Python SDK

- Prerequisites
- 1. Create an API token
- 2. Set up your Python environment
- 3. Install the Cloudflare Python SDK
- 4. Set your credentials
- 5. Create an AI Search instance
- 6. Upload content
- 7. Search your instance
- Next steps

import { LinkCard, DashButton } from "~/components";

This guide walks you through creating an AI Search instance, uploading content, and querying it from a Python application using the Cloudflare Python SDK.

Prerequisites

- Python 3.8 or later.
- Your account ID.

This guide uses the `default` namespace, which exists automatically on every account. To group instances into your own namespace, create one with `client.aisearch.namespaces.create()`.

1. Create an API token

You need an API token with **AI Search:Edit** and **AI Search:Run** permissions.

1. In the Cloudflare dashboard, go to **My Profile** > **API Tokens**.

2. Select **Create Token**.
3. Select **Create Custom Token**.
4. Enter a **Token name**, for example `AI Search Python`.
5. Under **Permissions**, add two permissions:
   - **Account** > **AI Search:Edit**
   - **Account** > **AI Search:Run**
6. Select **Continue to summary**, then select **Create Token**.
7. Copy and save the token value. This is your `API_TOKEN`.

2. Set up your Python environment

Create a project directory and a virtual environment to isolate your dependencies.

```
mkdir ai-search-python && cd ai-search-python
python3 -m venv .venv
source .venv/bin/activate
```

On Windows, activate the virtual environment with `.venv\Scripts\activate` instead.

3. Install the Cloudflare Python SDK

Install the official `cloudflare` package:

```
pip install cloudflare
```

4. Set your credentials

Export your account ID and API token as environment variables.

```
export CLOUDFLARE_ACCOUNT_ID="<ACCOUNT_ID>"
export CLOUDFLARE_API_TOKEN="<API_TOKEN>"
```

5. Create an AI Search instance

Create a file named `quickstart.py`. The following code sets up a client and creates an instance named `my-instance` in the `default` namespace. Because no data source is specified, the instance uses built-in storage, so you can upload files to it directly.

```
import os

from cloudflare import Cloudflare

client = Cloudflare(api_token=os.environ["CLOUDFLARE_API_TOKEN"])
account_id = os.environ["CLOUDFLARE_ACCOUNT_ID"]

instance = client.aisearch.namespaces.instances.create(
    name="default",
    account_id=account_id,
    id="my-instance",
)

print(f"Created instance: {instance.id}")
```

:::note Creating an instance is a one-time action. If you run the script again, remove this step, because an instance name must be unique within a namespace. :::

6. Upload content

Add the following to `quickstart.py` to upload a document. Setting `wait_for_completion` to `True` waits for indexing before returning so the file is ready to search. If indexing is still finishing, `item.status` may be `running`; the file continues indexing in the background and becomes searchable shortly after.

```
item = client.aisearch.namespaces.instances.items.upload(
    id="my-instance",
    account_id=account_id,
    name="default",
    file={
        "file": (
```

```
            "getting-started.md",
            b"AI Search indexes uploaded content for retrieval.",
            "text/markdown",
        ),
        "wait_for_completion": True,
    },
)

print(f"Uploaded item status: {item.status}")
```

7. Search your instance

Add the following to `quickstart.py` to run a query against your indexed content.

```
results = client.aisearch.namespaces.instances.search(
    id="my-instance",
    account_id=account_id,
    name="default",
    query="How does AI Search handle uploaded content?",
)

if results.chunks:
    print(results.chunks[0].text)
else:
    print("No results yet — your content may still be indexing. Try again in a moment.")
```

Run the script:

```
python quickstart.py
```

If the search returns no results, the content may still be indexing. Wait a moment, then run the search again.

Next steps

□□□ #2
□ □□□ □□□ 2026-10-31 23:48:43 UTC

□ □□□ □□□ 2026-10-31 23:48:45 UTC