

# Integrando Aplicação

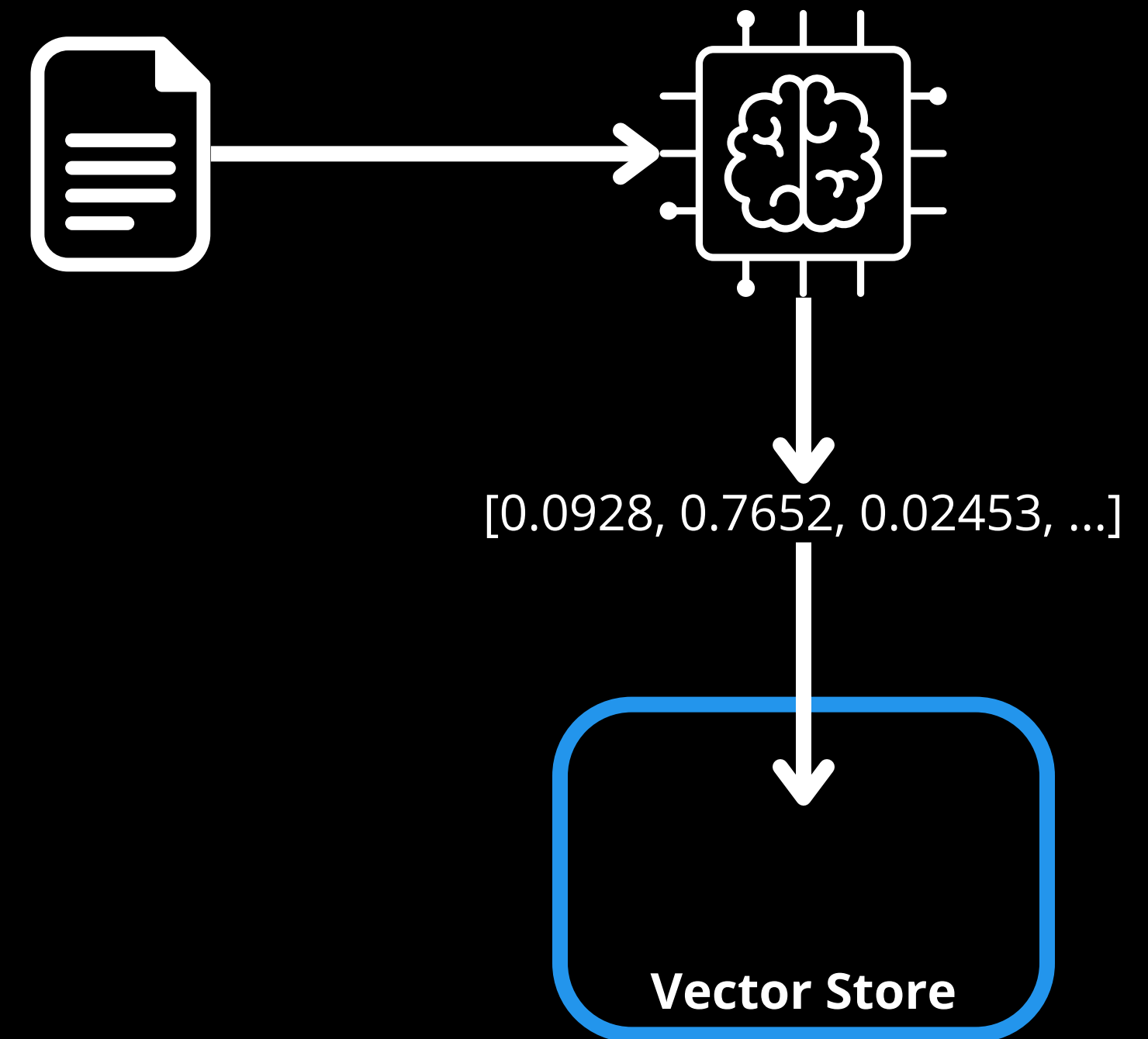


# Objetivo

- Criar chat para conversar sobre um documento
  - Usar streamlit para criar uma interface web que permita subir um documento para o servidor
  - Após subir o documento devemos ser capazes de fazer perguntas sobre ele para o modelo, que o recebe como contexto
-

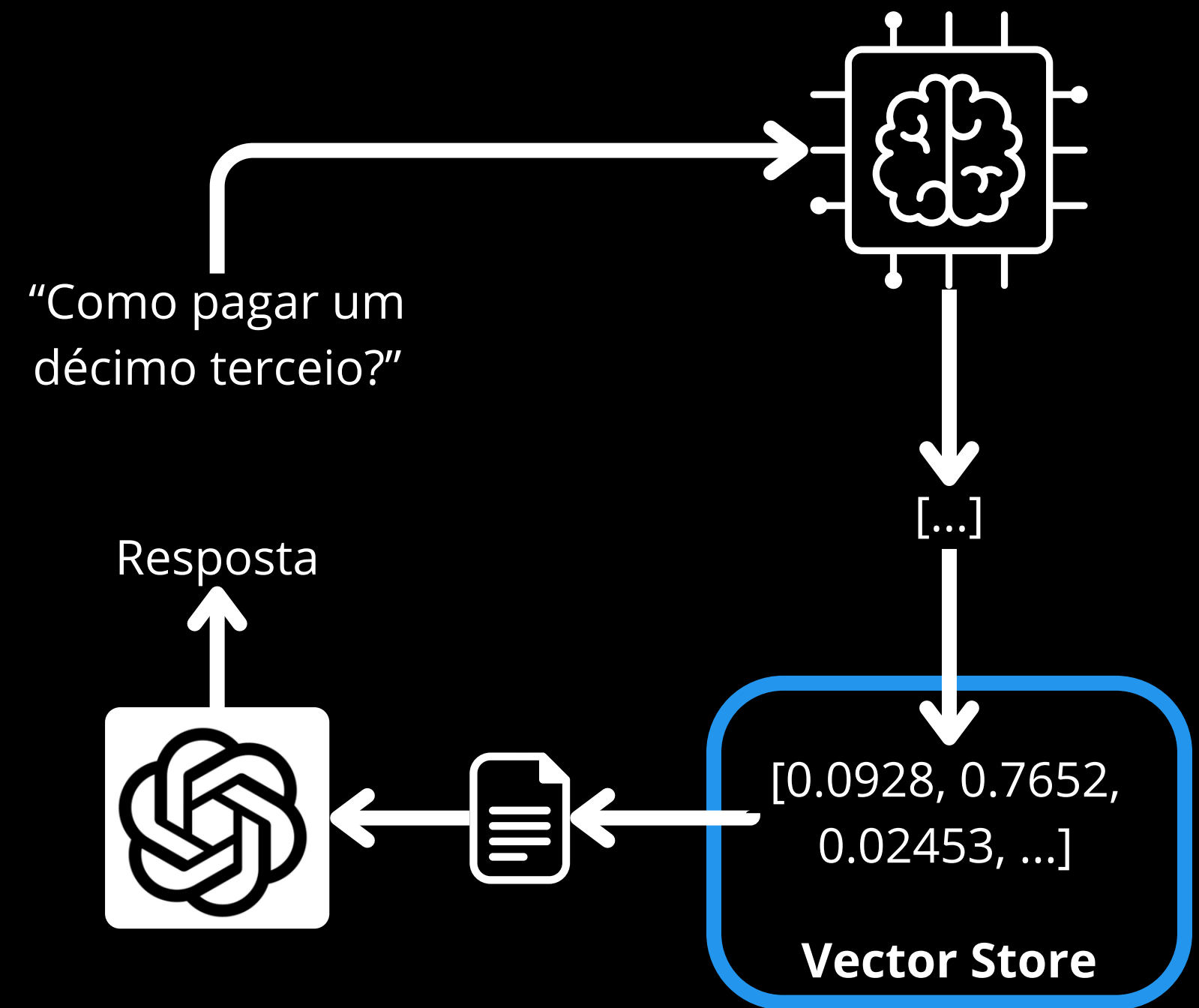
# RAG: Retrieval-Augmented Generation

- Metodologia que permite recuperar dados de um ou vários documentos
- Utiliza um modelo de embeddings para armazenar dados em banco vetorial
- Mesmo modelo transforma pergunta do usuário para conseguir pesquisar no banco os documentos que mais se aproximam dea para responder



# RAG: Retrieval-Augmented Generation

- A recuperação de documentos é feita usando a similaridade entre os vetores gerados
- Os documentos com vetores mais similares são recuperados e enviados como contexto para o modelo que gera a resposta



# Stack



Streamlit

**Interface**



**Chroma**

**Vector  
Store**



LangChain

**Text  
Tools**



Qwen

**Modelo**

---

# Aplicações: vLLM package

- Arquivo app.py possui a importação do package do vLLM com o modelo
- Útil para criar aplicação única e desenvolver rapidamente

```
1  import streamlit as st
2  import os
3  from PyPDF2 import PdfReader
4  import chromadb
5  from vllm import LLM
6  import numpy as np
7
8  llm = LLM(model="meta-llama/Llama-2-7b-chat-hf", gpu_memory_utilization=0.9, max_model_len=752)
9
10 response = llm.generate("Oi, pode me ajudar?")
```



Interface



Modelo

# Aplicações: vLLM package

- **Problema:** A aplicação demora para subir
- Sempre que a aplicação é inicializada, o modelo deve ser baixado (se não salvo em cache) e carregado para memória principal
- Dificulta desenvolvimento: alterações na interface ou backend demoram para atualizar
- Dificulta deploy: subir feature ou corrigir bugs terá maior tempo de propagação
- Nosso caso já possui o modelo rodando, então não precisamos subir outra instancia



Interface



Modelo

---

# Aplicações: vLLM client

- Implementa chamada HTTP para API do vLLM que está rodando na porta 8000
- Aplicação não precisa subir o modelo novamente
- Prática **modulariza** componentes da aplicação

```
def get_answer_from_llm(question, context):  
    prompt = prompt_template.format(context=context, question=question)  
    print("prompt", prompt)  
    url = "http://localhost:8000/v1/chat/completions"  
    headers = {"Content-Type": "application/json"}  
    payload = {  
        "model": "clibrain/Llama-2-7b-ft-instruct-es-gptq-4bit",  
        "messages": [  
            {"role": "system", "content": "Você é um assistente útil."},  
            {"role": "user", "content": prompt},  
        ],  
        "max_tokens": 512  
    }  
}
```



Streamlit

Interface

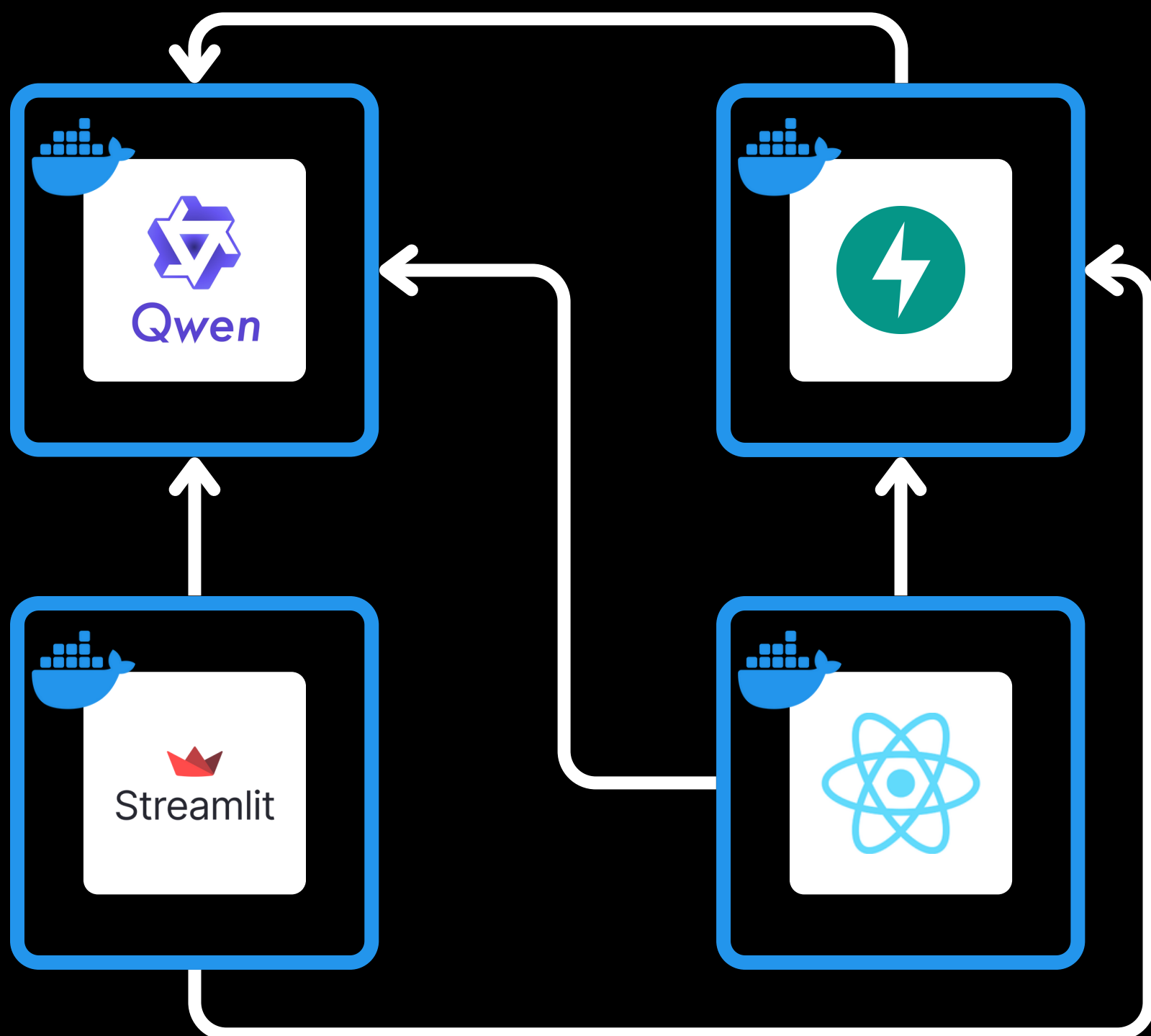


Qwen

Modelo



# Aplicações: Monolítico vs Microserviço



**O que são microserviços? | AWS**

Microserviços são uma abordagem arquitetônica e organizacional do desenvolvimento de software na qual o software consiste em pequenos serviços independentes que se comunicam usando APIs bem definidas....

 Amazon Web Services

# Aplicações: vLLM client

- O cliente precisa esperar toda a resposta ficar pronta para conseguir algum retorno
- Diferente de uma geração de imagem, não é necessário de um texto inteiro para começar a interpretá-lo



Streamlit

**Interface**

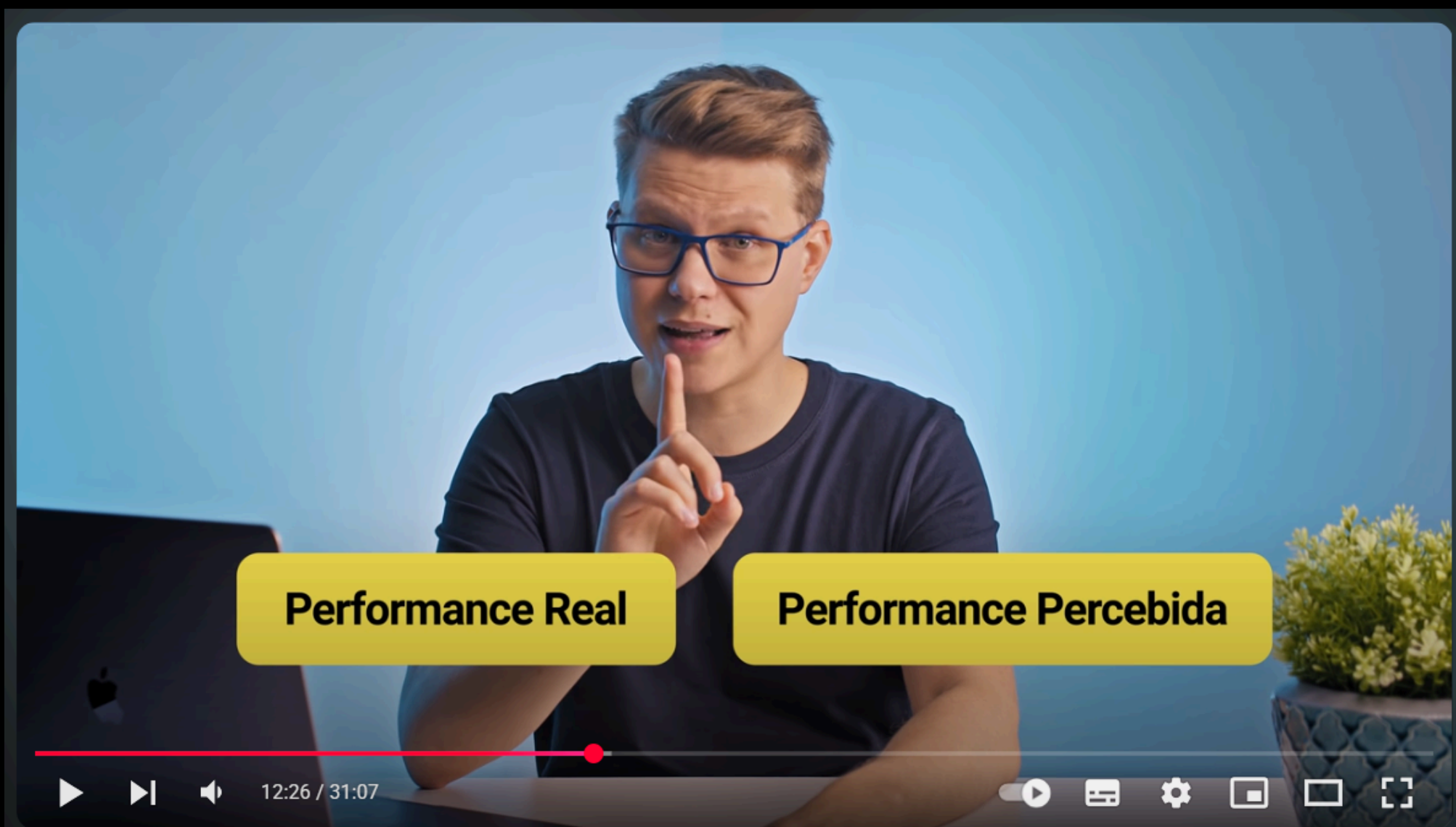


Qwen

**Modelo**

---

# Aplicações: Real vs Percebido



Performance Real

Performance Percebida

Como Eu Programo e Hospedo Sites da Forma Mais Moderna que Existe [GUIA DEFINITIVO]

Filipe Deschamps ✓  
800 mil inscritos

Seja membro Inscrever-se

45 mil

Compartilhar


500 mil visualizações há 4 anos

Duas aplicações podem gerar respostas no mesmo intervalo de tempo, mas se você percebe os resultados de uma primeiro, essa é considerada mais rápida

<https://www.youtube.com/watch?v=EW7m2WlvFgQ>

# Aplicações: Stream

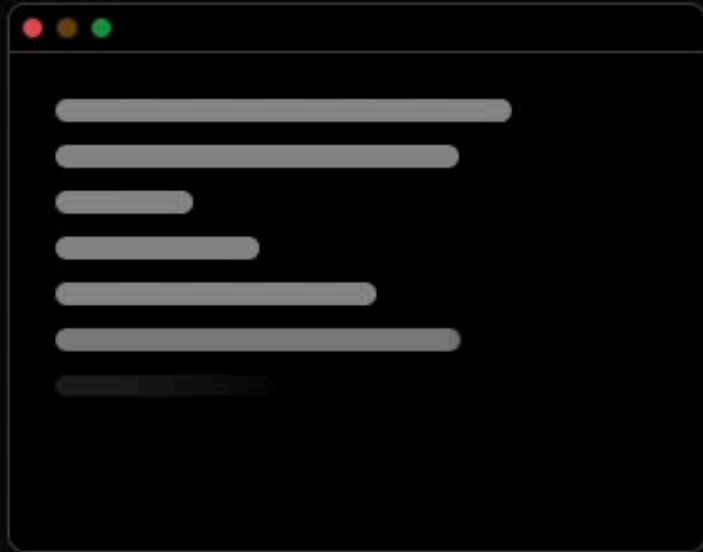
- Conceito amplo para definir um fluxo de dados contínuo



A diagram of a web browser window with a single, empty rectangular area, representing a blocking user interface where no content is visible until the full response is received.

**Blocking UI**

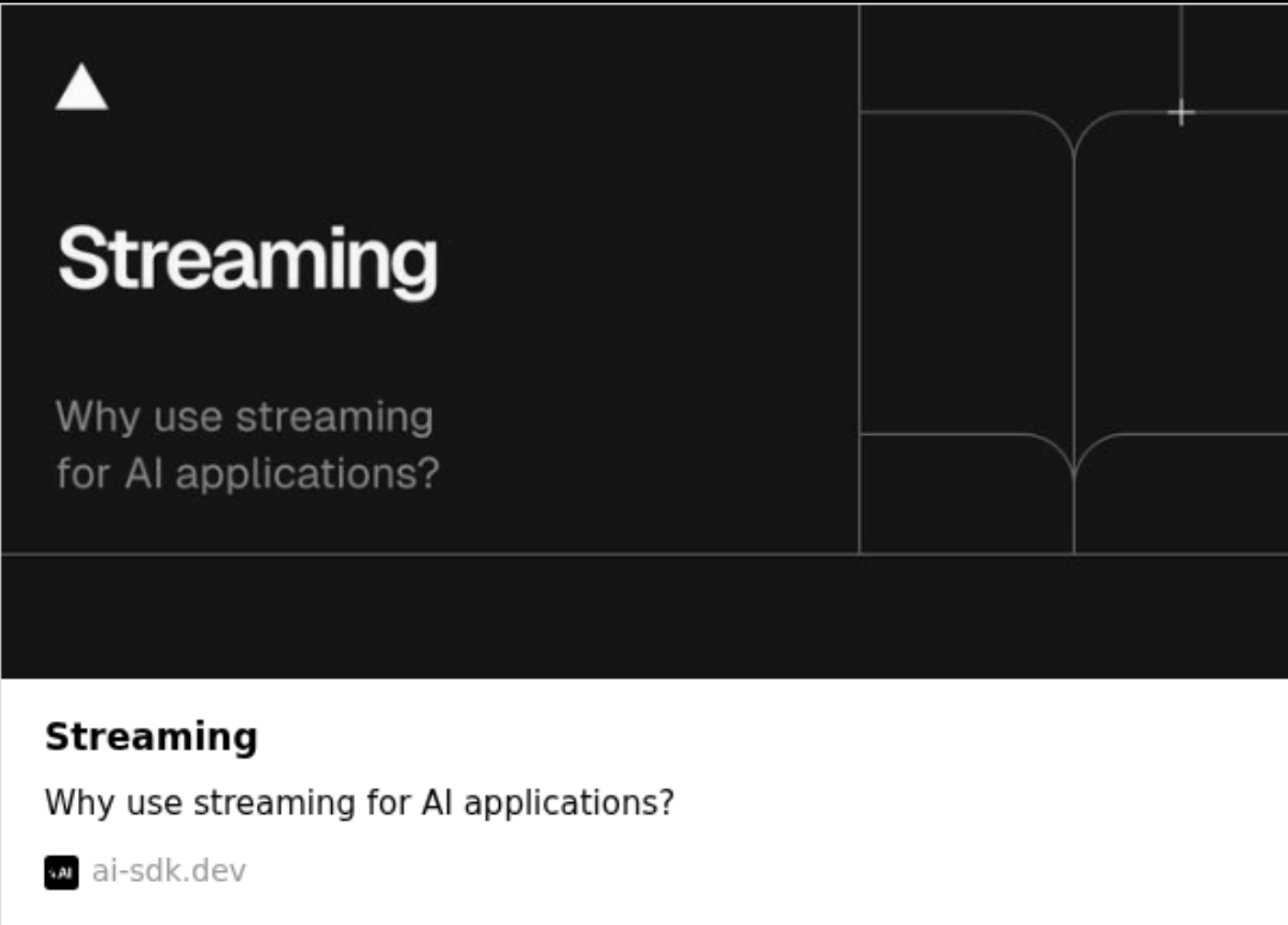
Blocking responses wait until the full response is available before displaying it.



A diagram of a web browser window showing multiple horizontal bars of varying lengths, representing a streaming user interface where content is displayed incrementally as it becomes available.

**Streaming UI**

Streaming responses can transmit parts of the response as they become available.




A presentation slide titled 'Streaming' with a subtitle 'Why use streaming for AI applications?'. The slide features a dark background with a white title and subtitle. A small white triangle is in the top left corner. The bottom of the slide has a white footer with the text 'Streaming' and 'Why use streaming for AI applications?' followed by a small logo and the URL 'ai-sdk.dev'.

**Streaming**

Why use streaming for AI applications?

**Streaming**

Why use streaming for AI applications?

 ai-sdk.dev

# Aplicações: Stream

- Aplicação modularizada e com consumo feito em stream para melhoria do desempenho percebido em app\_stream.py

```
def get_answer_from_llm(question, context):
    prompt = prompt_template.format(context=context, question=question)
    print("prompt", prompt)
    url = "http://localhost:8000/v1/chat/completions"
    headers = {"Content-Type": "application/json"}
    payload = {
        "model": "unsloth/Qwen3-8B-bnb-4bit",
        "messages": [
            {"role": "system", "content": "Você é um assistente útil."},
            {"role": "user", "content": prompt},
        ],
        "stream": True,
        "max_tokens": 2048
    }
    response = requests.post(url, headers=headers, data=json.dumps(payload), stream=True)
    if response.status_code == 200:
        for line in response.iter_lines():
```



Streamlit

**Interface**



Qwen

**Modelo**



# Aplicações: Stream

Digite sua pergunta:

Sobre o que é esse documento?

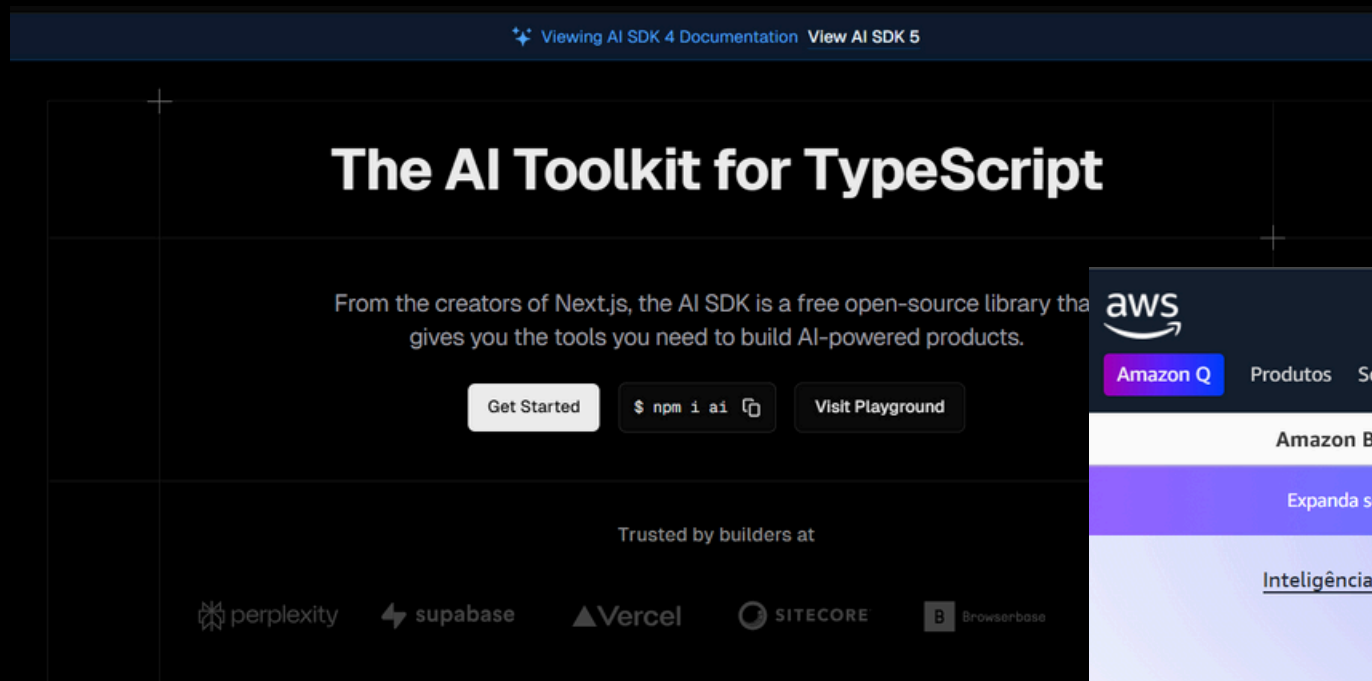
O documento fornecido parece ser um relatório de projeto que descreve uma solução para melhorar o desempenho de algoritmos de processamento de dados massivos. O documento discute diferentes técnicas de paralelismo e clusterização, e como elas podem ser utilizadas para reduzir o tempo de processamento necessário.

O relatório apresenta uma metodologia para implementar essas técnicas em grandes conjuntos de dados, e como isso pode melhorar o desempenho dos algoritmos. Além disso, o documento discute a possibilidade de utilizar frameworks e bibliotecas existentes para implementar essas técnicas, como CUDA, OpenMP, Apache Spark, etc.

Em resumo, o documento parece ser um relatório detalhado sobre uma solução tecnológica para melhorar o desempenho de algoritmos de processamento de dados massivos, incluindo a descrição da metodologia e dos recursos utilizados para implementá-la.

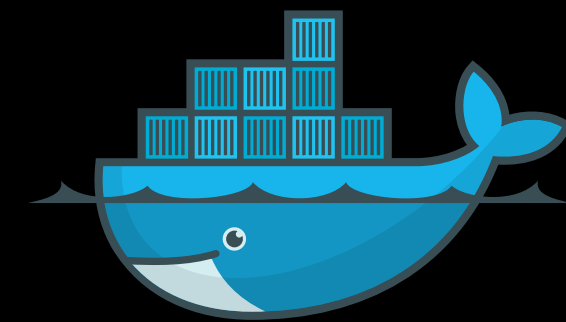
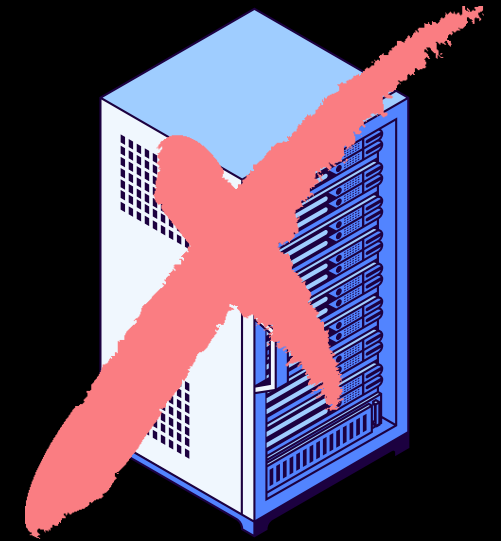
---

# Aplicações: Suporte



# Aplicações: Realidade

- Muitas aplicações feitas para escala não utilizam servidores alocados
- Serverless e auto scaling foram muito adotados nos últimos anos devido ao aproveitamento de recursos e possibilidade de escalar automaticamente de acordo com a demanda
- Mas recursos de GPU ainda não eram inclusos até pouco tempo



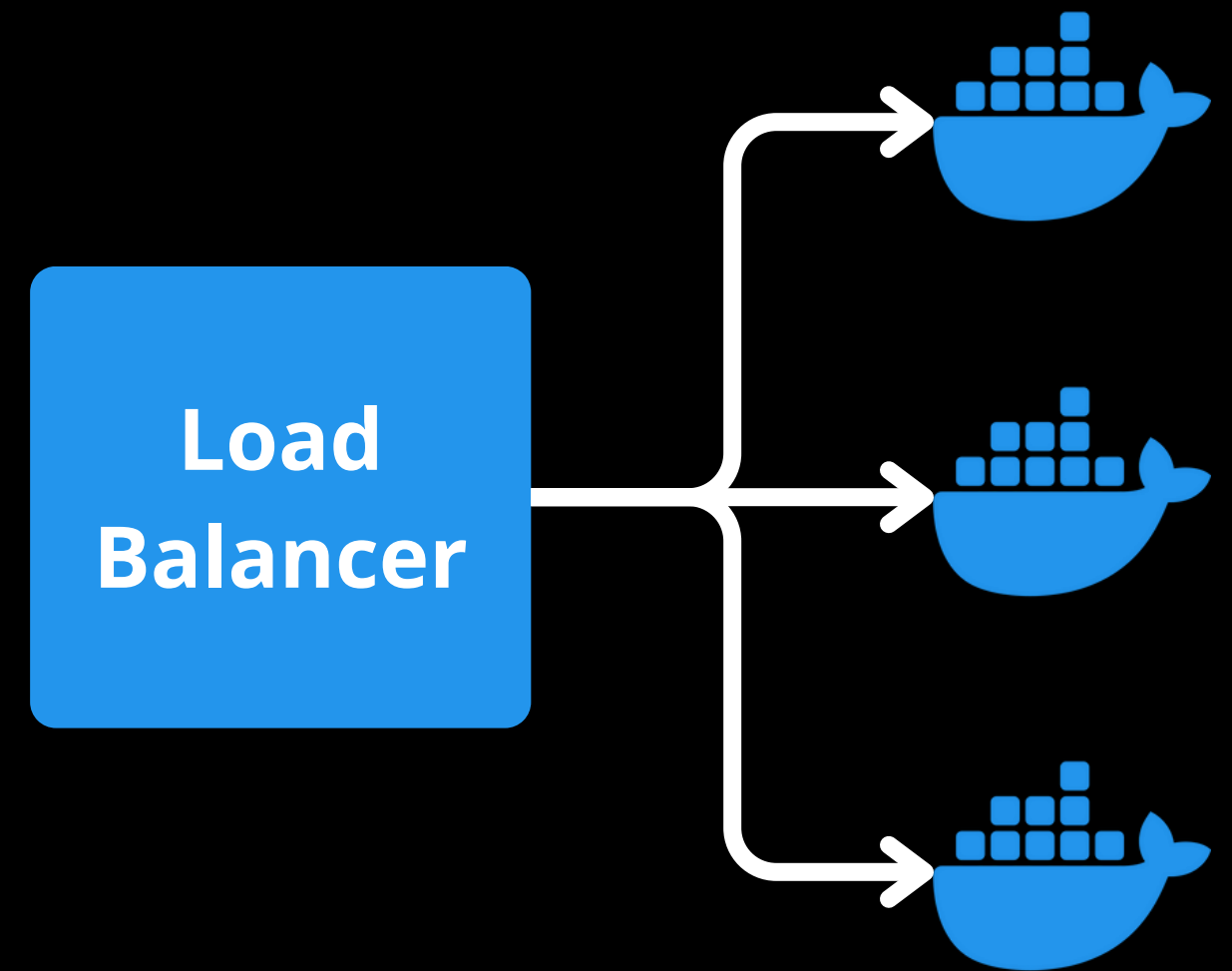


# Aplicações: Serverless

- Conjunto de serviços para alocar recursos sem a necessidade de um servidor alocado estaticamente
  - Não é mais necessário alocar uma máquina inteira para rodar uma aplicação única
-

# Aplicações: Serverless

- Aplicações auto escaláveis de containers
- Sobem a quantidade de containers de acordo com a demanda
- Balanceador de carga cuida para que as requisições sejam distribuídas entre containers
- Vantagens na utilização de APIs que seguem o padrão REST
- Cobrança por instancia + tempo + processamento

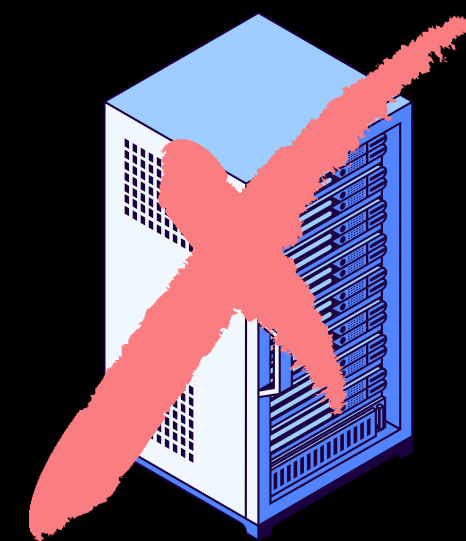


# Aplicações: Serverless

- Functions alocadas estaticamente para serem executadas apenas quando chamadas
- Principal forma de cobrança é por execução (quantidade de vezes em que ela foi chamada)



# Aplicações: Realidade



- Mas recursos de GPU ainda não eram inclusos até pouco tempo



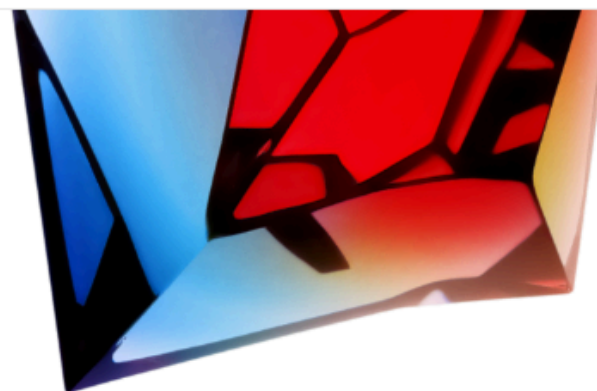
Serverless

# Cloud Run GPUs, now GA, makes running AI workloads easier for everyone

June 2, 2025



# Cloud Run



# Referências

- <https://docs.vllm.ai/en/latest/>
  - <https://aws.amazon.com/what-is/retrieval-augmented-generation/>
  - <https://aws.amazon.com/pt/microservices/#:~:text=Microservi%C3%A7os%20s%C3%A3o%20uma%20abordagem%20arquitet%C3%B4nica,comunicam%20usando%20APIs%20bem%20definidas.>
  - <https://www.youtube.com/watch?v=EW7m2WlvFgQ>
  - <https://ai-sdk.dev/docs/foundations/streaming>
  - <https://github.com/kamranahmedse/developer-roadmap>
  - <https://huggingface.co/Qwen/Qwen2-1.5B-Instruct>
-