
📑 Log Parsing Example: Extracting HTTP Status Codes

Search Query:

_sourceCategory=Labs/Apache/Access

| parse "HTTP/1.1\" * " as status_code

🔎 Explanation

1. **_sourceCategory=Labs/Apache/Access**
    - Filters logs to only include those from the Labs/Apache/Access category (Apache access logs).
    - This ensures you're only working with relevant data.
2. **parse "HTTP/1.1\" * " as status_code**
    - The parse operator looks at the raw log line and extracts the value that comes **after HTTP/1.1" and before the next space**.
    - In Apache access logs, this spot is where the **HTTP status code** (e.g., 200, 404, 500) appears.

Example raw log:

192.168.1.5 - - [19/Aug/2025:10:45:12 +0000] "GET /index.html HTTP/1.1" 200 2326

- After "HTTP/1.1" → the next value is **200** → that gets extracted as status_code.

3. **Result:**
    - Each log entry now has a new field called status_code.
    - You can run further analysis on this field (like grouping, counting, or filtering).

📊 Example Extended Query

_sourceCategory=Labs/Apache/Access

| parse "HTTP/1.1\" * " as status_code

| count by status_code

| sort by _count desc

Explanation:

- count by status_code → Groups logs by status codes (200, 404, 500, etc.).
- sort by _count desc → Shows the most common status codes first.

Result Example:

status_code _count

---------- ------

200 12500

404 730

500 120

✅ **Why This Matters**

- **Status codes** tell you how your web server is responding:
  - 200 = OK (healthy traffic)
  - 404 = Not Found (possible broken links or scans)
  - 500 = Server Errors (something is wrong)
- By parsing and counting them, you can:
  - Detect spikes in 404 (possible recon attempts).
  - Monitor 500 errors (system health issues).
  - Track overall traffic quality.
