenya/generative-ai-with-testcontainers/blob/main/08-testing

5. Conclusions

Conclusions

- **Langchaingo**: contribute!
- **Testcontainers Go** + **Docker Model Runner**: a really powerful and easy-to-use combo for local development experience.
- Using **Evaluators** (models with a very strict system prompt) helps us in identifying if the model our application is using responds correctly:
  - Helps us tuning up our application: e.g. choosing a different model, a different vector store, or even modifying the metrics used to classify/correlate the responses at test time.
- Different models can produce different responses:
  - E.g. Llama3.2:3b can excel in one task locally, but its response could be different than using OpenAI + o4 (220b???).
- **Integration tests** will give you enough confidence so you can make progress with speed, but you still need to test against the real thing, e.g. with OpenAI.
  - Run lots of integration tests but you still need to run **some E2E tests** against the real thing!

docker

Thank you!

Mes de QA ‘25

Resources

https://github.com/mdelapenya/generative-ai-with-testcontainers
https://github.com/testcontainers/testcontainers-go
https://github.com/testcontainers/workshop-go
https://docs.docker.com/ai/model-runner/
https://dair-ai.thinkific.com/: Courses on AI

@mdelapenya everywhere