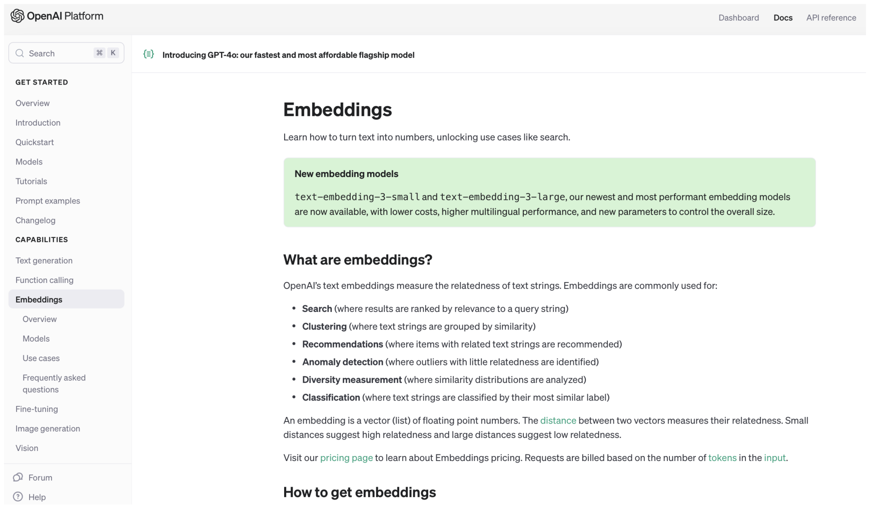
02-Creating an Embedding

1- Go to platform.openai.com and go on api reference tab and select embedding (https://platform.openai.com/docs/api-reference/embeddings)



Introducing GPT-4o: our fastest and most affordable flagship model

Embeddings

Learn how to turn text into numbers, unlocking use cases like search.

New embedding models

`text-embedding-3-small` and `text-embedding-3-large`, our newest and most performant embedding models are now available, with lower costs, higher multilingual performance, and new parameters to control the overall size.

What are embeddings?

OpenAI's text embeddings measure the relatedness of text strings. Embeddings are commonly used for:

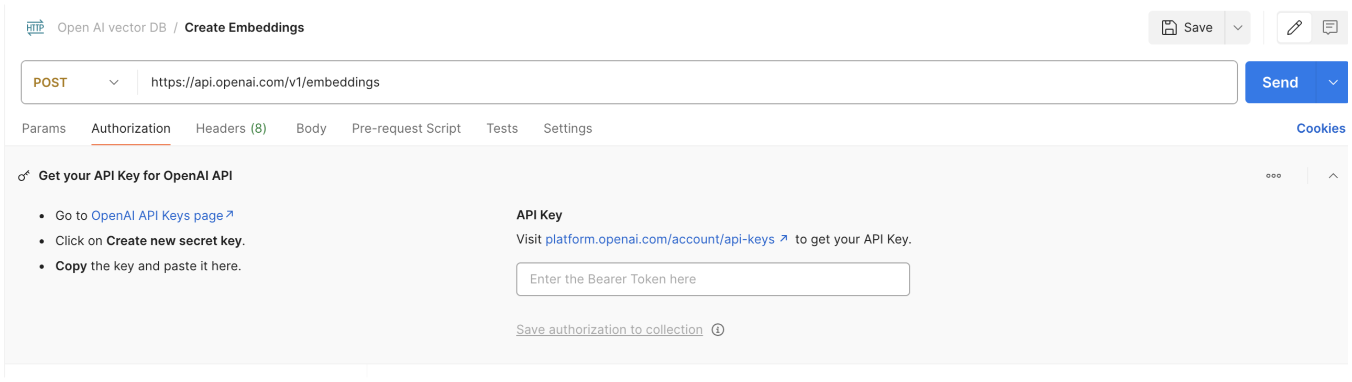
- **Search** (where results are ranked by relevance to a query string)
- **Clustering** (where text strings are grouped by similarity)
- **Recommendations** (where items with related text strings are recommended)
- **Anomaly detection** (where outliers with little relatedness are identified)
- **Diversity measurement** (where similarity distributions are analyzed)
- **Classification** (where text strings are classified by their most similar label)

An embedding is a vector (list) of floating point numbers. The distance between two vectors measures their relatedness. Small distances suggest high relatedness and large distances suggest low relatedness.

Visit our pricing page to learn about Embeddings pricing. Requests are billed based on the number of tokens in the input.

How to get embeddings

2- Create the postman request and add the secret Key by fetching it from openAI.



POST https://api.openai.com/v1/embeddings

Get your API Key for OpenAI API

- Go to OpenAI API Keys page
- Click on **Create new secret key**.
- **Copy** the key and paste it here.

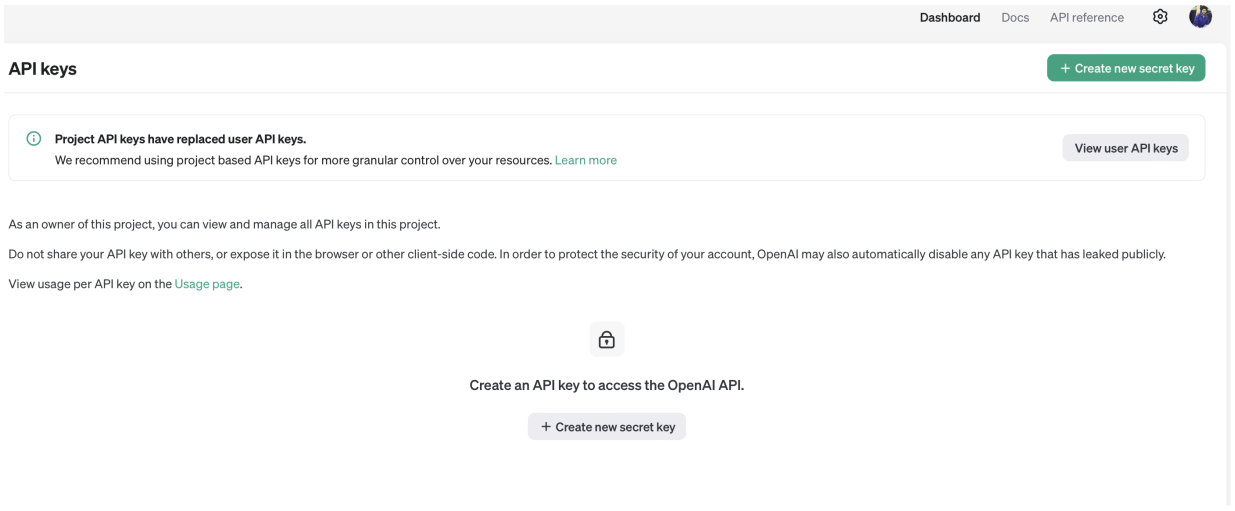
API Key

Visit platform.openai.com/account/api-keys to get your API Key.

Enter the Bearer Token here

Save authorization to collection

Create secret key from openAI



API keys

+ Create new secret key

Project API keys have replaced user API keys.
We recommend using project based API keys for more granular control over your resources. Learn more

View user API keys

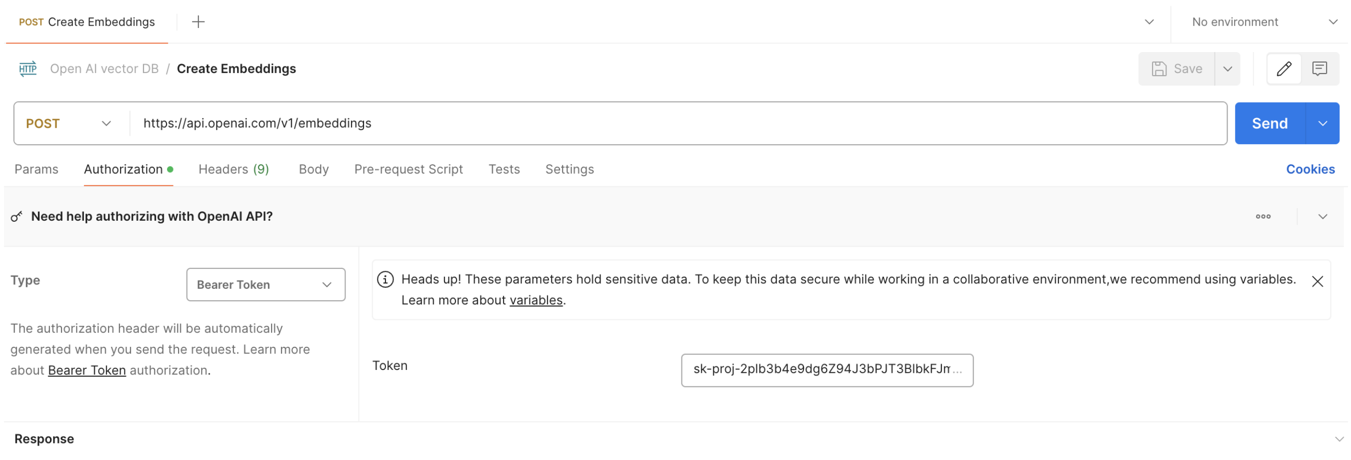
As an owner of this project, you can view and manage all API keys in this project.

Do not share your API key with others, or expose it in the browser or other client-side code. In order to protect the security of your account, OpenAI may also automatically disable any API key that has leaked publicly.

View usage per API key on the Usage page.

Create an API key to access the OpenAI API.

+ Create new secret key



POST https://api.openai.com/v1/embeddings

Need help authorizing with OpenAI API?

Type: Bearer Token

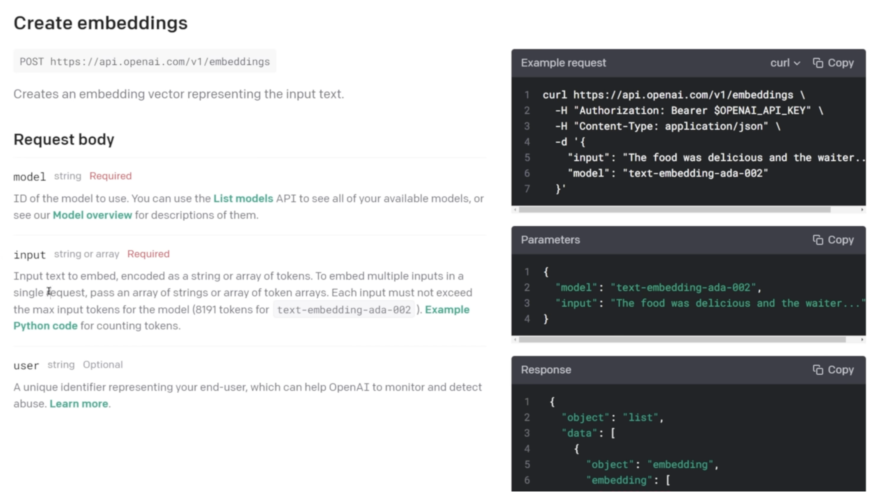
The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

Token: sk-proj-2plb3b4e9dg6Z94J3bPJT3BlbkFJn...

Response

Looking at the documentation, we only require model and input param as necessary for the api.


Create embeddings

POST https://api.openai.com/v1/embeddings

Creates an embedding vector representing the input text.

Request body

model string Required
ID of the model to use. You can use the List models API to see all of your available models, or see our Model overview for descriptions of them.

input string or array Required
Input text to embed, encoded as a string or array of tokens. To embed multiple inputs in a single request, pass an array of strings or array of token arrays. Each input must not exceed the max input tokens for the model (8191 tokens for `text-embedding-ada-002`). Example Python code for counting tokens.

user string Optional
A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse. Learn more.

Example request (curl)

```
curl https://api.openai.com/v1/embeddings \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "input": "The food was delicious and the waiter..
    "model": "text-embedding-ada-002"
  }'
```

Parameters

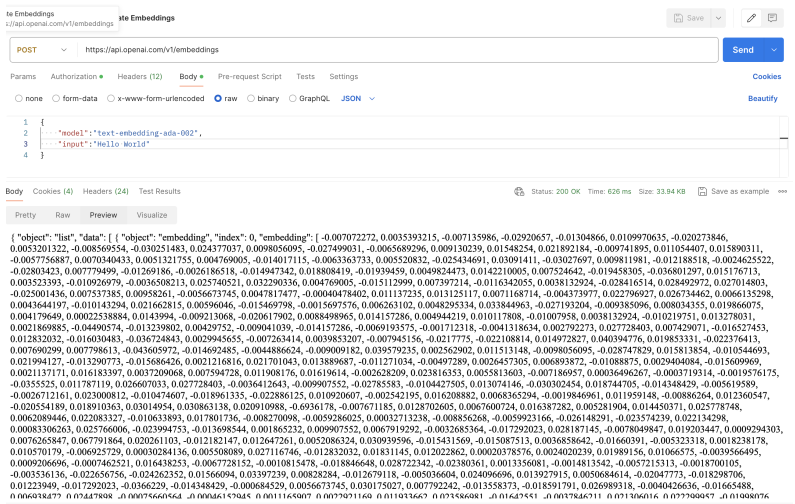
```
{
  "model": "text-embedding-ada-002",
  "input": "The food was delicious and the waiter..."
}
```

Response

```
{
  "object": "list",
  "data": [
    {
      "object": "embedding",
      "embedding": [
```

<u>**Ensure you have credit in your account**</u>

3- create body request and send to get the embedding.



POST https://api.openai.com/v1/embeddings

```
{
    "model":"text-embedding-ada-002",
    "input":"Hello World"
}
```

Status: 200 OK Time: 626 ms Size: 33.94 KB

{ "object": "list", "data": [{ "object": "embedding", "index": 0, "embedding": [-0.007072272, 0.0035393215, -0.007135986, -0.02920657, -0.01304866, 0.0109970635, -0.020273846,
0.0053201322, -0.008569554, -0.030251483, 0.024377037, 0.0098056095, -0.027499031, -0.0065689296, 0.009130239, 0.01548254, 0.021892184, -0.009741895, 0.011054407, 0.015890311,
-0.0057756887, 0.0070340433, 0.0051321755, 0.004769005, -0.014017115, -0.0063363733, 0.005520832, -0.025434691, 0.03091411, -0.03027697, 0.009811981, -0.012188518, -0.0024625522,
-0.02803423, 0.007779499, -0.01269186, -0.0026186518, -0.014947342, 0.018808419, -0.01939459, 0.0049824473, 0.0142210005, 0.007524642, -0.019458305, -0.036801297, 0.015176713,
0.003523393, -0.010926979, -0.0036508213, 0.025740521, 0.032290336, 0.004769005, -0.015112999, 0.007397214, -0.0116342055, 0.0038132924, -0.028416514, 0.028492972, 0.027014803,
-0.025001436, 0.007537385, 0.00958261, -0.0056673745, 0.0047817477, -0.00040478402, 0.011137235, 0.013125117, 0.0071168714, -0.004373977, 0.022796927, 0.026734462, 0.0066135298,
0.0043644197, -0.010143294, 0.021662815, 0.00596046, -0.015469798, -0.0015697576, 0.006263102, 0.0048295334, 0.033844963, -0.027193204, -0.009385096, 0.008034355, 0.019866075,
0.004179649, 0.00022538884, 0.0143994, -0.009213068, -0.020617902, 0.0088498965, 0.014157286, 0.004944219, 0.010117808, -0.01007958, 0.0038132924, -0.010219751, 0.013278031,
0.0021869885, -0.04490574, -0.013239802, 0.00429752, -0.009041039, -0.014157286, -0.0069193575, -0.001712318, -0.0041318634, 0.002792273, 0.027728403, 0.007429071, -0.016527453,
0.012832032, -0.016030483, -0.036724843, 0.0029945655, -0.007263414, 0.0039853207, -0.007945156, -0.0217775, -0.022108814, 0.014972827, 0.040394776, 0.019853331, -0.022376413,
0.007690299, 0.007798613, -0.043605972, -0.014692485, -0.0044886624, -0.009009182, 0.039579235, 0.002562902, 0.011513148, -0.0098056095, -0.028747829, 0.015813854, -0.010544693,
0.021994127, -0.013290773, -0.015686426, 0.0021216816, 0.021701043, 0.013889687, -0.011271034, -0.00497289, 0.0026457305, 0.006893872, -0.01088875, 0.0029404084, -0.015609969,
0.0021137171, 0.016183397, 0.0037209068, 0.007594728, 0.011908176, 0.01619614, -0.002628209, 0.023816353, 0.0055813603, -0.007186957, 0.00036496267, -0.0003719314, -0.0019576175,
-0.0355525, 0.011787119, 0.026607033, 0.027728403, -0.0036412643, -0.009907552, -0.02785583, -0.0104427505, 0.013074146, -0.030302454, 0.018744705, -0.014348429, -0.005619589,
-0.0026712161, 0.023000812, -0.010474607, -0.018961335, -0.022886125, 0.010920607, -0.002542195, 0.016208882, 0.0068365294, -0.0019846961, 0.011959148, -0.00886264, 0.012360547,
-0.020554189, 0.018910363, 0.03014954, 0.030863138, 0.020910988, -0.6936178, -0.007671185, 0.0128702605, 0.0067600724, 0.016387282, 0.005281904, 0.014450371, 0.025778748,
0.0062089446, 0.022083327, -0.010633893, 0.017801736, -0.008270098, -0.0059286025, 0.00032713238, -0.008856268, -0.0059923166, -0.026148291, -0.023574239, 0.022134298,
0.00083306263, 0.025766006, -0.023994753, -0.013698544, 0.001865232, 0.009907552, 0.0067919292, -0.0032685364, -0.017292023, 0.028187145, -0.0078049847, 0.019203447, 0.0009294303,
0.0076265847, 0.067791864, 0.020261103, -0.012182147, 0.012647261, 0.0052086324, 0.030939596, -0.015431569, -0.015087513, 0.0036858642, -0.01660391, -0.005323318, 0.0018238178,
0.010570179, -0.006925729, 0.00030284136, 0.005508089, 0.027116746, -0.012832032, 0.01831145, 0.012022862, 0.00020378576, 0.0024020239, 0.01989156, 0.01066575, -0.0039566495,
0.0009206696, -0.0007462521, 0.016438253, -0.0067728152, -0.0010815478, -0.018846648, 0.028722342, -0.02380361, 0.0013356081, -0.0014813542, -0.0057215313, -0.0018700105,
-0.003536136, -0.022656756, -0.024262352, 0.01566094, 0.03397239, 0.00828284, -0.012679118, -0.005036604, 0.024096696, 0.013927915, 0.0050684614, -0.02047773, -0.018298706,
0.01223949, -0.017292023, -0.0366229, -0.014348429, -0.000684529, 0.0056673745, 0.030175027, 0.007792242, -0.013558373, -0.018591791, 0.026989318, -0.0040426636, -0.01665488,
0.006938472, 0.02447898, -0.00075660564, -0.00046152945, 0.0011165907, 0.0022921169, 0.011933662, 0.023586981, -0.01642551, -0.0037846211, 0.021306016, 0.022299957, -0.01998076,

you can send 1 word, 2 words or large chunks of paragraph and get the embeddings. (why are they almost of the same size?)
These embeddings can then be used to search in database.

"text-embedding-ada-002" model takes 8192 token which is approx around (8192*4= 32764 letters) (around 10 pages of pdf approx)