**Here's a User Guide for your Automated API Monitoring Script:**

**User Guide: cURL Automated API Monitoring Script**

**Overview:**

**This script is designed to automatically monitor and check the status and performance of multiple APIs. It allows you to monitor the response times and HTTP status codes of different APIs based on a configuration file. The script sends HTTP requests to the listed APIs using curl, collects response times and status codes, and logs the results periodically.**

**Key Features:**

* **HTTP Method Support: Supports both GET and POST methods for making API requests.**
* **Custom Headers and Data: Allows the configuration of headers and request body for POST requests.**
* **Monitoring Interval: Monitors APIs at regular intervals defined in the configuration file.**
* **Background Monitoring: Runs monitoring in a separate thread to keep the main program alive.**
* **Logging: Logs the status code, response time, and response content of each API request to the console.**

**Pre-requisites:**

1. **Python 3.x:**
   * **Make sure you have Python 3 installed. You can check by running:**
   * **python3 --version**
2. **Required Libraries:**
   * **The script uses the following libraries:**
     + **subprocess: For executing system commands (in this case, curl).**
     + **json: For reading and processing the configuration file.**
     + **threading: For running monitoring in a separate thread.**
     + **time: For controlling the intervals between API checks.**

**These are built-in Python libraries, so no additional installations are required.**

1. **Curl:**
   * **The script relies on curl, a command-line tool to make HTTP requests. Ensure curl is installed and available in your system's PATH.**

**Configuration File:**

**The script requires a JSON configuration file (config.json) that defines which APIs to monitor, the HTTP method to use, optional headers, and request data. Below is an example of how the config.json should be structured:**

**{**

**"monitoring\_interval": 60,**

**"apis": [**

**{**

**"url": "https://api.example.com/endpoint1",**

**"method": "GET",**

**"headers": {**

**"Authorization": "Bearer YOUR\_API\_KEY"**

**}**

**},**

**{**

**"url": "https://api.example.com/endpoint2",**

**"method": "POST",**

**"headers": {**

**"Content-Type": "application/json"**

**},**

**"data": {**

**"key": "value"**

**}**

**}**

**]**

**}**

**Configuration Details:**

* **monitoring\_interval: Defines the number of seconds between each monitoring cycle (e.g., 60 seconds).**
* **apis: A list of API endpoints to be monitored.**
  + **url: The URL of the API endpoint to monitor.**
  + **method: The HTTP method to use (GET or POST).**
  + **headers: (Optional) A dictionary of headers to include in the request.**
  + **data: (Optional) The data to send in the body of a POST request. This should be a dictionary.**

**How It Works:**

1. **Load Configuration:**
   * **The script loads the configuration file (config.json) and reads the list of APIs to monitor, along with the required HTTP methods and headers.**
2. **API Request Sending:**
   * **The send\_request function uses curl to send the defined request to each API. It captures the response time, HTTP status code, and response data (if available).**
3. **Logging Results:**
   * **For each API request, the script logs the following details to the console:**
     + **The URL of the monitored API.**
     + **The response time in seconds.**
     + **The HTTP status code returned by the API (e.g., 200 for success).**
     + **The response content (the raw response, useful for debugging).**
4. **Monitoring Loop:**
   * **The script runs in a loop, checking each API at the defined interval (monitoring\_interval).**
   * **It logs the results for each API, waits for the defined interval, and then repeats the process indefinitely.**
5. **Background Monitoring:**
   * **The script runs the monitoring process in a background thread, which allows the main program to stay alive without blocking.**

**How to Use:**

1. **Prepare the Configuration File:**
   * **Create a config.json file in the same directory as the script. Define the APIs you want to monitor, along with their HTTP method, headers, and data (if applicable).**
2. **Run the Script:**
   * **To start the monitoring, simply run the Python script:**
   * **python3 monitor\_apis.py**
   * **The script will automatically start the monitoring process and log the results to the console.**
3. **Monitoring and Logging:**
   * **The script will print the monitoring results for each API, including response time, status code, and response content.**

**Sample Output:**

**Monitoring URL: https://api.example.com/endpoint1**

**Response Time: 0.234 seconds**

**Status Code: 200**

**Response: {"status":"success", "data": {...}}**

**--------------------------------------------------**

**Monitoring URL: https://api.example.com/endpoint2**

**Response Time: 0.523 seconds**

**Status Code: 400**

**Response: {"error":"Invalid request"}**

**--------------------------------------------------**

**Advanced Configuration:**

* **Custom Headers: Add custom headers for authentication, content type, etc.**
* **POST Data: For POST requests, include a dictionary of data that will be sent in the body of the request.**

**Troubleshooting:**

* **Curl Not Found: If you encounter an error saying curl is not found, you need to install it. On most Linux distributions, you can install it using:**
* **sudo apt-get install curl**

**On macOS, it should be pre-installed. On Windows, you can download it from the** [**official website**](https://curl.se/download.html)**.**

* **Invalid JSON in Configuration File: Ensure that the config.json file is properly formatted and all keys are correctly specified.**

**Script Termination:**

**To stop the script, you can press Ctrl+C in the terminal to safely terminate the process.**

**This guide should help you set up, configure, and run the Automated API Monitoring Script for continuous monitoring of your APIs.**