docker

Testing GenAI apps in Go

DevTools Day Bengaluru
Jan 18th, 25

About me

You can find me as **@mdelapena** 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
- First time in India!!! 🇮🇳

I'm STILL learning about GenAI/AI/ML

I’m a software developer in love with software quality: products & workflows.

What we are going to see today:

01. GenAI in today's software
02. The Cloud analogy
03. Gen AI Tooling in Go
04. Testing approach to GenAI
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
- ➔ Mistral
- ➔ ...

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

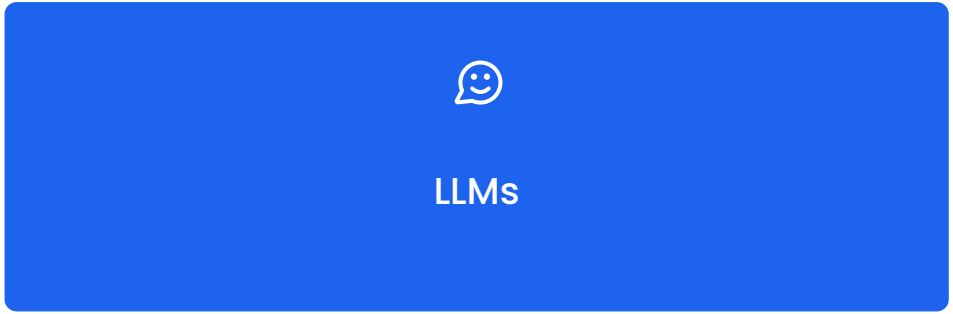
Develop with LLMs

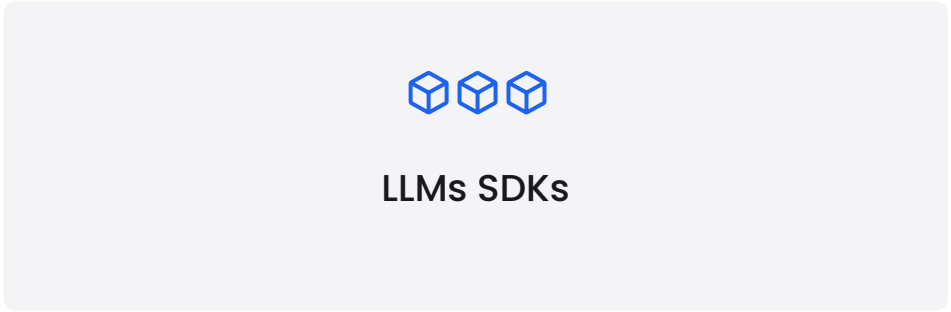
Langchain (Python, Node), **Langchain4j** (Java), **LlamaIndex**, and many more tools:

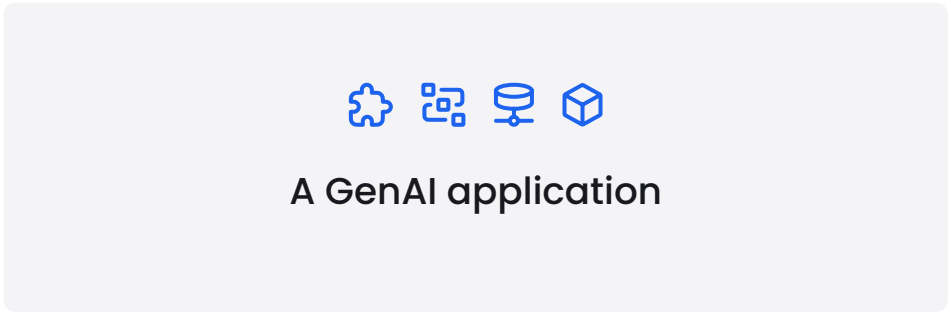
- Allow you to talk to LLMs
- Design prompts
- Create chats, tools and agents
- Talk to Vector databases

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. The Cloud analogy

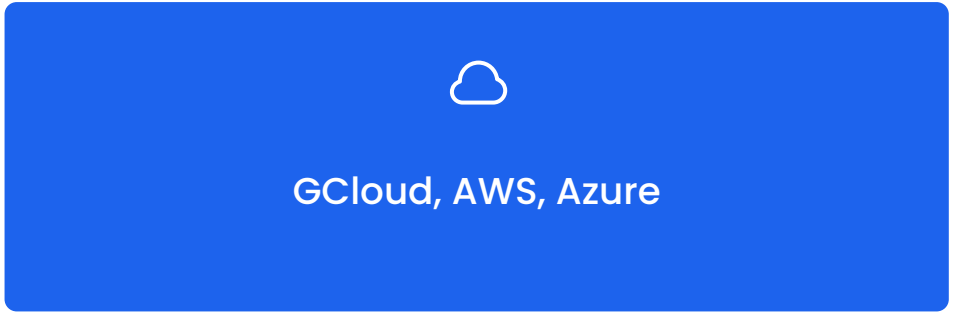
Develop with the Cloud

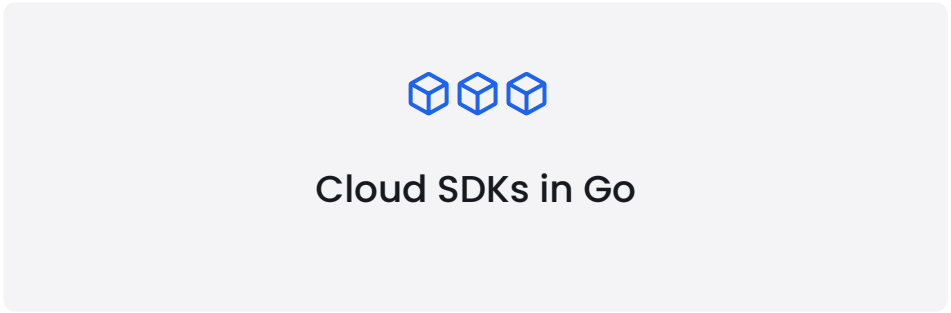
How it works

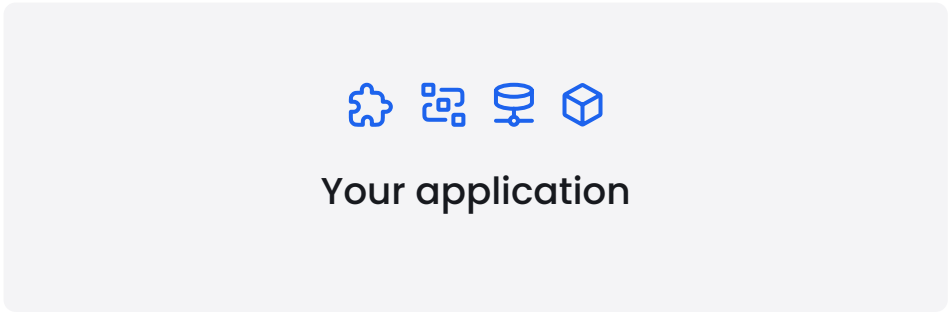
- Our company uses a given Cloud provider
- We setup that Cloud's SDKs into your project
- We configure the credentials
- We start coding...

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

But how do you test these applications?



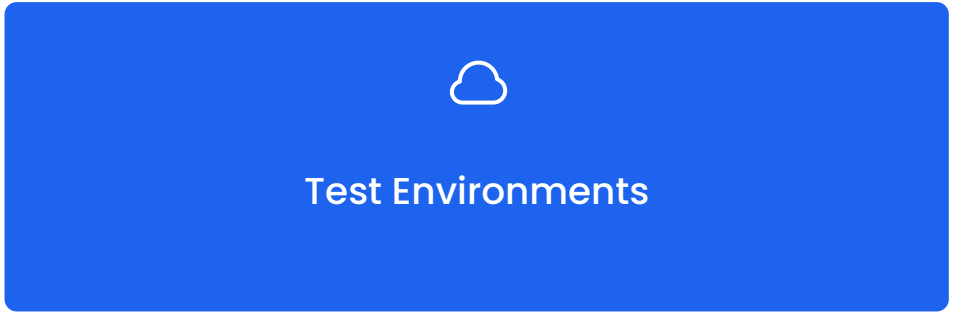


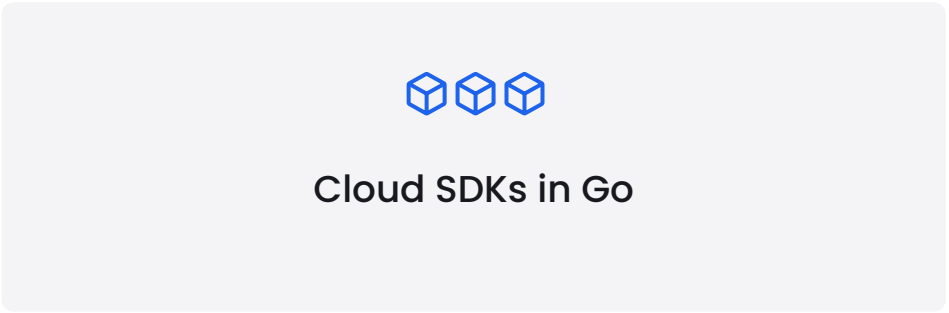


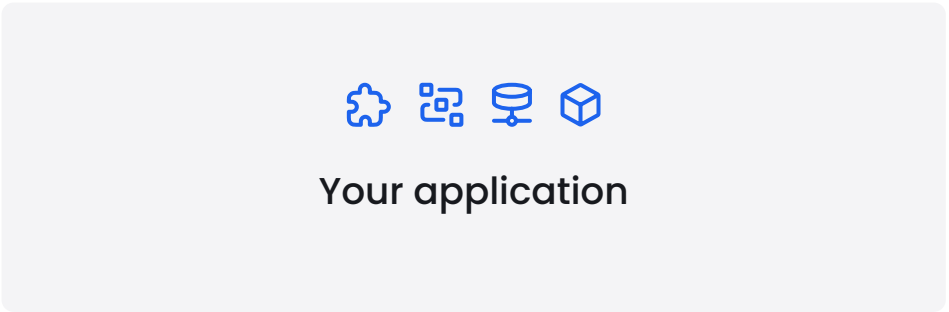
Testing Cloud applications

- 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?

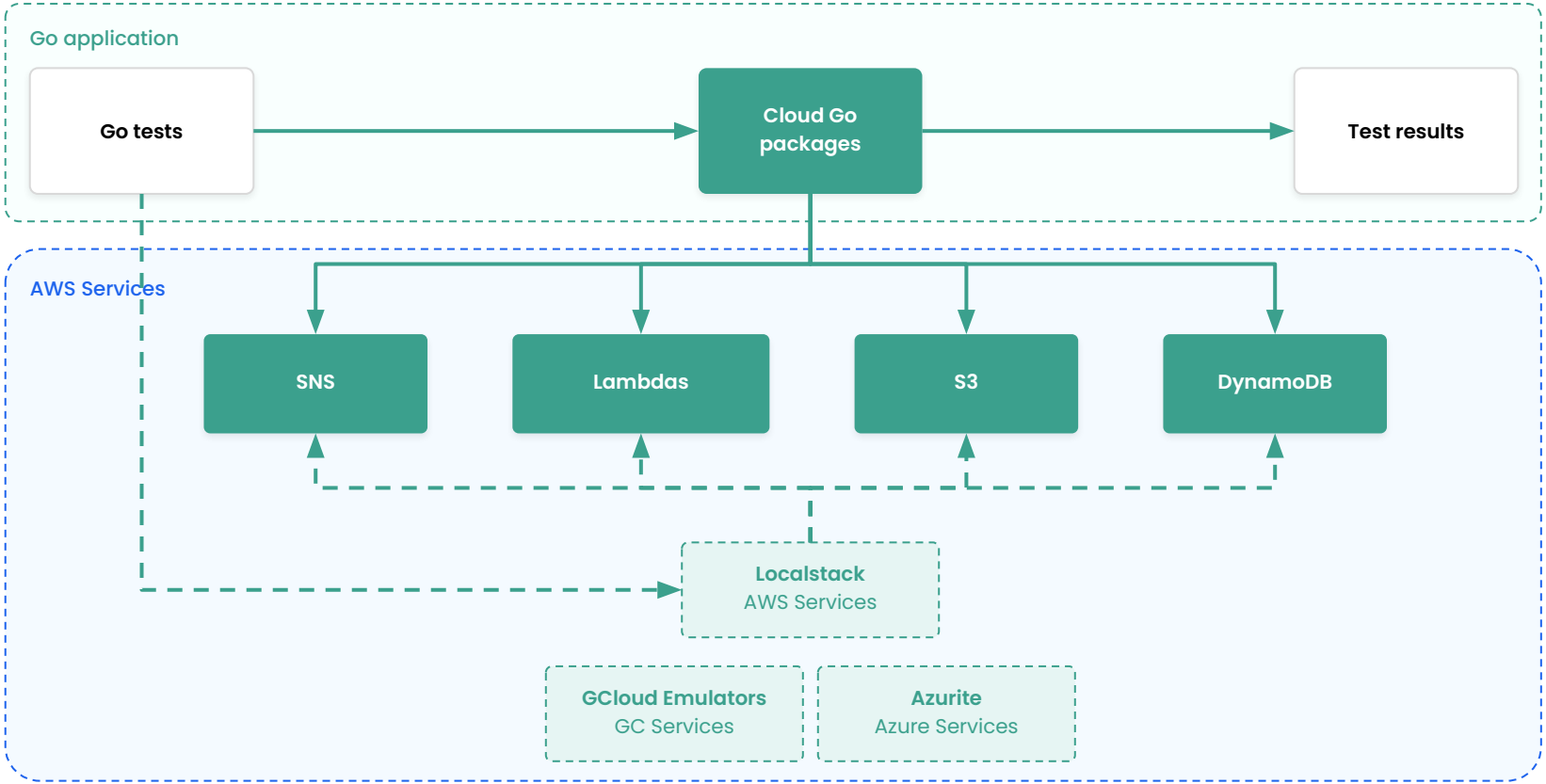
- Do you know Localstack, Google Cloud and Azurite emulators?
  - You can run a Docker container representing those cloud services.
  - They work like a charm!







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 enrich LLM responses (Retrieval Augmented Generation)
    - ◆ Chroma, Milvus, pgVector, Pinecone, Qdrant, Weaviate...

- ➔ 3 DEMOS

langchaingo: completions

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

It comes with APIs to abstract the LLM creation and obtain if 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 (vectors) of words, phrases or concepts, 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.

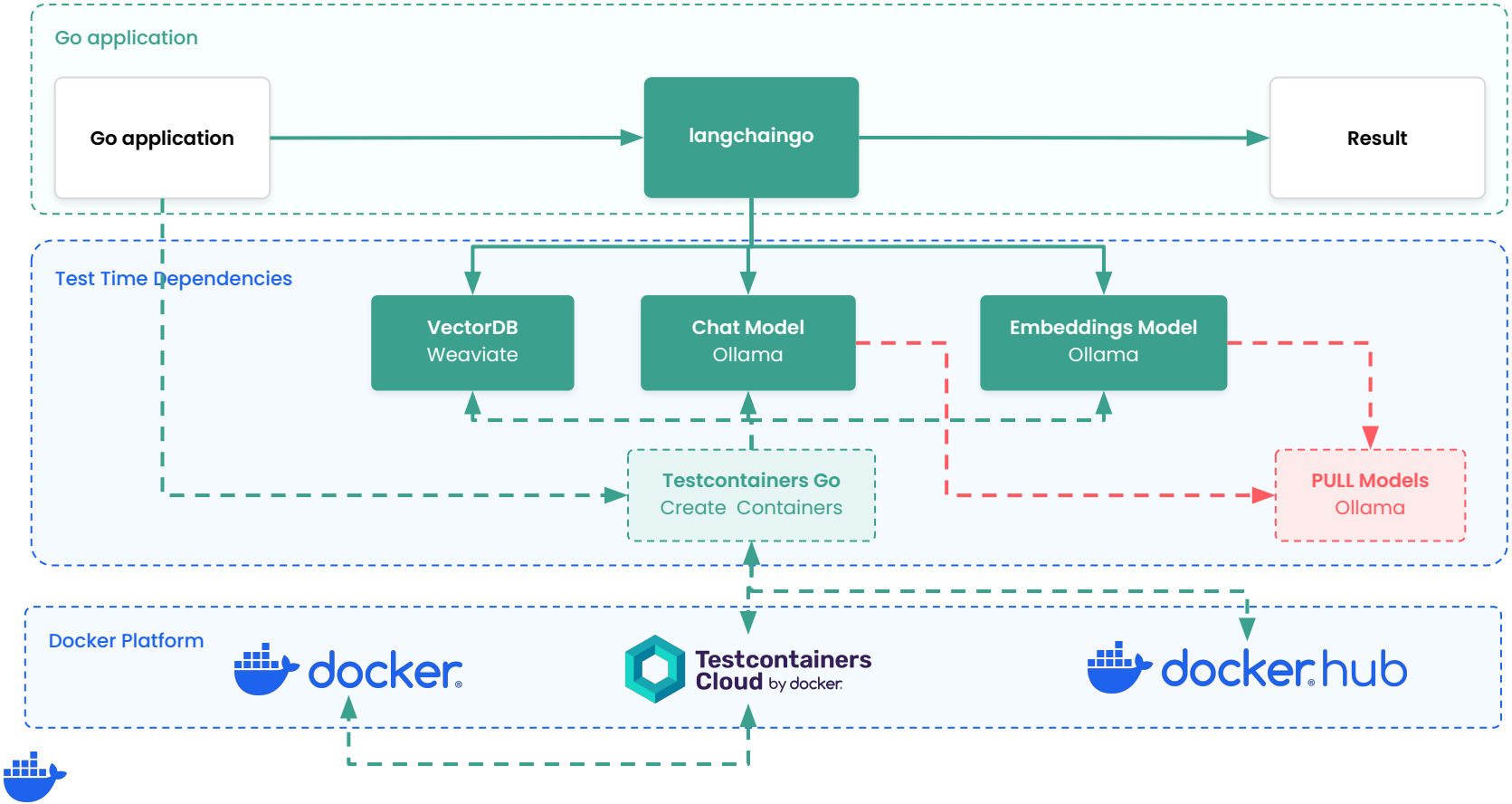
It's possible to pass a vector of embeddings to a vector database, and leverage the power of these systems to obtain relevant documents to enrich the response from the LLM.

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)
}
```

How did those examples work?

Dockerised workflow



Ollama

Inference Engine: https://github.com/ollama/ollama

Like Docker, but for running models! Wrapper of Llama.cpp written in Go

- ➔ ollama pull $MODEL
- ➔ ollama run $MODEL
- ➔ It has Modelfiles for customising models (temperature, system messages…)
- ➔ It has a native app for Linux, Mac and Windows…
- ➔ … or you can run it as a Docker container
  - ◆ used in the demos, thanks to https://github.com/mdelapenya/dockerize-ollama-models/
  - ◆ https://hub.docker.com/u/mdelapenya

➔ Ollama has native access to the GPUs of the host, which is key for speed.

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

GenericContainer

Creates a container from an image, exposing the container port in a random free port in the host.

** Wait strategies live in the "wait" package.*

```
pgCtr, err := testcontainers.GenericContainer(ctx,
testcontainers.GenericContainerRequest{
  ContainerRequest: testcontainers.ContainerRequest {
    Image: "postgres:14",
    ExposedPorts: []string{"5432/tcp"},
    WaitingFor: wait.ForLog("database system is ready to accept
connections").WithOccurrence(2),
  },
  Started: true,
})
if err != nil {
  log.Print("Container failed to start")
  return
}
defer func() {
  if err := testcontainers.TerminateContainer(ctx, pgCtr); err != nil {
    log.Print("Container failed to start")
    return
  }
}
// test my stuff
```

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**: Ollama
- Keycloak, OpenFGA, Vault...

Go package: testcontainers-go

Go packages: testcontainers-go/modules

Your Go app

With modules!

Ready-to-use Go packages wrapping GenericContainer with APIs specific to the underlying technology.

Package: “…/modules/postgres”

https://testcontainers.com/modules

```go
// pgCtr, err := postgres.Run(ctx, "postgres:14")
pgCtr, err := postgres.Run(ctx, "postgres:14",
  postgres.WithDatabase("my-database"),
  postgres.WithUsername("gopher"),
  postgres.WithPassword("p4ssw0rd!"),
  postgres.WithInitScripts("testdata/sql/init.sql"),
  postgres.BasicWaitStrategies(),
)
if err != nil {
  log.Print("Container failed to start")
  return
}
defer func() {
  if err := testcontainers.TerminateContainer(ctx, pgCtr); err != nil {
    log.Print("Container failed to start")
    return
  }
}
// test my stuff
conn, err := pgCtr.ConnectionString(ctx, "ssl=disabled")
```

Remember Ollama?

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

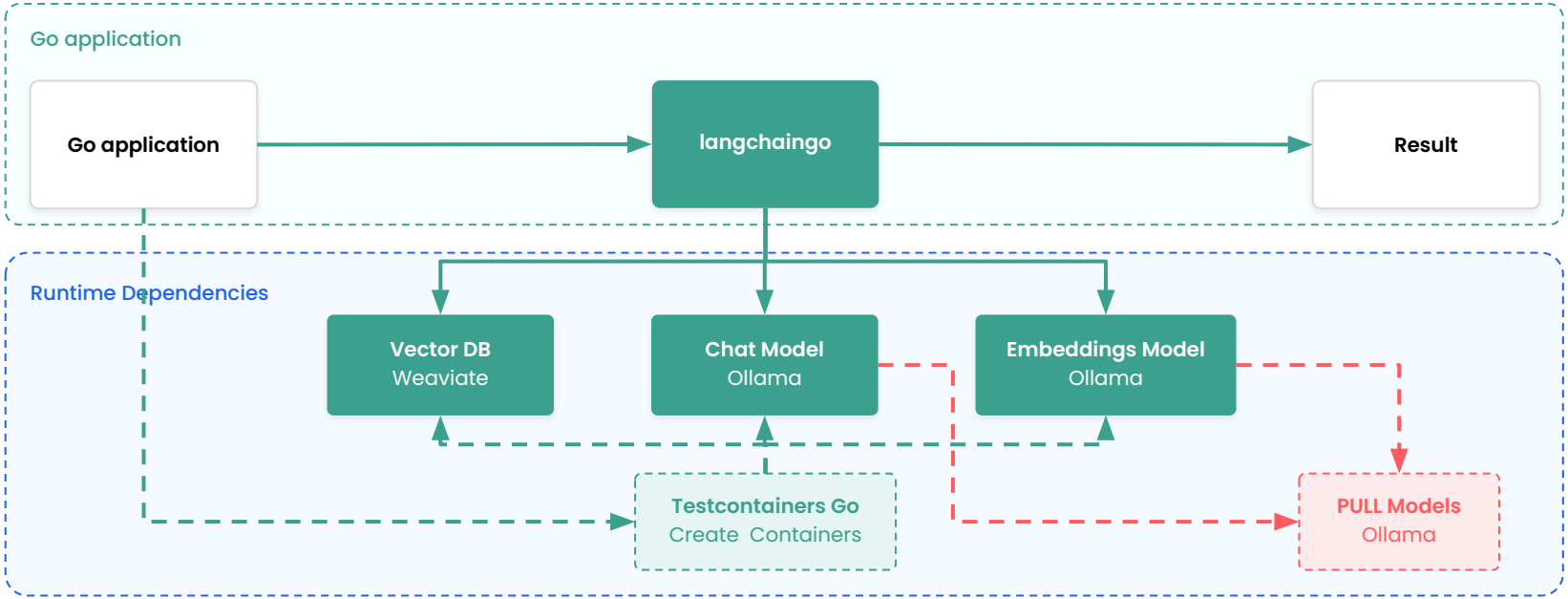
Since v0.35.0, it's possible to interact with the local Ollama process as it was a container, honoring the create/start/stop/terminate container lifecycle.

```
ollamaContainer, err := ollama.Run(ctx, "ollama/ollama:0.5.4")
if err != nil {
      log.Printf("failed to start container: %s", err)
      return
}
code, reader, err := ollamaContainer.Exec(ctx, []string{"ollama", "pull", "all-minilm"})
```

4. Testing approach to GenAI

Could Ollama be considered the Localstack for LLMs?

Please remember...



Demo 1: strings comparison

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

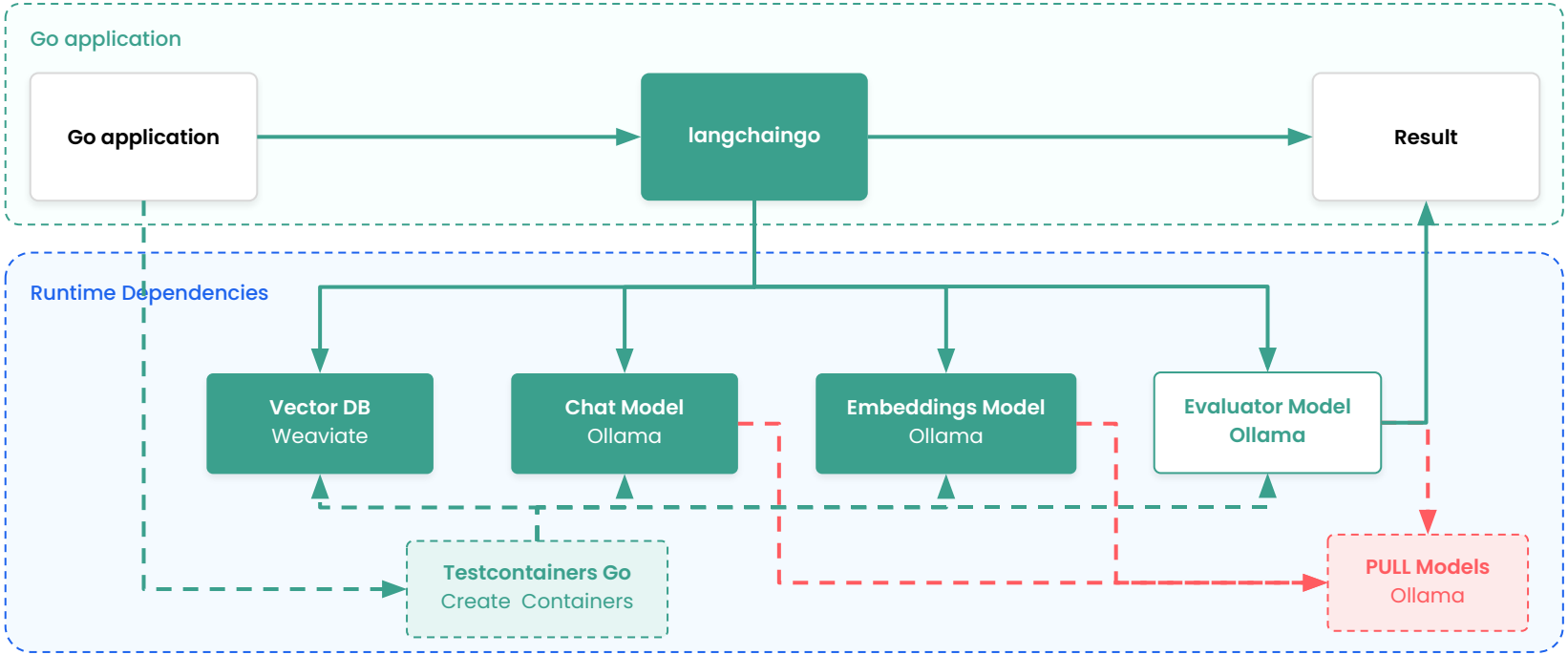
Demo 2: cosine similarity

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: response format,
  - Provide reference examples
- Define a very strict **User** Prompt:
  - Provide a detailed format: *### 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



Demo 3: using an Evaluator

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

Considerations

- ➔ Testcontainers Go + Ollama: a really powerful and easy-to-use local development experience.
- ➔ Using specialised models with a very strict system prompt helps us in identifying if the model our application is using responds correctly:
  - ◆ We can automate the test execution
  - ◆ Adjust our application based on that: e.g. choosing a different model, vector store, or even modifying the metrics used to classify/correlate the responses at test time.
- ➔ Each model has its own idiosyncrasies, so models from different providers can produce different responses.
  - ◆ E.g. Ollama + Llama3.2:3b can excel in one task, but its response could be different than using OpenAI + o4.
- ➔ Integration tests will give you 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 don't forget to add some E2E tests.

5. Conclusions

Conclusions

- ➔ We have very powerful tools in the Go ecosystem to work with LLMs
  - ◆ **Langchaingo** is becoming the reference for that in Go
- ➔ We can use local models to interact with LLMs thanks to **Ollama**
  - ◆ Pull and Run models, even from Huggingface!
  - ◆ Replicate with *enough* confidence what external services can do.
- ➔ **Testcontainers Go** can provide the runtime dependencies in a programmatic manner, enabling a local development experience that comes with increased trust and development speed.
- ➔ **Evaluators** with a very consistent system prompt can help in enhancing your testing activities.

docker

शुक्रिया Bengaluru!!

DevTools Day Bengaluru
Jan 18th, 25