Documentation

search

Search

- *rocket_launch*

  Get started

  - Installation
    add
  - Fundamentals
    add
  - First steps
    add
- *code*

  Develop

  - Concepts
    add
  - API reference
    remove
    - PAGE ELEMENTS

    - Write and magic
      add
    - Text elements
      add
    - Data elements
      add
    - Chart elements
      add
    - Input widgets
      add
    - Media elements
      add
    - Layouts and containers
      add
    - Chat elements
      add
    - Status elements
      add
    - Third-party components*open_in_new*
    - APPLICATION LOGIC

    - Navigation and pages
      add
    - Execution flow
      add
    - Caching and state
      add
    - Connections and secrets
      remove
      - SECRETS

star

Tip

This page only contains the `st.connection` API. For a deeper dive into creating and managing data connections within Streamlit apps, read Connecting to data.

st.connection

Streamlit Version Version 1.41.0

Create a new connection to a data store or API, or return an existing one.

Configuration options, credentials, and secrets for connections are combined from the following sources:

- The keyword arguments passed to this command.
- The app's `secrets.toml` files.
- Any connection-specific configuration files.

The connection returned from `st.connection` is internally cached with `st.cache_resource` and is therefore shared between sessions.

Function signature[source]

st.connection(name, type=None, max_entries=None, ttl=None, **kwargs)

Parameters	
name (str)	The connection name used for secrets lookup in `secrets.toml`. Streamlit uses secrets under `[connections.<name>]` for the connection. `type` will be inferred if `name` is one of the following: `"snowflake"`, `"snowpark"`, or `"sql"`.
type (str, connection class, or None)	The type of connection to create. This can be one of the following: • `None` (default): Streamlit will infer the connection type from `name`. If the type is not inferrable from `name`, the type must be specified in `secrets.toml` instead. • `"snowflake"`: Streamlit will initialize a connection with `SnowflakeConnection`. • `"snowpark"`: Streamlit will initialize a connection with `SnowparkConnection`. This is deprecated. • `"sql"`: Streamlit will initialize a connection with `SQLConnection`. • A string path to an importable class: This must be a dot-separated module path ending in the importable class. Streamlit will import the class and initialize a connection with it. The class must extend `st.connections.BaseConnection`. • An imported class reference: Streamlit will initialize a connection with the referenced class, which must extend `st.connections.BaseConnection`.
max_entries (int or None)	The maximum number of connections to keep in the cache. If this is `None` (default), the cache is unbounded. Otherwise, when a new entry is added to a full cache, the oldest cached entry is removed.
ttl (float, timedelta, or None)	The maximum number of seconds to keep results in the cache. If this is `None` (default), cached results do not expire with time.
Returns	
(Subclass of BaseConnection)	An initialized connection object of the specified `type`.

Function signature[source]

st.connection(name, type=None, max_entries=None, ttl=None, **kwargs)

**kwargs (any)	Connection-specific keyword arguments that are passed to the connection's `._connect()` method. `**kwargs` are typically combined with (and take precendence over) key-value pairs in `secrets.toml`. To learn more, see the specific connection's documentation.

Returns

(Subclass of BaseConnection)	An initialized connection object of the specified `type`.

Examples

Example 1: Inferred connection type

The easiest way to create a first-party (SQL, Snowflake, or Snowpark) connection is to use their default names and define corresponding sections in your `secrets.toml` file. The following example creates a `"sql"`-type connection.

`.streamlit/secrets.toml`:

```
[connections.sql]
dialect = "xxx"
host = "xxx"
username = "xxx"
password = "xxx"
```

Your app code:

```
import streamlit as st
conn = st.connection("sql")
```

Example 2: Named connections

Creating a connection with a custom name requires you to explicitly specify the type. If `type` is not passed as a keyword argument, it must be set in the appropriate section of `secrets.toml`. The following example creates two `"sql"`-type connections, each with their own custom name. The first defines `type` in the `st.connection` command; the second defines `type` in `secrets.toml`.

`.streamlit/secrets.toml`:

```
[connections.first_connection]
dialect = "xxx"
host = "xxx"
username = "xxx"
password = "xxx"

[connections.second_connection]
type = "sql"
dialect = "yyy"
host = "yyy"
username = "yyy"
password = "yyy"
```

Your app code:

```
import streamlit as st
conn1 = st.connection("first_connection", type="sql")
```

```
conn2 = st.connection("second_connection")
```

Example 3: Using a path to the connection class

Passing the full module path to the connection class can be useful, especially when working with a custom connection. Although this is not the typical way to create first party connections, the following example creates the same type of connection as one with `type="sql"`. Note that `type` is a string path.

`.streamlit/secrets.toml`:

```
[connections.my_sql_connection]
url = "xxx+xxx://xxx:xxx@xxx:xxx/xxx"
```

Your app code:

```
import streamlit as st
conn = st.connection(
    "my_sql_connection", type="streamlit.connections.SQLConnection"
)
```

Example 4: Importing the connection class

You can pass the connection class directly to the `st.connection` command. Doing so allows static type checking tools such as `mypy` to infer the exact return type of `st.connection`. The following example creates the same connection as in Example 3.

`.streamlit/secrets.toml`:

```
[connections.my_sql_connection]
url = "xxx+xxx://xxx:xxx@xxx:xxx/xxx"
```

Your app code:

```
import streamlit as st
from streamlit.connections import SQLConnection
conn = st.connection("my_sql_connection", type=SQLConnection)
```

For a comprehensive overview of this feature, check out this video tutorial by Joshua Carroll, Streamlit's Product Manager for Developer Experience. You'll learn about the feature's utility in creating and managing data connections within your apps by using real-world examples.

←Previous: secrets.tomlNext: SnowflakeConnection→
forum

Still have questions?

Our forums are full of helpful information and Streamlit experts.
