**Summary of Learnings and How It Relates to the Exercise**

\* **API Interaction is Key:** The core of this exercise was successfully interacting with the Cortex XSIAM API. I identified the correct v2 endpoint, headers (`Authorization` and `x-xdr-auth-id`), and payload structure for pagination and sorting. The initial `500 Internal Server Error` was due to using an incorrect API hostname or payload structure, which I debugged..

\* **Credentials Matter** The `NoCredentialsError` is a very common issue when working with AWS SDKs. Fixed permissions and used the right S3 access and secret keys

\* **Iterative Development & Debugging:** I went through several iterations to fix issues:

\* Incorrect API endpoints/payloads.

\* The `ModuleNotFoundError` (Python environment).

\* `NoCredentialsError` (AWS auth).

\* Linter errors.

\* **Logging for Clarity:** The detailed logs are crucial in seeing what the script was doing at each step, what data it was sending, what responses it received, and where errors occurred.

\* **Step 2: Pull Alerts from Cortex XSIAM**

\* API Endpoint Identification: `POST /public\_api/v2/alerts/get\_alerts\_multi\_events/` is correctly used.

\* API Calls & Pagination: The logs show the script fetching page 0. Since `total\_count` was 11 and `search\_to` was 100, it correctly determined there were no more pages (`Next page token for page 0: None`). If there were more than 100 alerts, my`while True` loop with `next\_page\_token` would have handled it.

\* Documentation: The logs themselves provide a live "document" of the request payload.

\*  **Step 3: Store Alerts in Cloud Storage**

\* Cloud Service: AWS S3 (`gbcortex` bucket).

\* Script for Upload: `main.py` with `boto3`. The logs clearly show the "Attempting to clear alerts folder," "Attempting to upload alert...", and "Successfully uploaded alert..." messages.

\* Structured Format: Alerts are stored as individual JSON files (e.g., `alerts/310.json`).

\* Documentation: The script (`clear\_s3\_alerts\_folder`, `upload\_alert\_to\_s3` functions) and the logs serve as documentation.

[ ]: