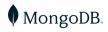
ANNOUNCEMENT: Voyage Al joins MongoDB to power more accurate and trustworthy Al applications on Atlas.







Docs Home

MongoDB Atlas

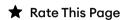
- Get Started
- Create & Connect to Clusters
- ▶ Configure Security Features
- ▶ Configure UI Access
- ▶ Migrate or Import Data
- Interact with Data
- Query Federated Data
- Atlas Search
- ▼ Atlas Vector Search

Quick Start

Create Embeddinas









=

Docs Home / MongoDB Atlas / Atlas Vector Search

Atlas Vector Search Quick Start

This quick start describes how to load sample documents that contain vector embeddings into an Atlas cluster or local Atlas deployment, create an Atlas Vector Search index on those embeddings, and then perform semantic search to return documents that are similar to your query.

Time required: 15 minutes



Work with a runnable version of this tutorial as a Python notebook.

Select your language



On this page

Objectives

Create a Vector Search Index

Run a Vector Search Query

Learning Summary

Next Steps

Objectives

In this quick start, you will do the following steps:

- Create an index definition for the sample_mflix.embedded_movies collection that indexes the plot_embedding field as the vector type.
 The plot_embedding field contains embeddings created using OpenAI's text-embedding-ada-002 embedding model. The index definition specifies 1536 vector dimensions and measures similarity using dotProduct.
- 2. Run an Atlas Vector Search query that searches the sample sample_mflix.embedded_movies collection. The query uses the \$vectorSearch stage to search the plot_embedding field, which contains embeddings created using OpenAl's text-embedding-ada-002 embedding model. The query searches the plot_embedding field using vector embeddings for the string time travel. It considers up to 150 nearest neighbors, and returns 10 documents in the results.

To learn more, see Learning Summary.

Create a Vector Search Index

➤ To set the client you use to run the examples on this page, use the **Select your language** drop-down menu in the right navigation pane.

In this section, you create an Atlas Vector Search index on sample data that you load into an Atlas cluster or a deployment hosted on your local computer:

Atlas Cluster

Local Deployment



Set up your Atlas cluster.

- a. Create a free Atlas account or sign in to an existing account.
- b. If you don't yet have an Atlas cluster, create a free
 M0 cluster. To learn more about creating an Atlas cluster, see Create a Cluster.

NOTE

If you are working with an existing cluster, you must have Project Data Access Admin or higher access to your Atlas project.

If you create a new cluster, you have the necessary permissions by default.

You can create only one MO Free cluster per project.

- c. In the left sidebar, click **Atlas Search**. Choose your cluster from the **Select data source** menu and click **Go to Atlas Search**.
- d. If you haven't yet loaded the sample dataset onto your cluster, click Load a Sample Dataset. In the Load Sample Dataset dialog box, click Load Sample Dataset to confirm.

If you already loaded the sample dataset, check that the sample_mflix database contains the embedded_movies collection. If it doesn't, drop the sample databases and reload the sample dataset.

Loading the **sample dataset** can take several minutes to complete.

2 Create a Vector Search index.

NOTE

You can use the mongosh command or driver helper methods to create Atlas Vector Search indexes on all Atlas cluster tiers. For a list of supported driver versions, see Supported Clients.

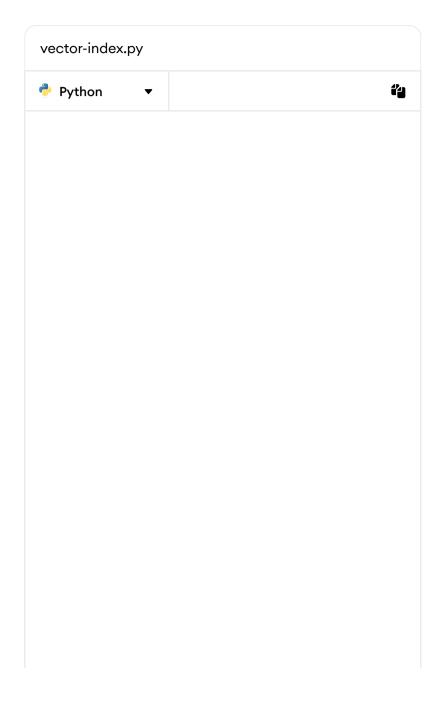
a. Add the PyMongo Driver as a dependency in your project:



For more detailed installation instructions, see the MongoDB Python Driver documentation.

b. Define the index.

Create a file named vector-index.py. Copy and paste the following code into the file.



```
from pymongo.mongo_client import Mongo(
1
    from pymongo.operations import SearchIr
2
    import time
3
4
    # Connect to your Atlas deployment
5
    uri = "<connectionString>"
6
    client = MongoClient(uri)
7
8
    # Access your database and collection
9
    database = client["sample_mflix"]
   collection = database["embedded_movies"]
12
    # Create your index model, then create
   search_index_model = SearchIndexModel(
15
     definition={
16
        "fields": [
17
            "type": "vector",
18
            "path": "plot_embedding",
19
            "numDimensions": 1536,
20
            "similarity": "dotProduct",
21
            "quantization": "scalar"
22
23
24
25
     },
     name="vector_index",
26
```

```
type="vectorSearch"
27
28
29
   result = collection.create_search_index
   print("New search index named " + resul
32
   # Wait for initial sync to complete
33
    print("Polling to check if the index is
   predicate=None
  if predicate is None:
     predicate = lambda index: index.get('
37
38
   while True:
39
     indices = list(collection.list_search
40
     if len(indices) and predicate(indices
41
        break
42
     time.sleep(5)
43
    print(result + " is ready for querying.
45
46 client.close()
```

This index definition:

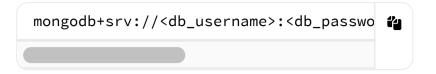
- Indexes the plot_embedding field as the vector type. This field contains vector embeddings that represent the summary of a movie's plot.
 - Specifies 1536 vector dimensions.
 - Measures similarity using dotProduct similarity.
 - Enables automatic quantization of the vectors.

This code also includes a polling mechanism to check if the index is ready to use.

c. Specify the <connection-string>.

Replace <connection-string> with the connection string for your Atlas cluster or local deployment, and then save the file.

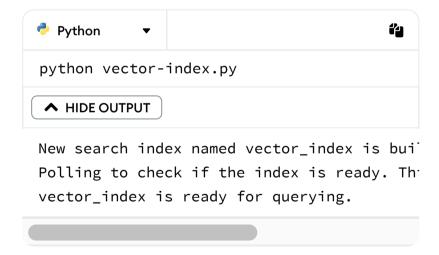
Your connection string should use the following format:



NOTE

Ensure that your connection string includes your database user's credentials. To learn more about finding your connection string, see Connect via Drivers.

d. Run the following command to create the index.



Run a Vector Search Query

In this section, you run a sample vector search query on your indexed embeddings.



This query searches for documents that include text in the plot field that is semantically related to the term "time travel".

- a. Create a file named atlas-vector-search-quick-start.py.
- b. Copy and paste the following sample query into the atlas-vector-search-quick-start.py file:



```
import pymongo
1
2
    # connect to your Atlas cluster
3
   client = pymongo.MongoClient("<connect:</pre>
4
5
6
    # define pipeline
    pipeline = [
7
8
9
        '$vectorSearch': {
          'index': 'vector_index',
10
          'path': 'plot_embedding',
11
          'queryVector': [-0.0016261312, -0
12
          'numCandidates': 150,
13
          'limit': 10
14
15
16
     }, {
        '$project': {
17
          '_id': 0,
18
          'plot': 1,
19
20
          'title': 1,
21
          'score': {
22
            '$meta': 'vectorSearchScore'
23
24
25
26
```

```
27
28 # run pipeline
29 result = client["sample_mflix"]["embedo
30
31 # print results
32 for i in result:
33    print(i)
34
```

This query uses the \$vectorSearch stage to:

- Compare vector embeddings of the search term against vector embeddings of movie plots in the plot_embedding field of the sample_mflix.embedded_movies collection.
- Consider up to the 150 most similar movie plots and return the top 10 results.

It uses the \$project stage to:

• Only include the movie plot and title fields in the results.

• Add a score field to show the relevance of each result to the search term.

To learn more about this pipeline stage, see Run Vector Search Queries.

Specify the <connection-string>.

Replace <connection-string> with the connection string for your Atlas cluster or local deployment, and then save the file.

Your connection string should use the following format:

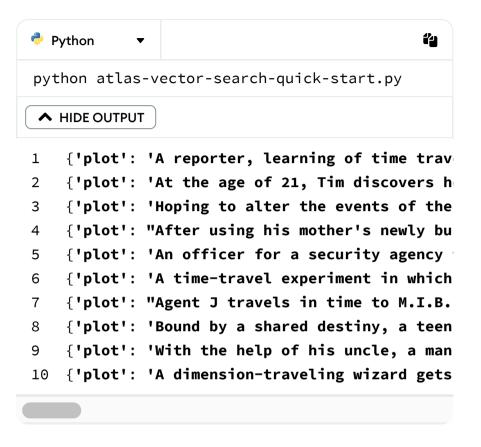
mongodb+srv://<db_username>:<db_password>@-

NOTE

Ensure that your connection string includes your database user's credentials. To learn more about finding your connection string, see Connect via Drivers.

3 Run your query.

Run the following command to query your collection:



Learning Summary

This quick start focused on retrieving documents from your Atlas cluster that contain text that is semantically related to a provided query. However, you can create a vector search index on embeddings that represent any type of data that you might write to an Atlas cluster, such as images or videos.

Sample Data

This quick start uses the sample_mflix.embedded_movies collection which contains details about movies. In each document in the collection, the plot_embedding field contains a vector embedding that represents the string in the plot field. For more information on the schema of the documents in the collection, see Sample Mflix Dataset.

By storing your source data and its corresponding vector embeddings in the same document, you can leverage both fields for complex queries or hybrid search. You can even store vector embeddings generated from different embedding models in the same document to streamline your workflow as you test the performance of different vector embedding models for your specific use case.

Vector Embeddings

The vector embeddings in the

sample_mflix.embedded_movies collection and in the example query were created using the OpenAI text-embedding-ada-002 embedding model. Your choice of embedding model informs the vector dimensions and vector similarity function you use in your vector search index. You can

use any **embedding model** you like, and it is worth experimenting with different models as accuracy can vary from model to model depending on your specific use case.

To learn how to create vector embeddings of your own data, see How to Create Vector Embeddings.

Vector Index Definition

An index is a data structure that holds a subset of data from a collection's documents that improves database performance for specific queries. A vector search index points to the fields that contain your vector embeddings and includes the dimensions of your vectors as well as the function used to measure similarity between vectors of queries and vectors stored in the database.

Because the text-embedding-ada-002 embedding model used in this quick start converts data into vector embeddings with 1536 dimensions and supports the cosine function, this vector search index specifies the same number of vector dimensions and similarity function.

Vector Search Query

The query you ran in this quick start is an aggregation pipeline, in which the \$vectorSearch stage performs an Approximate Nearest Neighbor (ANN) search followed by a \$project stage that refines the results. To see all the options for a vector search query, including using Exact Nearest Neighbor (ENN) or how to narrow the scope of your vector search with the filter option, see Run Vector Search Queries.

Next Steps

- To learn how to create embeddings from data and load them into Atlas, see Create Embeddings.
- To learn how to implement retrieval-augmented generation (RAG), see Retrieval-Augmented Generation (RAG) with Atlas Vector Search.
- To integrate Atlas Vector Search with popular AI frameworks and services, see Integrate Vector Search with AI Technologies.
- To build production ready AI chatbots using Atlas Vector Search, see the MongoDB Chatbot Framework.[™]

• To learn how implement <u>RAG</u> without the need for <u>API</u> keys or credits, see <u>Build a Local RAG Implementation</u> with Atlas Vector Search.

English **About** Careers Investor Relations GitHub Legal Security Information Trust Center Connect with Us **Support**

Customer Portal

Contact Us

Atlas Status	Customer Support
Manage Cookies	
Deployment Options	
MongoDB Atlas	Enterprise Advanced
Community Edition	
© 2024 MongoDB, Inc.	