Setting up Twitter API

First, create a Twitter developer account and an app to acquire the Bearer token. Then, utilize UrlFetchApp in Google Apps Script to access Twitter API, incorporating "Authorization: Bearer ..." in the request header.

Below is a step-by-step guide on how to use Google Apps Script (GAS) to connect to the Twitter API and retrieve the latest tweets mentioning the word “workspace.” In this example we’ll use Twitter’s API v2 with a Bearer Token for authentication. (Note that Twitter’s API policies and endpoints may change, so always refer to the Twitter API documentation for the latest details.)

Important:

- You must have a Twitter Developer account and create an application to obtain your API credentials.
- This example uses the Bearer Token (for OAuth 2.0) which is the simplest method for accessing public endpoints like “recent search.”
- Make sure your Twitter app has the necessary permissions for the endpoint you wish to call.

Step 1. Get Your Twitter API Credentials

1. Apply for a Developer Account:
   If you haven’t already, apply for a Twitter Developer account at Twitter Developer Portal.
2. Create a Twitter App:
   Once approved, create an app in the Twitter Developer Dashboard to obtain your API keys and tokens.
3. Obtain Your Bearer Token:
   Navigate to the “Keys and Tokens” section for your app and copy the Bearer Token. This token will be used for authentication in your API requests.

Step 2. Write the Google Apps Script Code

1. Open Google Apps Script:
   Go to script.google.com and create a new project.

```
function getTweets() {
 const BEARER_TOKEN = CONFIG.BEARER_TOKEN;
 const query = encodeURIComponent('Workspace');
 const url = "https://api.twitter.com/2/tweets/search/recent?query=" + query;
 const options = {
```

```
  "method" : "get",
  "headers" : {
    "Authorization":"Bearer " + BEARER_TOKEN
  },
  "muteHttpExceptions": true
};
try {
  const response = UrlFetchApp.fetch(url,options);
  const responseData = response.getResponseCode();
  if(responseData === 200){
    const data = JSON.parse(response.getContentText());
    Logger.log(JSON.stringify(data,null,2));
  }else{
    Logger.log('Error: Response Code '+responseData)
  }
} catch(e){
  Logger.log("Error occured: "+e);
}
}
```

This Google Apps Script function, `getTweets()`, fetches recent tweets containing the word `"Workspace"` using Twitter's API v2. Below is a detailed breakdown of each part of the code.

1. Function Definition

`function getTweets() {`

- `getTweets()` is a user-defined function that retrieves recent tweets containing the keyword `"Workspace"`.

2. Retrieve Bearer Token

`const BEARER_TOKEN = CONFIG.BEARER_TOKEN;`

- `CONFIG.BEARER_TOKEN`:
    - The script assumes that `CONFIG` is an object that stores API credentials.

    - The Bearer Token is needed for authentication when making API requests to Twitter.
    - This ensures that the API call is authenticated and authorized.

3. Construct the Search Query

```
const query = encodeURIComponent('Workspace');
```

- `'Workspace'` is the search term for retrieving tweets.
- `encodeURIComponent()`:
    - Ensures that special characters (like spaces) are properly encoded in the URL.
    - Converts `"Workspace"` to `"Workspace"` (safe for URL use).

4. Build the API Endpoint URL

```
const url = "https://api.twitter.com/2/tweets/search/recent?query=" +
query;
```

- Constructs the full API request URL by appending the encoded query.
- Final API URL:
    - `"https://api.twitter.com/2/tweets/search/recent?query=Workspace"`
    - This will search for the most recent tweets containing `"Workspace"`.

5. Define API Request Options

```
const options = {
  "method" : "get",
  "headers" : {
    "Authorization": "Bearer " + BEARER_TOKEN
  },
  "muteHttpExceptions": true
};
```

- `options` object specifies how the request should be made:
    - `method: "get"` → Specifies a GET request to retrieve data.
    - `headers`:
        - `"Authorization": "Bearer " + BEARER_TOKEN`:
            - Uses Bearer Token authentication to access the Twitter API.
            - `"Bearer "` must be prepended to the token.
    - `muteHttpExceptions: true`:

    - Prevents the script from stopping if the request fails.
    - Allows handling of errors manually.

6. Try-Catch Block (Error Handling)

```
try {
```

- Ensures the script doesn't crash if an error occurs.
- If an API request fails, the script logs the error instead of stopping execution.

7. Make the API Request

```
const response = UrlFetchApp.fetch(url, options);
```

- Uses `UrlFetchApp.fetch()` to send the API request.
- Stores the response received from the Twitter API.

8. Get the Response Code

```
const responseData = response.getResponseCode();
```

- Retrieves the HTTP status code of the API response.
- The response code helps determine whether the request was successful or failed.

9. Handle the Response

Case 1: Success (Response Code 200)

```
if(responseData === 200){
  const data = JSON.parse(response.getContentText());
  Logger.log(JSON.stringify(data,null,2));
}
```

- If response code is `200` (OK):
    - Parses the response JSON data into a JavaScript object.
    - Logs the formatted JSON response using `Logger.log()`.

Case 2: Error (Non-200 Response)

```
else{
  Logger.log('Error: Response Code ' + responseData);
}
```

- If the response code is not 200, an error message is logged.

10. Catch Errors

```
} catch(e){
  Logger.log("Error occurred: " + e);
}
```

- If an unexpected error occurs (e.g., network issues, invalid API token), it is caught and logged.

How to Use This Code

1. Set Up Twitter API Credentials

- You need a Twitter Developer Account and a Bearer Token.
- Store the token securely in Google Apps Script using a `CONFIG` object.

2. Create the `CONFIG` Object

If `CONFIG` is not defined, create a separate `config.gs` file and add:

```
const CONFIG = {
  BEARER_TOKEN: "YOUR_TWITTER_BEARER_TOKEN_HERE"
};
```

3. Run the Script

- Open Google Apps Script (`script.google.com`).
- Paste the script and Run `getTweets()`.
- Check View > Logs to see the retrieved tweets.

Example API Response (Twitter API v2)

If successful, `Logger.log()` will output something like:


```
{
  "data": [
    {
      "id": "1886210042065834261",
```

```
      "text": "RT @Google: Millions of small businesses use AI in
Google Workspace..."
    },
    {
      "id": "1886208813898121423",
      "text": "RT @Dwriteway: A cluttered workspace is a cluttered
mind..."
    }
  ]
}
```

Summary of Key Points

Feature	Description
Authentication	Uses Twitter API v2 with Bearer Token
Search Query	Searches for recent tweets containing `"Workspace"`
API Request	Uses `UrlFetchApp.fetch()` with a GET request
Response Handling	Checks for `200` status, logs tweets, handles errors
Error Handling	Uses `try-catch` to prevent script failures

Populate the Tweet data to a Spreadsheet

```
const SHEETID = '1y7*****PGw';

function fetchAndStore(){

   const BEARER_TOKEN = CONFIG.BEARER_TOKEN;

   if(!BEARER_TOKEN){

     Logger.log("Error need token");

   }
```

```
const query = encodeURIComponent('Workspace');

const url = "https://api.twitter.com/2/tweets/search/recent?query=" + query;

const options = {

  "method" : "get",

  "headers" : {

    "Authorization":"Bearer " + BEARER_TOKEN

  },

  "muteHttpExceptions": true

};

try {

  const response = UrlFetchApp.fetch(url,options);

  const responseData = response.getResponseCode();

  if(responseData === 200){

    const data = JSON.parse(response.getContentText());

    if(data && data.data){

      storeTweetsInSheets(data.data);

    }else{

      Logger.log("No tweets found.");

    }

  }else{
```

```
    Logger.log('Error: Response Code '+responseData)

  }

 }catch(e){

  Logger.log("Error occured: "+e);

 }

}

function storeTweetsInSheets(tweets){

 const sheet = getOrCreate("Twitter Data");

 if(sheet.getLastRow() === 0){

  sheet.appendRow(["Tweet ID","Tweet Text","Edit History","Tweet URL"]);

 }

 Logger.log(tweets);

 tweets.forEach(tweet => {

  const tweetID = tweet.id;

  const tweetText = tweet.text;

  const tweetHistory = tweet.edit_history_tweet_ids ?
tweet.edit_history_tweet_ids.join(", "):"NA";

  const tweetURL = "https://twitter.com/user/status/"+tweetID;

  sheet.appendRow([tweetID,tweetText,tweetHistory,tweetURL]);

 })

}
```

```
function getOrCreate(sheetName){

 const ss = SpreadsheetApp.openById(SHEETID);

 let sheet = ss.getSheetByName(sheetName);

 if(!sheet){

   sheet = ss.insertSheet(sheetName);

 }

 return sheet;

}
```

This Google Apps Script **fetches tweets** mentioning `"Workspace"`, processes the tweet data, and **stores the results into a Google Sheet**.

1. Constant for Google Sheet ID

```
const SHEETID = '1y7kgMKfxmyXCW8tUSPw5kes-mxtuebobKDPeYcokPGw';
```

- This stores the **Google Spreadsheet ID** where the fetched tweets will be saved.
- The ID is a **unique identifier** for a Google Sheet.
- This ID is used in `SpreadsheetApp.openById(SHEETID)` to access the specific spreadsheet.

2. Main Function: `fetchAndStore()`

```
function fetchAndStore() {
```

- This function **fetches recent tweets** from Twitter's API and **saves them to a Google Sheet**.

3. Retrieve Bearer Token & Validate

```
const BEARER_TOKEN = CONFIG.BEARER_TOKEN;
if (!BEARER_TOKEN) {
```

```
  Logger.log("Error need token");
}
```

- **Fetches the API token** from the `CONFIG` object.
- If the **token is missing**, logs an error and the function execution **continues**.

4. Encode Search Query & Build API URL

```
const query = encodeURIComponent('Workspace');
const url = "https://api.twitter.com/2/tweets/search/recent?query=" +
query;
```

- **Encodes `"Workspace"`** to make it safe for URL use.
- Constructs the **Twitter API endpoint URL** to search for recent tweets containing `"Workspace"`.

5. Define API Request Options

```
const options = {
  "method": "get",
  "headers": {
    "Authorization": "Bearer " + BEARER_TOKEN
  },
  "muteHttpExceptions": true
};
```

- **Specifies request type**: `GET`
- **Sets headers**:
  - **Authorization:** Uses the `BEARER_TOKEN` to authenticate the request.
- **muteHttpExceptions: true**:
  - Prevents the script from stopping on an API error.
  - Allows **manual error handling** instead.

6. Try-Catch Block to Handle API Request

```
try {
  const response = UrlFetchApp.fetch(url, options);
  const responseData = response.getResponseCode();
```

- Uses `UrlFetchApp.fetch()` to send the request to Twitter's API.
- Stores the **HTTP response code** (`200` for success).

7. Handle Response Data

Success: HTTP 200

```
if (responseData === 200) {
  const data = JSON.parse(response.getContentText());
  if (data && data.data) {
    storeTweetsInSheets(data.data);
  } else {
    Logger.log("No tweets found.");
  }
}
```

- **Parses the API response JSON**.
- Checks if the response **contains tweets**.
- If tweets are found, **calls `storeTweetsInSheets()`** to save them.

Failure: Non-200 Response

```
else {
  Logger.log('Error: Response Code ' + responseData);
}
```

- Logs an error message if the API request fails.

8. Handle Unexpected Errors

```
} catch (e) {
  Logger.log("Error occurred: " + e);
}
```

- Catches and logs unexpected errors (e.g., network issues, API failure).

9. Function: `storeTweetsInSheets(tweets)`

```
function storeTweetsInSheets(tweets) {
```

- **Receives the list of tweets** and saves them to the **Google Sheet**.

10. Get or Create the Google Sheet

```
const sheet = getOrCreate("Twitter Data");
```

- Calls `getOrCreate("Twitter Data")` to **get or create** the sheet named `"Twitter Data"`.

11. Add Column Headers (Only if Sheet is Empty)

```
if (sheet.getLastRow() === 0) {
  sheet.appendRow(["Tweet ID", "Tweet Text", "Edit History", "Tweet URL"]);
}
```

- If the sheet is **empty**, adds **column headers**.

12. Loop Through Each Tweet and Extract Data

```
Logger.log(tweets);
tweets.forEach(tweet => {
  const tweetID = tweet.id;
  const tweetText = tweet.text;
  const tweetHistory = tweet.edit_history_tweet_ids ? tweet.edit_history_tweet_ids.join(", ") : "NA";
  const tweetURL = "https://twitter.com/user/status/" + tweetID;
  sheet.appendRow([tweetID, tweetText, tweetHistory, tweetURL]);
});
```

- Loops through each **tweet object** and extracts:
  - **`tweet.id`** → Unique tweet ID.
  - **`tweet.text`** → Tweet content.
  - **`tweet.edit_history_tweet_ids`** → List of tweet edits (converted to a string).
  - **Constructs `tweetURL`** → Direct link to the tweet.
- **Appends the extracted data** into the Google Sheet.

13. Function: `getOrCreate(sheetName)`

```
function getOrCreate(sheetName) {
  const ss = SpreadsheetApp.openById(SHEETID);
```

```
  let sheet = ss.getSheetByName(sheetName);
  if (!sheet) {
    sheet = ss.insertSheet(sheetName);
  }
  return sheet;
}
```

- **Opens the Google Spreadsheet** using `SHEETID`.
- **Checks if the sheet exists**:
  - If **not**, creates a new sheet.
- Returns the **Google Sheet object**.

How the Code Works

1. **Fetches recent tweets** containing `"Workspace"` from Twitter API.
2. **Parses the API response** to check for valid tweets.
3. **If tweets exist**:
   - Calls `storeTweetsInSheets()` to save the tweets.
   - Retrieves or creates a Google Sheet.
   - If the sheet is empty, adds **column headers**.
   - **Appends tweet details**: Tweet ID, text, edit history, and URL.
4. **If the request fails**:
   - Logs an **error message**.

How to Use This Code

1. Store the Bearer Token Securely

- Open **Google Apps Script Editor**.
- Add the `CONFIG` object in a separate `config.gs` file:

```
const CONFIG = {
  BEARER_TOKEN: "YOUR_TWITTER_BEARER_TOKEN_HERE"
};
```

2. Set the Google Sheet ID

- Replace **`SHEETID`** with your actual **Google Spreadsheet ID**.

3. Create a Google Sheet

- Open Google Sheets.
- Copy its **ID** from the URL (`docs.google.com/spreadsheets/d/XXXXXX/edit`).
- Name the sheet `"Twitter Data"`.

4. Run the Script

- Open **Apps Script Editor**.
- Run `fetchAndStore()`.
- Open the Google Sheet to see the tweets.

Expected Output in Google Sheets

Tweet ID	Tweet Text	Edit History	Tweet URL
1886210042065834261	RT @Google: Millions of small businesses use AI in Google Workspace...	1886210042065834261	View Tweet
1886208813898121423	RT @Dwriteway: A cluttered workspace is a cluttered mind...	1886208813898121423	View Tweet

Key Features

✅ **Authenticates with Twitter API** using a Bearer Token
✅ **Fetches recent tweets** containing `"Workspace"`
✅ **Handles errors gracefully** using try-catch
✅ **Saves tweets in Google Sheets** (appends new data)
✅ **Handles missing sheets** (creates if not found)