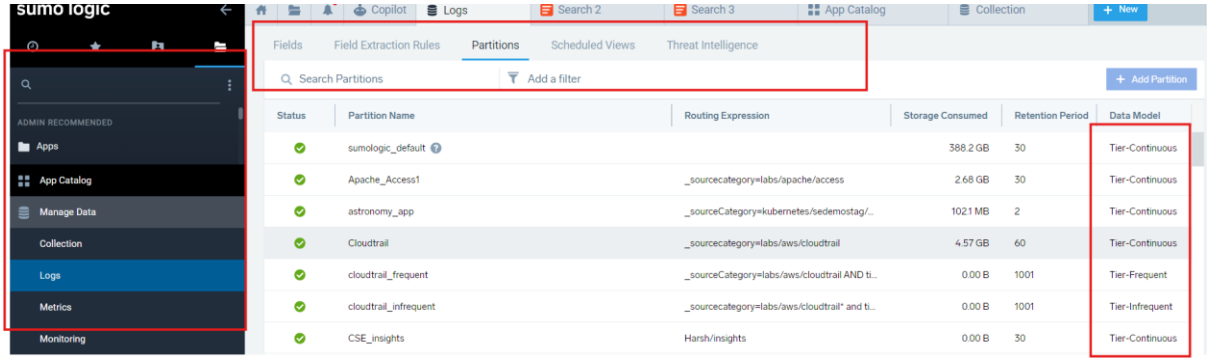
Data Searching and Log Parsing

Introduction

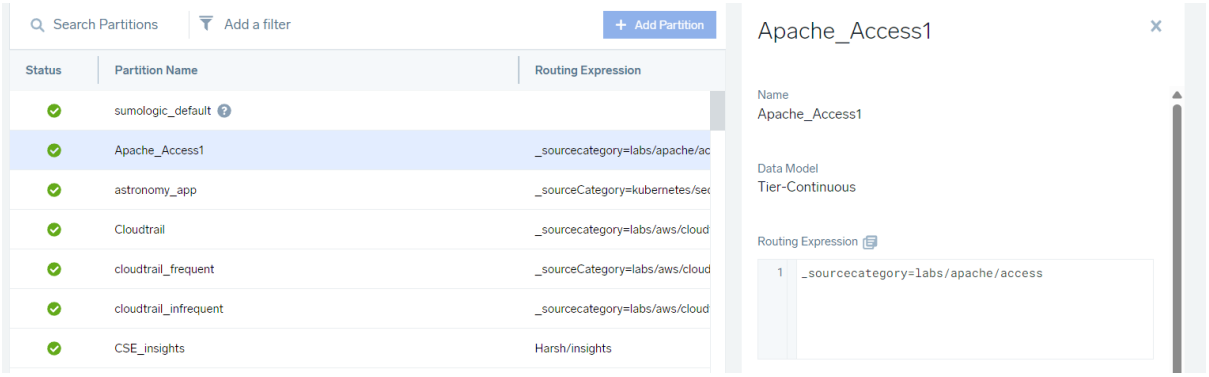
In this lab, I explored how to search, parse, and analyse log data to extract meaningful insights. I learned the difference between raw and parsed logs, built queries in Basic Mode, and understood how parsing improves the readability and efficiency of data analysis. Through this exercise, I gained practical experience in converting unstructured log data into a structured format that is easier to search, interpret, and utilize effectively.



First, I opened the logs from the left-side panel by navigating to **Manage Data → Logs**. Then, on the top panel, I selected the **partition** where I found the logs.

These logs were divided into three data tiers:

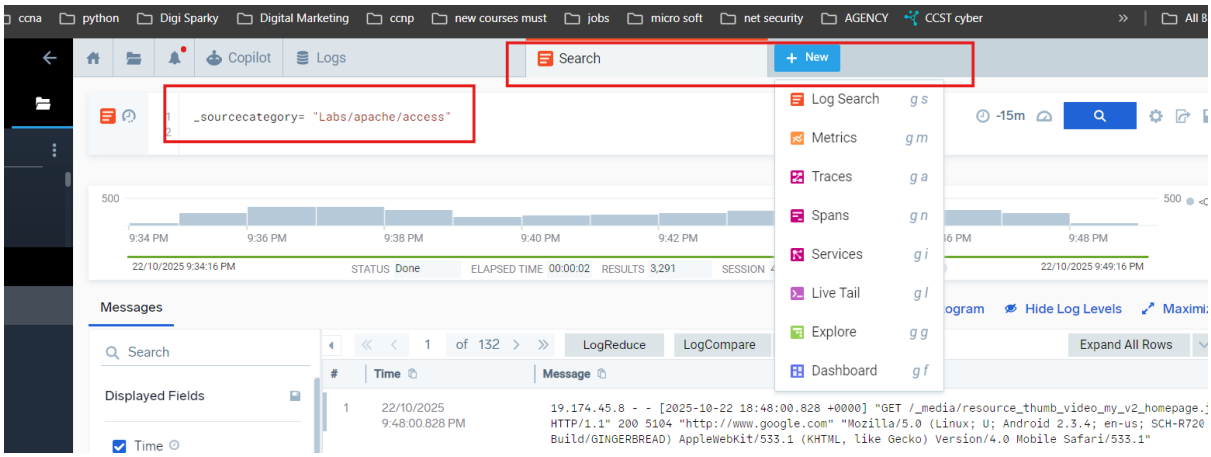
- **Tier – Continuous:** Stores real-time or constantly updated logs used for live monitoring and quick analysis.
- **Tier – Frequent:** Contains logs that are accessed regularly but not in real time, optimized for daily operations and queries.
- **Tier – Infrequent:** Holds older or rarely accessed logs, mainly kept for long-term storage and compliance at a lower cost.



Next, I selected the **Apache data** stored in the **Tier – Continuous** section.
On the right side of the panel, I used the **routing expression** to filter the specific dataset:

```
_sourcecategory = labs/apache/access
```

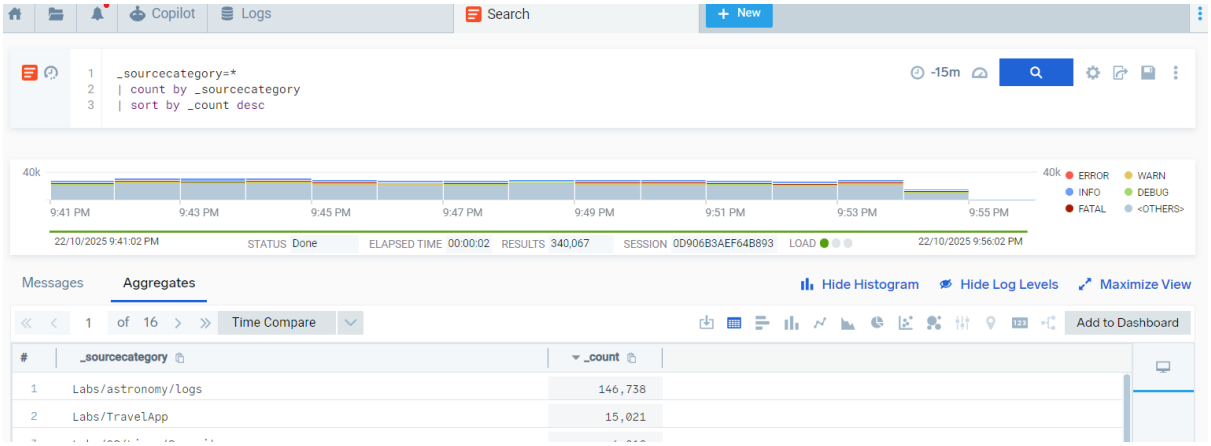
This expression helped me narrow down the logs to only those related to Apache access activity, making the data easier to analyze and interpret.



Next, on the **top panel**, I clicked the **plus (+)** icon to start a new **Log search**.
A new search window opened, where I pasted the same **routing expression** into the search bar:

```
_sourcecategory = labs/apache/access
```

After running the query, all the logs related to Apache appeared in the results section. However, at this stage, the logs were displayed in their **raw form**, making them difficult to read and interpret directly.



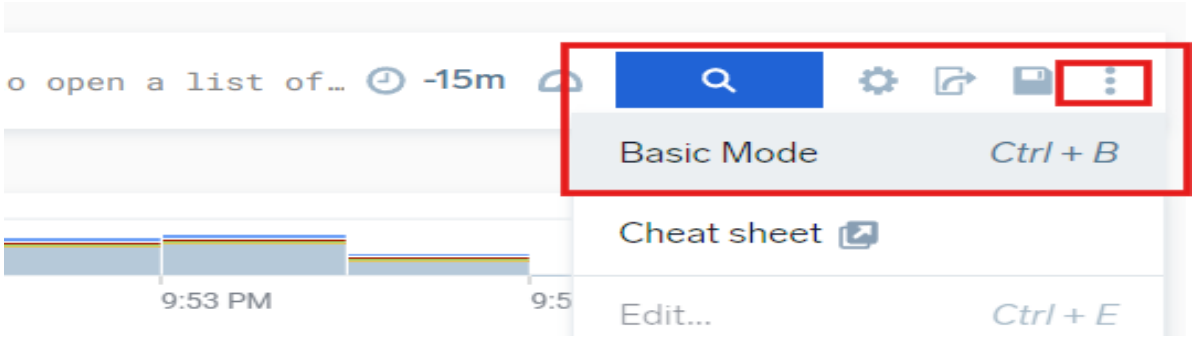
As the search was in **Advanced Mode**, I entered the following query to retrieve the logs:

```
_sourcecategory = *
| count by _sourcecategory
| sort by _count desc
```

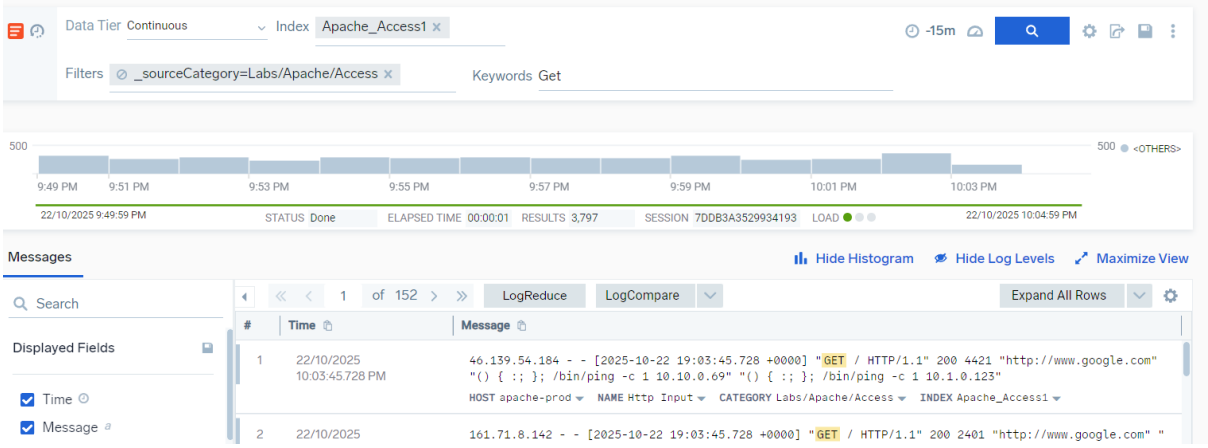
This query lists all available data sources by counting the number of log entries under each **_sourcecategory**.
The command **`count by _sourcecategory`** groups the logs based on their source category, while **`sort by _count desc`** arranges them in descending order according to the number of

log entries.
This helps identify which sources generate the most log data.

After executing the query, the required data was displayed as shown in the above screenshot.



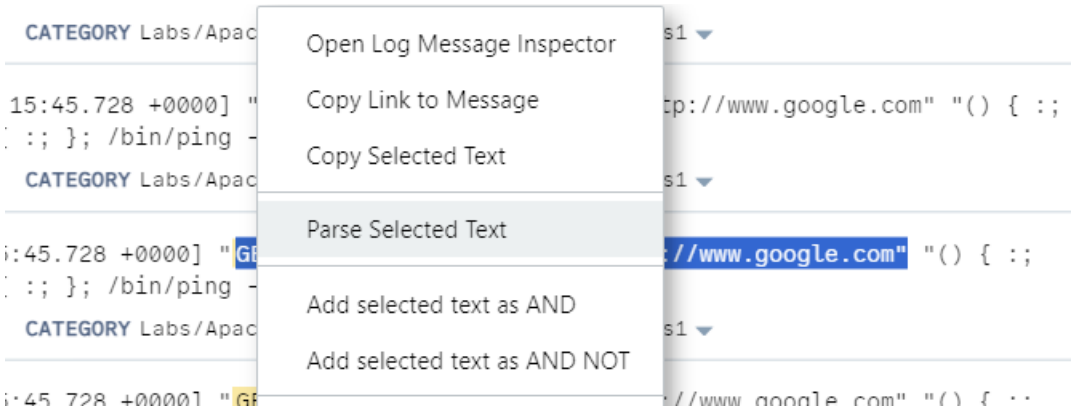
Next, I converted the search to **Basic Mode** by clicking on the **three-dot (⋮) menu** on the right side of the search bar and selecting **"Basic Mode"** from the options.
This action switched the interface from advanced query writing to a simpler visual mode, as shown in the screenshot above.



In **Basic Mode**, the search bar layout changed to include several fields:

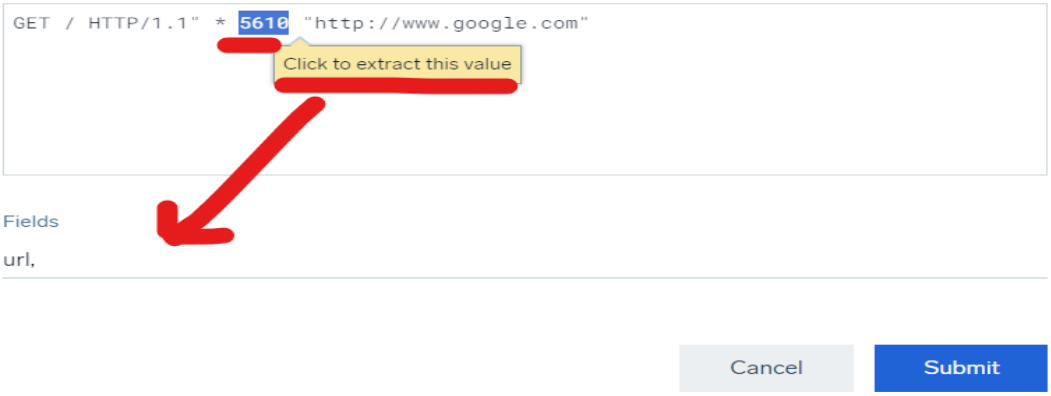
- **Data Tier:** Allows selection from the three available data tiers (Continuous, Frequent, Infrequent).
- **Index:** Specifies the dataset or log source to search within.
- **Filters:** Provides options to narrow down the search results based on specific criteria.
- **Keywords:** I entered **GET** here to filter logs containing this keyword.

Even after applying these selections, the targeted data was still displayed in **raw form**, which required **parsing** to make it more readable and structured.



I selected the portion of the data starting from **GET** to the end of the link.
Then, I **right-clicked** on the selection and chose **"Parse Selected Text"**.

This action allowed me to convert the raw log entry into a structured format, making it easier to read, analyze, and extract meaningful information.



A new window opened where the copied log was displayed and ready for parsing. The **Parse Text** dialog allows me to select portions of text to extract as fields, which are then automatically added to the search box to build the query.

1. I highlighted the **URL** and clicked **"Click to extract this value"**. In the **Fields** box, I entered `URL` followed by a comma.
2. Next, I highlighted the **status code (200)** and extracted it, entering `status_code` in the Fields box. I used an underscore to follow best practices for naming fields.
3. I then highlighted the **file size**, extracted it, and entered `size` in the Fields box.
4. Finally, I highlighted the **referring URL** (excluding quotation marks), extracted it, and added `referrer` in the Fields box.

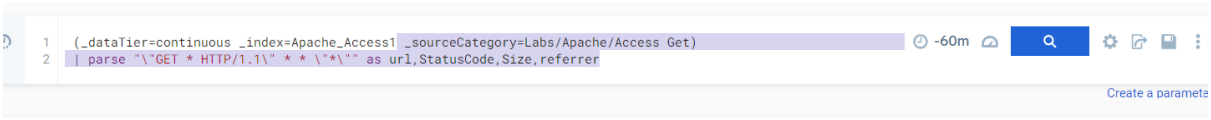
After clicking **Submit**, the parsing information was added to my query. Although I could have typed the query manually, the **Parse Text** dialog simplifies the process without needing to remember the exact syntax.

Note: This parsed query can only be executed in **Advanced Mode**.

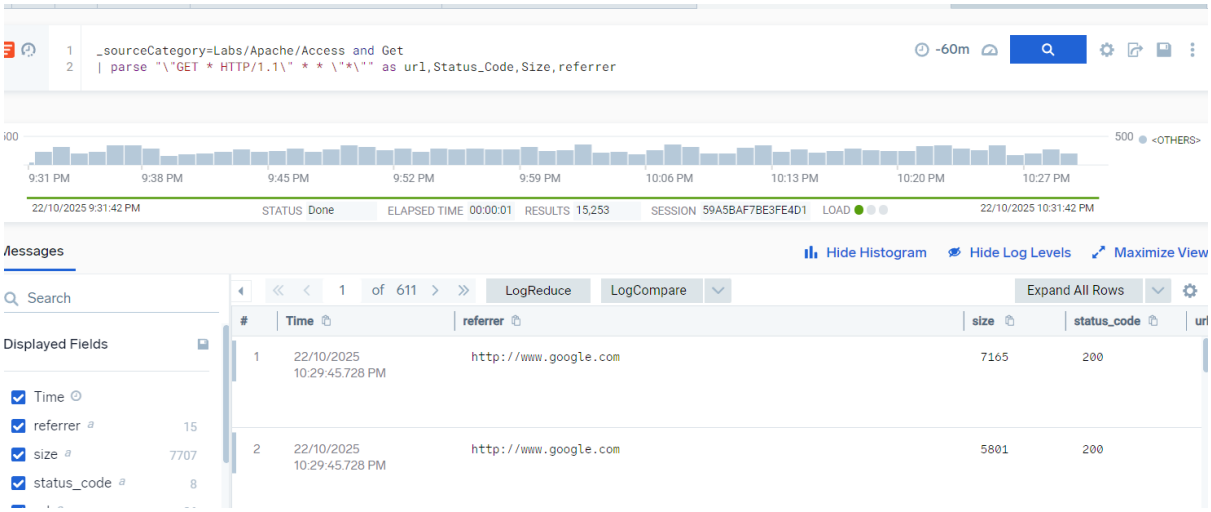
The final query became:

```
_sourceCategory=Labs/Apache/Access and GET
| parse "\"GET * HTTP/1.1\" * * \"*\" " as url,status_code,size,referrer
```

This query now converts raw log entries into structured fields, making it easier to analyze specific components of each log.



After building the parsed query, I **copied it** and pasted it into the **Advanced Search** bar.
I set the **time range** to the last **60 minutes** to focus on recent log data.



Upon running the query, the logs were displayed in a **clear and readable format**, with the extracted fields (`url`, `status_code`, `size`, `referrer`) organized for easy analysis, as shown in the screenshot above.

To save the query for future use, I clicked the **More Actions** icon and selected **Save As**.
I entered a name for the search, in this case **"Apache Status Codes"**. Adding a description is optional. The **query** and **time range** were automatically filled in, but I had the option to modify them before saving.

By default, the saved search was added to my **Personal** folder. I also had the option to select a subfolder or create a **new folder** by clicking **+ New Folder**.

Finally, I clicked **Save** to add the search to the library. The saved search name also appeared on the **top tab bar**, making it easy to access later.

Conclusion

In this lab, I learned how to search, filter, and parse log data effectively. I understood the difference between **raw logs**, which are unstructured and hard to read, and **parsed logs**, which are organized into meaningful fields for easier analysis. By using both **Basic** and **Advanced Mode**, I practiced building queries, extracting specific data, and saving searches for future use.

This exercise helped me see the **benefits of parsing data**, including improved readability, faster analysis, and the ability to focus on relevant information. Overall, the lab strengthened my skills in working with log data and preparing it for meaningful insights.