Lab Report — Working with Time in Splunk

Name: Dhruvish

Date: October 22, 2025

Platform: Splunk Cloud Trial

Objective

The objective of this lab was to master how time is handled in Splunk by exploring the time range picker, relative time modifiers (earliest and latest), data aggregation using timechart and bin, time formatting using strftime, and identifying ingestion delays using _time versus _indextime.

Tools Used

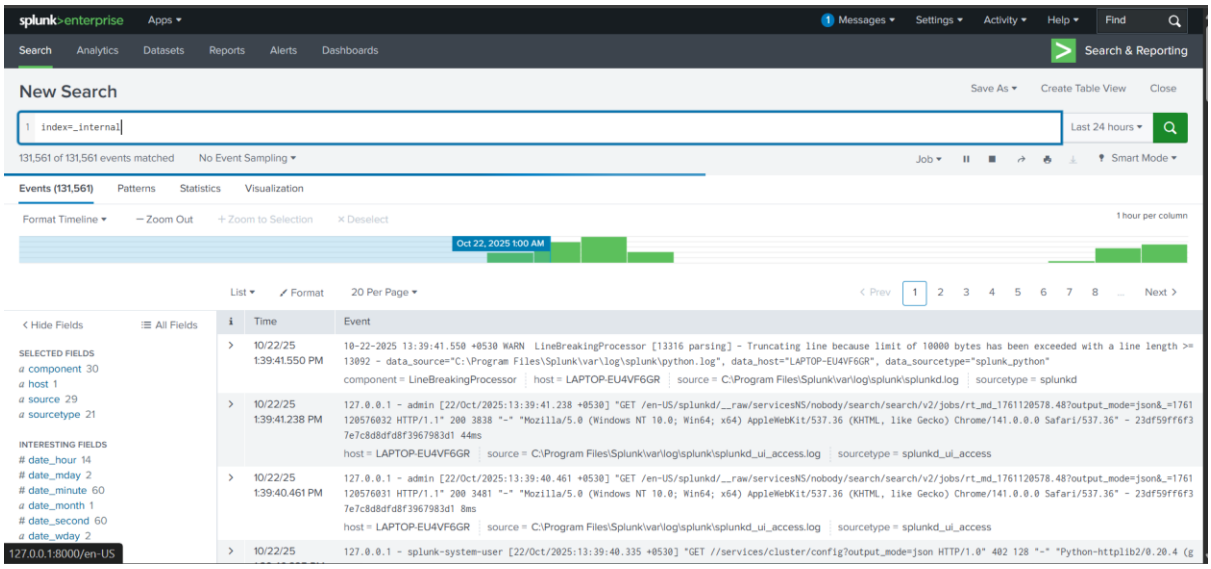
- **Splunk Cloud Trial** (or local Splunk Free)
- **Web Browser** (Chrome)

Step 1: Time Picker Basics

- Accessed *Apps → Search & Reporting*.
- Set the time range picker to **Last 24 Hours**.
- Ran the search:

  index=_internal

- Observed that the event list and timeline adjust dynamically based on the selected range, showing event distribution over time.



Step 2: Relative Time Modifiers (earliest/latest)

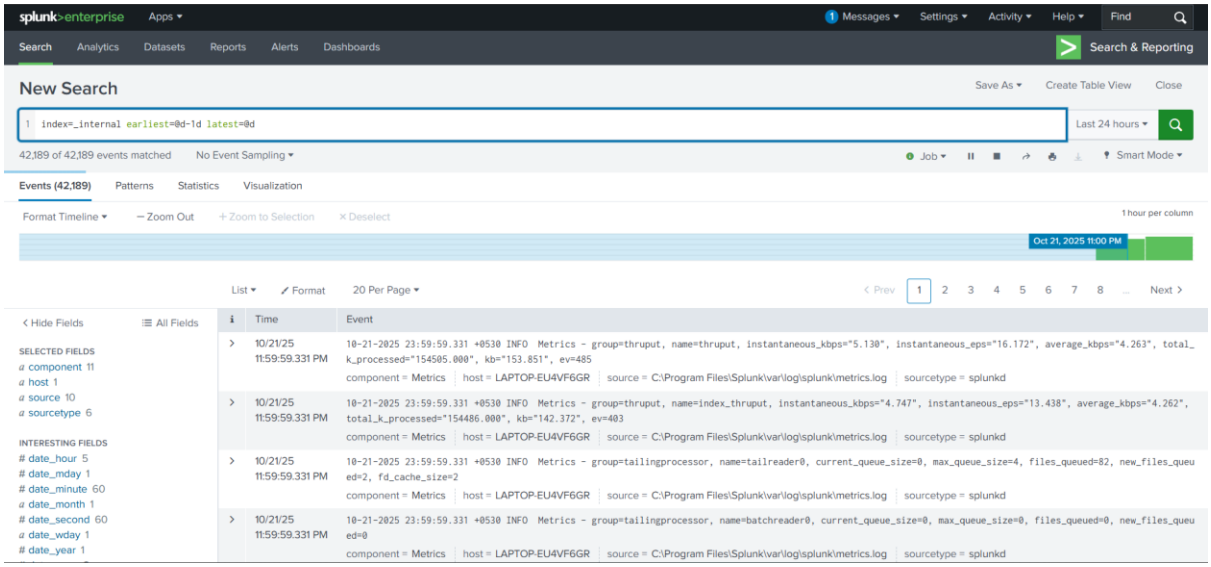
Explored earliest and latest time modifiers to override the picker directly in SPL:

- Last 2 hours (snapped to hour):

  index=_internal earliest=-2h@h latest=@h

- Yesterday only:

  index=_internal earliest=@d-1d latest=@d



- Last 7 full days (excluding today):

  index=_internal earliest=-7d@d latest=@d

These modifiers allowed precise time alignment and made temporal comparisons more accurate.

Step 3: Trends with timechart

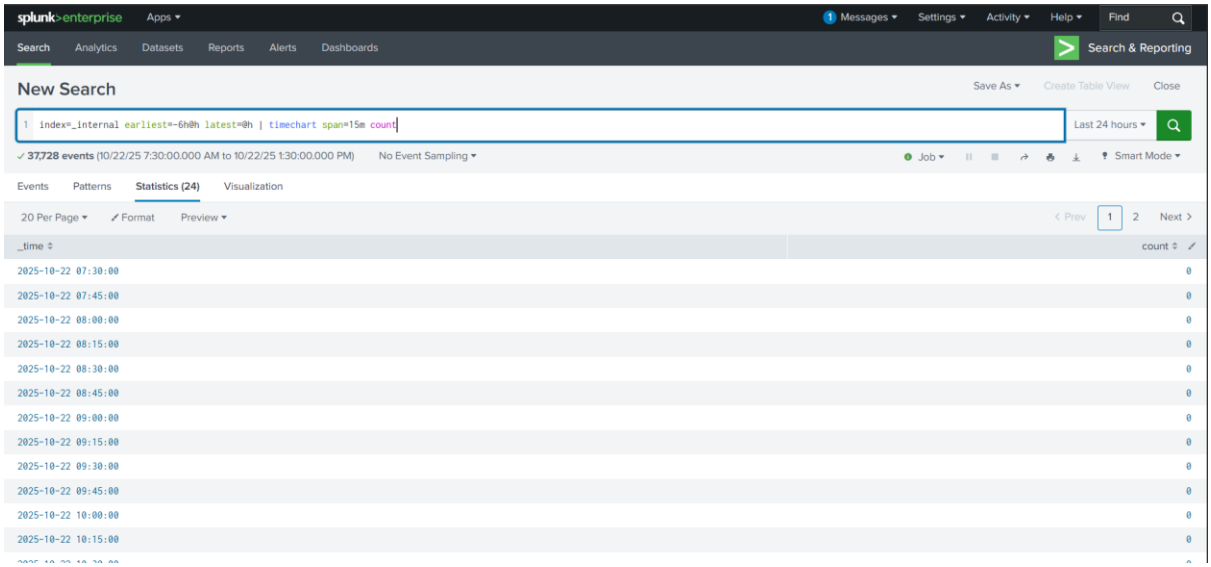
Used timechart to visualize data trends:

- Hourly event count:

  index=_internal earliest=-24h@h latest=@h | timechart span=1h count

- Compared different bucket spans:

  index=_internal earliest=-6h@h latest=@h | timechart span=15m count



index=_internal earliest=-6h@h latest=@h | timechart span=1h count

- Split counts by sourcetype:

  index=_internal earliest=-24h@h latest=@h | timechart span=1h count by sourcetype

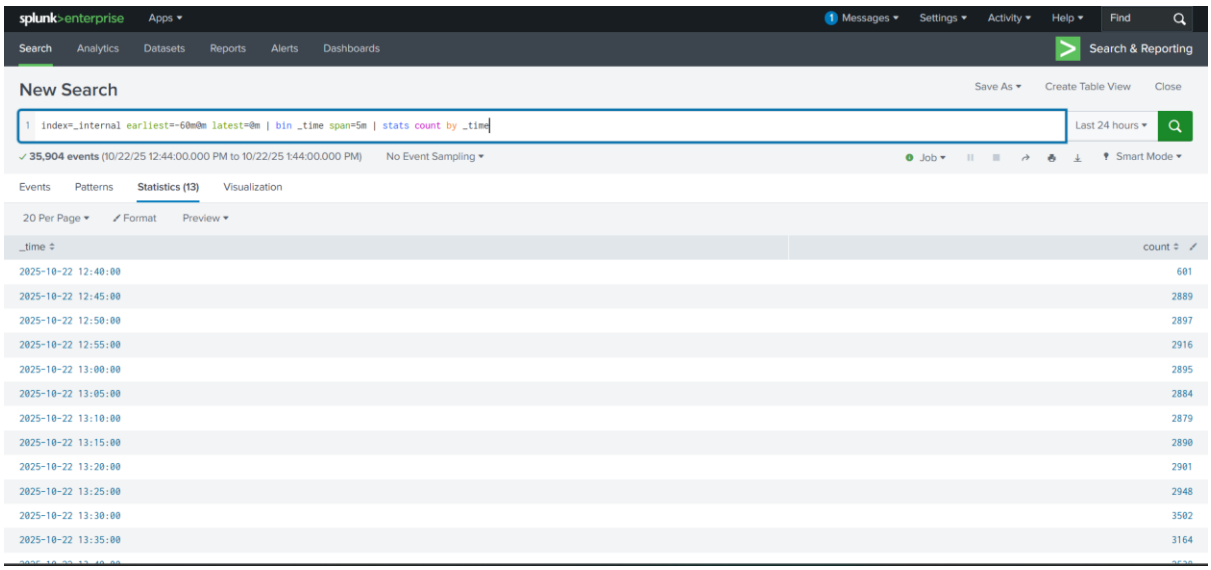
Observation:
Smaller spans (e.g., 15m) showed detailed fluctuations, while larger spans (1h) provided smoother trends for higher-level reporting.

Step 4: Manual Bucketing with bin

Manually aligned timestamps into fixed intervals for customized aggregation:

- 5-minute buckets:

  index=_internal earliest=-60m@m latest=@m | bin _time span=5m | stats count by _time



- Added a split by source:

  index=_internal earliest=-60m@m latest=@m sourcetype=splunkd | bin _time span=5m | stats count by _time source

Result: Created structured time-based buckets that improved event correlation accuracy.

Step 5: Formatting and Extracting Time Components

Used strftime to format timestamps and extract parts of time:

- Converted timestamps:

  index=_internal | eval ts=strftime(_time,"%F %T") | table _time ts host source

- Extracted hour and day:

  index=_internal earliest=-24h@h latest=@h

  | eval hour=strftime(_time,"%H"), dow=strftime(_time,"%a")

  | stats count by dow hour | sort dow hour

Observation: Helped identify hourly and daily event distribution trends.

Step 6: Event Time vs Index Time

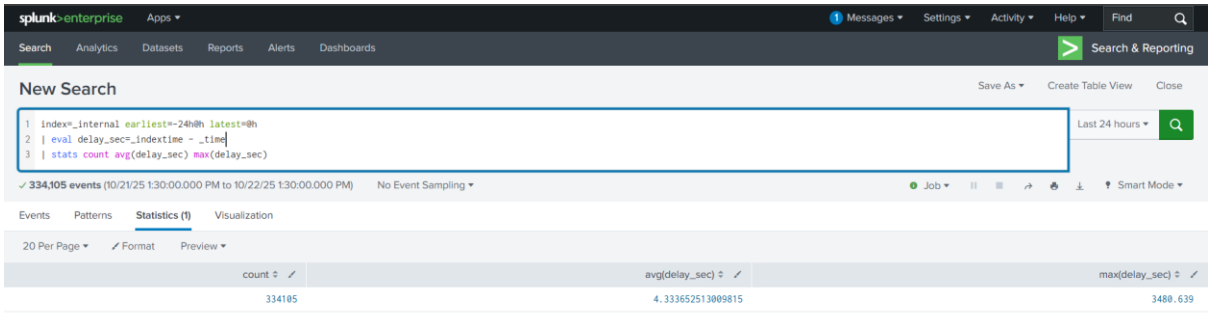
Compared event generation time (_time) vs indexing time (_indextime) to detect delays:

- Calculated delay:

  index=_internal earliest=-24h@h latest=@h

  | eval delay_sec=_indextime - _time

  | stats count avg(delay_sec) max(delay_sec)



- Found delayed events (>60 seconds):

  index=_internal earliest=-24h@h latest=@h

  | eval delay_sec=_indextime - _time

  | where delay_sec > 60

| eval event_time=strftime(_time,"%F %T"), index_time=strftime(_indextime,"%F %T")

| table event_time index_time delay_sec host source sourcetype

Observation: Identified latency between event creation and indexing—useful for diagnosing pipeline issues.

Step 7: Time Zone and Preferences

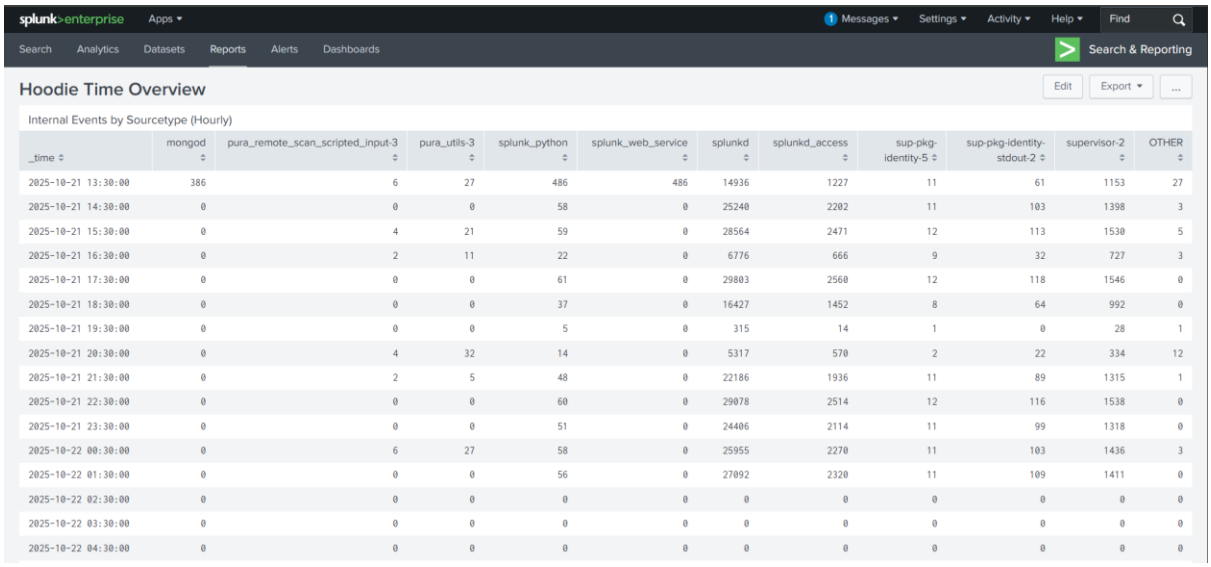
Changed time zone settings under *User Preferences* and observed shifts in time labels within visualizations, confirming that time display adapts to user configuration.

Step 8: Save a Time-Focused Dashboard Panel

Created and saved a panel:

index=_internal earliest=-24h@h latest=@h | timechart span=1h count by sourcetype

- Saved to Dashboard: **Hoodie Time Overview**
- Panel Name: **Internal Events by Sourcetype (Hourly)**
- Set **Refresh Interval:** 60 seconds



_time	mongod	pura_remote_scan_scripted_input-3	pura_utils-3	splunk_python	splunk_web_service	splunkd	splunkd_access	sup-pkg-identity-5	sup-pkg-identity-stdout-2	supervisor-2	OTHER
2025-10-21 13:30:00	386	6	27	486	486	14936	1227	11	61	1153	27
2025-10-21 14:30:00	0	0	0	58	0	25240	2202	11	103	1398	3
2025-10-21 15:30:00	0	4	21	59	0	28564	2471	12	113	1530	5
2025-10-21 16:30:00	0	2	11	22	0	6776	666	9	32	727	3
2025-10-21 17:30:00	0	0	0	61	0	29803	2560	12	118	1546	0
2025-10-21 18:30:00	0	0	0	37	0	16427	1452	8	64	992	0
2025-10-21 19:30:00	0	0	0	5	0	315	14	1	0	28	1
2025-10-21 20:30:00	0	4	32	14	0	5317	570	2	22	334	12
2025-10-21 21:30:00	0	2	5	48	0	22186	1936	11	89	1315	1
2025-10-21 22:30:00	0	0	0	60	0	29078	2514	12	116	1538	0
2025-10-21 23:30:00	0	0	0	51	0	24406	2114	11	99	1318	0
2025-10-22 00:30:00	0	6	27	58	0	25955	2270	11	103	1436	3
2025-10-22 01:30:00	0	0	0	56	0	27092	2320	11	109	1411	0
2025-10-22 02:30:00	0	0	0	0	0	0	0	0	0	0	0
2025-10-22 03:30:00	0	0	0	0	0	0	0	0	0	0	0
2025-10-22 04:30:00	0	0	0	0	0	0	0	0	0	0	0

Deliverables

- Search using earliest and latest
- Two timechart comparisons (15m vs 1h span)
- bin + stats table output
- Delay table (_time vs _indextime)
- Dashboard panel showing internal events by sourcetype (hourly)

Reflection

- **Preferred command:** timechart for visual trend analysis; bin + stats when custom aggregation or calculations are needed.
- **Ingestion delay:** Analyzing _indextime - _time helps detect data latency or indexing bottlenecks.
- **Snap modifiers:** @h (hour) and @d (day) are the most used for aligning searches to time boundaries.

Lab Summary

This lab deepened understanding of Splunk's time management.
I practiced precise time control using SPL modifiers, visualized temporal data using timechart, aligned events via bin, formatted timestamps with strftime, and analyzed ingestion delays.
Result: Enhanced efficiency and accuracy in interpreting time-based data in Splunk environments.