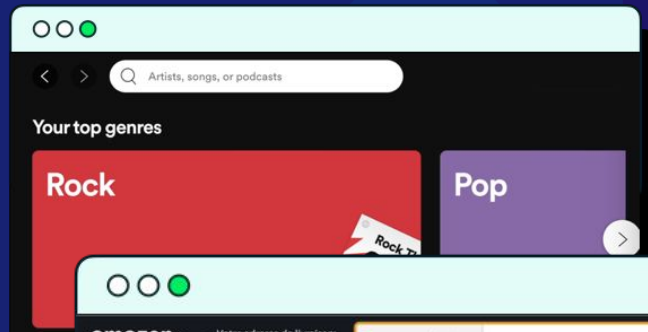
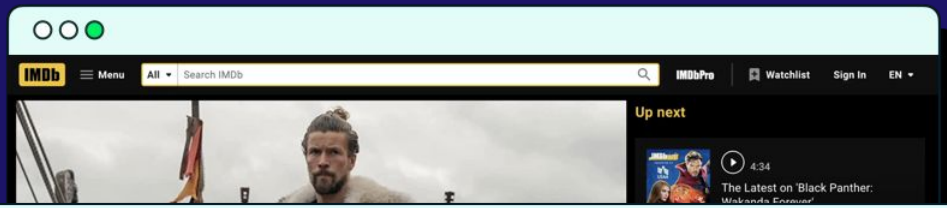
Weaviate

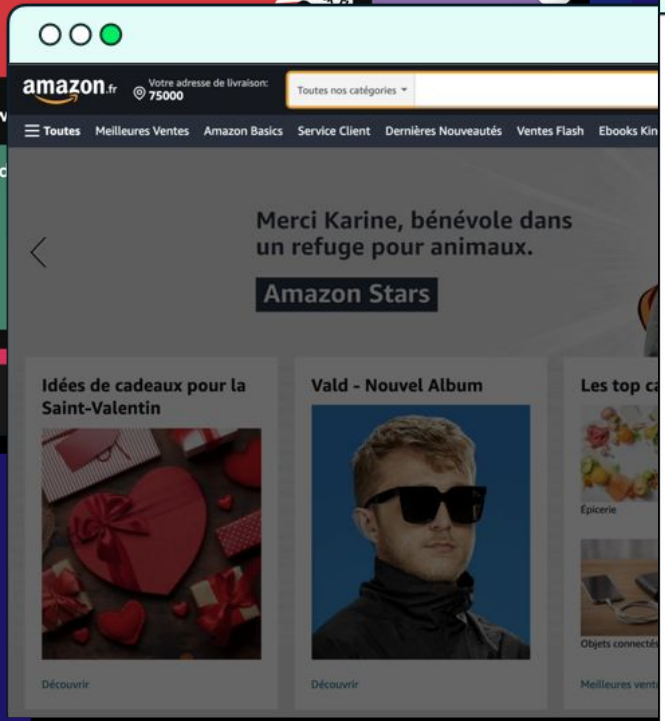
Weaviate Workshop

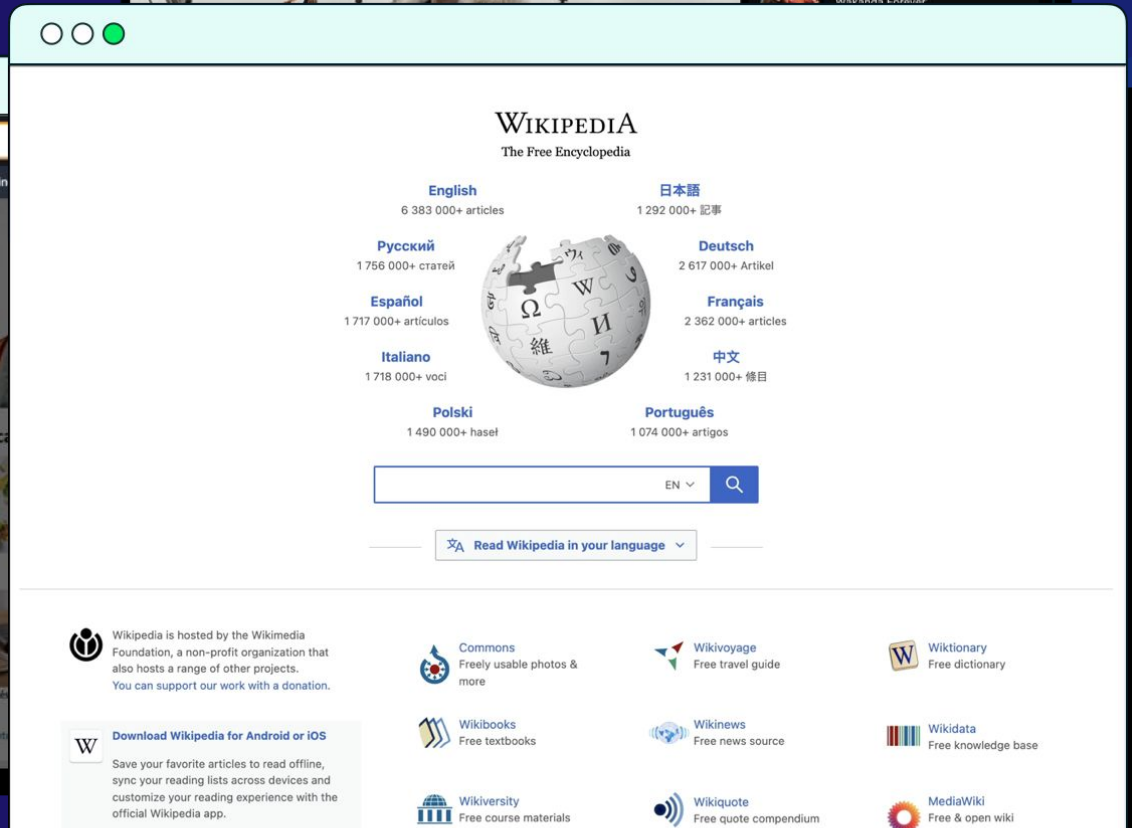
Sebastian Witalec
Head of DevRel at Weaviate

Everything we do
(on the internet)
Starts with Search


Artists, songs, or podcasts
Your top genres
Rock
Pop

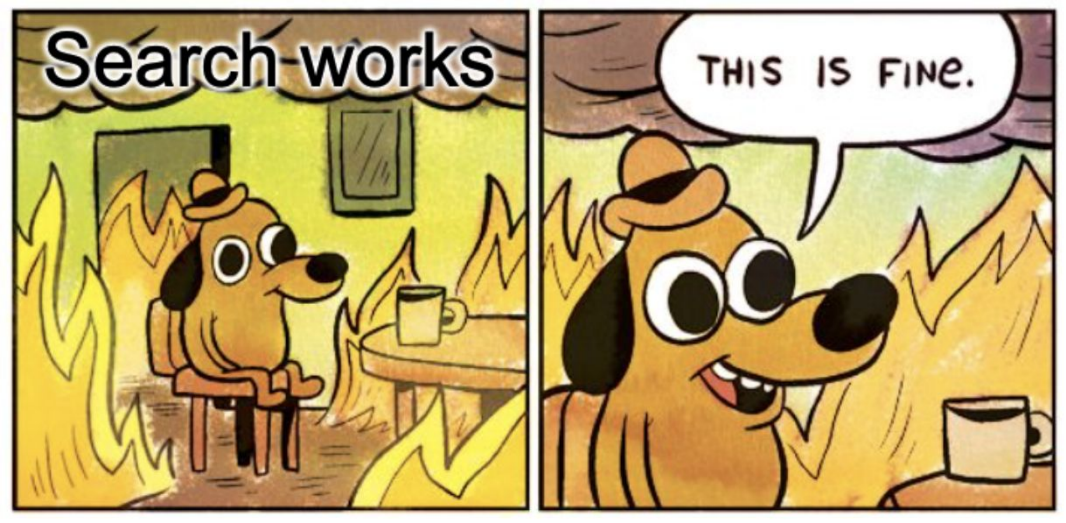

IMDb
Menu
All
Search IMDb
IMDbPro
Watchlist
Sign In
EN
Up next
4:34
The Latest on 'Black Panther:


amazon.fr
Votre adresse de livraison:
75000
Toutes nos catégories
Toutes
Meilleures Ventes
Amazon Basics
Service Client
Dernières Nouveautés
Ventes Flash
Merci Karine, bénévole dans un refuge pour animaux.
Amazon Stars
Idées de cadeaux pour la Saint-Valentin
Découvrir
Vald - Nouvel Album
Découvrir


WIKIPEDIA
The Free Encyclopedia
English
6 383 000+ articles
日本語
1 292 000+ 記事
Русский
1 756 000+ статей
Deutsch
2 617 000+ Artikel
Español
1 717 000+ artículos
Français
2 362 000+ articles
Italiano
1 718 000+ voci
中文
1 231 000+ 條目
Polski
1 490 000+ haseł
Português
1 074 000+ artigos
EN
Read Wikipedia in your language
Wikipedia is hosted by the Wikimedia Foundation, a non-profit organization that also hosts a range of other projects.
You can support our work with a donation.
Download Wikipedia for Android or iOS
Save your favorite articles to read offline, sync your reading lists across devices and customize your reading experience with the official Wikipedia app.
Commons
Freely usable photos & more
Wikivoyage
Free travel guide
Wiktionary
Free dictionary
Wikibooks
Free textbooks
Wikinews
Free news source
Wikidata
Free knowledge base
Wikiversity
Free course materials
Wikiquote
Free quote compendium
MediaWiki
Free & open wiki

What is the problem?

What is the problem?



The challenges of traditional search

Why do airplanes fly?

Why you should **fly** with ExpensiveAir's **airplanes!**
...fly with ExpensiveAir's modern fleet of **airplanes** to over 4 destinations. **Why do** you hesitate? Book now!

The challenges of traditional search (the keywords match)

Why do airplanes fly?

Why you should fly with ExpensiveAir's airplanes.
...**fly** with ExpensiveAir's modern fleet of **airplanes** to over 4 destinations. **Why do** you hesitate? Book now!

In summary

Semantic Search to the rescue!

Why do airplanes fly?

Dynamics of Flight - NASA

Airplane wings are shaped **to make air move faster over the top of the wing**. When air moves faster, the pressure of the air decreases.

In summary

How does this work?

How does this work?

“You don’t need to understand it, but it is pretty cool” 😉

How does this work?

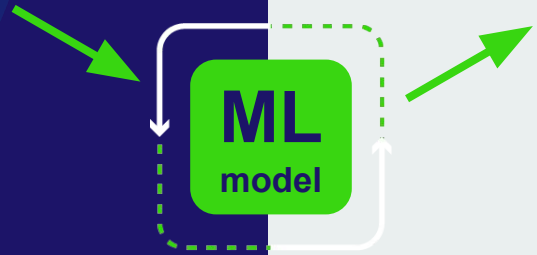
“You don’t need to understand it, but it is pretty cool” 😉

Machine Learning models

[0.23, 0.45, ..., 0.84, 0.23]

Vector Embeddings

"Fly with ExpensiveAir's modern fleet of airplanes to over 4 destinations. Why do you hesitate? Book now!"



[0.23, 0.45, ..., 0.84, 0.23]

Vector Embeddings

"Fly with ExpensiveAir's modern fleet of airplanes to over 4 destinations. Why do you hesitate? Book now!"

Airplane wings are shaped to make air move faster over the top of the wing. When air moves faster, the pressure of the air decreases.

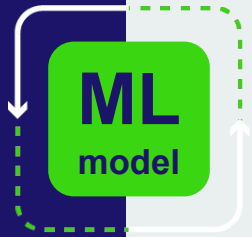
[0.23, 0.45, ..., 0.84, 0.23]

[0.26, 0.31, ..., 0.12, 0.44]

Vector Embeddings

"Fly with ExpensiveAir's modern fleet of airplanes to over 4 destinations. Why do you hesitate? Book now!"

Airplane wings are shaped to make air move faster over the top of the wing. When air moves faster, the pressure of the air decreases.



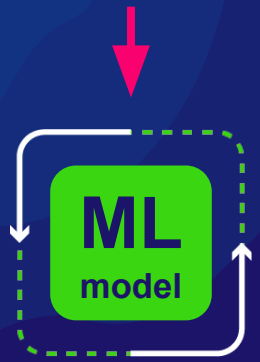
[0.23, 0.45, ..., 0.84, 0.23]

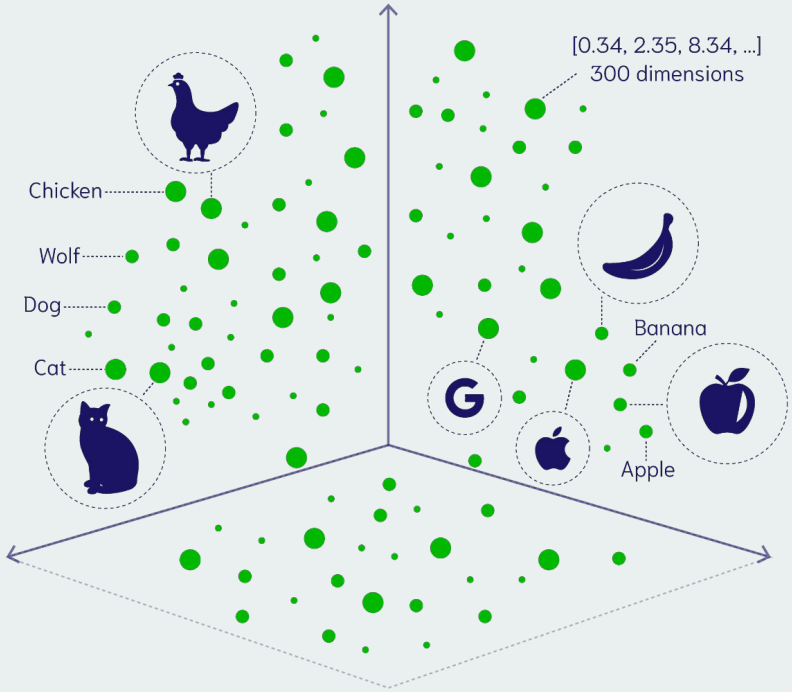
[0.26, 0.31, ..., 0.12, 0.44]

Vector Space

Vector Search

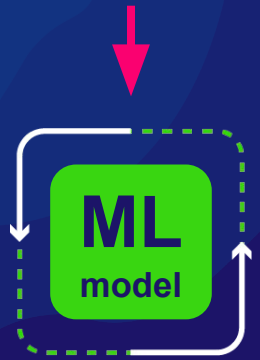
Why do airplanes fly?



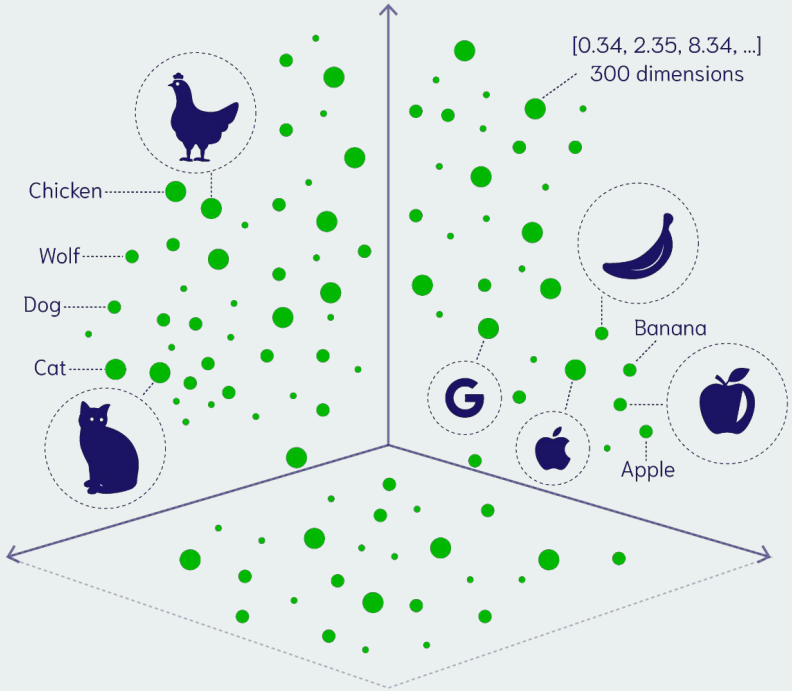


Vector Search

Why do airplanes fly?



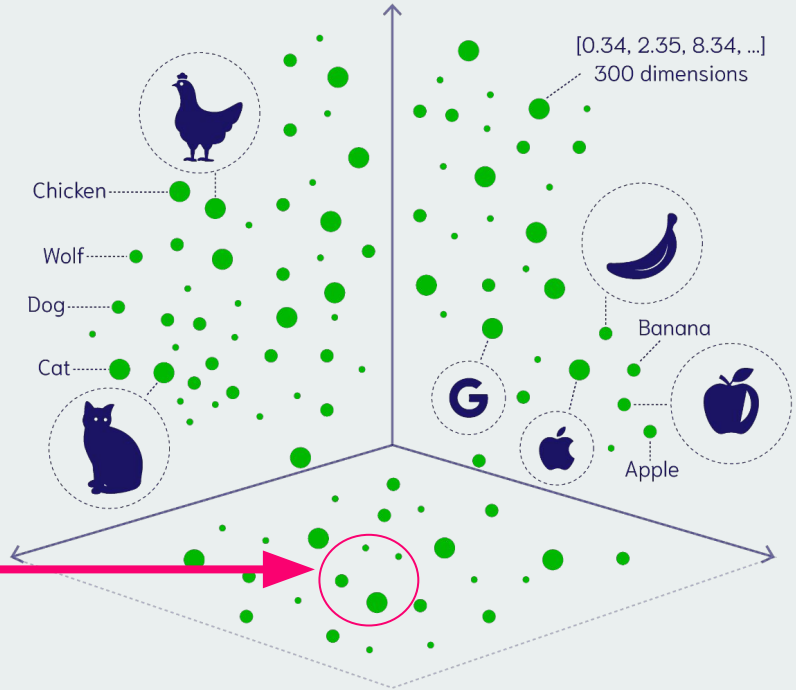
[0.24, 0.36, ..., 0.16, 0.46]



Vector Search

Why do airplanes fly?

[0.24, 0.36, ..., 0.16, 0.46]



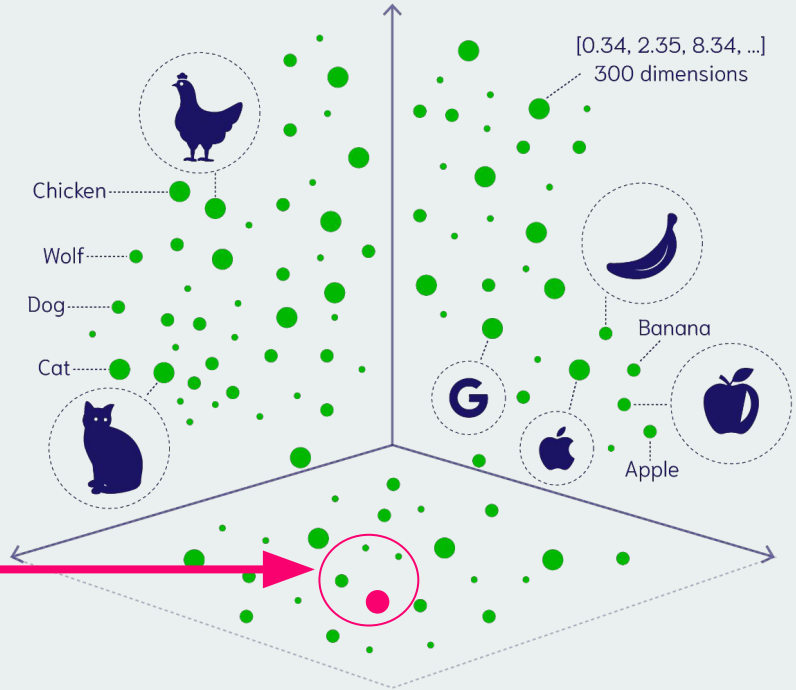
Vector Search

Why do airplanes fly?

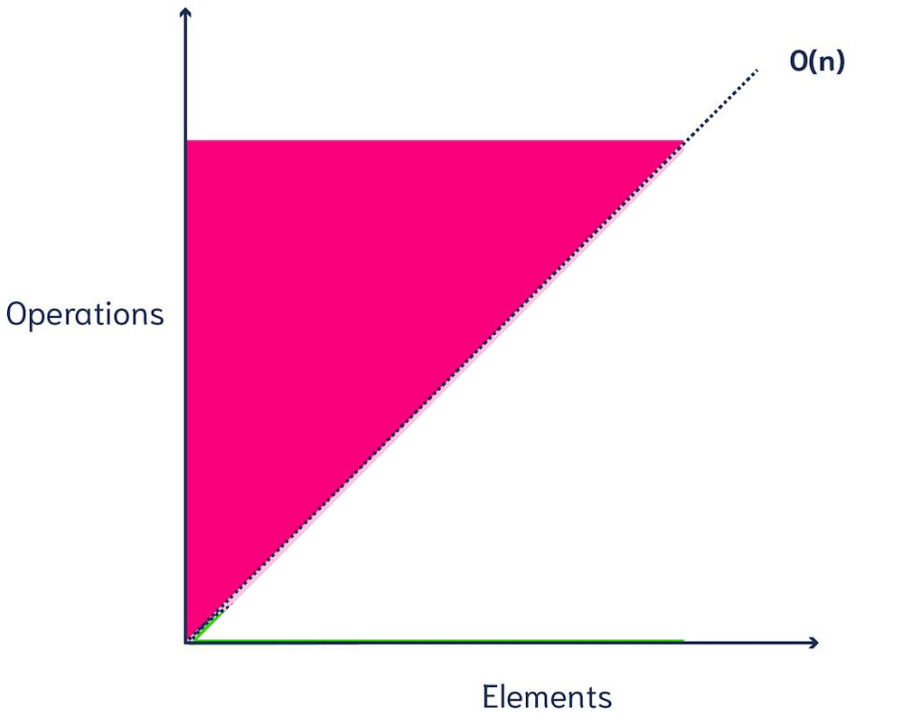
[0.24, 0.36, ..., 0.16, 0.46]

[0.26, 0.31, ..., 0.12, 0.44]

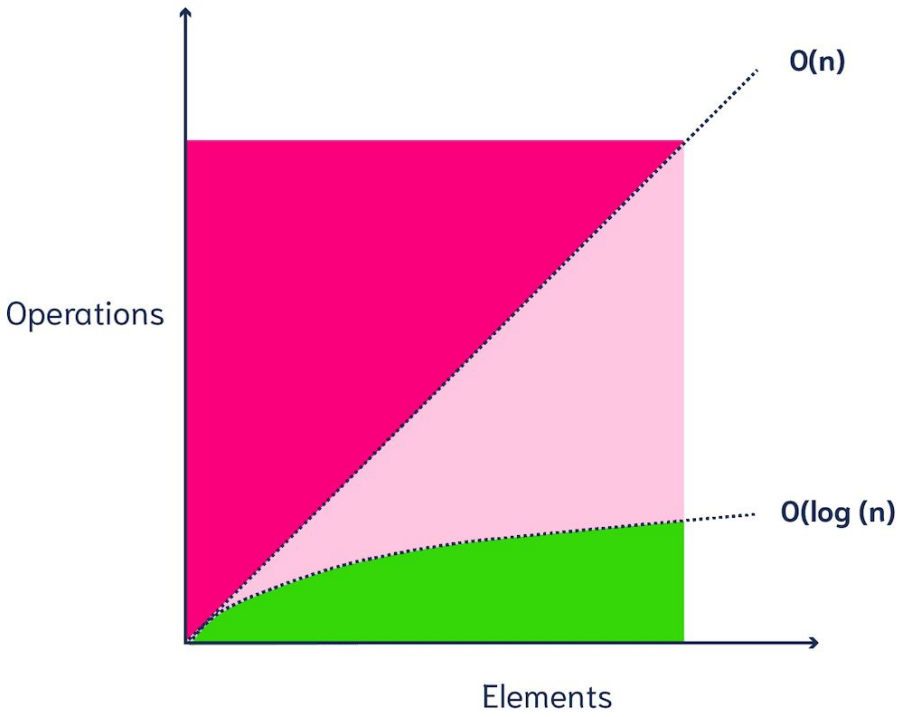
The NASA article



Traditional indexes don’t work with vector embeddings



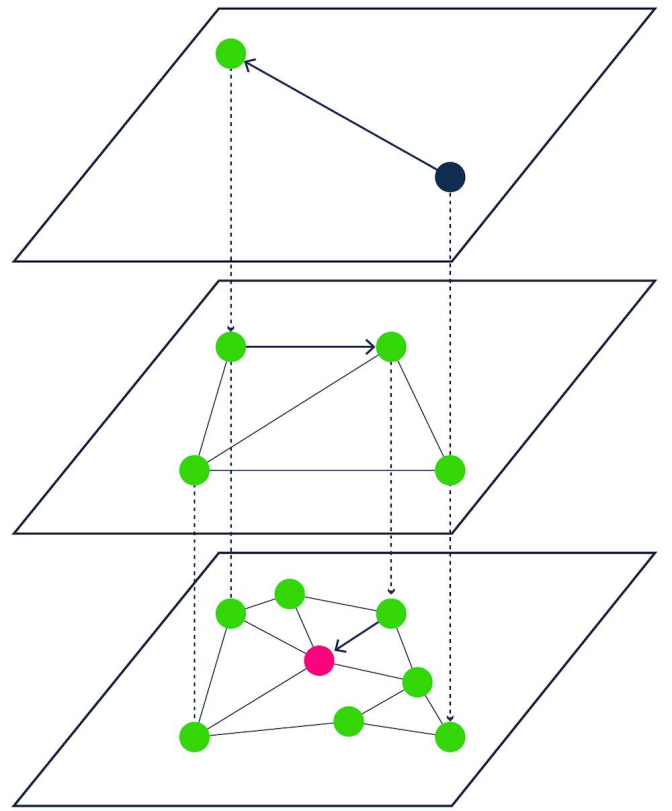
KNN - brute force (exact)



KNN - brute force (exact)

ANN - be smart about it (approximate)

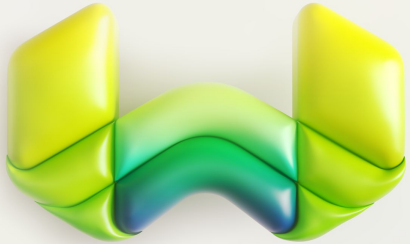
Hierarchical Navigable Small World (HNSW)



https://weaviate.io/developers/weaviate/benchmarks/ann

How do I implement that?

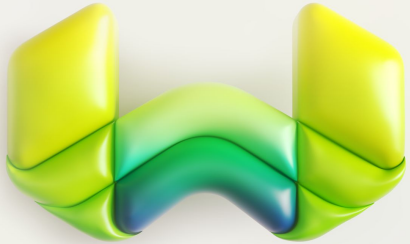
You don’t!
Enter a Vector Database



Weaviate

The AI Native Vector Database

LESSON 1
end-to-end



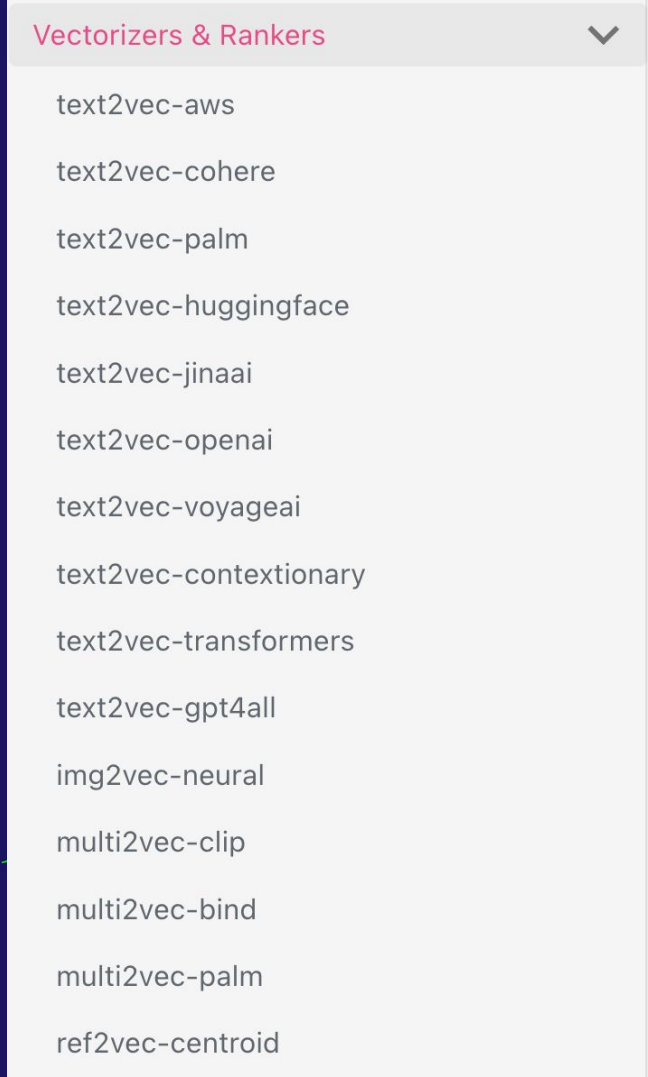
Weaviate

The AI Native Vector Database

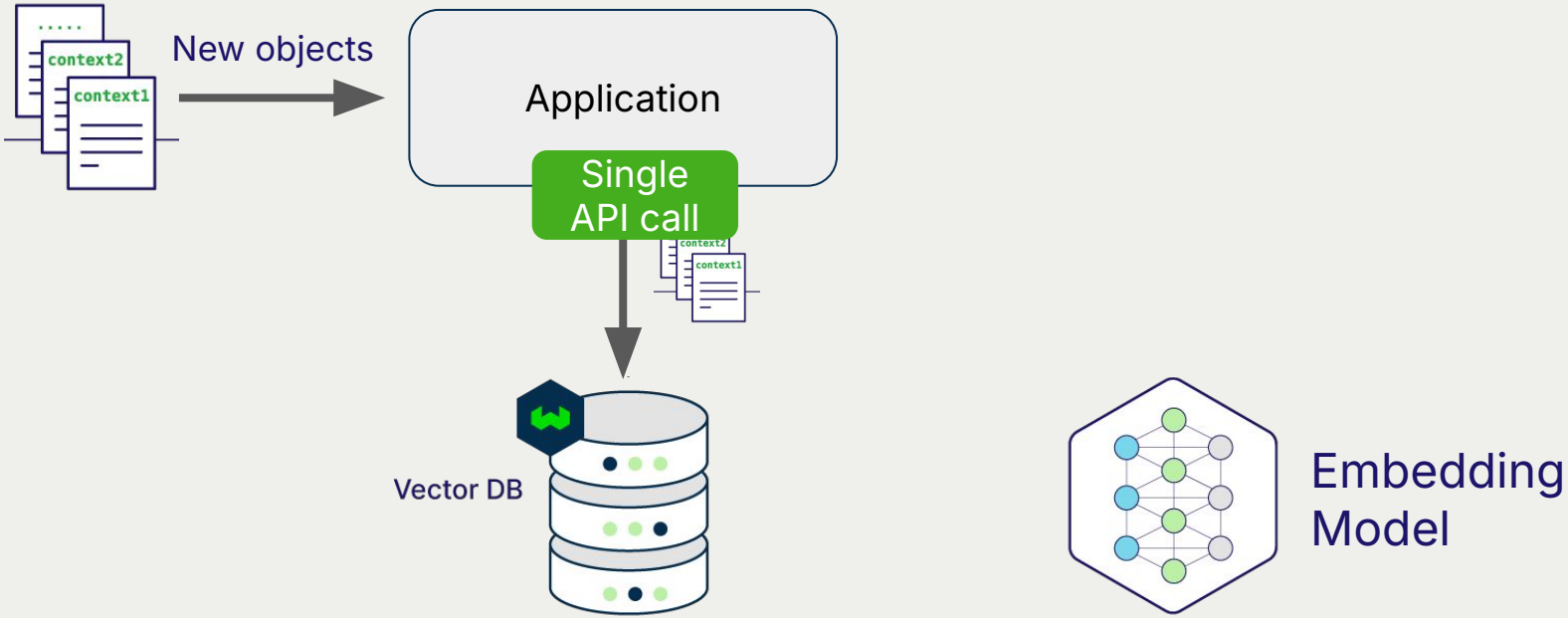
Built for scale 🚀

Running at **billion scale** since 2022

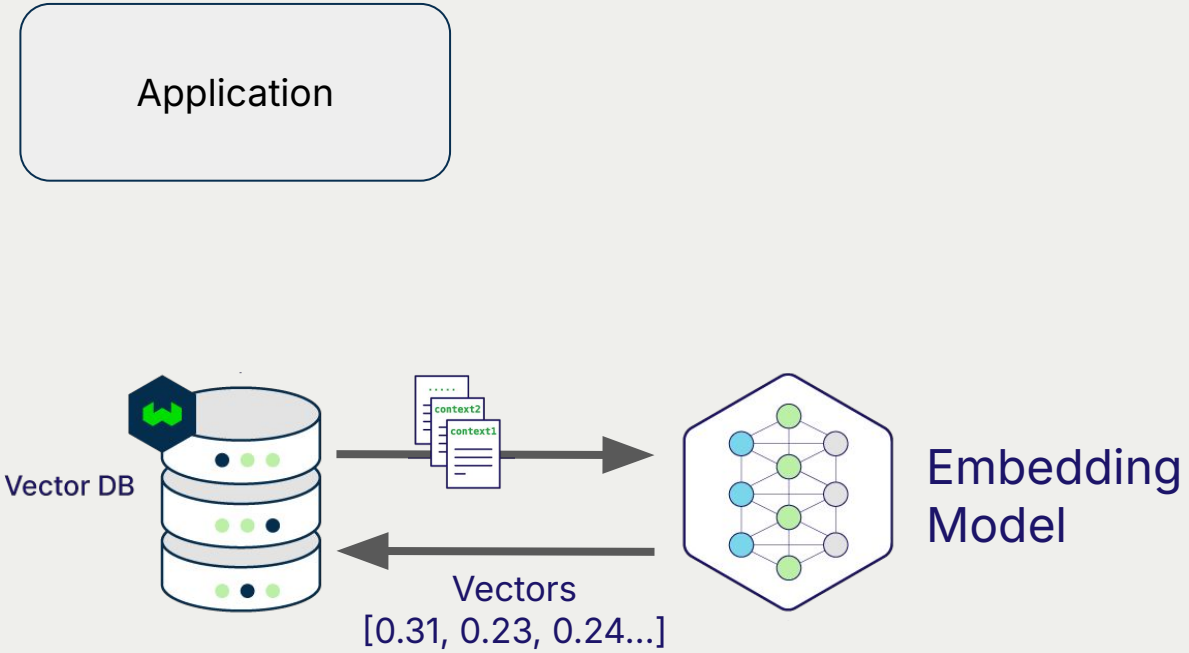
Already integrated with many Models



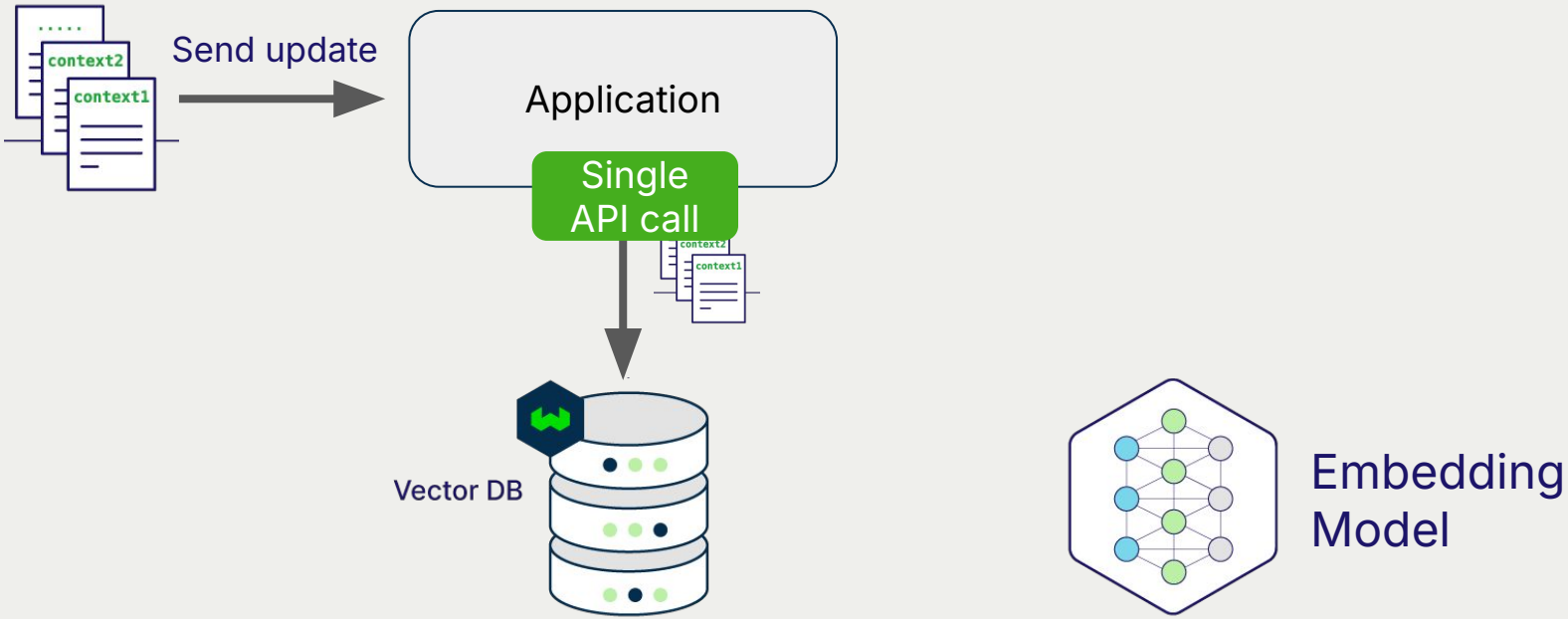
Data operations (i.e. data import)



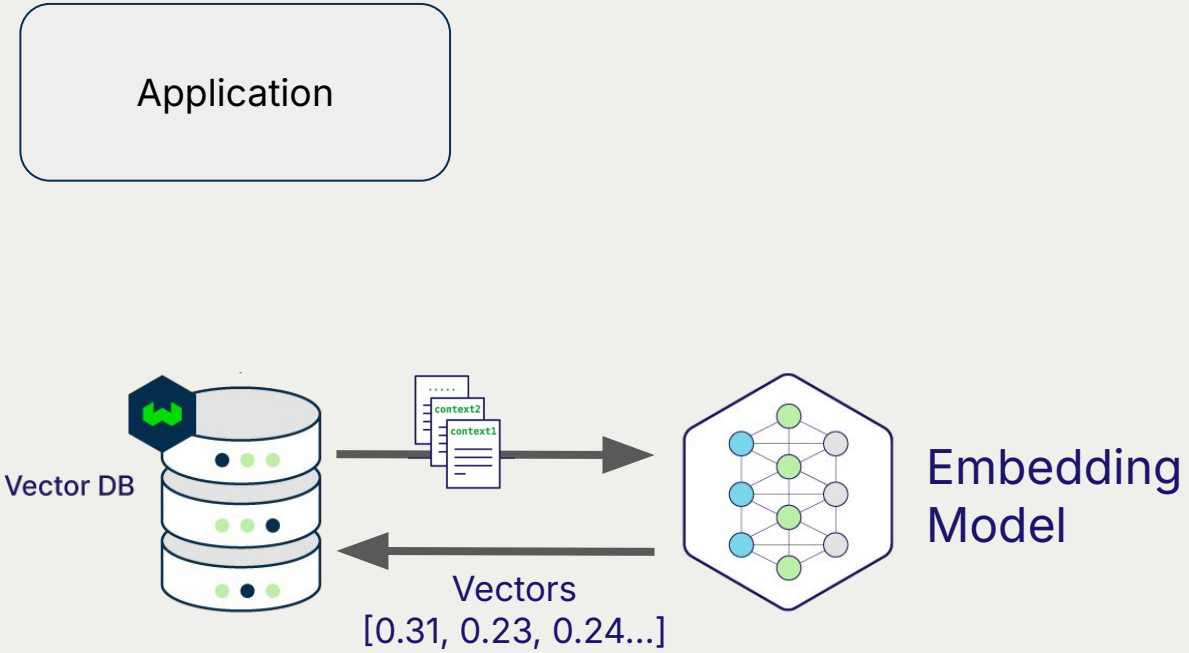
Data operations (i.e. data import)



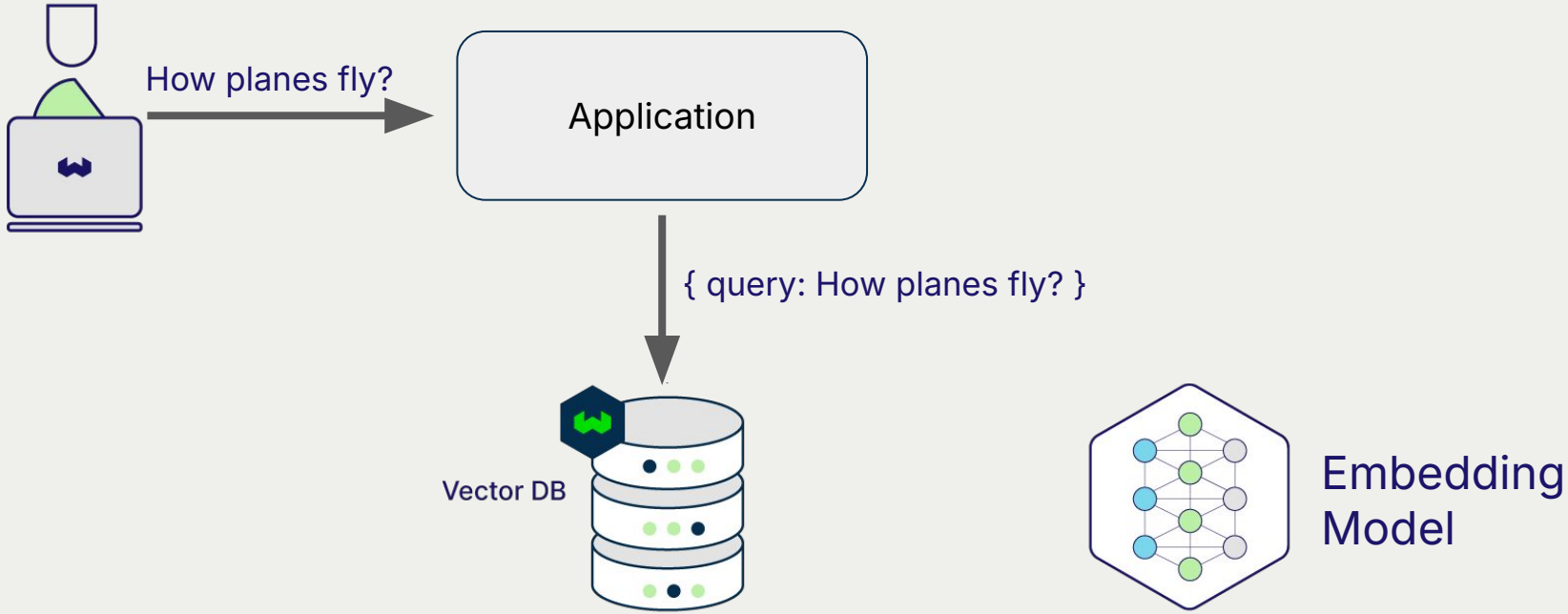
Data operations (i.e. update)



Data operations (i.e. update)

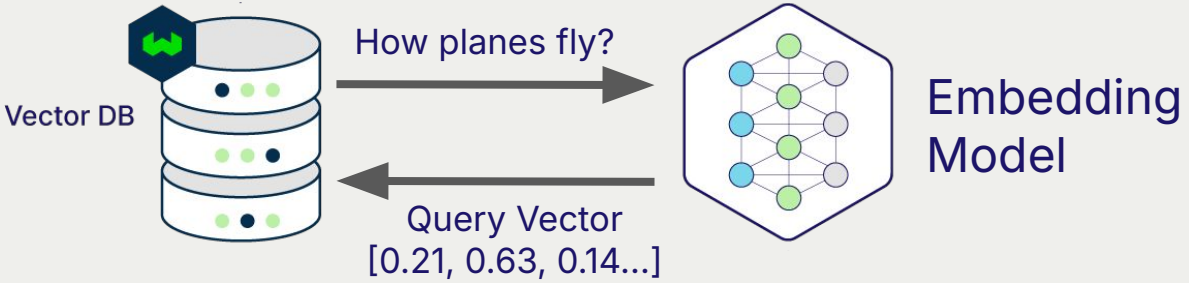


Query

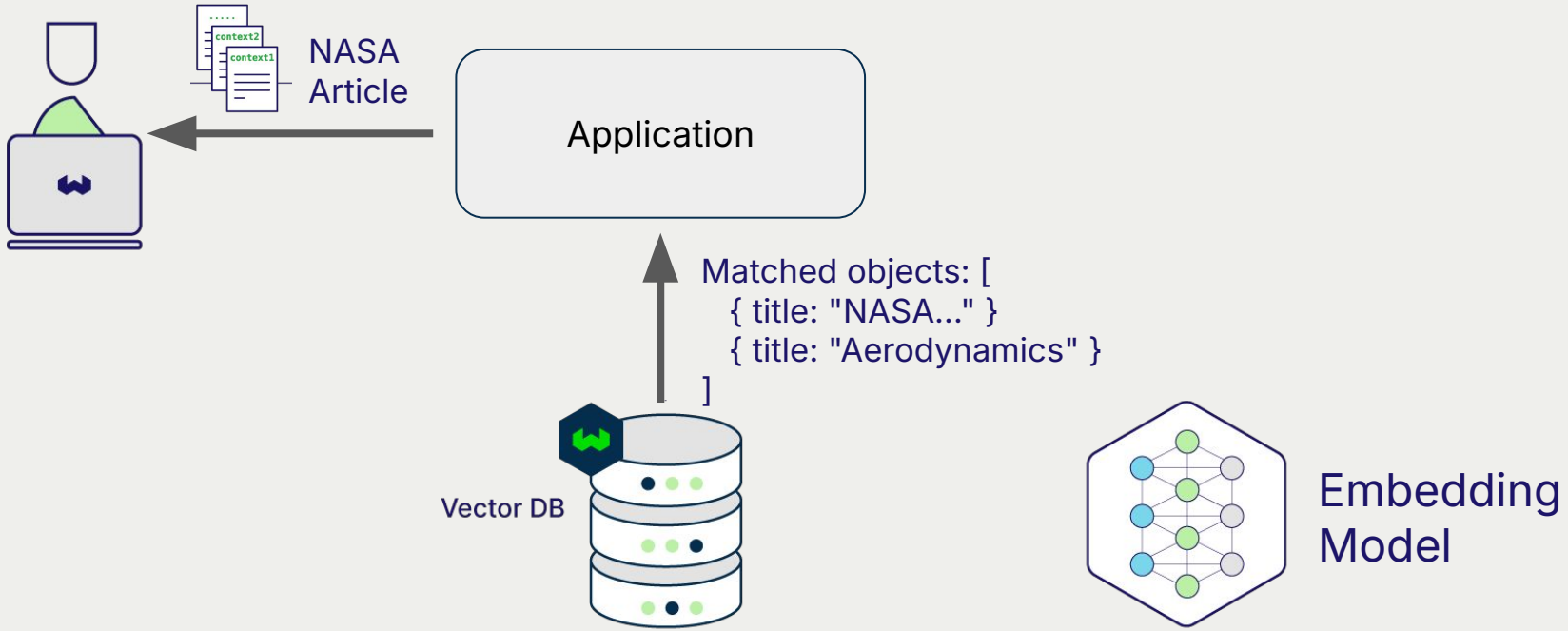

How planes fly?
Application
{ query: How planes fly? }
Vector DB
Embedding Model

Query

Application



Query


context2
context1
NASA
Article
Application
Matched objects: [
{ title: "NASA..." }
{ title: "Aerodynamics" }
]
Vector DB
Embedding
Model

Vector Search with Text

```
articles = client.collections.get("Articles")

response = articles.query.near_text(
    query="animals in movies",
    limit=2,
)
```

Search with Filter

```
articles = client.collections.get("Articles")

response = articles.query.near_text(
    query="animals in movies",
    limit=2,
    filters=Filter.by_property("category")
                  .equal("Media"),
)
```

Keyword Search (BM25)

```
articles = client.collections.get("Articles")

response = articles.query.bm25(
    query="part C2F20",
    limit=2,
)
```

Hybrid Search

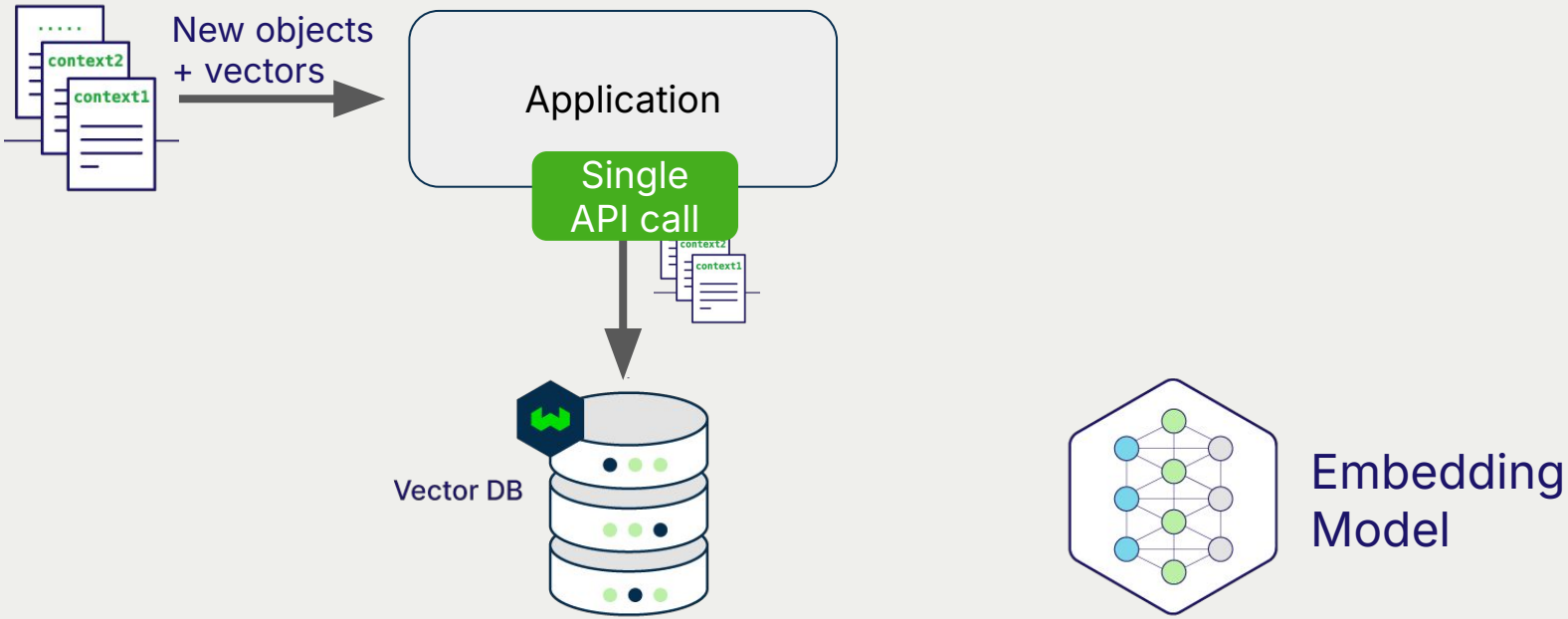
```
articles = client.collections.get("Articles")

response = articles.query.hybrid(
    query="repair parts model C2F20",
    aplpha=0.7,
    limit=2,
)
```

Hands on

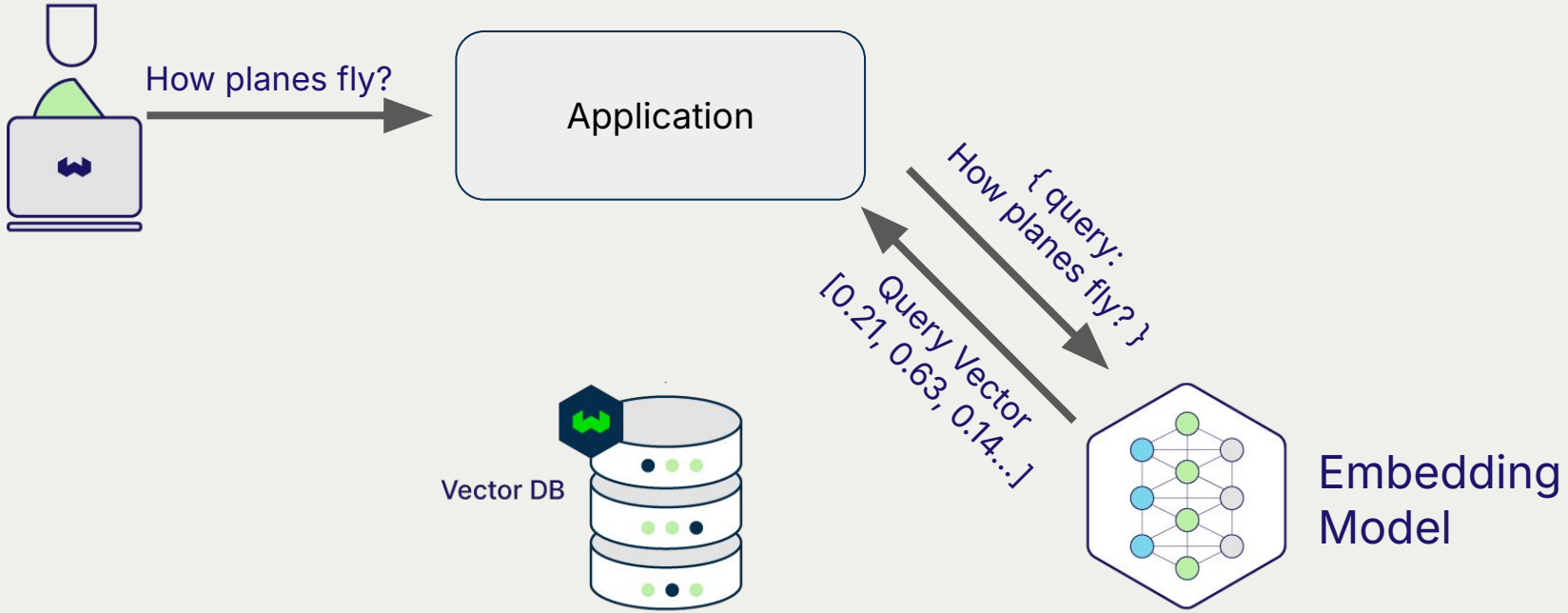
Part 1
end-to-end

LESSON 2
Bring Your Own Vectors

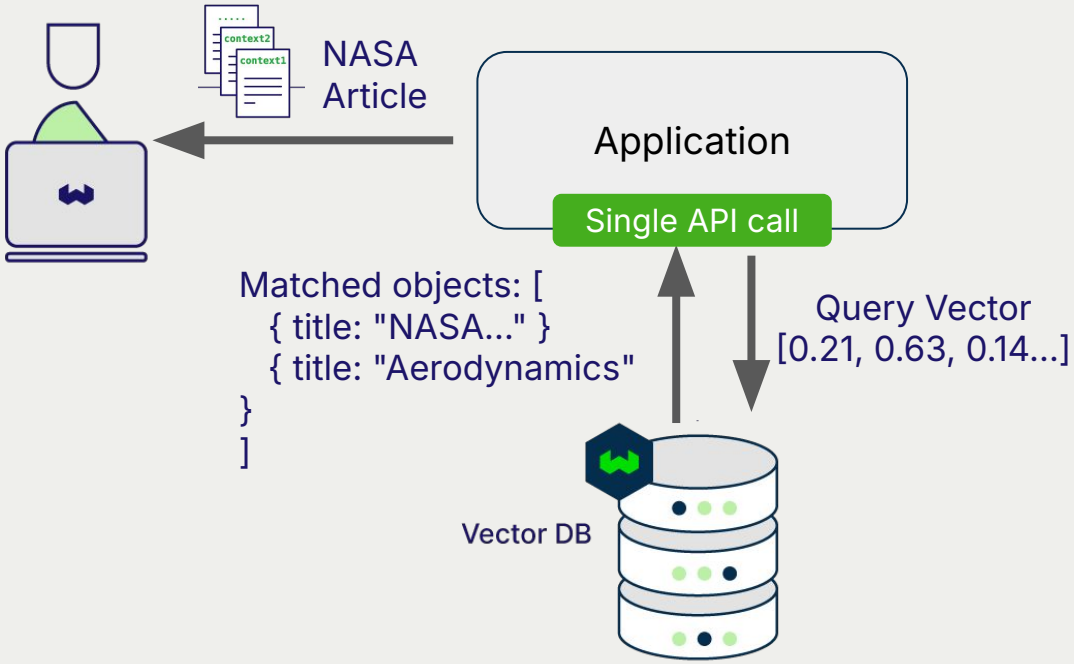
Import Data with Vectors
(no vectorizer)

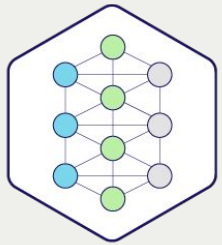


Query
(no vectorizer)



Query
(no vectorizer)





Embedding
Model

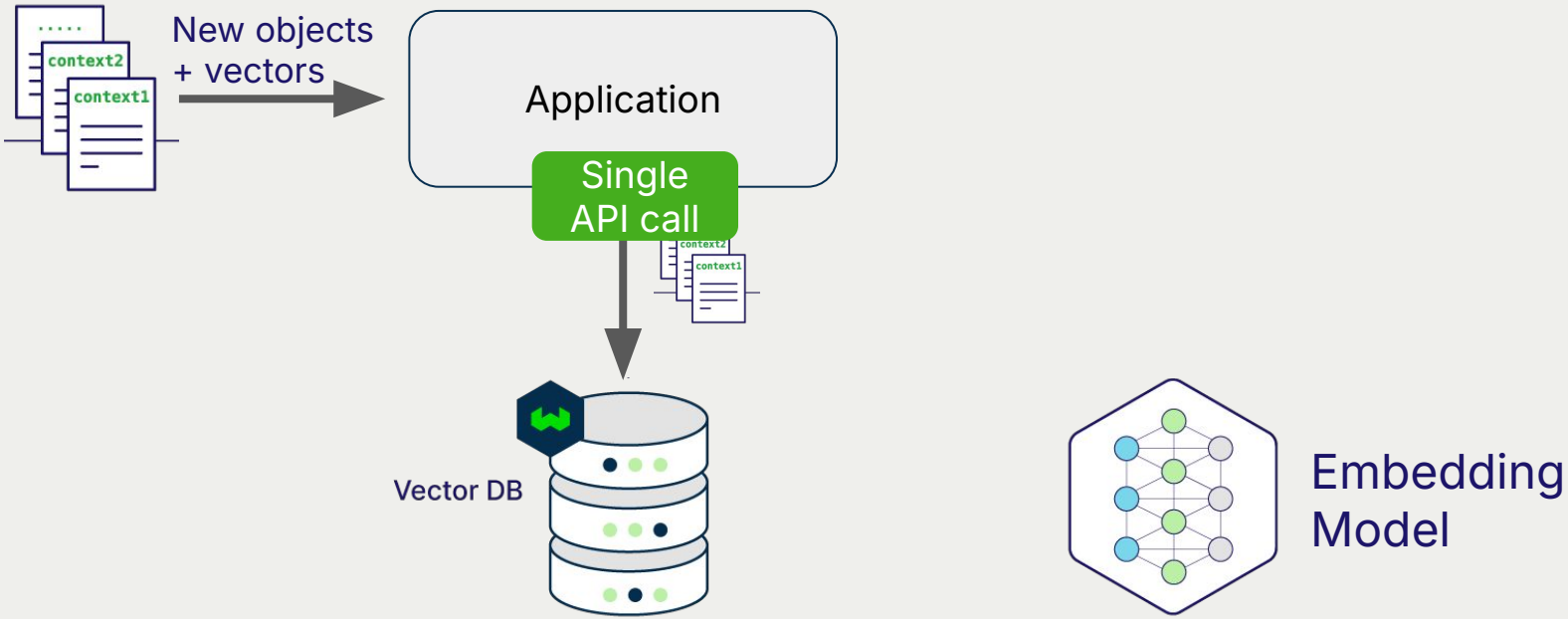
Find Similar Objects*

```
articles = client.collections.get("Articles")

response = articles.query.near_object(
    near_object="a1dd67f9-bfa7-45e1-b45e-26eb8c52e9a6",
    limit=2,
)
```

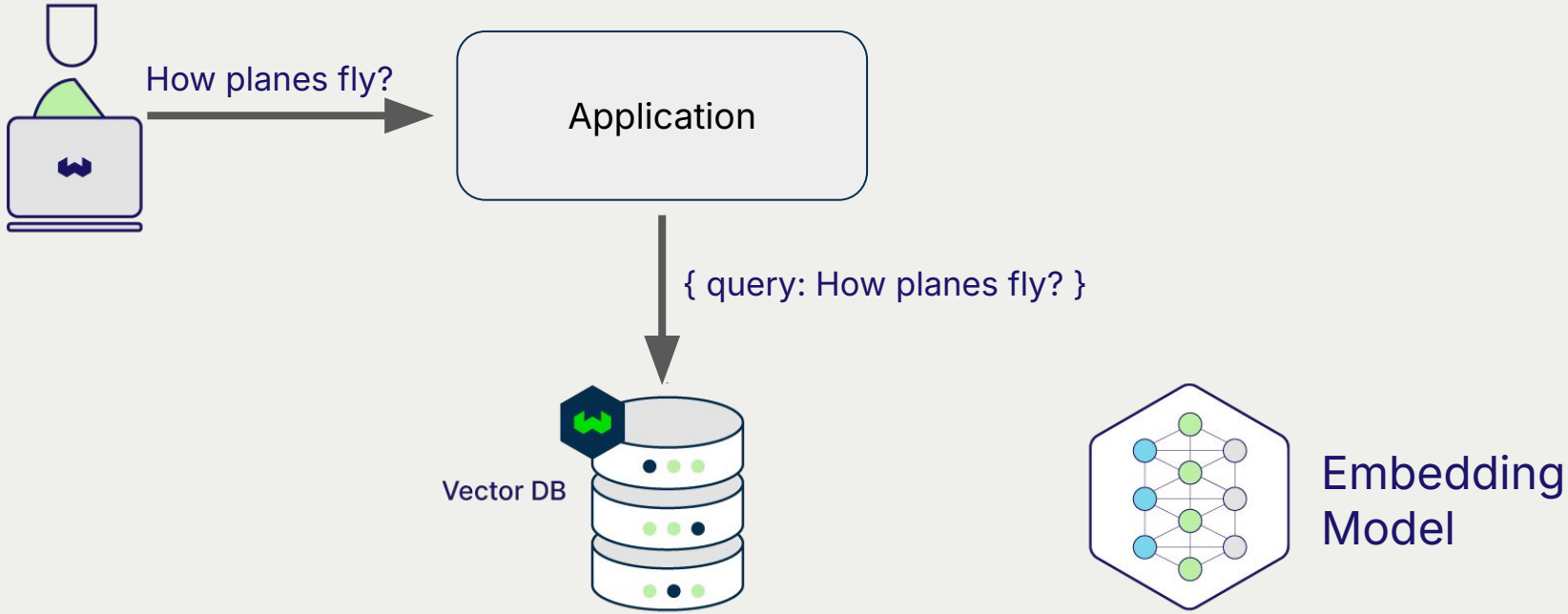
Search with Vector*

```
articles = client.collections.get("Articles")

response = articles.query.near_vector(
    near_vector=[0.131, -0.321, 0.631, ...],
    limit=2,
)
```

Import Data with Vectors
(with vectorizer)

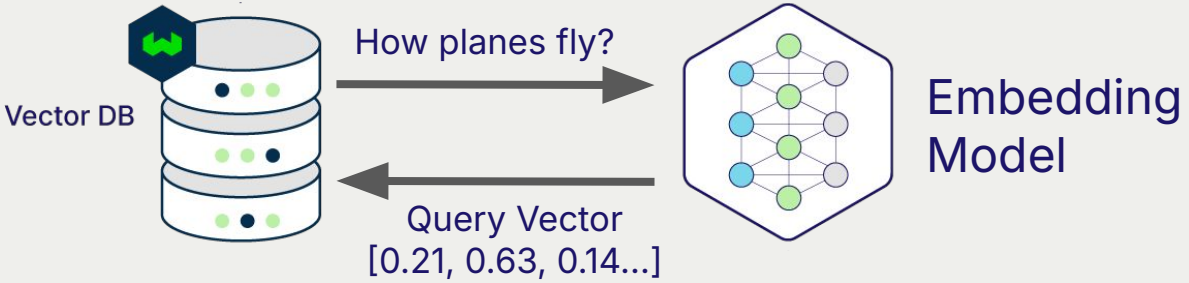


Query
(with vectorizer)

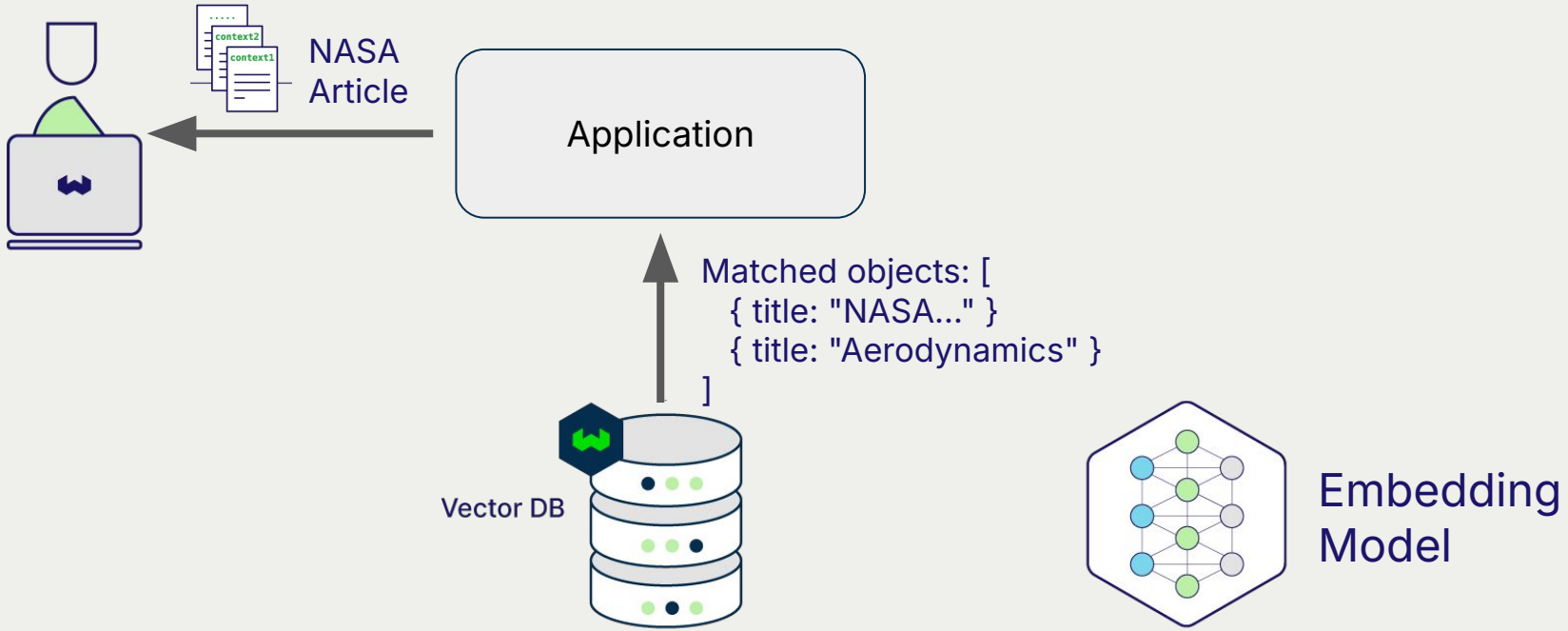


Query
(with vectorizer)

Application



Query
(with vectorizer)



Hands on

Part 2
Import vectorized data

LESSON 3
Retrieval Augmented Generation (RAG)

Vector Database

Large Language Model

One Key Difference

Vector Database (Stateful)

Large Language Model (Stateless)

Vector Database (Stateful)

- Insert data into the database, and generate vectors.

Large Language Model (Stateless)

- The knowledge base is trained into the model.

Vector Database (Stateful)

- Insert data into the database, and generate vectors.
- Data changes on the fly.

Large Language Model (Stateless)

- The knowledge base is trained into the model.
- The model doesn’t learn on the fly.

Vector Database (Stateful)

- Insert data into the database, and generate vectors.
- Data changes on the fly.
- CRUD operations to update the state.

Large Language Model (Stateless)

- The knowledge base is trained into the model.
- The model doesn’t learn on the fly.
- Requires fine-tuning to update its worldview.

This is not a story of VDBs vs LLMs

This is a story of VDBs + LLMs

This is a story of VDBs + LLMs

Focus on content:

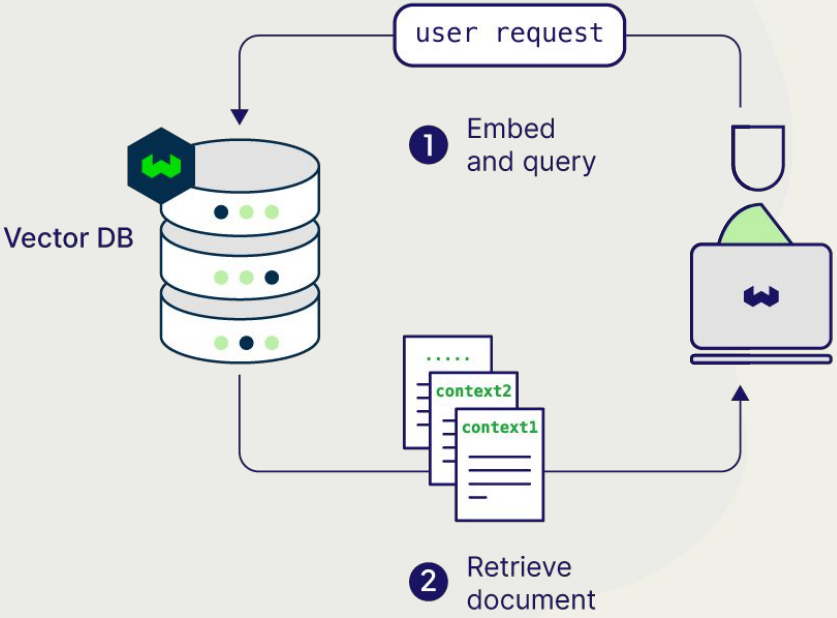
- Keep the data up to date
- Provide accurate context to LLMs

Focus on response:

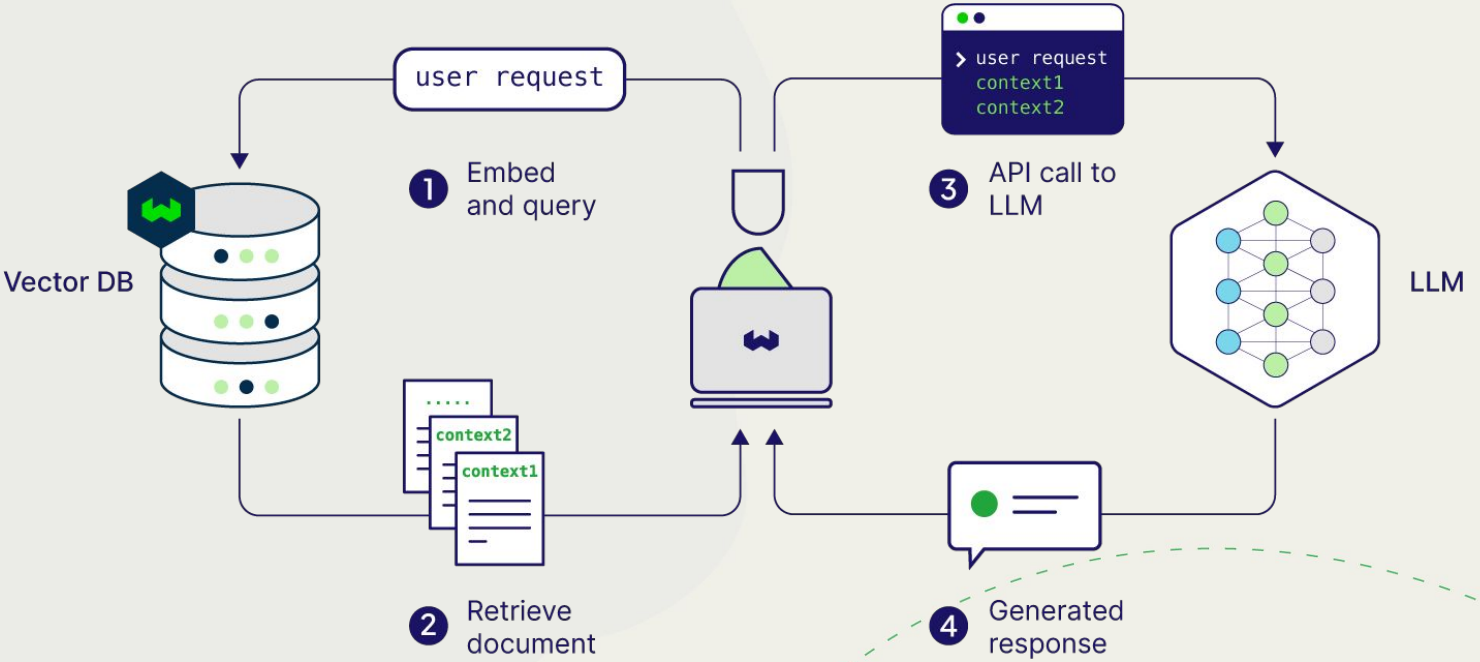
- Speak with your brand voice
- Boost understanding of the underlying context

Built-in RAG workflow

Retrieval (R)


user request
1 Embed and query
Vector DB
context2
context1
2 Retrieve document

Retrieval-Augmented Generation (RAG)



RAG – Single Prompt

```
article = client.collections.get("Article")
response = article.generate.near_text(
    query="data stores",
    limit=3,
    single_prompt="Please explain how {title} works based on {content}"
)
```

RAG – Grouped Task

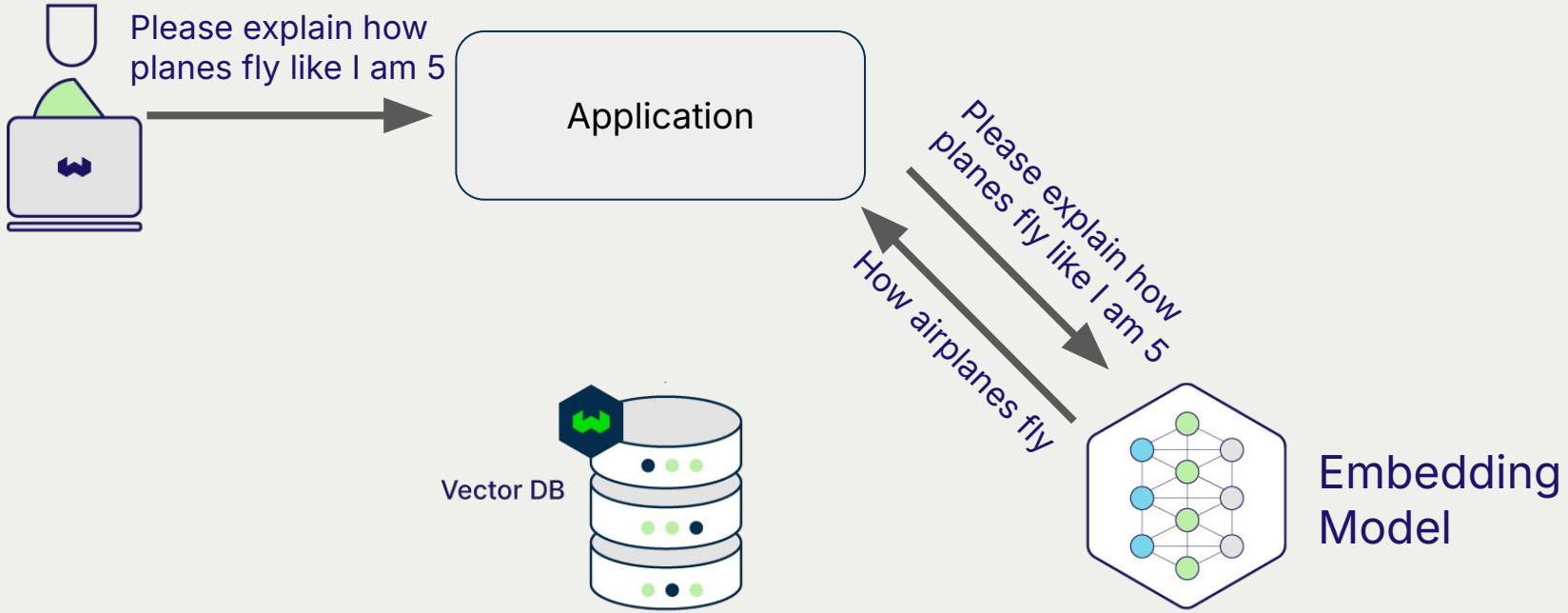
```
article = client.collections.get("Article")
response = article.generate.near_text(
    query="data stores",
    limit=3,
    grouped_task="Please summarise pros and cons of these data stores."
)
```

RAG – Single & Grouped Task

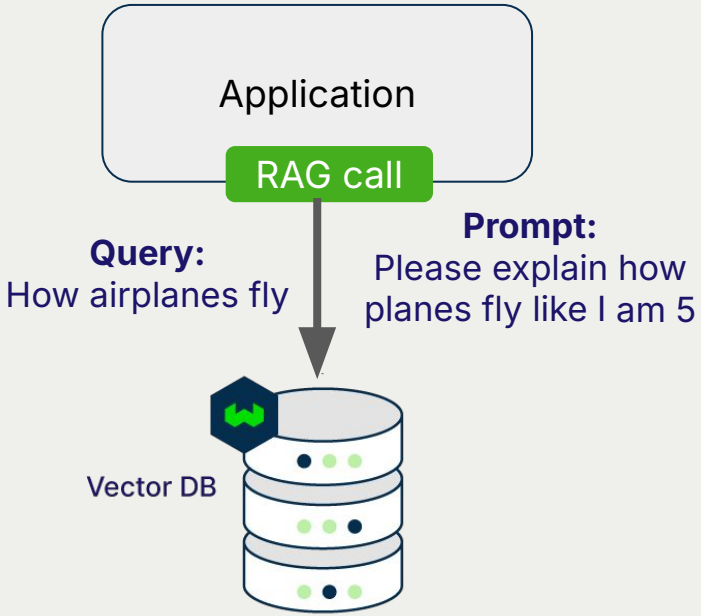
```
article = client.collections.get("Article")
response = article.generate.near_text(
    query="data stores",
    limit=3,
    single_prompt="Please explain how {title} works based on {content}",
    grouped_task="Please summarise pros and cons of these data stores."
)
```

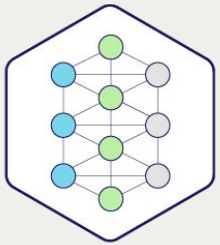
Bonus
Use an LLM to generate a query

Query from Prompt



Query from Prompt





Embedding Model

Hands on

Part 3
RAG

LESSON 4
Multi-tenancy

Split Data per (Domain)

User
Each user with their own data

Customer
White labelled-solutions

Team / Department
Separate HR docs from legal docs

Split Data per (Resource)

PDF
Tenant for PDF chunks

Video
Tenant for Video chunks

Group of Documents
Tenant per topic

Videos
Tenant per set of videos
(i.e. course lessons)

Split Data per (Time)

Daily
Tenant per day

Monthly
Tenant per month

Many collections

"Create a new collection per group"

✅ Separate indexes
⇒ faster queries

❌ Hard to manage
❌ Collections use more resources

Filters

"Add a property per group"

✅ Easier to manage

❌ One big index
⇒ slower queries

Built-in Multi-tenancy

Smaller Indexes
Index per tenant

Faster queries
Less to search

GDPR compliant
For User tenants

Tenant memory management
Hot and Cold

Flat Index - friendly
Save RAM

Create a tenant-ready collection

```
client.collections.create(
    name="MyCollection",
    multi_tenancy_config=Configure.multi_tenancy(enabled=True)
)
```

Create tenants

```python
my_collection = client.collections.get("MyCollection")

my_collection.tenants.create([
    Tenant(name='tenantA'),
    Tenant(name='tenantB')
])
```

Add data to a tenant

```
tenant_a = my_collection.with_tenant("tenantA")

tenant_a.data.insert(
    properties={"title": "About tenant inserts"}
)
```

Query tenant data

```
tenant_a = my_collection.with_tenant("tenantA")

tenant_a.query.near_text(
    query="how to search in a tenant"
)
```

Hands on

Part 4
Multi-tenancy

LESSON 5
Multi-vector

Multiple Vectors

Named Vectors

Many ways to see the data

```
{
  "title": "Product name",
  "description": "A short description of the product",
  "cover-image": "base64/adfogsiagr",
  "price:": 20.5
}
```

Many ways to see the data

Default

object vector

```
{
  "title": "Product name",
  "description": "A short description of the product",
  "cover-image": "base64/adfogsiagr",
  "price:": 20.5
}
```

Many ways to see the data

Different modalities

text vec

img vec

```
{
  "title": "Product name",
  "description": "A short description of the product",
  "cover-image": "base64/adfogsiagr",
  "price:": 20.5
}
```

Many ways to see the data

Different purpose

title vec

desc vec

Use different models (to compare)

Create a new collection

```
collection = client.collections.create(
    name="Article",
    vectorizer_config=[
        Configure.NamedVectors.text2vec_aws(
            name= "title_vector",
            source_properties=["title"],
            model="amazon.titan-embed-text-v1",
            region="eu-central-1",
        ),
        Configure.NamedVectors.text2vec_aws(
            name= "content_vector",
            source_properties=["title", "description"],
            model="cohere.embed-english-v3",
            region="eu-central-1",
        )
    ],
)
```

Insert data

```
data = [{
  "title": "Named Vectors",
  "description": "Used for multiple vectors"
},{
  "title": "Weaviate",
  "description": "An open source vector database"
}]

article = client.collections.get("Article")
article.data.insert_many(data)
```

Query

```
article = client.collections.get("Article")
response = article.query.near_text(
    query="data stores",
    target_vector="title_vector", # vector name
    limit=3
)
```

Hands on

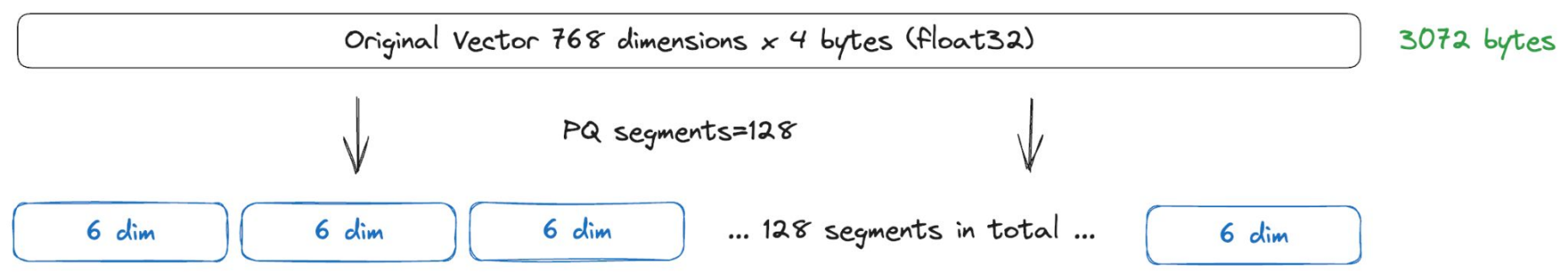
Part 5
Multi-vector

LESSON 6
Vector Compression

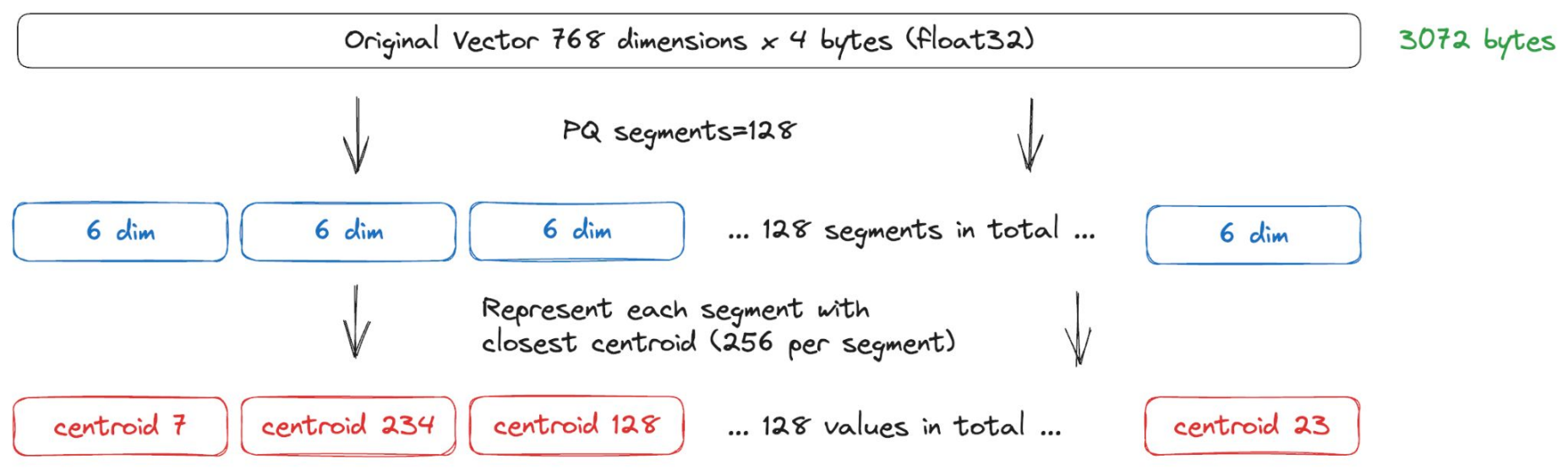
Product Quantization (PQ)

Vector Compression

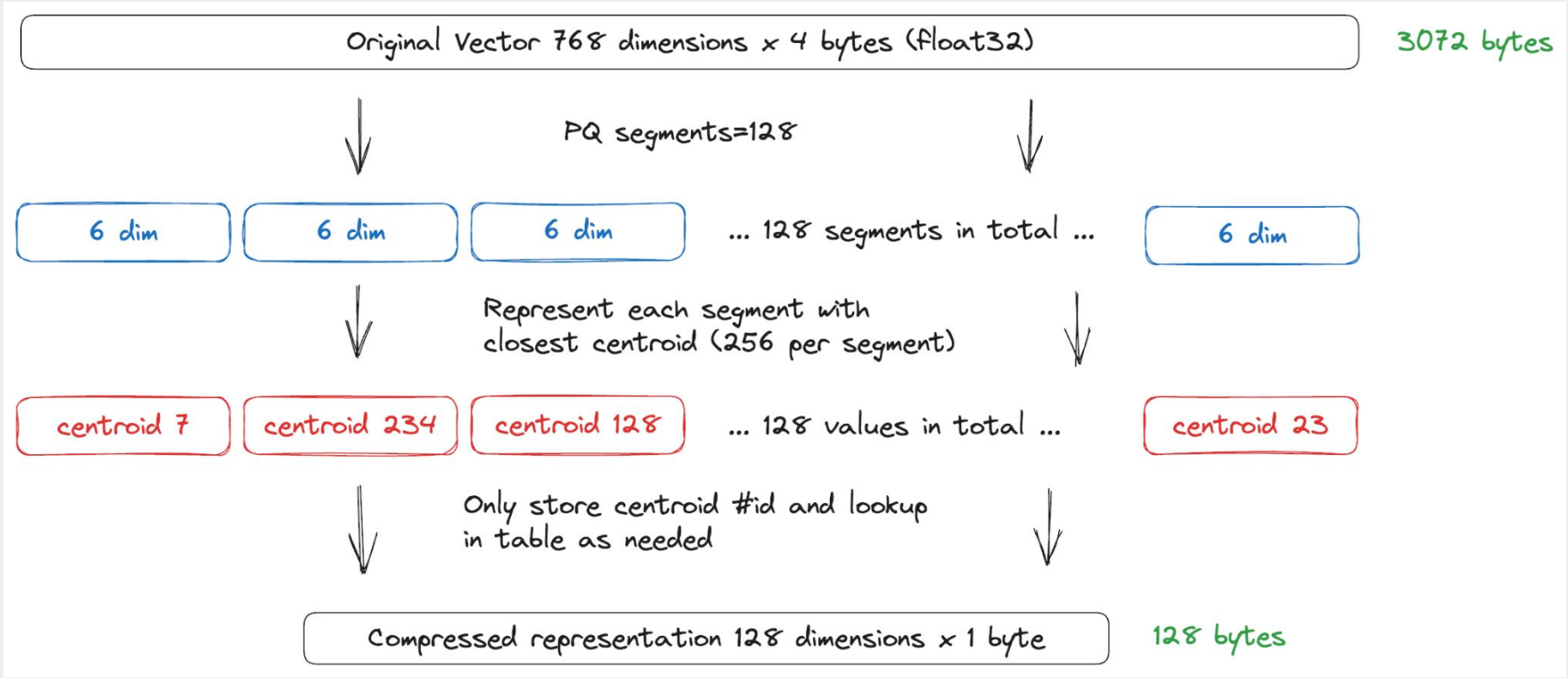
PQ compression in steps



PQ compression in steps



PQ compression in steps



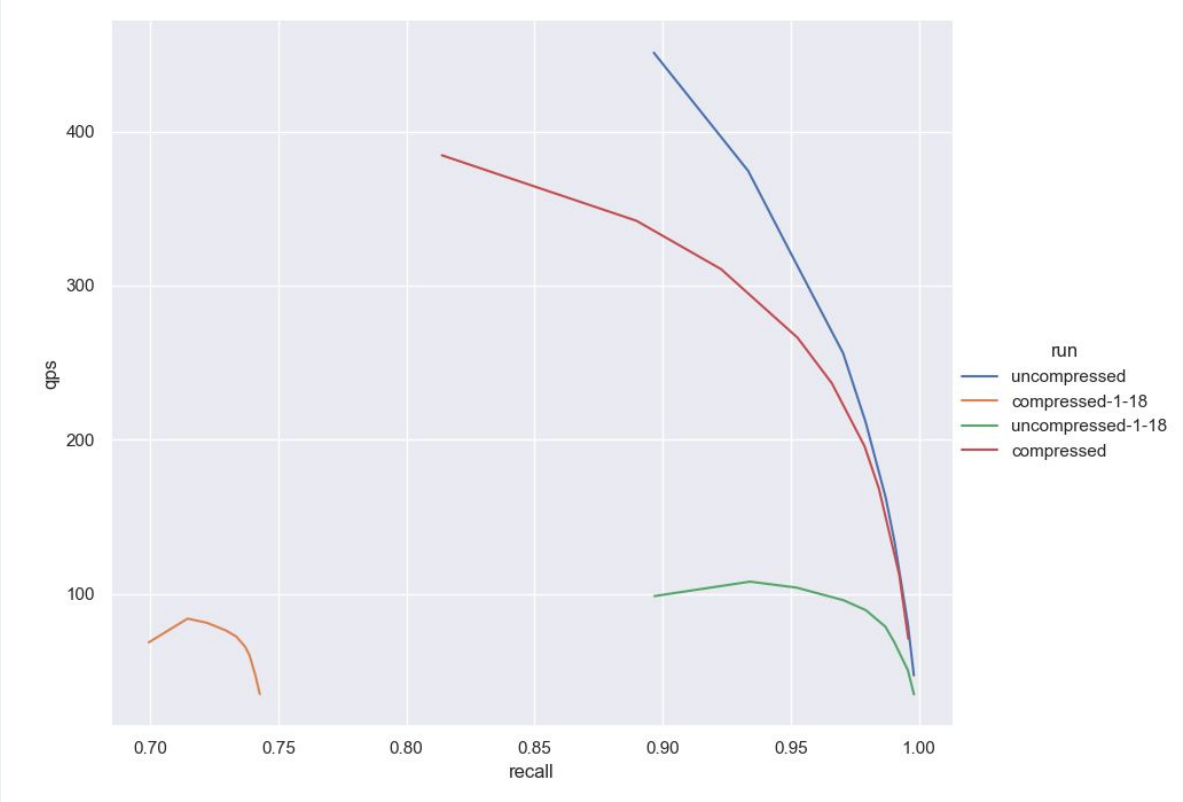
Memory Savings

Dataset	Compressed (MB)	Uncompressed (MB)
Sphere	725 (15%)	5014
DBPedia	900 (14%)	6333

Indexing is faster

Dataset	Compressed (h)	Uncompressed (h)
Sphere	0:26:42	0:30:03
DBPedia	0:51:34	0:52:44

Some Recall price



How to use PQ?
As easy as 1–2–3

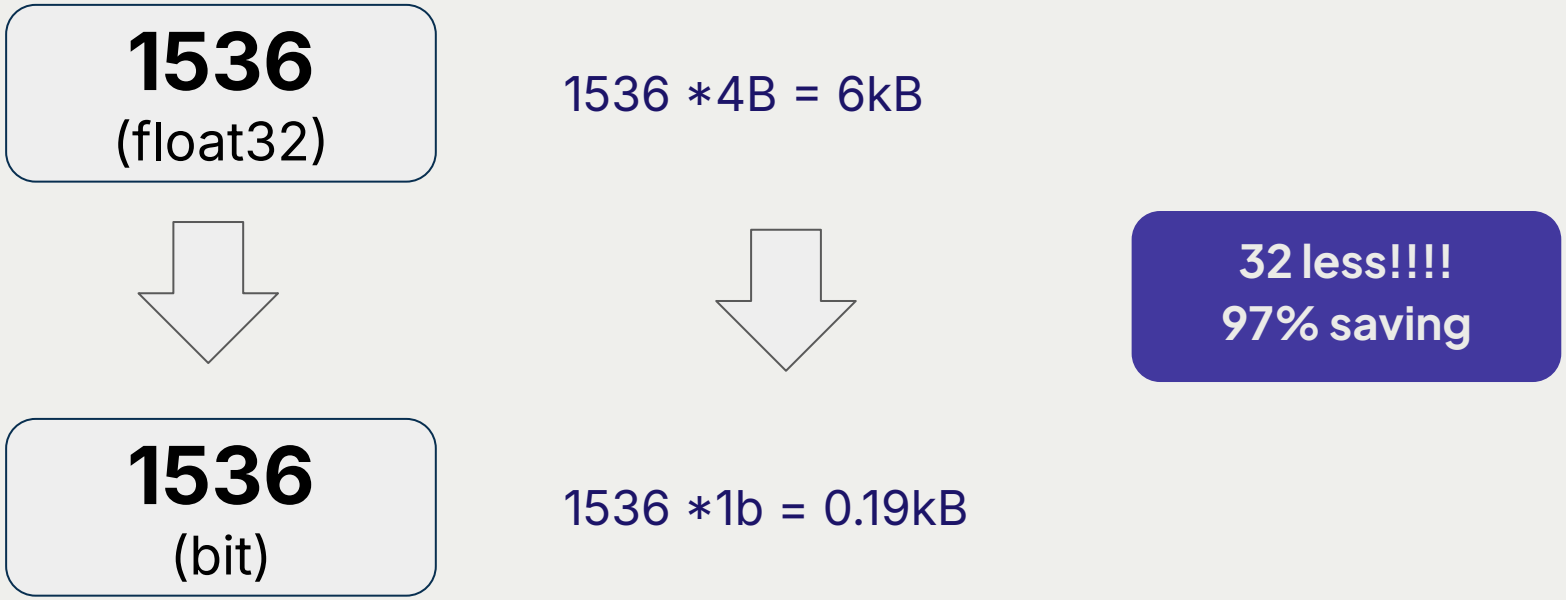
Enable PQ

```
articles.config.update(
    vector_index_config=Reconfigure.vector_index(
        pq_enabled=True,
        pq_training_limit=10000, # No of training vectors. Def: 100k
        pq_segments=96 # No of segments to break the vector into
    )
)
```

Binary Quantization (BQ)

BQ

Compress vectors into binary representation



Binary Quantization (BQ)

Vector Compression with Flat index

BQ

For small indexes
Under 100k

Great with tenants
For small tenants

Minimal RAM required
Vectors stored on disk

Enable BQ

```
client.collections.create(
    name="Articles",
    vectorizer_config=wvc.config.Configure.Vectorizer.text2vec_aws(),

    vector_index_config=wvc.config.Configure.VectorIndex.flat(
        distance_metric=wvc.config.VectorDistances.COSINE,
        vector_cache_max_objects=1000000,
        quantizer=Configure.VectorIndex.Quantizer.bq()
    ),
)
```

Hands on

Part 6
Wikipedia with PQ

Thank you

Connect with us!

weaviate.io

weaviate/weaviate

weaviate_io

Where to learn more

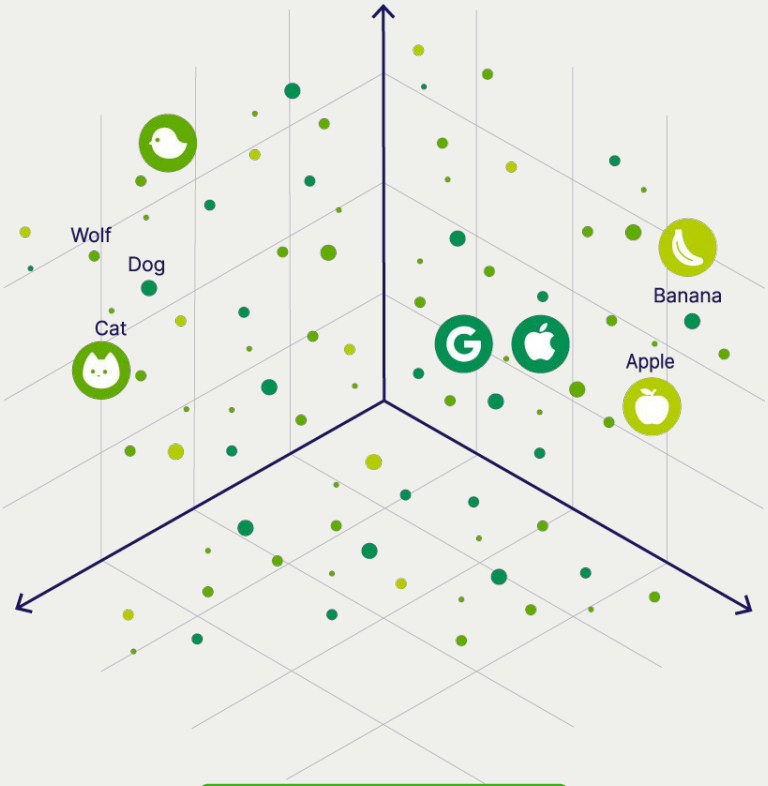
- weaviate.io/developers/weaviate/quickstart
- weaviate.io/developers/academy
- weaviate.io/blog

- github.com/weaviate/recipes
- github.com/weaviate/Verba
- github.com/weaviate-tutorials/multimodal-workshop

BONUS MultiModal Search

Multi-modal Search

Multi-modal Search

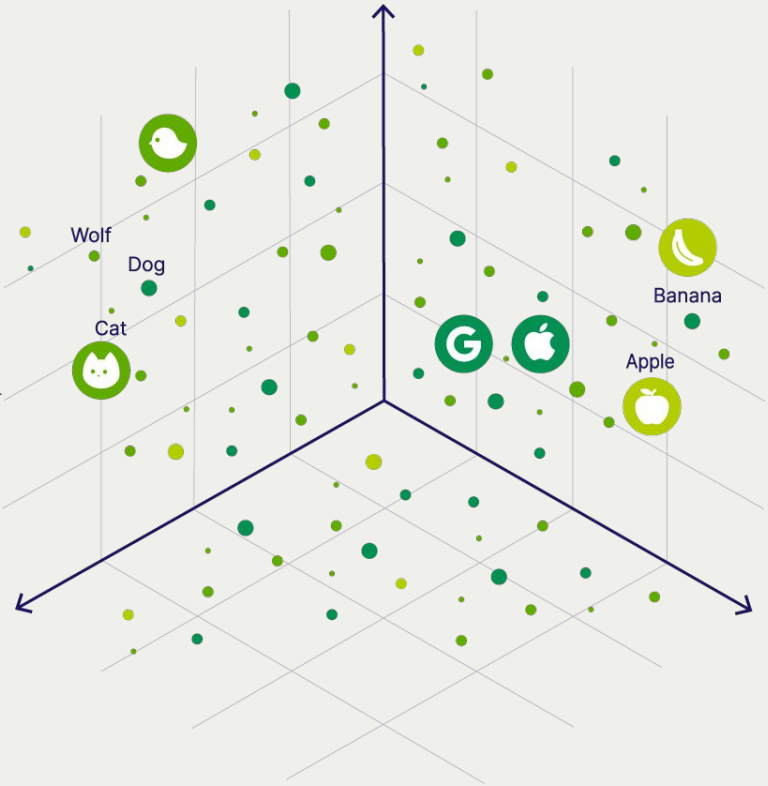


Vector Space

Multi-modal Search

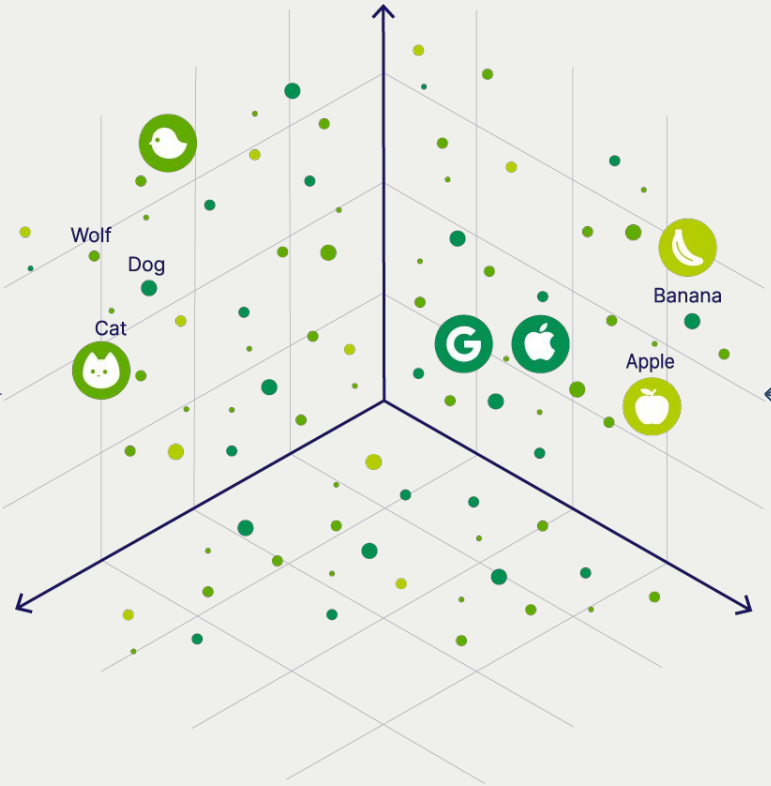
Query

"Lions roam the savannas"



Multi-modal Search

"Lions roam the savannas"

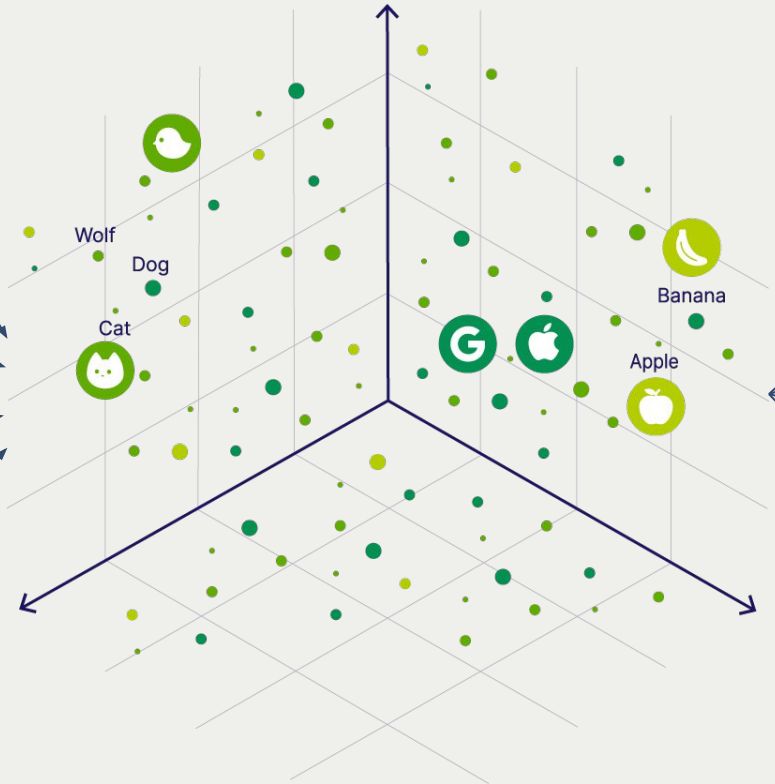


Results

Multi-modal Search

any to any

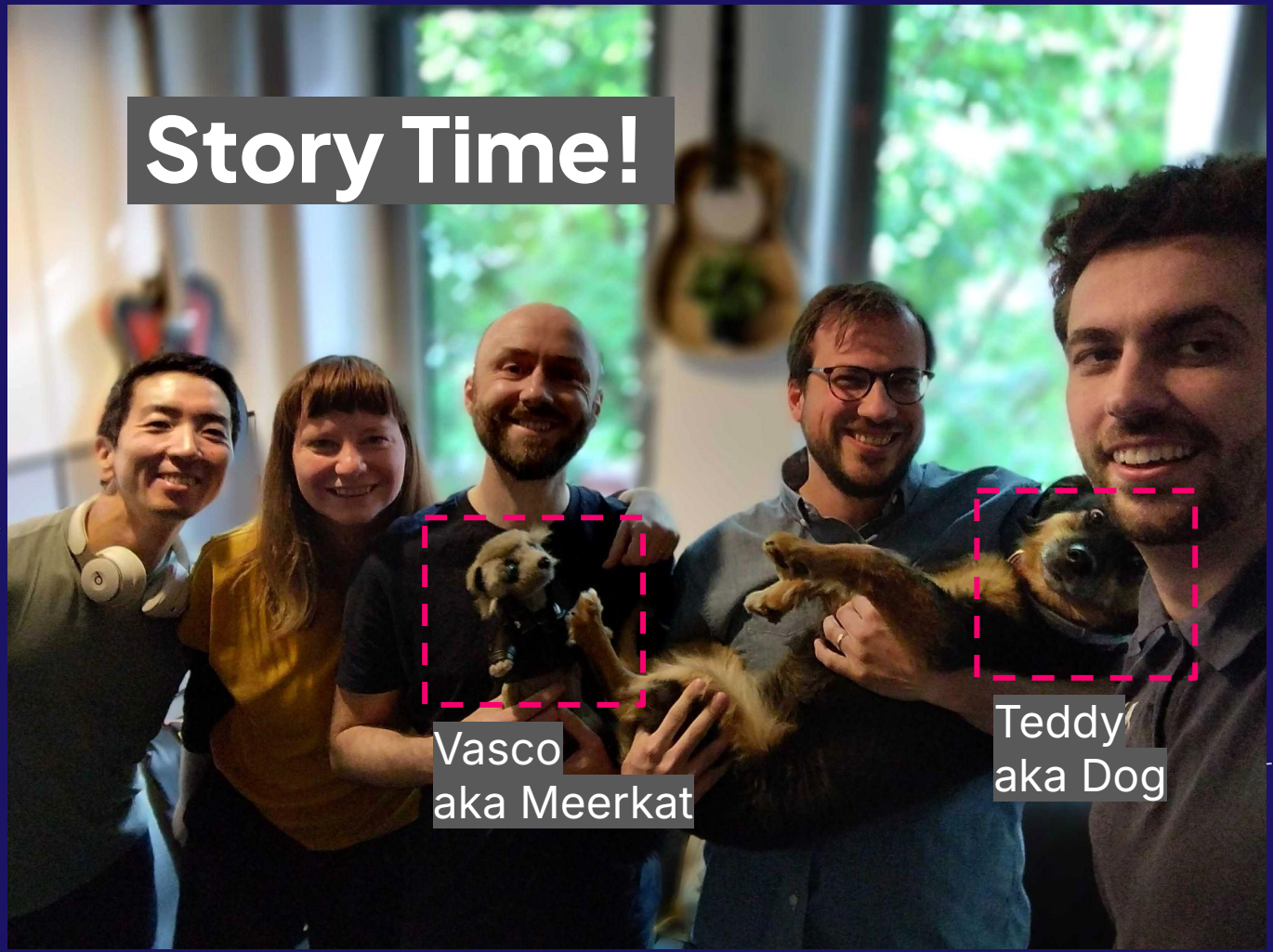
"Lions roam the savannas"



Story Time!

Story Time!

Story Time!



Text input (same thing)

```
res = client.collections.get("MyResources")

response = res.query.near_text(
    query="lions roaming the savanah",
    limit=4,
)
```

Image input

```
from pathlib import Path
res = client.collections.get("MyResources")

response = res.query.near_image(
    near_image=Path("./happy-puppy.png"),
    limit=4
)
```

Video input

```
from pathlib import Path
res = client.collections.get("MyResources")

response = res.query.near_media(
    near_image=Path("./running-lion.mp4"),
    media_type=NearMediaType.VIDEO,
    limit=4
)
```

Deployment Options

(Open Source)
Docker & Kubernetes

AWS Marketplace

Weaviate Cloud Services (WCS)

Embedded Weaviate

ML exists beyond Python

Language Clients

- Python*
- JavaScript/TypeScript*
- Go
- Java

Many ways to Search

Demo!

MM Demo 👨‍🍳
MM RAG

NSW / HNSW
A closer look

https://docs.google.com/presentation/d/1WdkcAWyWWt0UQi42pQi26s0urrl8lSSL

Lower

ef
(query param)

Higher

❌ Lower Recall (query accuracy)

✅ Faster Queries

✅ Higher Recall (accuracy)

❌ Slower Queries

Over 512 - diminishing return

Lower

efConstruction
(index param)

Higher

✅ Faster import

❌ Smaller impact on Query Recall

❌ Slower import

✅ Improves Query Recall
(without sacrificing query speed)

✅ Outcome ⇒ can reduce ef

Lower

maxConnections
(index param)

Higher

✅ Faster import
✅ Less RAM

✅ (A bit) Faster Queries
Worse Recall

❌ Slower import
❌ More RAM

❌ Slower Queries
✅ Better Recall

Tuning the setup