AI BUILDERS SUMMIT

Workshop | RAG

JP Hwang

Technical Curriculum Developer
Weaviate

Database Patterns for RAG: Single Collections vs Multi-tenancy

VIRTUAL JAN 15 - FEB 6

Database Patterns for RAG: Single Collections vs Multi-tenancy:

Prerequisites

- **Docker Desktop**
  - https://www.docker.com/products/docker-desktop
- **Python 3.9+**
  - Jupyter Notebook
- **Ollama (Optional) or Cohere API access**
  - If neither is available, the instructor will share a temporary Cohere API key during the session
- **Instructions available online**
  - https://github.com/weaviate-tutorials/odsc-ai-builders-2025

Agenda

1 RAG overview

Retrieval augmented generation - the *what* and *why*

2 The two patterns

Introduction to single collection vs multi-tenancy architectures

3 Hands-on 1

Build a RAG system with a single collection type architecture

4 Hands-on 2

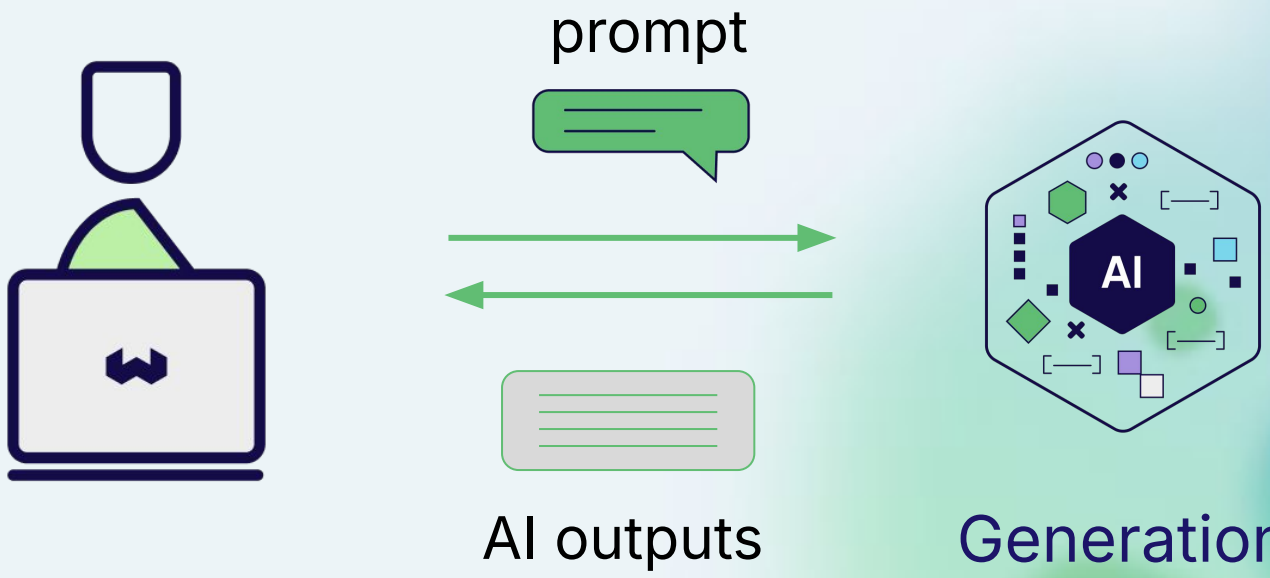
Build a RAG system with a multi-tenant architecture

5 Recap & comparisons

When to choose each one?
Implications of choices

RAG: Overview

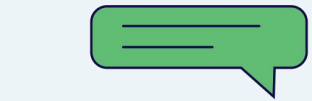
Large Language Model

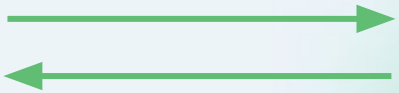


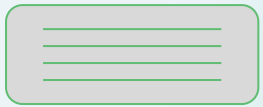
Large Language Model

Explain vector search like I am 5!

prompt



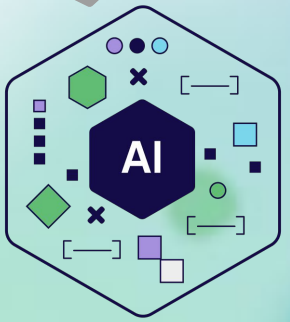




AI outputs

It's like giving each toy a special "fingerprint" ...

A magic helper that can quickly look at ALL the toys' "fingerprints" at once and tell you which ones are most similar to your query

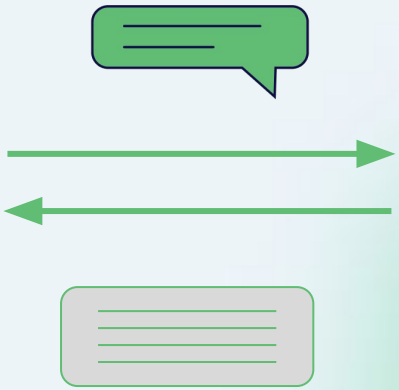


Generation

Large Language Model

How many world championships has Max Verstappen won?

prompt



AI outputs

Max Verstappen has won 3 Formula One World Championships.

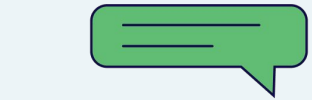
He won his first championship in 2021 ... He then dominated the 2022 and 2023 seasons.

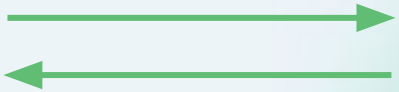
Generation

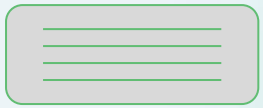
Large Language Model

Who is JP Hwang?

prompt





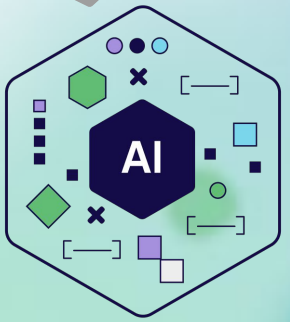


AI outputs

A South Korean-born, American entrepreneur and businessman.

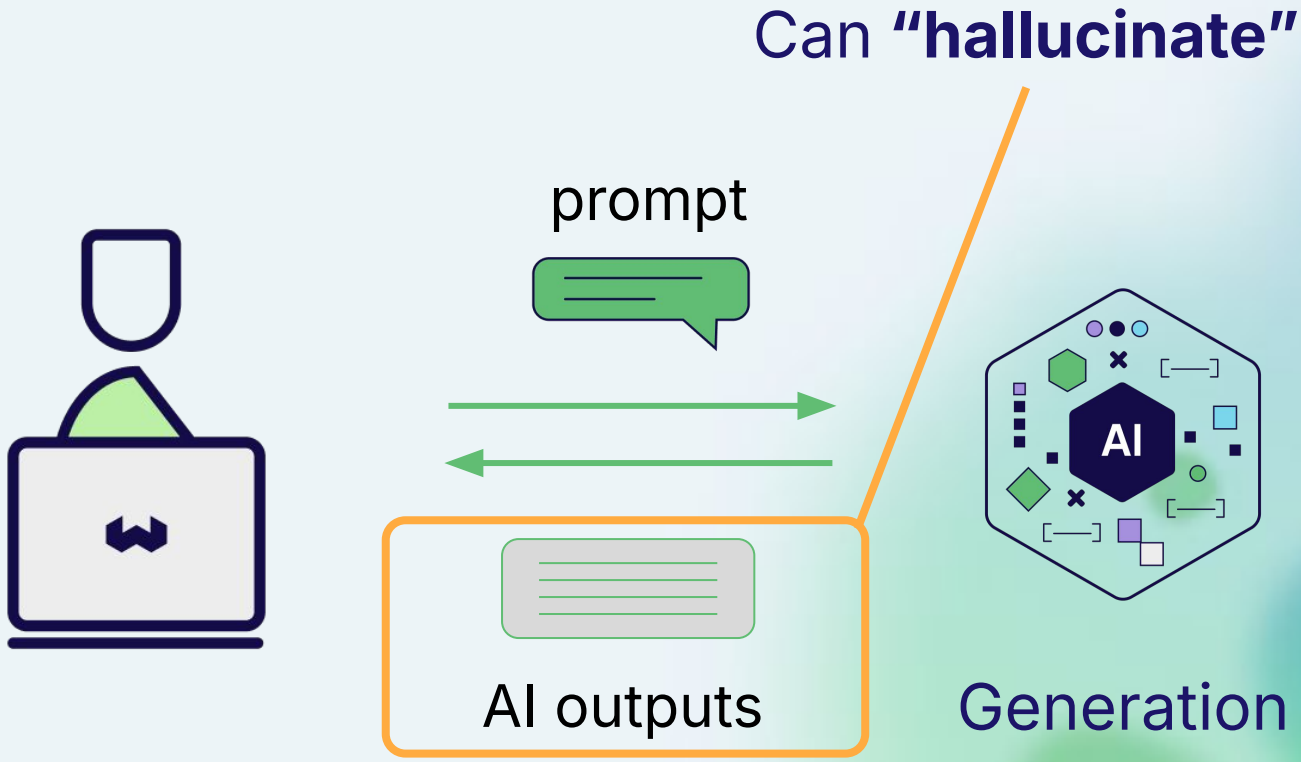
The founder and CEO of Superb Tickets, an online ticket marketplace.

...

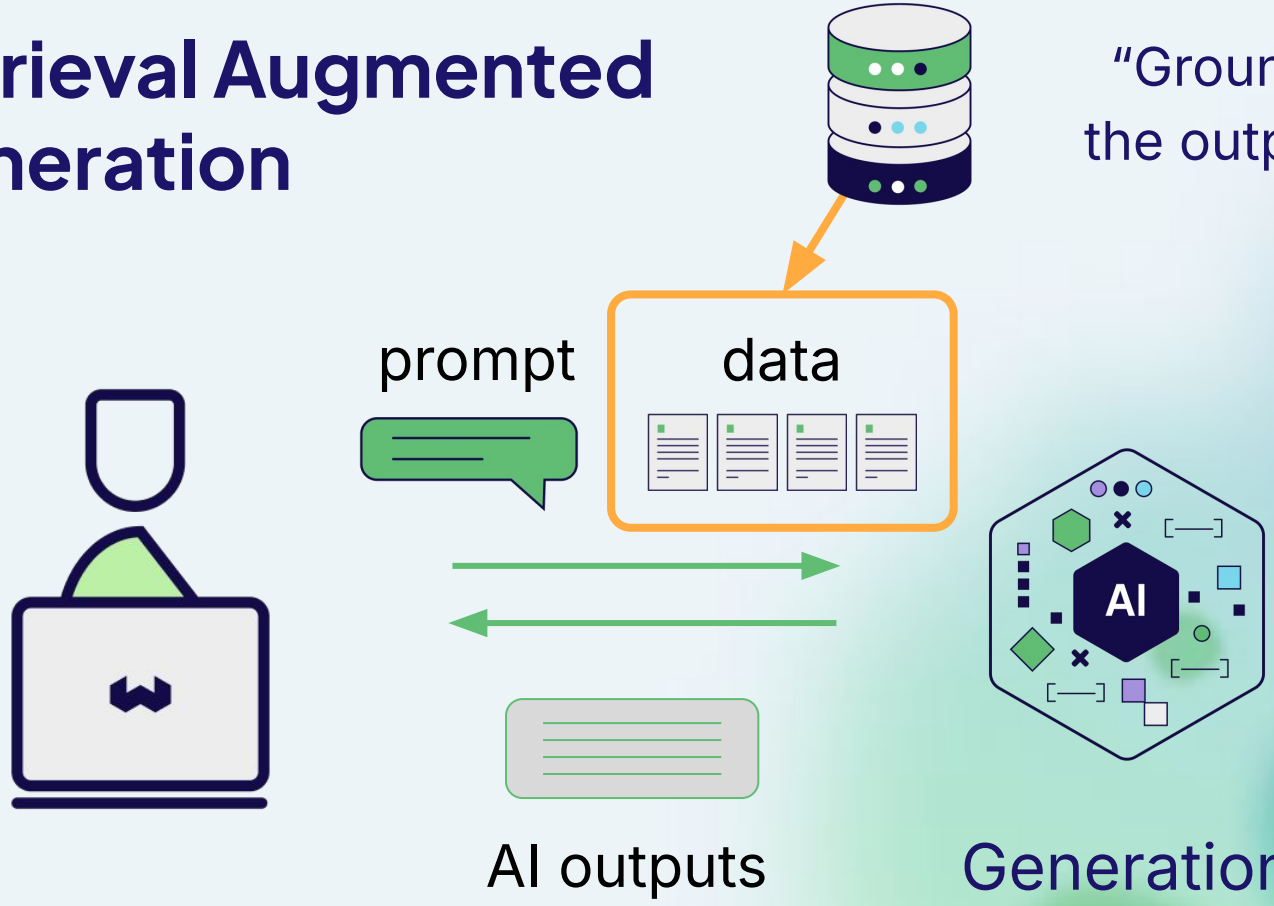


Generation

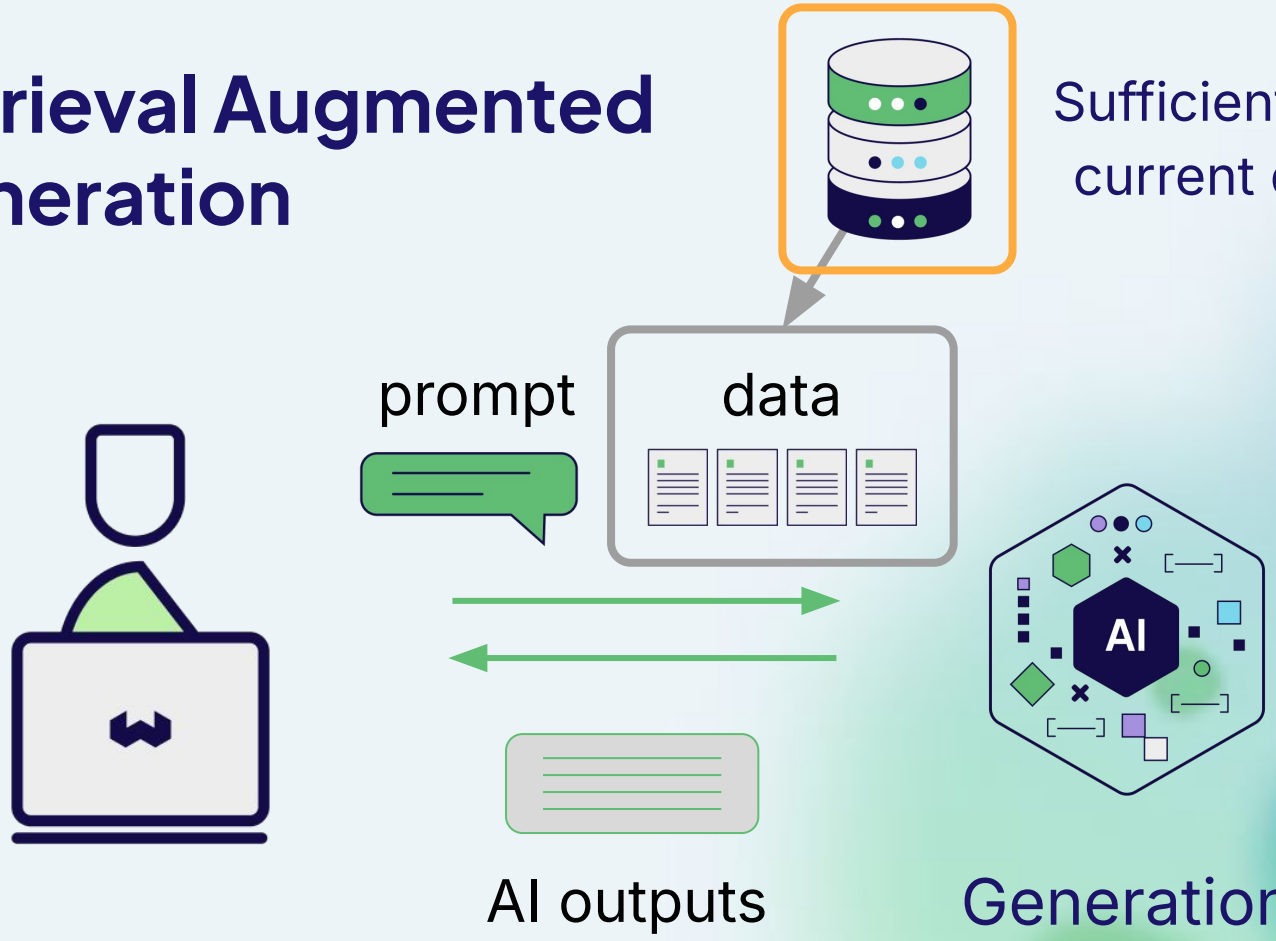
Large Language Model



Retrieval Augmented Generation



Retrieval Augmented Generation



Retrieval Augmented Generation

prompt

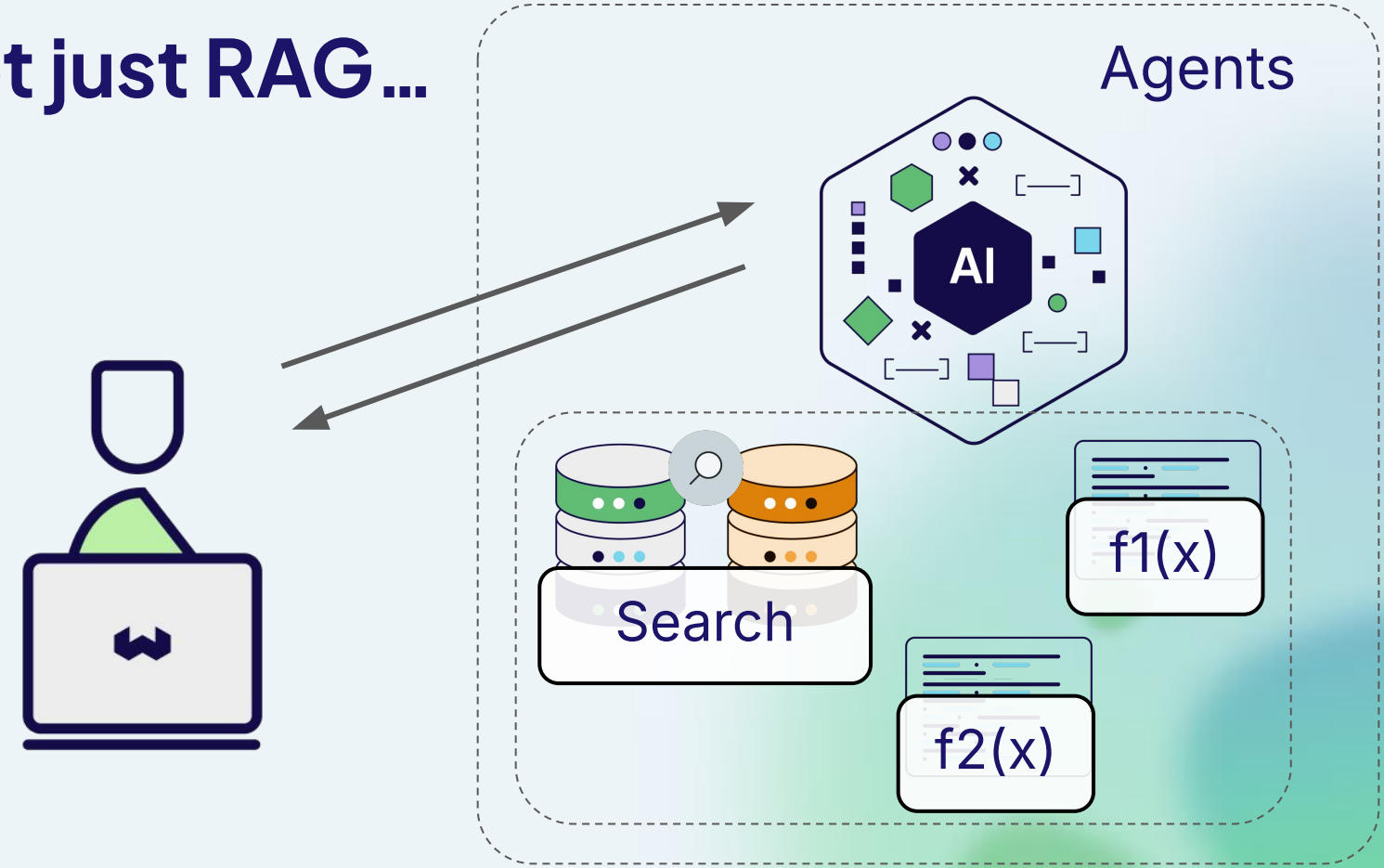
data

AI

AI outputs

Generation

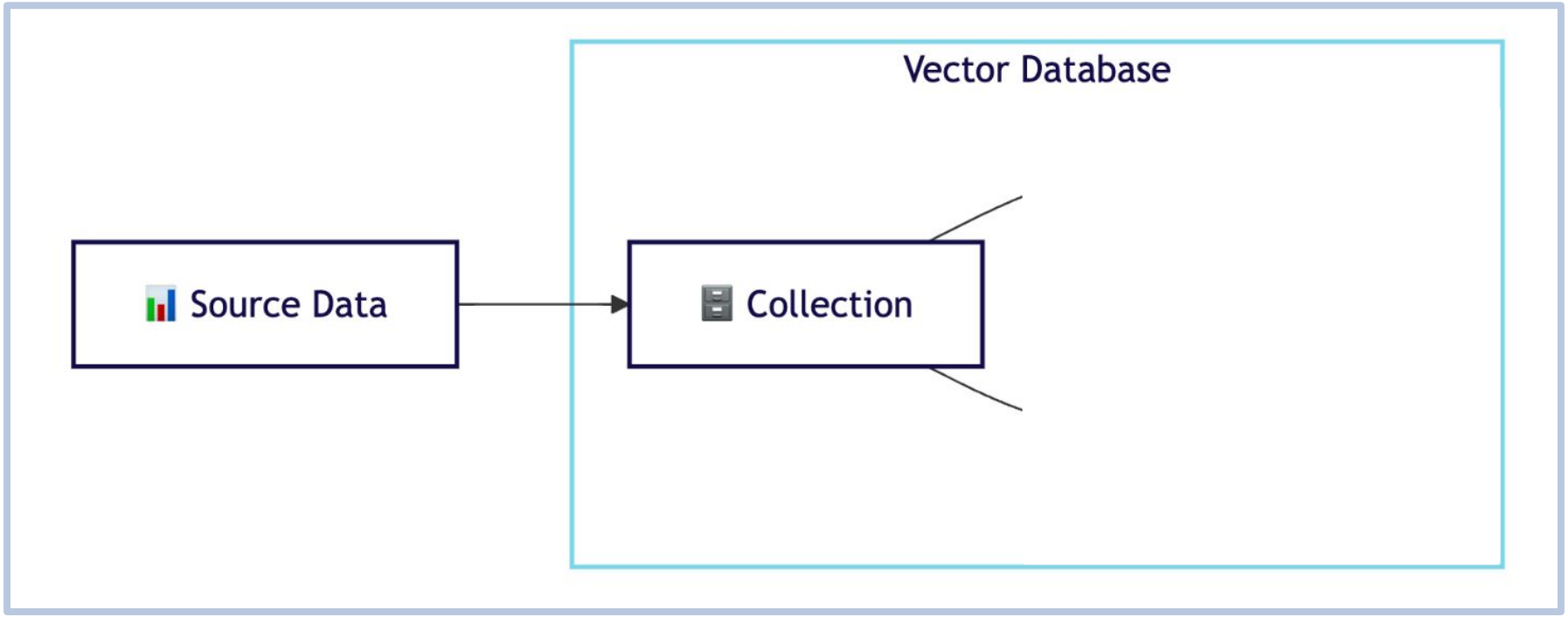
Not just RAG...



Vector Database Patterns

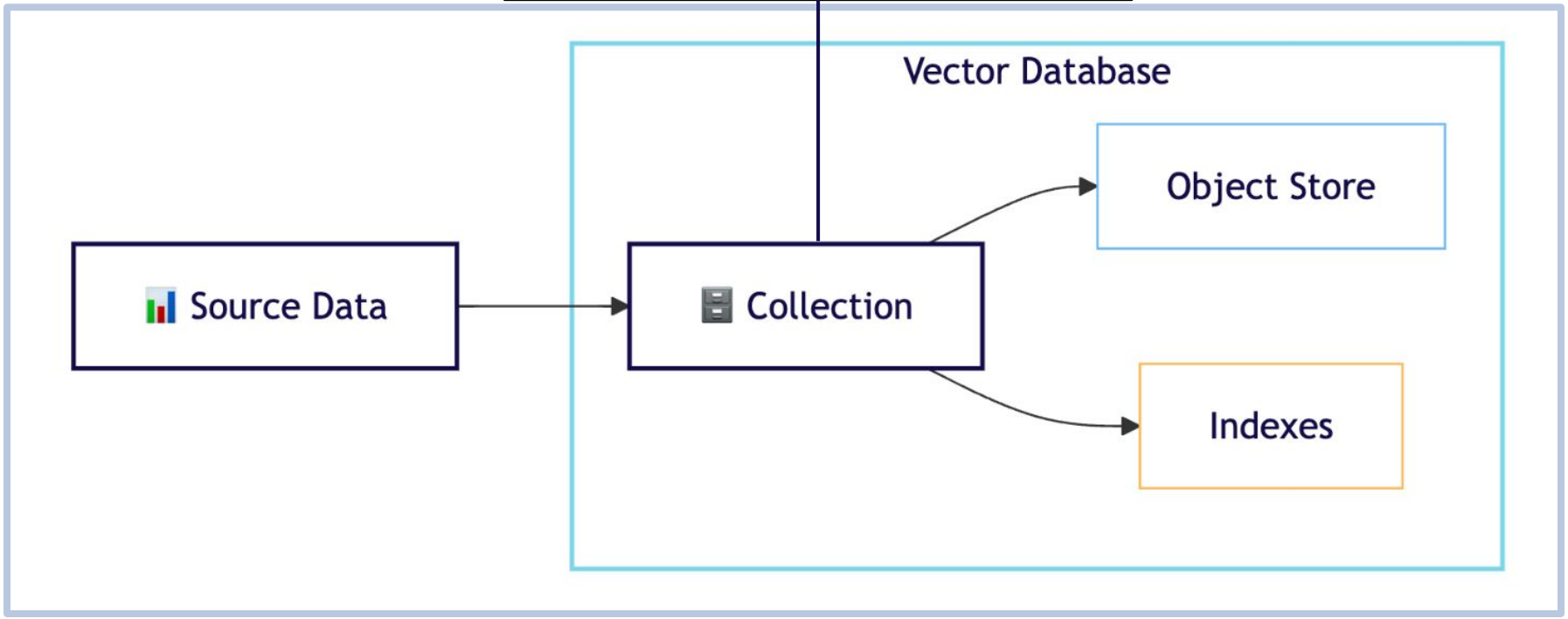
Single Collection

Single Collection

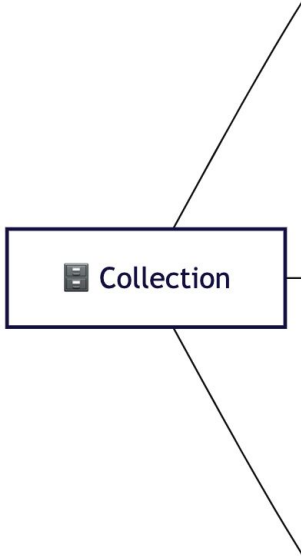


Single Collection

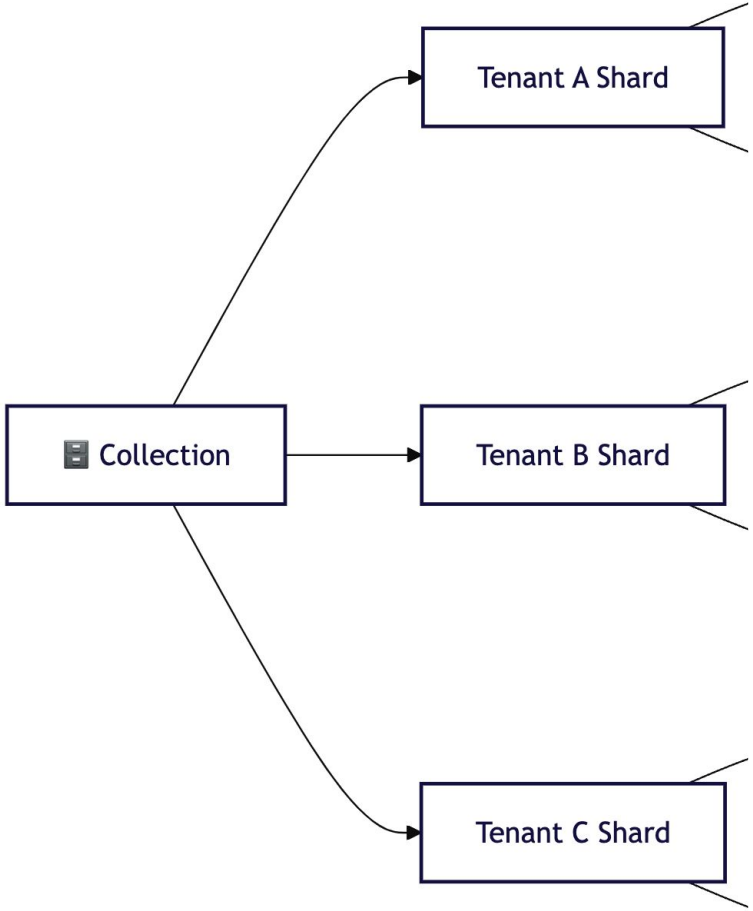
- Data schema
- Embedding model definition
- Index configurations
- Replication config
- etc.



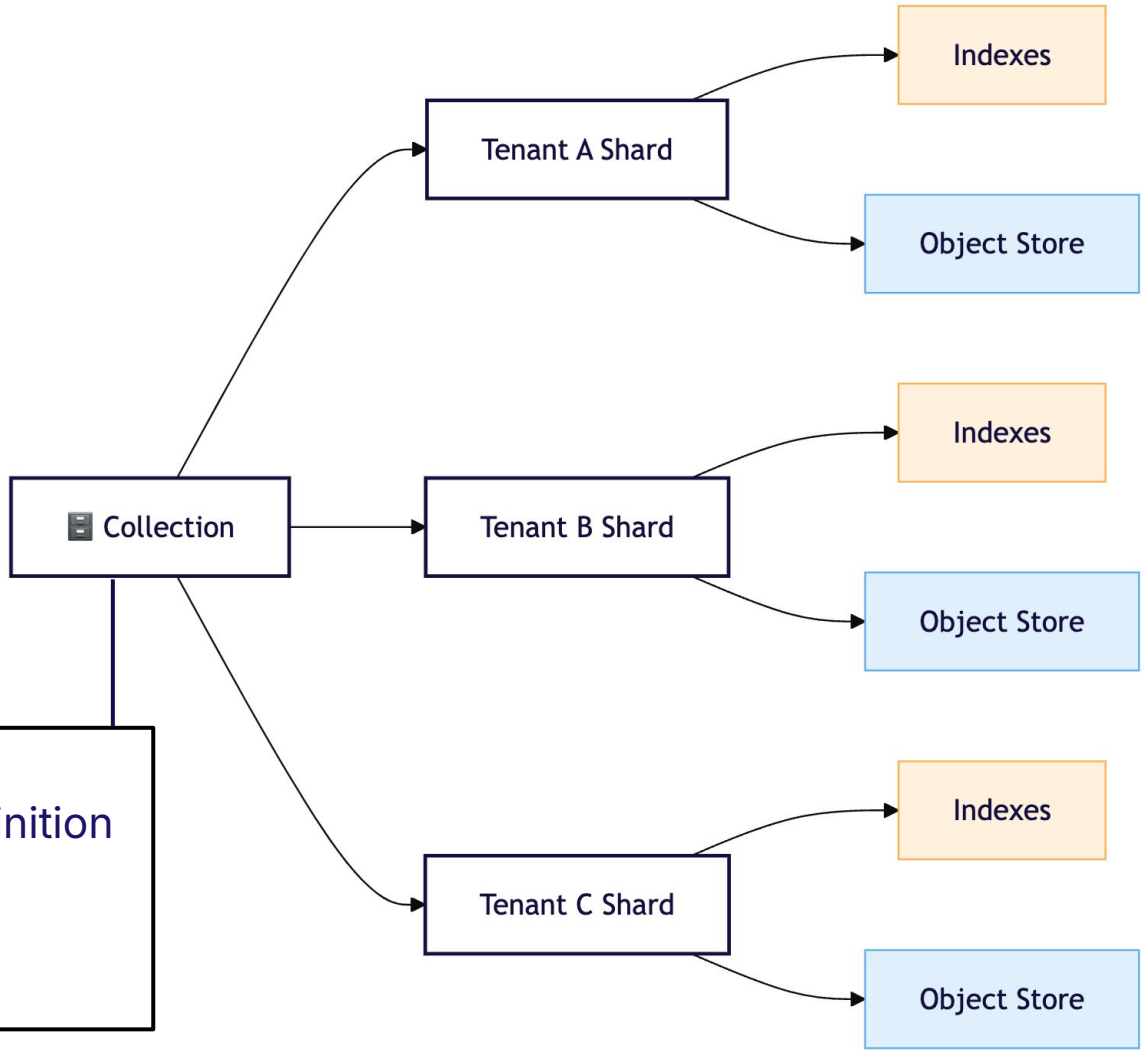
Multi-tenant Collection



Multi-tenant Collection

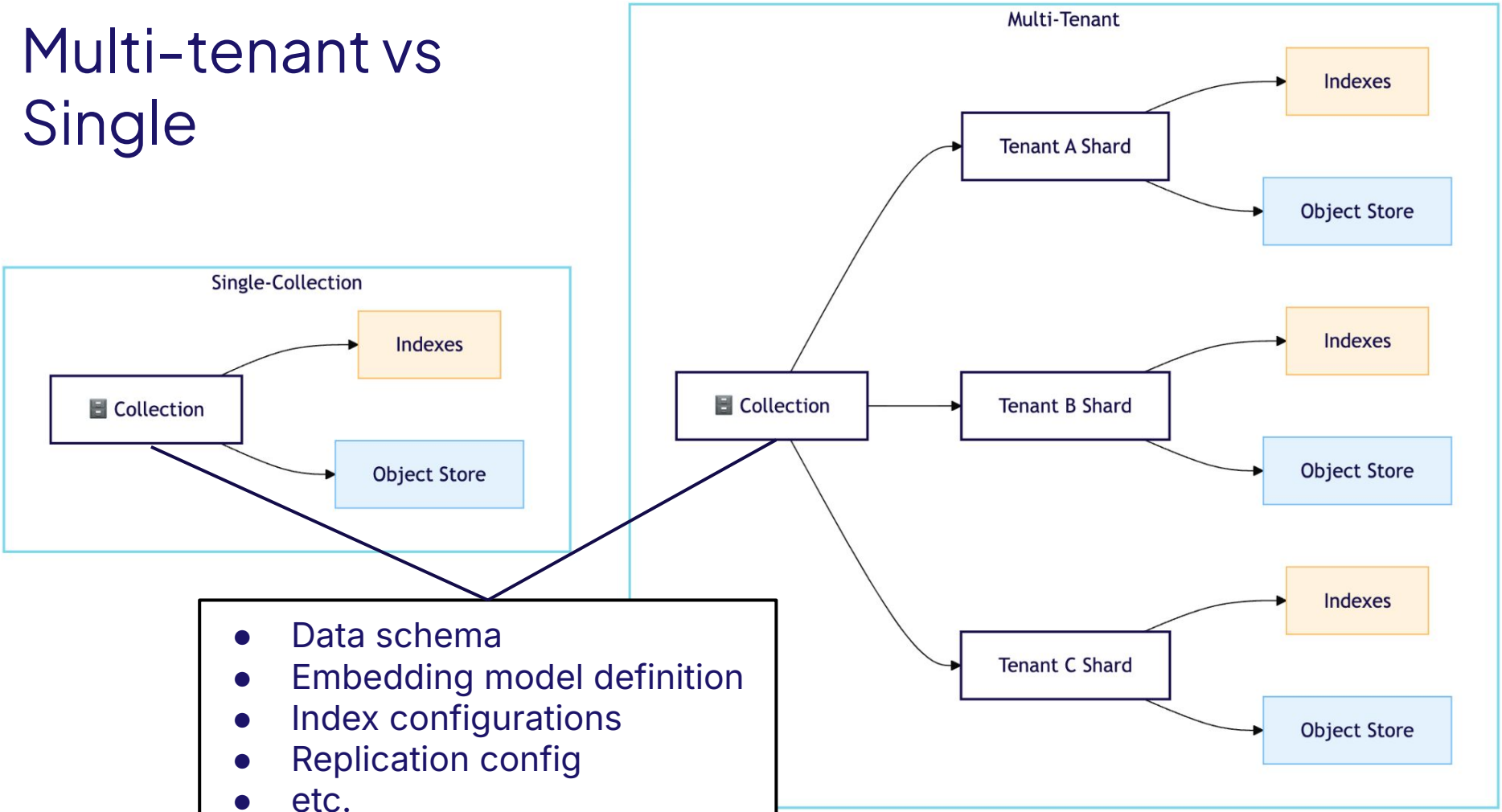


Multi-tenant Collection



- Data schema
- Embedding model definition
- Index configurations
- Replication config
- etc.

Multi-tenant vs Single

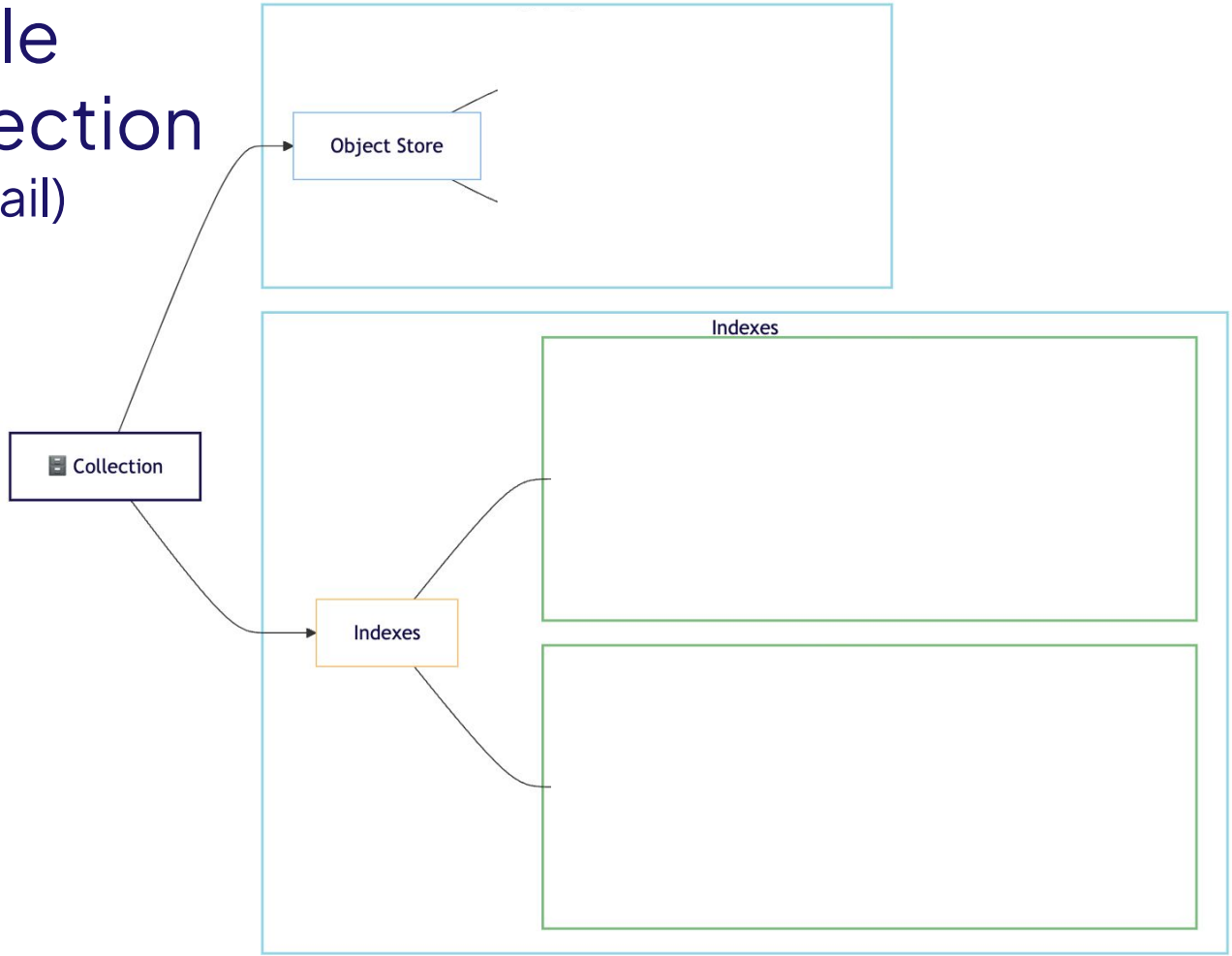


- Data schema
- Embedding model definition
- Index configurations
- Replication config
- etc.

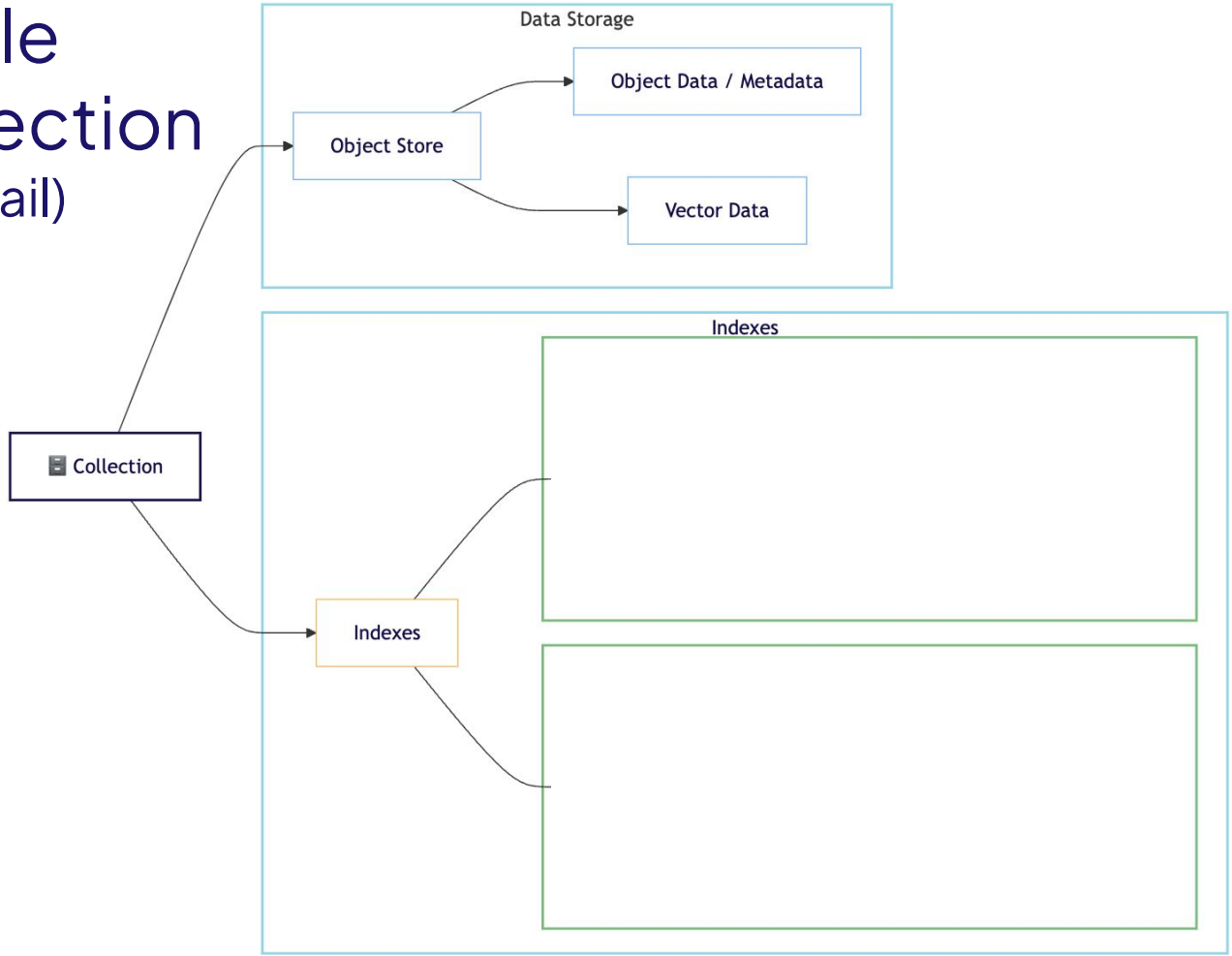
Single Collection Pattern

In detail

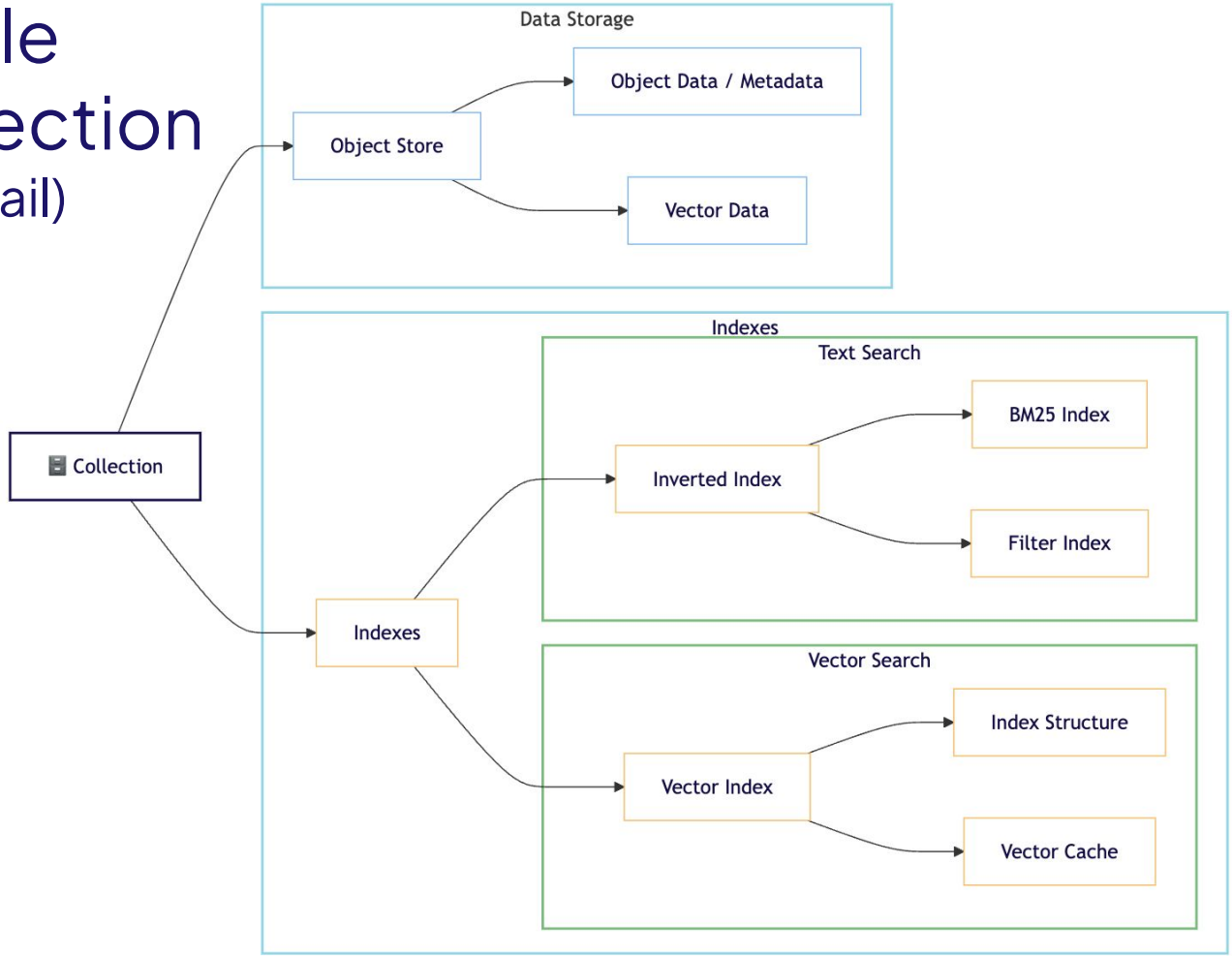
Single Collection
(in detail)



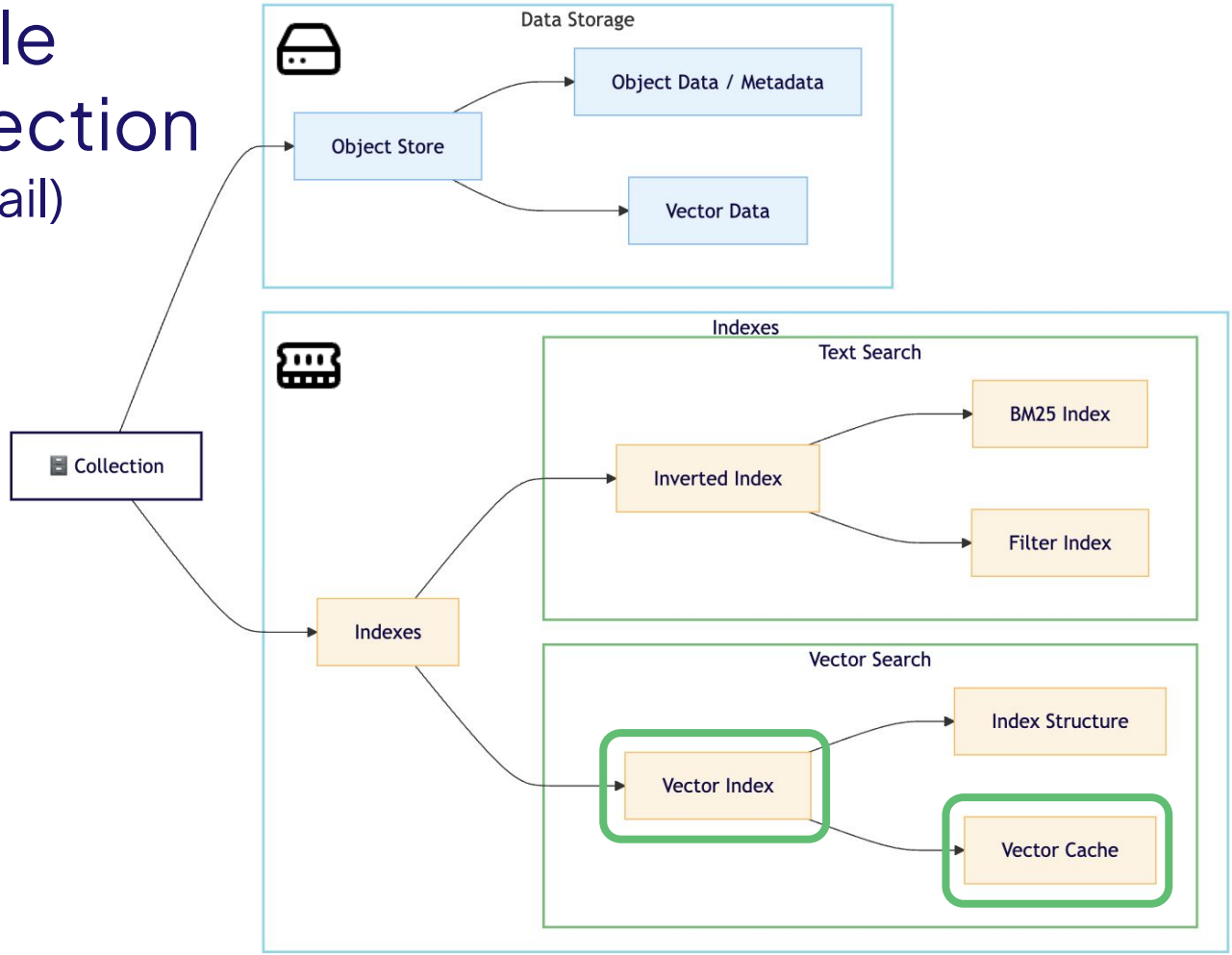
Single Collection
(in detail)



Single Collection
(in detail)



Single Collection
(in detail)



Hands-on section 1!

Single-collection DB patterns

Challenges of scale

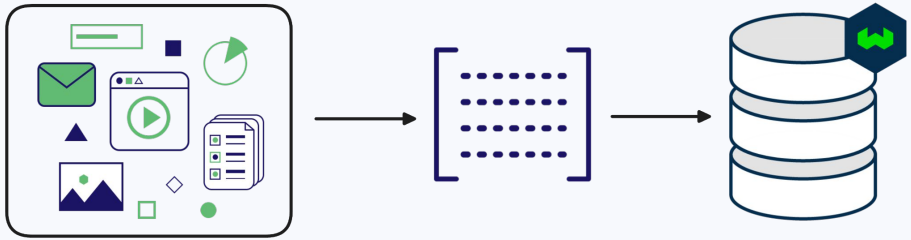
Scale: Considerations

- **Object count**: Memory & storage
- **User count**: Data management & compliance
- **Server load**: Distribute load

Managing resource requirements

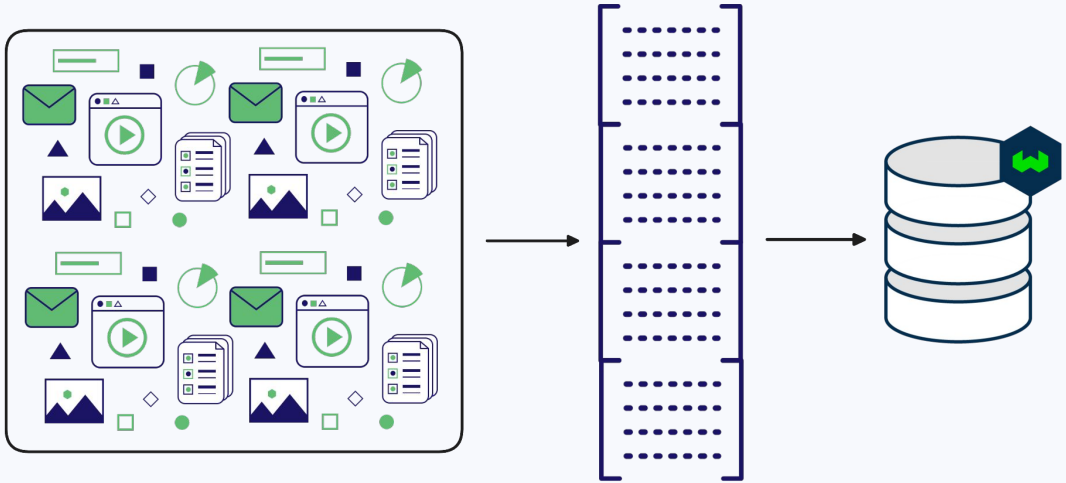
Scale: Solutions

Growth



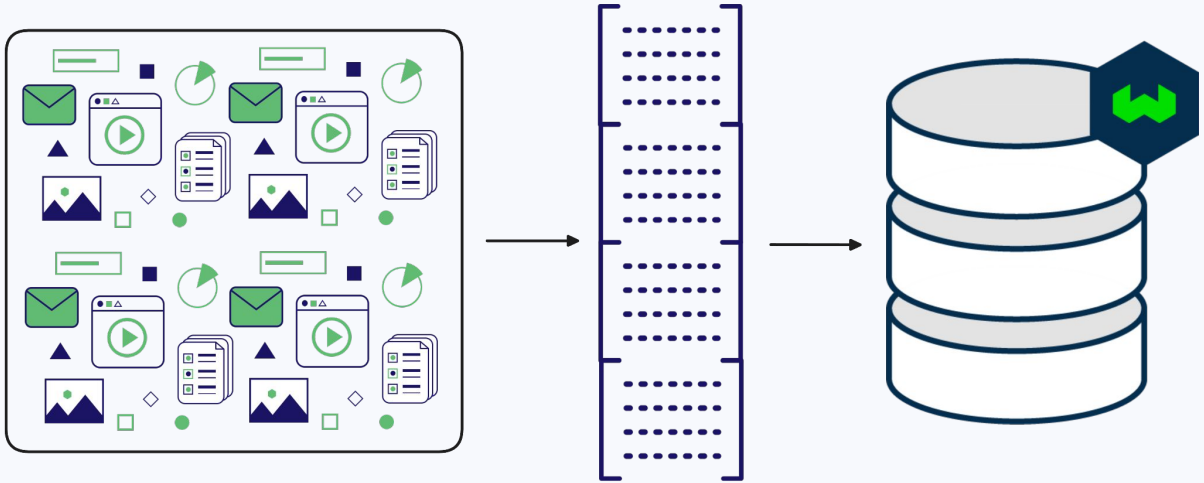
Scale: Solutions

Growth



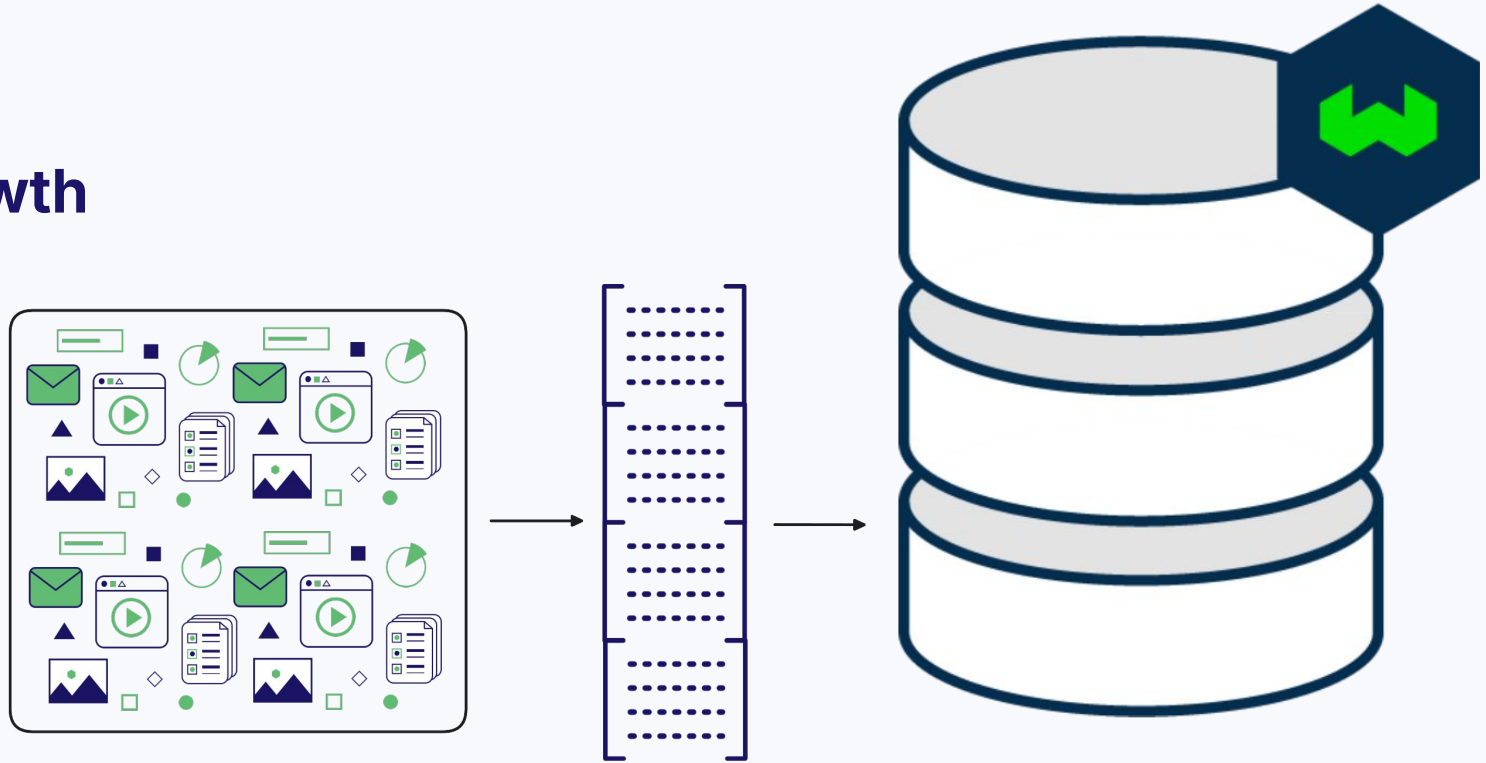
Scale: Solutions

Growth



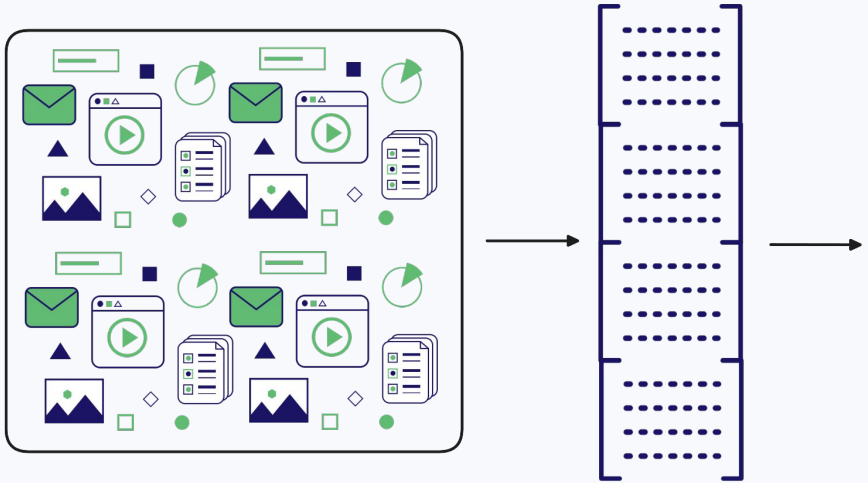
Scale: Solutions

Growth



Scale: Solutions

Growth



- Single point of failure
- Efficiency
- Upgrades
- Costs

1024 gib

19 matches

Instance name	On-Demand hourly rate	vCPU	Memory	Storage
x2gd.metal	$5.344	64	1024 GiB	2 x 1900 SSD
x2gd.16xlarge	$5.344	64	1024 GiB	2 x 1900 SSD
hpc6id.32xlarge	$5.70	64	1024 GiB	4 x 3800 NVMe SSD
x2iedn.8xlarge	$6.669	32	1024 GiB	1 x 950 NVMe SSD
x2idn.16xlarge	$6.669	64	1024 GiB	1 x 1900 NVMe SSD

AWS Spot pricing, Nov 2024

1024 gib

19 matches

Instance name	On-Demand hourly rate	orage
x2gd.metal	$5.344	1900 SSD
x2gd.16xlarge		
hpc6id.32xlarge		
x2iedn.8xlarge		
x2idn.16xlarge		

120 * 365 = **$43,800 / year!**

AWS Spot pricing, Nov 2024

Vector indexing options

Scale: Solutions

Improve efficiency - indexing

Scale: Solutions

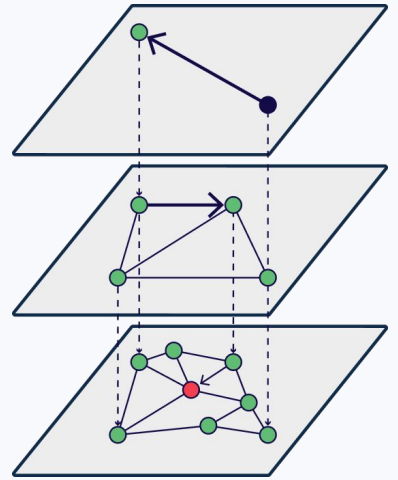
Improve efficiency - indexing

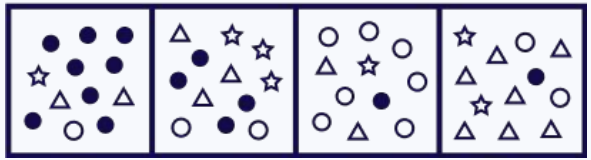
- **HNSW** index (default)
- **Flat** index

Scale: Solutions

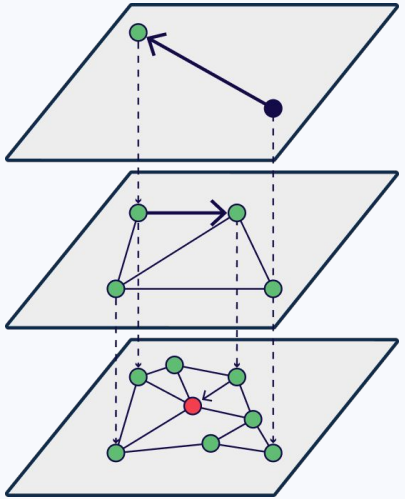
Indexes - comparison

- **HNSW**: fast + scalable
- **Flat**: tiny footprint; ~100k objs





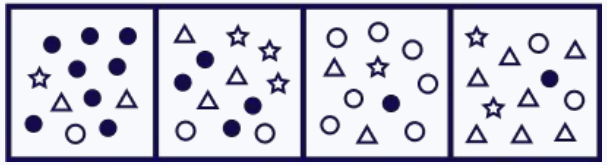
Scale: Solutions



How to choose?

- **Start** with **HNSW**

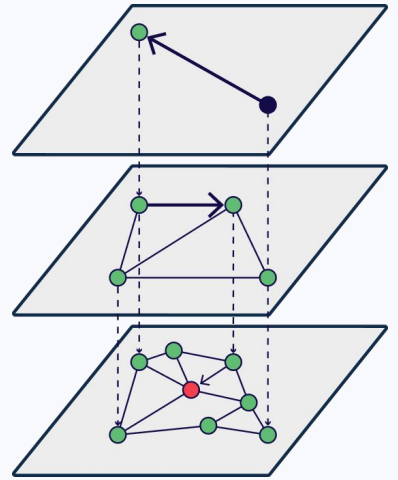
  (Tune speed / size / accuracy)
- Multi-tenancy?
  - Try **dynamic**

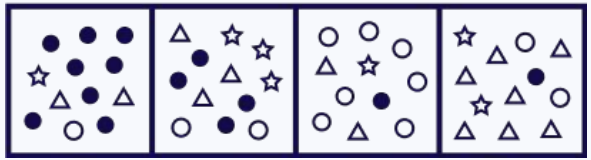


Scale: Solutions

Improve efficiency - indexing

- **HNSW** index (default)
- **Flat** index
- **Dynamic** index
  - Flat → HNSW @ threshold

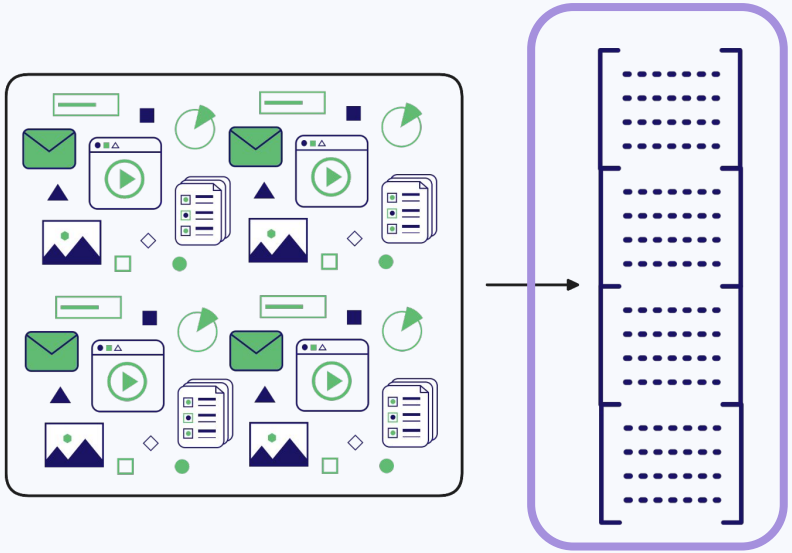




Vector quantization

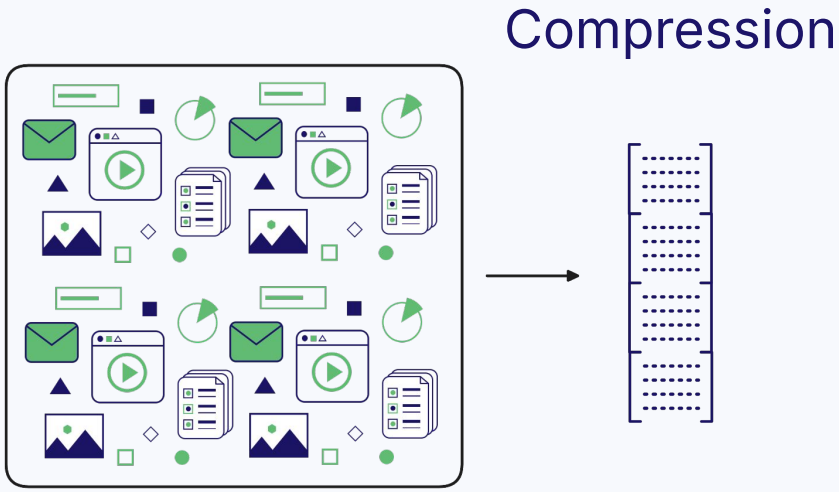
Scale: Solutions

Improve efficiency



Scale: Solutions

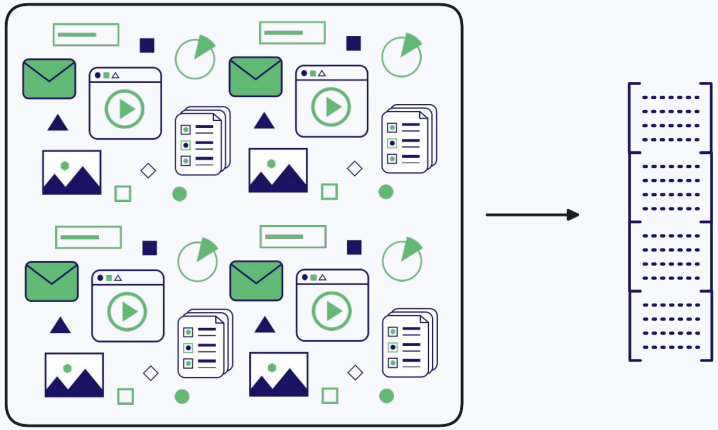
Improve efficiency



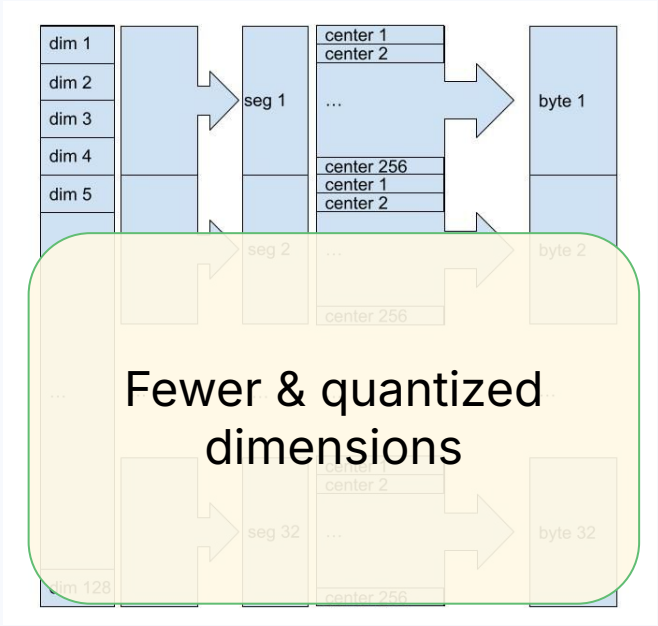
Scale: Solutions

Improve efficiency

Compression

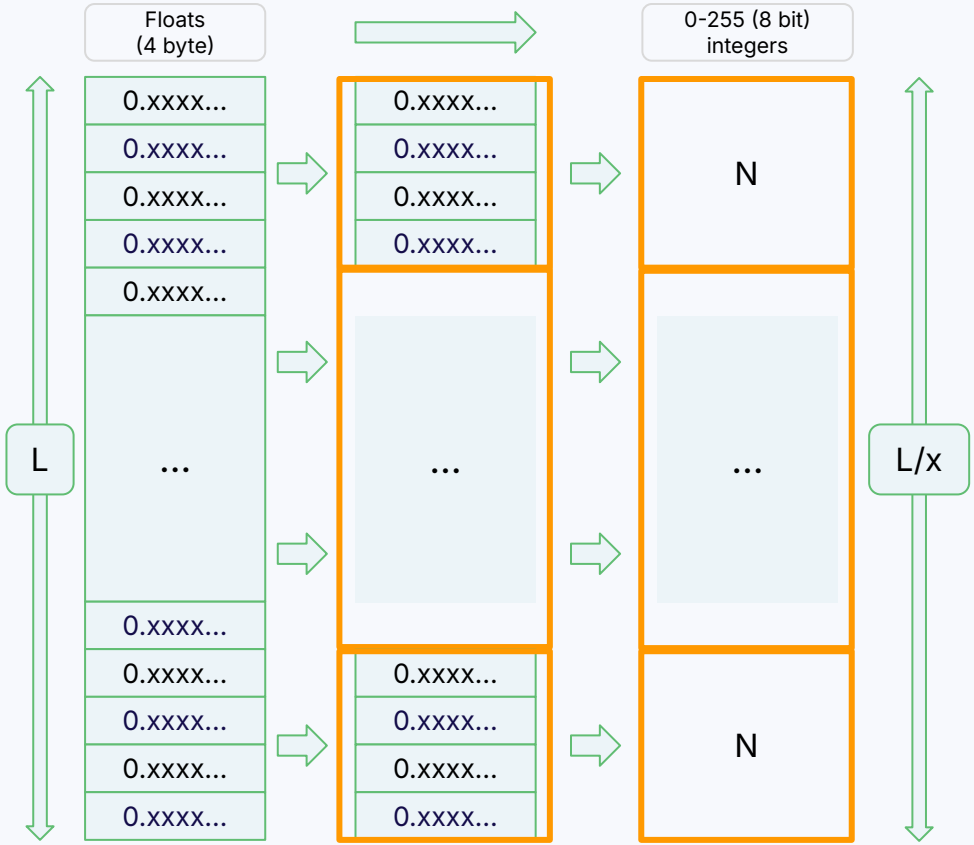


Product quantization



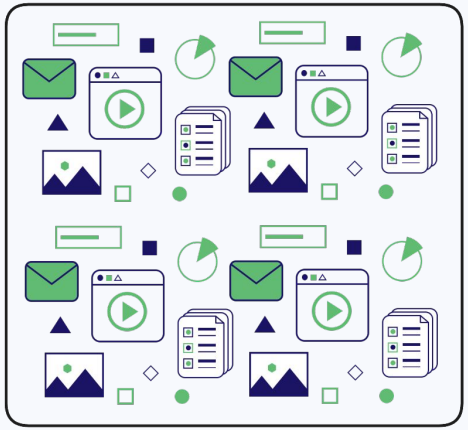
Customisable compression

(e.g. 128 floats → 32 bytes: 16x)

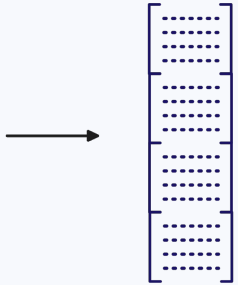

Floats
(4 byte)
0-255 (8 bit)
integers
0.xxxx...
0.xxxx...
0.xxxx...
0.xxxx...
0.xxxx...
...
0.xxxx...
0.xxxx...
0.xxxx...
0.xxxx...
0.xxxx...
0.xxxx...
0.xxxx...
0.xxxx...
N
...
N
L
L/x

Scale: Solutions

Improve efficiency



Compression



Binary quantization

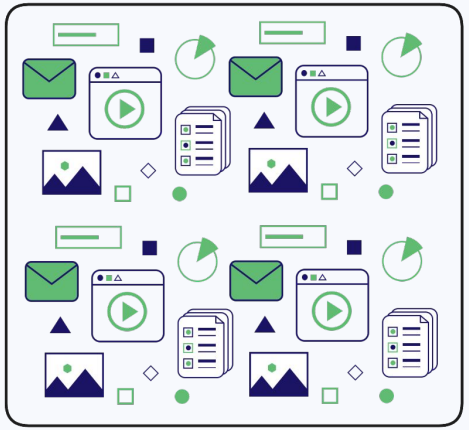
[-0.1324..., -0.9253...,
0.2389...,
...
0.3249..., 0.2390..., -0.4823...]

[0, 0, 1, 0, 1, 1,
...
0, 1, 0, 1, 1, 0]

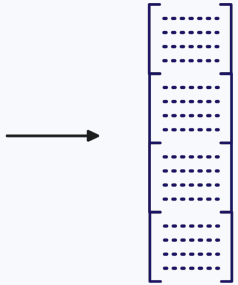
n floats → n bits
(32x reduction)

Scale: Solutions

Improve efficiency



Compression



Scalar quantization

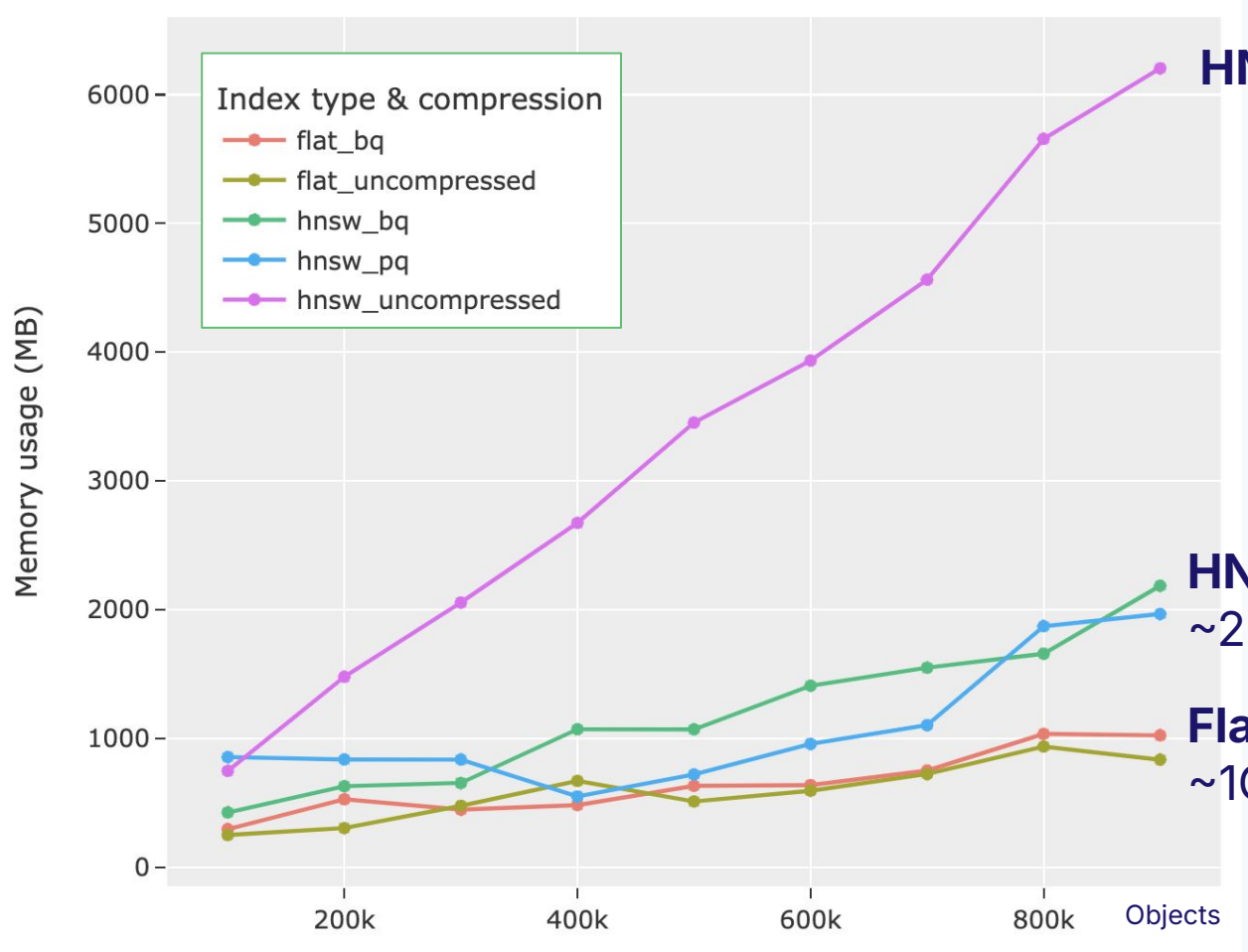
[-0.1324..., -0.9253...,
0.2389...,
...
0.3249..., 0.2390..., -0.4823...]

↓

[104, 12, 138,
...
152, 138 47]

n floats → n ints
(4x reduction)

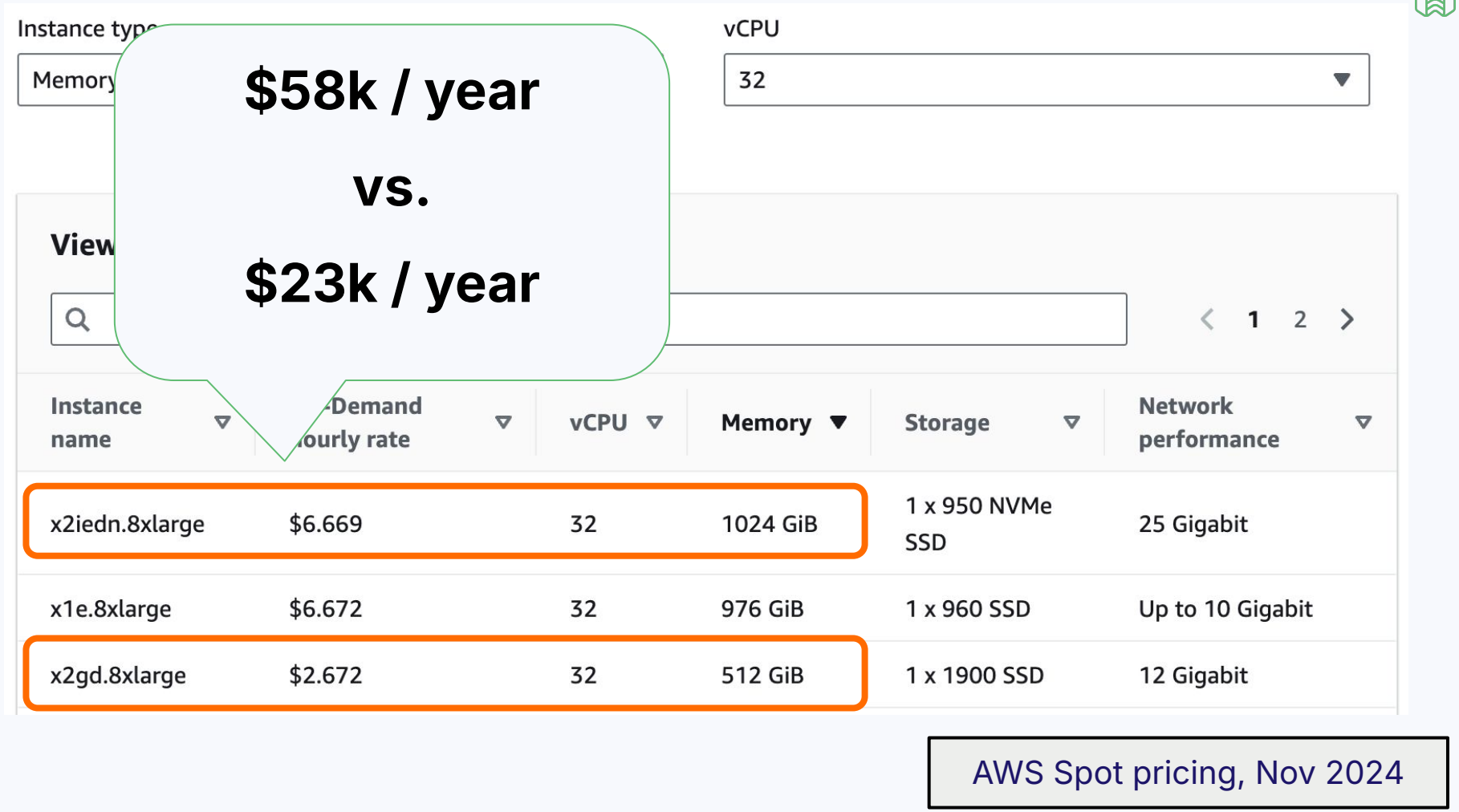
Example: Index type & quantization vs memory footprint



HNSW: ~6GB

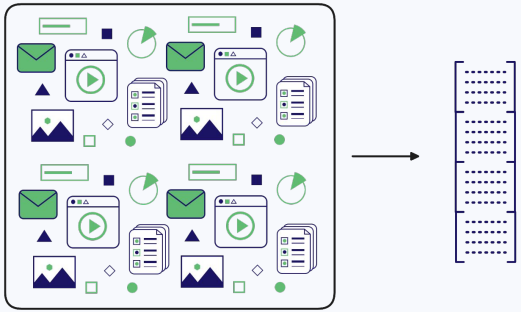
HNSW + BQ/PQ: ~2GB

Flat (+BQ): ~1GB



AWS Spot pricing, Nov 2024

Scale: Solutions



BQ / PQ / SQ compression

Search **quality** mitigated by over-fetching & rescoring

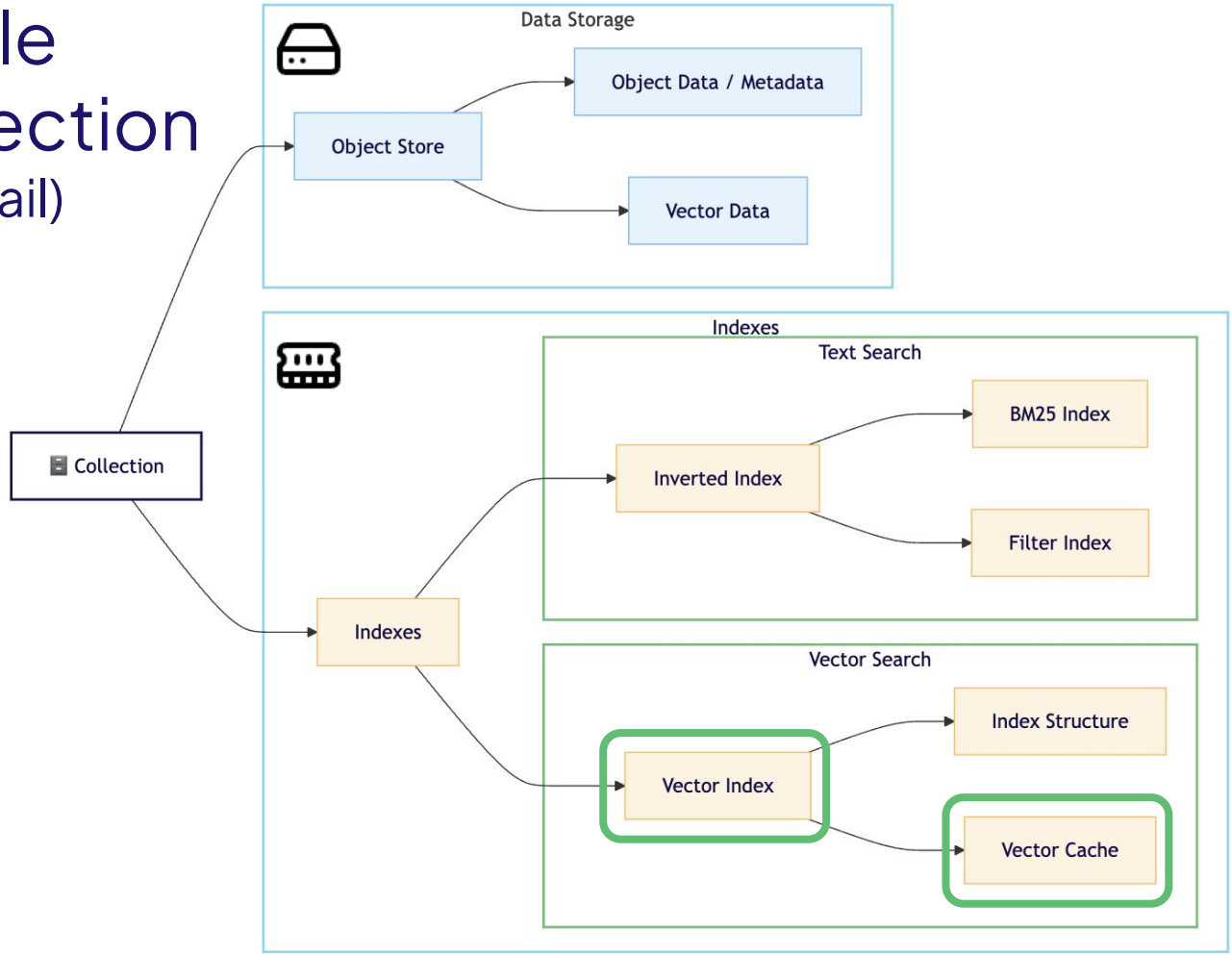
When to use which?

- Generally, try PQ first
- BQ: model-specific

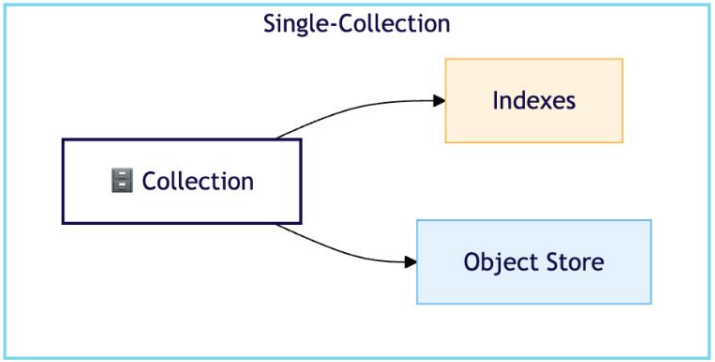
Multi-Tenancy Pattern

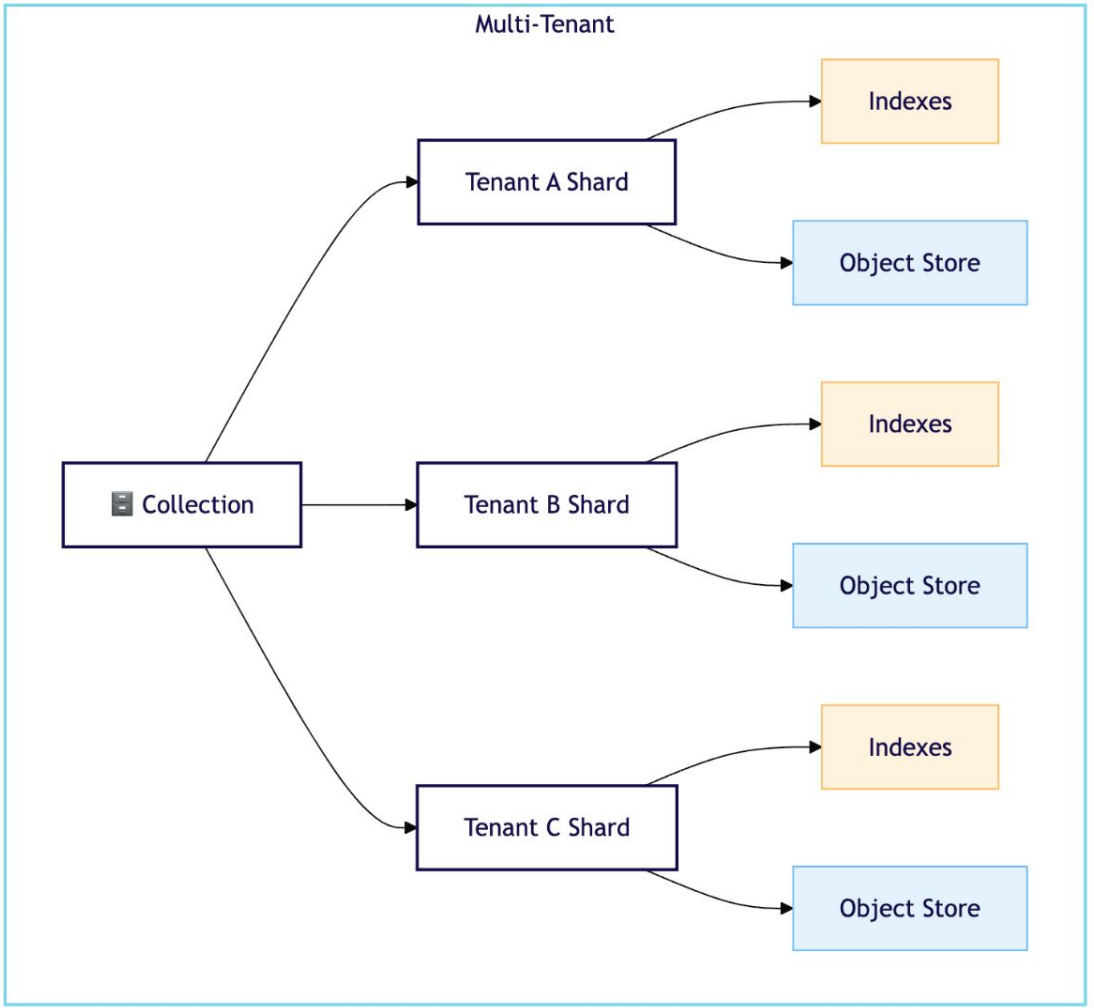
In detail

Single Collection
(in detail)



Multi-tenant vs Single



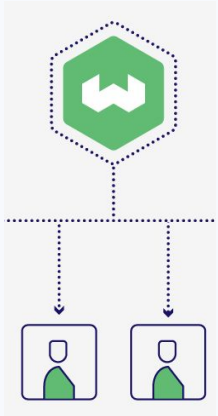


Hands-on section 2!

Multi-tenant DB patterns

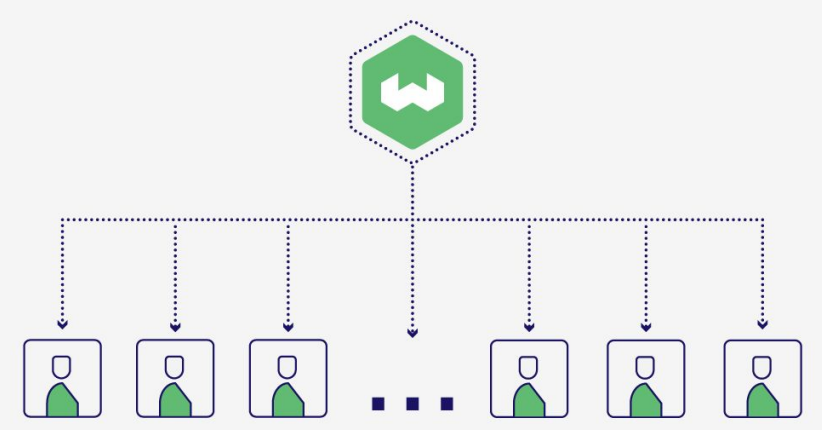
Scale: Solutions

End user growth



Scale: Solutions

End user growth



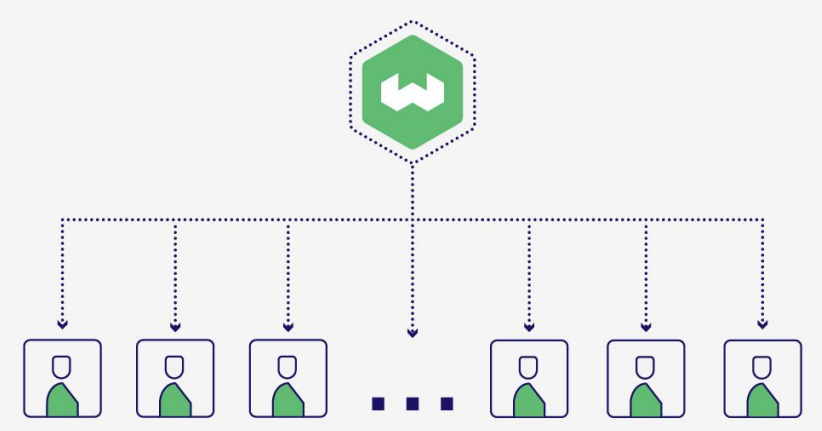
Challenges faced:

- Performance
- Data isolation
- Compliance

Developed: **Multi-tenancy**

Scale: Solutions

End user growth

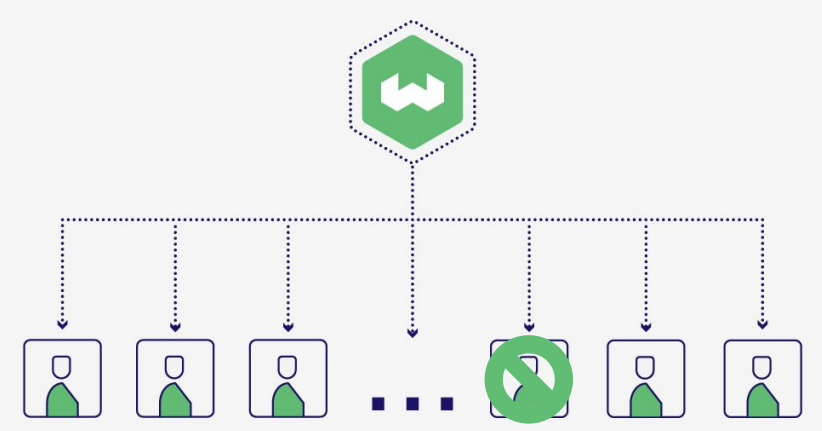


Multi-tenancy implementation

- 1000s per node
- Isolated
- Active/inactive/offloaded tenants

Scale: Solutions

End user growth

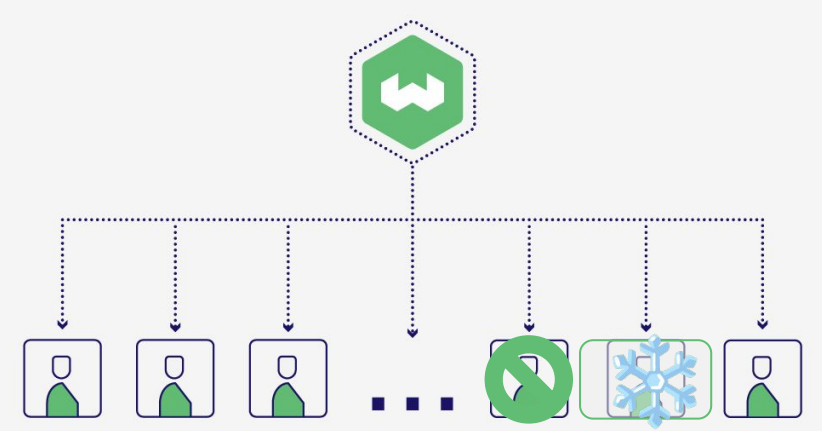


Multi-tenancy implementation

- 1000s per node
- **Isolated** (easy deletion & compliance)
- Active/inactive/offloaded tenants

Scale: Solutions

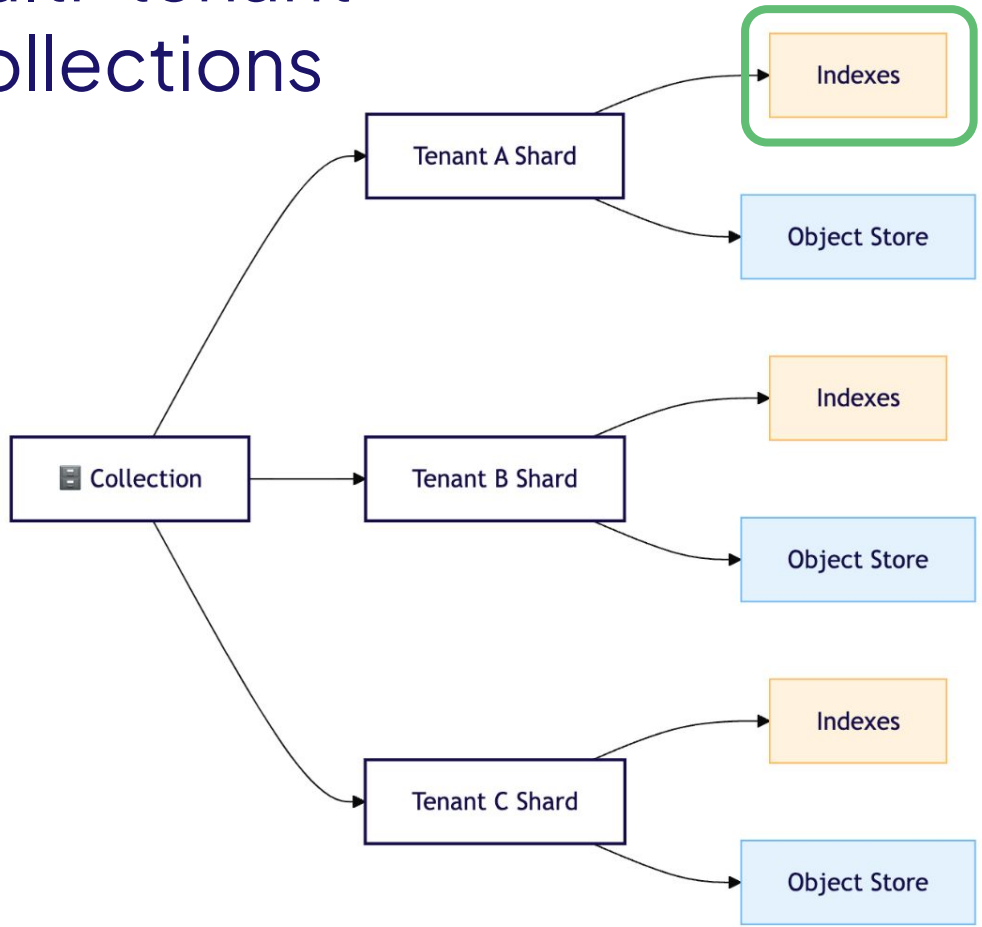
End user growth

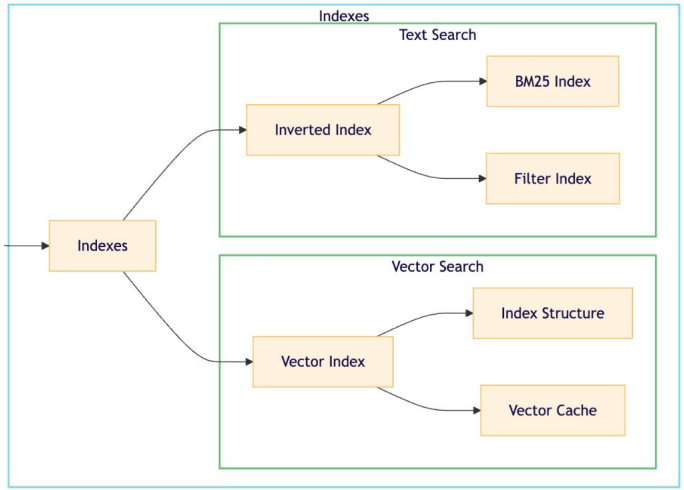


Multi-tenancy implementation

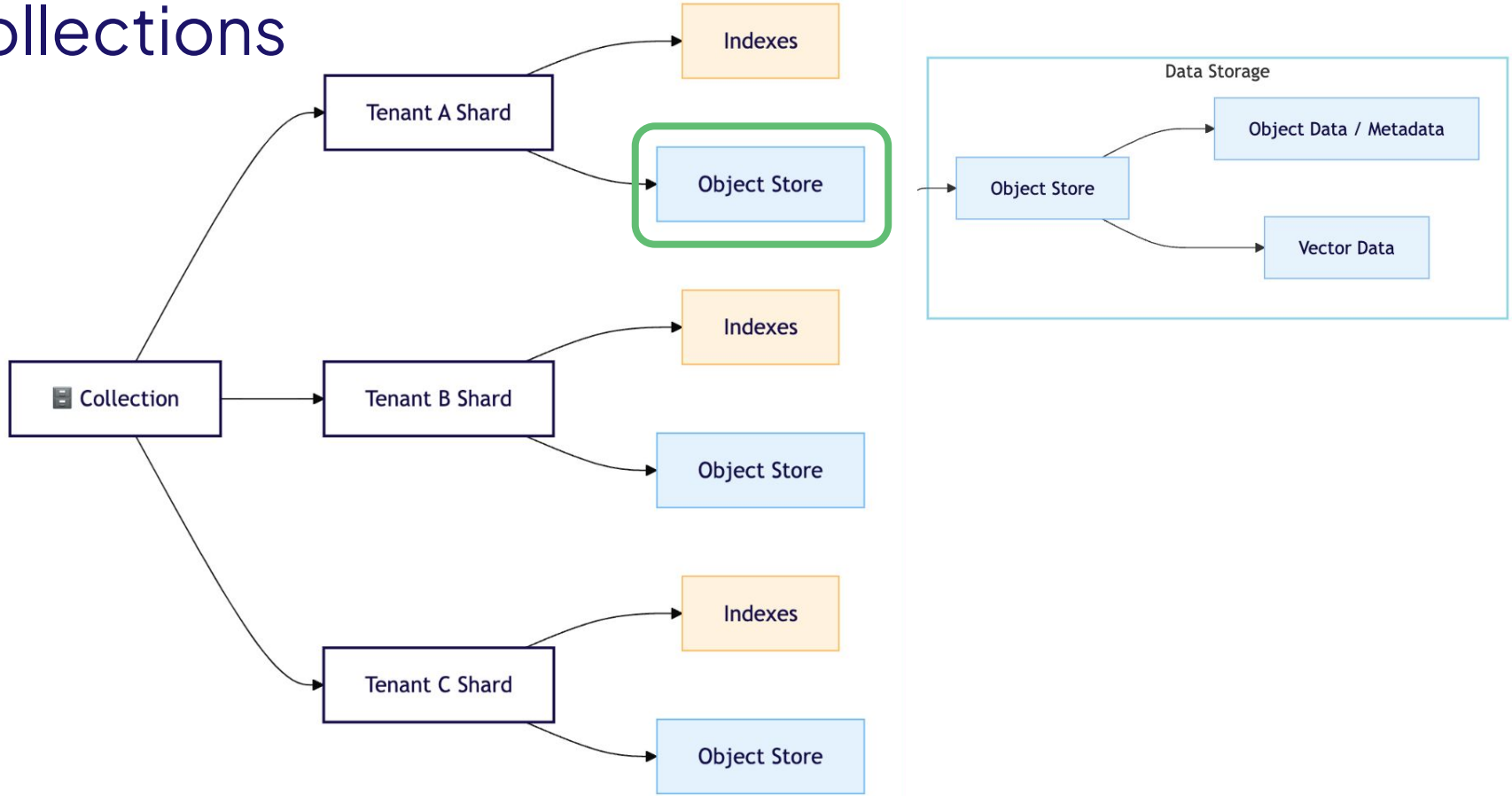
- 1000s per node
- Isolated
- **Active/inactive/offloaded tenants** (efficient)

Multi-tenant Collections

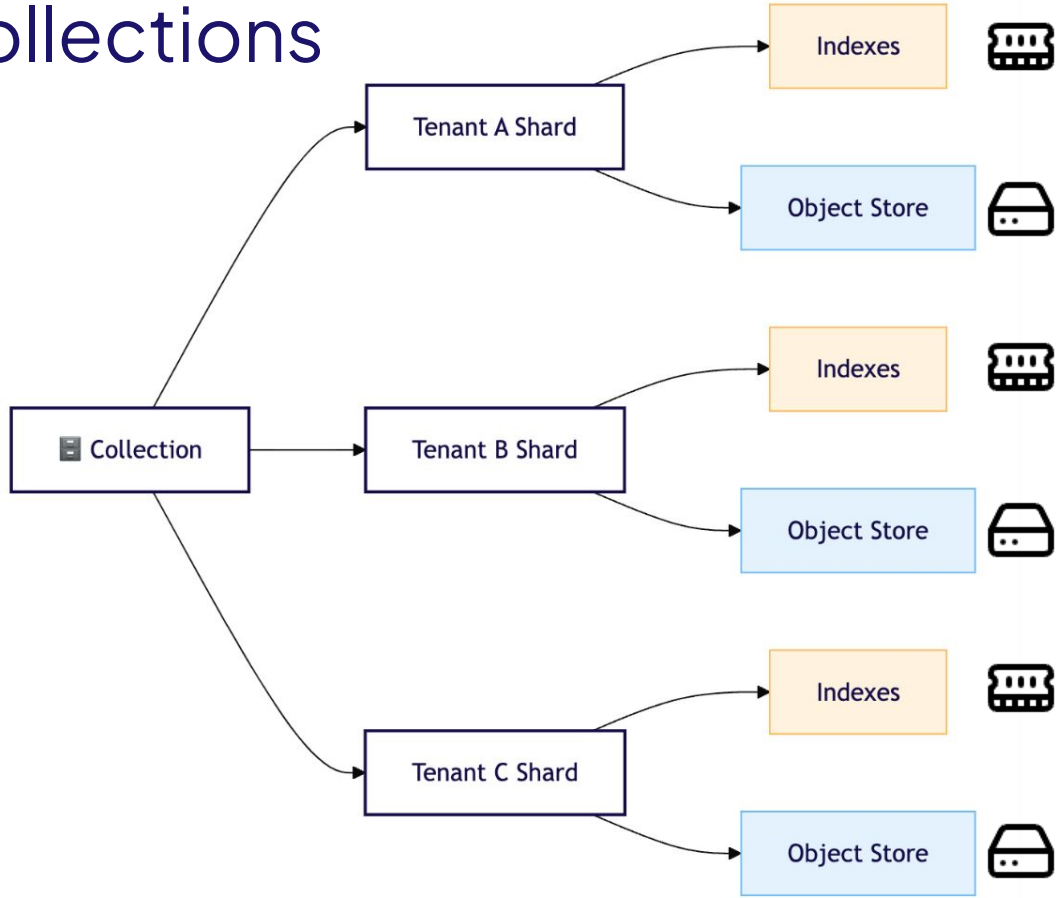


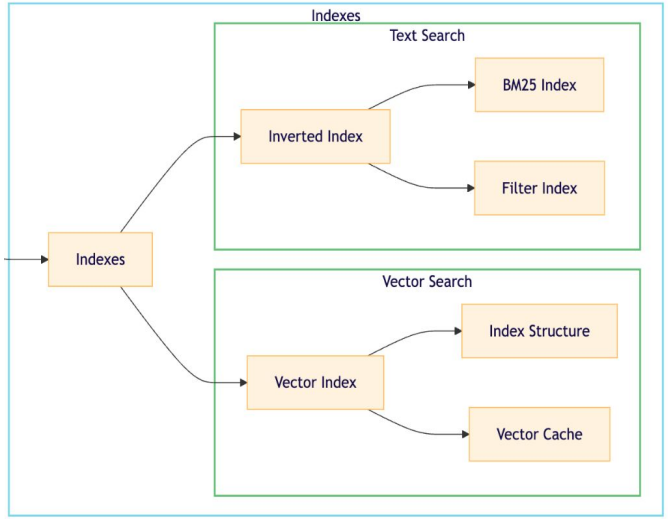


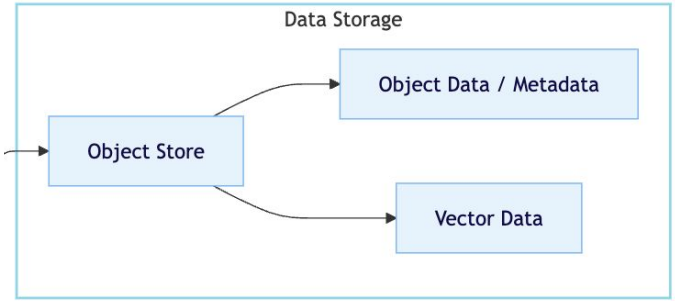
Multi-tenant Collections



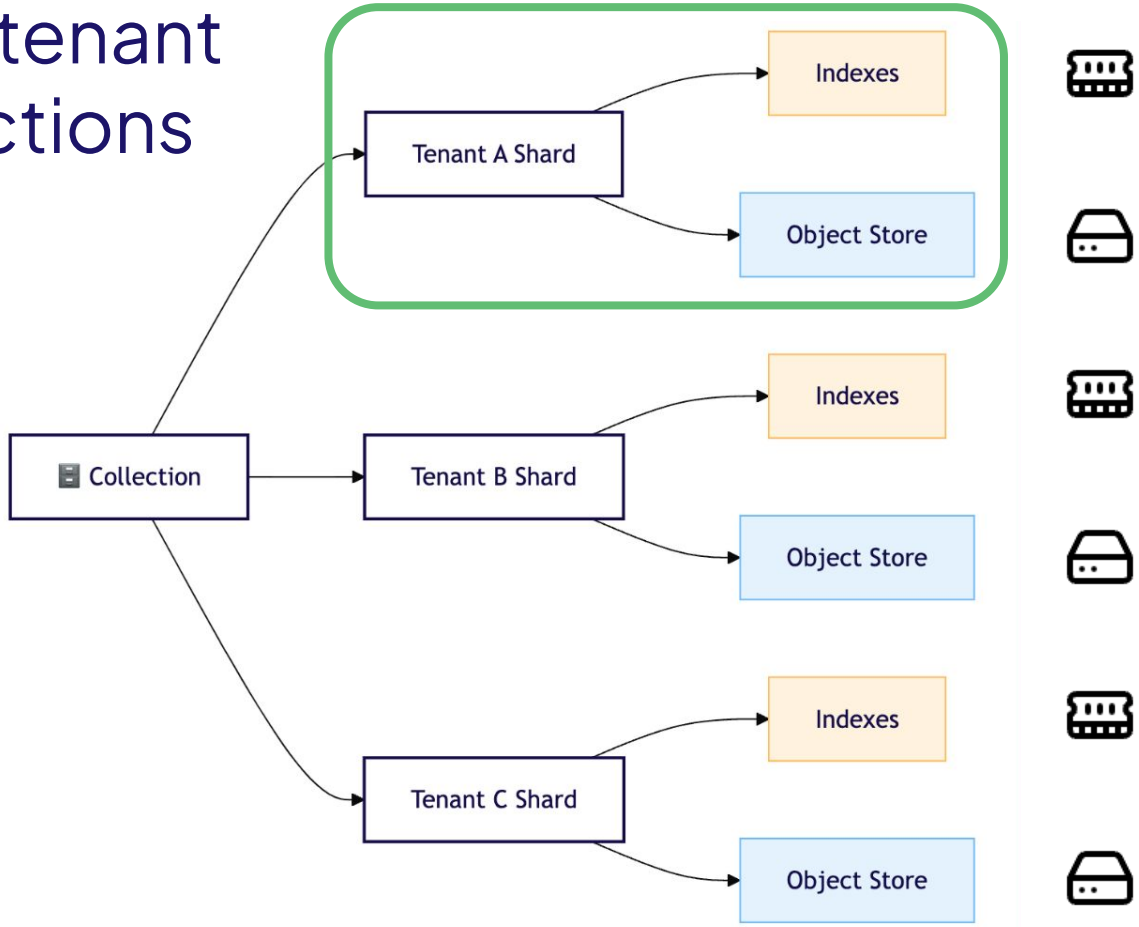
Multi-tenant Collections



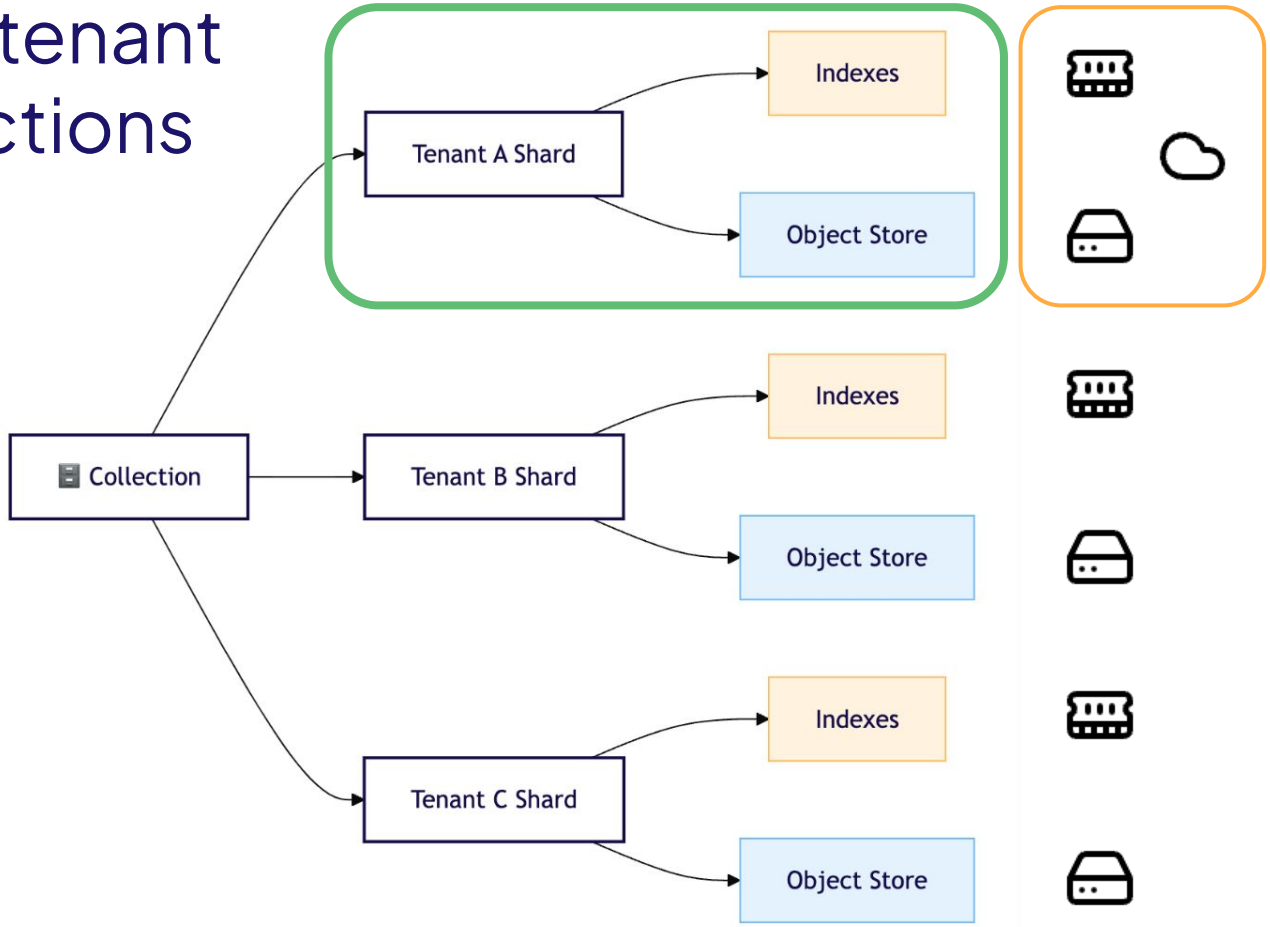




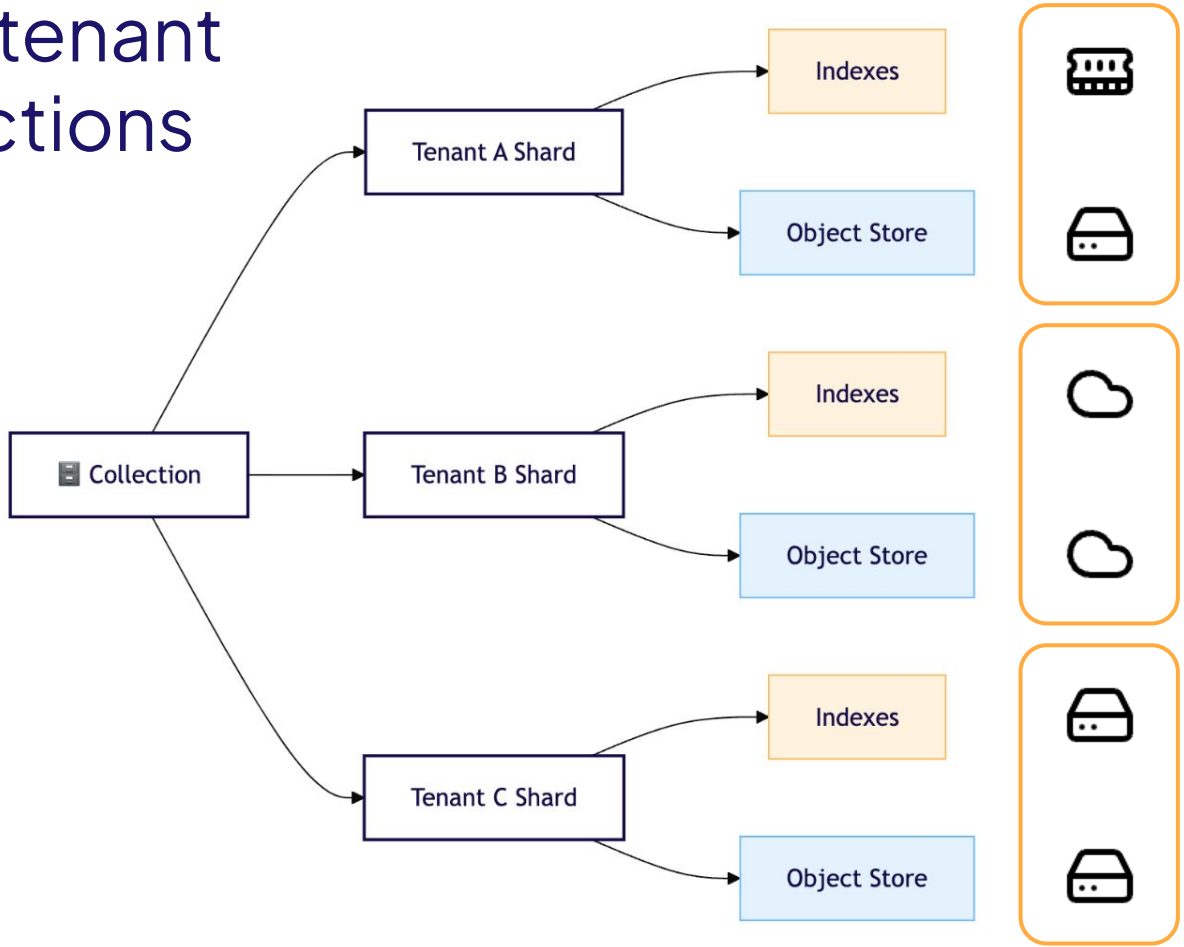
Multi-tenant Collections



Multi-tenant Collections



Multi-tenant Collections



Tenant status management

Single collection vs multi-tenant patterns

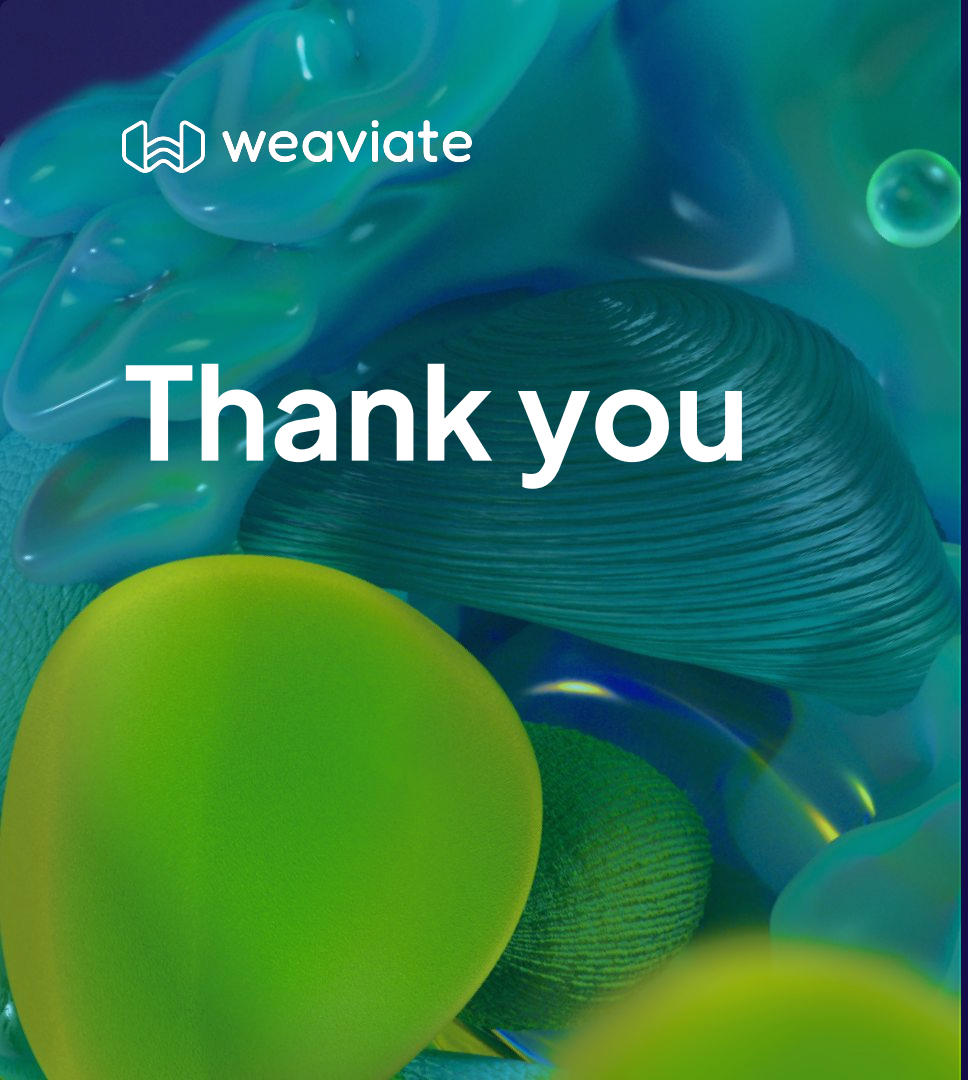
Single collection patterns

- **Data to be used at the same time:**
  - Search entire dataset
  - HNSW index for scalability
  - Try PQ quantization
- **Example use cases:**
  - E-commerce
  - One Wiki-type knowledge base

Single collection vs multi-tenant patterns

Multi-tenant patterns

- **Isolated datasets for each end user**
  - Subsets searched separately
  - Flat / Dynamic index
  - Try BQ quantization
  - Tenant state management
- **Example use cases:**
  - Online AI-powered apps on private data (wikis, services)



weaviate.io

weaviate/weaviate

JP Hwang