**Splunk Introduction (29-08-2025)**

**Module 1 – Intro to Splunk**

**Splunk Components**

Splunk is often described as a search engine for machine data. It ingests, indexes, and makes sense of large volumes of raw, unstructured data generated by systems, applications, and devices. Examples include login attempts, system errors, security events, or user activities.

Key components in Splunk deployments:

* **Indexer**: Collects incoming data, processes it, and stores it in an organized, searchable format.
* **Search Head**: Provides the interface for running searches, generating reports, and building dashboards.
* **Forwarder**: Lightweight agents installed on source systems to send data into Splunk.

In small environments, a single Splunk instance may perform all roles. In larger setups, these roles are separated across multiple instances for scalability.

**Basic Splunk Functions**

Splunk provides core functionality to work with machine data:

* **Data Ingestion**: Forwarders, network listeners (TCP/UDP), HTTP Event Collector (HEC), APIs, and connectors bring data into Splunk.
* **Indexing**: Raw data is parsed, enriched with metadata, and stored in a searchable format.
* **Search and Analysis**: Users can query data with Splunk’s Search Processing Language (SPL).
* **Dashboards and Reports**: Searches and visualizations can be saved and shared for monitoring and analysis.

Splunk operates across on-premises servers, cloud environments, and hybrid infrastructures, making it highly adaptable.

**Module 2 – Using Splunk**

**Define Splunk Apps**

A Splunk app is a collection of dashboards, reports, knowledge objects, and configurations packaged together for a specific use case. Examples include Splunk Enterprise Security and IT Service Intelligence.

**Understand Splunk User Roles**

Splunk uses role-based access control:

* **Admin**: Full privileges to manage users, apps, and system settings.
* **Power User**: Can create and share dashboards, reports, and alerts.
* **User**: Can search data, create private knowledge objects, and view shared content.

**Search & Reporting App**

The Search & Reporting app is the default interface for running searches, saving reports, and building dashboards. It is typically the starting point for most Splunk users.

**Splunk Web Interface**

The web interface provides access to:

* Search bar for running SPL queries.
* Tools for saving searches, creating alerts, and visualizations.
* Navigation to apps, reports, dashboards, and knowledge objects.

**Module 3 – Using Search**

**Run Basic Searches**

Basic searches begin with specifying an index or sourcetype:

index=security sourcetype=linux\_secure "failed password"

**Set the Time Range of a Search**

Searches can be scoped to a specific time range (last 15 minutes, last 24 hours, custom time ranges). This improves relevance and performance.

**Save Search Results**

Searches can be saved as reports for later use. Saved searches can also be scheduled.

**Identify the Contents of Search Results**

Search results are made up of **events** (raw log entries with fields). Each event has metadata such as timestamp, host, source, and sourcetype.

**Work with Events**

Events can be filtered, highlighted, or refined by adding/removing search terms.

**Share Search Jobs**

By default, a search job remains active for 10 minutes. Shared searches can remain active for 7 days.

**Export Search Results**

Results can be exported in CSV, JSON, XML, or raw text formats.

**Select Search Modes**

* **Fast Mode**: Prioritizes speed, returns essential fields only.
* **Verbose Mode**: Returns all fields, suitable for troubleshooting.
* **Smart Mode**: Balances speed and completeness by adjusting automatically.

**Control a Search Job**

Users can pause, stop, or extend the time-to-live of a search job.

**Module 4 – Exploring Events**

**Refine Searches**

Refining searches helps reduce noise and improve relevance. SPL commands and filters can narrow down results.

**Understand Timestamps**

Splunk assigns timestamps to events during ingestion. Accurate timestamps are essential for time-based searches and visualizations.

**Use the Events Tab**

The Events tab allows users to interactively add or remove search terms based on what appears in results.

**Module 5 – Search Processing Language (SPL)**

**Use Wildcards**

Wildcards make searches flexible:

* error\* → matches error, errors, error\_code
* \*fail\* → matches failed, failure, login\_fail

**Understand Case Sensitivity**

By default, searches are case-insensitive. Specific field values may be case-sensitive depending on configuration.

**Use Booleans**

Boolean operators refine searches:

* **AND**: Finds events containing all terms.
* **OR**: Finds events containing at least one term.
* **NOT**: Excludes terms.

Parentheses () group conditions, quotation marks "" search exact phrases.

**Use Special Characters**

Quotation marks, parentheses, and operators help structure more complex searches.

**Module 6 – What are Commands?**

**Anatomy of Splunk’s Search Language**

SPL searches are composed of:

* **Search terms**: Keywords or field-value pairs.
* **Commands**: Instructions to process results (e.g., stats, timechart).
* **Functions**: Applied inside commands (e.g., count(), avg()).
* **Arguments**: Options provided to functions/commands.
* **Clauses**: Define filters or groupings.

**Best Practices for Writing Searches**

* Start simple and refine incrementally.
* Use index and sourcetype early for efficiency.
* Use transforming commands (stats, timechart) for summarization.
* Avoid unnecessary wildcards for better performance.

**Module 7 – What are Knowledge Objects?**

**Categories of Knowledge Objects**

Knowledge objects enhance the usability of Splunk data.

1. **Data Interpretation** – Field extractions, event types.
2. **Data Classification** – Tags, event categorization.
3. **Data Enrichment** – Lookups that add external context.
4. **Data Normalization** – Data models that standardize fields.
5. **Data Models** – Structures enabling advanced features like Pivot.

**Types of Knowledge Objects**

* **Saved Searches/Reports**
* **Dashboards**
* **Tags**
* **Lookup Tables**
* **Event Types**
* **Data Models**

**Module 8 – Creating Reports and Dashboards**

**Save a Search as a Report**

Any SPL search can be saved as a report. Reports may run on-demand or be scheduled.

**Edit Reports**

Reports can be modified to update queries, scheduling, or sharing permissions.

**Use Transforming Commands**

Commands such as stats, chart, and timechart turn raw events into meaningful summaries and visualizations.

**Create a Dashboard**

Dashboards are collections of visualizations and reports.

* **Panels** represent individual visualizations or saved searches.
* **Classic Dashboards** are simple to configure.
* **Dashboard Studio** offers modern, customizable designs.

**Add a Report to a Dashboard**

Saved reports can be added to dashboards as panels. Drilldowns can link panels to deeper searches.

**Edit a Dashboard**

Dashboards can be customized by rearranging panels, updating queries, or modifying visualizations.

**Example Use Case**

As a security analyst monitoring login failures:

1. **Ingest Logs**: Linux authentication logs are ingested into Splunk.
2. **Run Search**:
3. index=auth "failed password"
4. **Save Report**: Store this search as *Failed Logins Report*.
5. **Create Dashboard**: Add panels showing:
   * Failed logins over 24 hours.
   * Top 5 usernames with failed logins.

**Share Dashboard**: Provide visibility to the team for real-time monitoring.

**Splunk Fields (01-09-2025)**

**1. Using Fields**

In Splunk, data is stored as events, and each event contains information that is organized into **fields**. Fields are essentially labels or categories that describe parts of an event. For example, a log line may contain a status field, a host field, or an action field.

* **Field Names vs. Values**
  + Field names are **case sensitive**.
  + Field values are **not case sensitive**.
* **Field Operators**  
  You can use operators in searches to filter results:
  + = and != work with both numbers and strings.
  + >, <, >=, and <= work only with numbers.

Example:

status=404

price>=100

These queries return only events that match the conditions.

* **Boolean Logic with Fields**  
  You can combine conditions with Boolean operators:
  + AND (implicit if you just use spaces)
  + OR (explicitly written)
  + NOT (explicitly written to exclude results)

Example:

index=security sourcetype=linux\_secure action=failure NOT (host="mail\*" OR host=www1)

* + This searches the security index, looks for events of type linux\_secure where the action is failure, and excludes hosts that begin with “mail” or are exactly “www1.”
* **Difference Between != and NOT**
  + != checks only events that have the field and compares the value.
  + NOT excludes all events with a certain value, even if some events do not have that field.

Example with three events:

status=200 user=john

status=404 user=jane

user=mary

* + status!=200 returns only the second event (because the third event does not have a status field to compare).
  + NOT status=200 returns the second and third events (it excludes only those with status=200).
* **OR Example**
* index=web (status=500 OR status=503 OR status=505)

This looks in the web index and finds events where the status is 500, 503, or 505.

* **IN Example**
* index=web status IN ("500", "503", "505")

This is a shorter way of writing multiple OR conditions.

* **Best Practice: Place fields Before stats**  
  When you use the stats command to calculate counts, sums, or averages, it is recommended to use the fields command before it. The fields command ensures only the required fields are processed, improving search performance.

Example:

index=web status IN ("500", "503", "505")

| fields status

| stats count by status

* **Including and Excluding Fields**
  + fields field1 field2 keeps only the listed fields.
  + fields - field1 removes a field from results.
  + You may also see fields +fieldname to emphasize inclusion.
* **Renaming Fields**  
  You can rename fields to make reports clearer:
* index=web status IN ("500", "503", "505")
* | stats count by status
* | rename status as "HTTP Status", count as "Number of Events"

This changes the column names in the results to make them easier to read.

**2. Using the Fields Sidebar**

When you run a search in Splunk, the Fields sidebar on the left helps you explore data more easily.

* **Selected Fields**  
  These are fields that are always shown in results (for example, \_time, \_raw, host, source, sourcetype).
* **Interesting Fields**  
  These are fields that occur in at least 20% of the events returned. They appear in the sidebar so you can quickly add them to your search.

From the sidebar, you can:

* Click on a field name to view its values.
* Click on a value to add it directly into your search query.

This makes it easier to refine searches without typing everything manually.

**3. Fields in Search Queries**

Fields are used directly in search queries to filter and refine results. Examples include:

* Exact match:
* status=200
* Inequality:
* duration>5
* Multiple conditions:
* action=login user=admin
* Boolean filters:
* NOT (host="web1" OR host="web2")
* IN operator for multiple values:
* status IN ("200", "404", "500")

The pipe symbol | is used to pass search results from one command to another, creating a pipeline of commands.

**4. Fields in Search Results**

When Splunk ingests data, it automatically extracts a set of default fields for every event. These include:

* host: The system or machine from which the event originated.
* source: The file, directory, or input source for the event.
* sourcetype: The format or type of the data (for example, syslog, apache logs, JSON).
* \_time: The timestamp of the event.
* \_raw: The full raw text of the log event.

These fields are automatically available for filtering and analysis as soon as the data is ingested.

**5. Enriching Data with Knowledge Objects**

Beyond the default fields, Splunk provides tools to enrich your data with additional knowledge objects:

* **Field Extractions**: Create custom fields by extracting values from raw data.
* **Field Aliases**: Assign alternative names to existing fields.
* **Calculated Fields (via eval)**: Use the eval command to create new fields based on existing ones.

Example with eval:

index=network sourcetype=cisco\_wsa\_squid

| stats sum(sc\_bytes) as Bytes by usage

| eval bandwidth = Bytes/1024/1024

Here, a new field called bandwidth is created by converting bytes into megabytes.

* **Lookups**: Enhance data by adding external information (for example, mapping IP addresses to geographic locations).
* **Tags and Event Types**: Organize and categorize events to make searches easier.

**Splunk - Visualization**

**1. Formatting Commands**

Formatting commands shape the data returned by Splunk searches, making it easier to view and analyze.

**fields**

* Used to include or exclude fields.
* Syntax:
  + fields field1 field2 → keeps only the specified fields.
  + fields - field1 field2 → removes both field1 and field2.
  + fields -field1 field2 → removes only field1.
* Improves search efficiency by reducing the amount of data returned.

**table**

* A transforming command used to create tabular views of results.
* Displays only the specified fields.
* Example:

index=web sourcetype=access\_combined product\_name=\*

| table JSESSIONID product\_name price

* Drops all fields except those explicitly listed.

**dedup**

* Removes duplicate events based on one or more fields.
* Splunk keeps the first unique combination and drops the rest.
* Example:

... | dedup JSESSIONID price

**addtotals**

* Adds row or column totals for numeric fields.
* Options:
  + row=true → adds a totals row.
  + col=true → adds a totals column.
  + label → customizes the totals row label.
  + labelfield → specifies the field where the label should appear.
  + fieldname → renames the totals column.
  + row=false → prevents adding a totals row.
* Example:

| addtotals col=true label="Total Sales" labelfield="product\_name" fieldname="Total By Product"

**fieldformat**

* Alters the display of field values without changing the raw data.
* Example:

... | fieldformat Total = "$" + tostring(Total, "commas")

* + Displays totals with a dollar sign and comma formatting.

**2. Visualizing Data**

Transforming commands prepare data for visualization.

**top**

* Finds the most common values in a field.
* Returns count and percentage by default.
* Default limit: 10.
* Options:
  + limit → change number of results.
  + countfield → rename count column.
  + percentfield → rename percent column.
  + showcount → show/hide count column.
  + showperc → show/hide percent column.
  + showother → include/exclude "Other".
* Example:

| top product\_name by Vendor limit=3 countfield="Number of Sales" showperc=false

**rare**

* Opposite of top.
* Finds the least common values in a field.

**stats**

* General aggregation function.
* Common functions:
  + sum
  + count
  + dc() → distinct count
  + avg
  + min
  + max
  + values
  + list
* Example:

| stats sum(price) as Total by product\_name

**chart**

* Similar to stats but optimized for two-dimensional aggregation.
* Works well with visualization tools.
* Example:

| chart sum(price) over product\_name by VendorCountry

**timechart**

* A time-aware version of chart.
* Automatically buckets events by time.
* The X-axis is always time.
* Options:
  + span → defines time intervals (e.g., 5m, 1h, 1d).
* Example:

| timechart span=1h count

**trendline**

* Adds moving averages to visualize trends.
* Types:
  + sma → Simple Moving Average (equal weights).
  + ema → Exponential Moving Average (recent values weighted more).
  + wma → Weighted Moving Average (manual linear weights).

**3. Generating Maps**

Splunk supports geographic visualization using specialized commands.

**Internal Fields**

* \_time → timestamp of the event.
* \_raw → raw event data.

**iplocation**

* Enriches IP addresses with geographic info.
* Fields added: Country, City, Region, Latitude, Longitude.

**geostats**

* Similar to stats, but geography-aware.
* Aggregates results by latitude and longitude.
* Key arguments:
  + latfield → latitude field.
  + longfield → longitude field.
  + globallimit → limit column count.

**geom**

* Draws geographic shapes (polygons) for choropleth maps.
* Works with geo lookup datasets (e.g., geo\_countries).

**Visualization Types**

* **Marker Map** → plots points.
* **Cluster Map** → groups nearby points.
* **Choropleth Map** → shades regions based on values.

**4. Single Value Visualizations**

* Display one key metric at a time.
* Often paired with formatting commands like fieldformat to improve readability.
* Example:

| stats sum(price) as TotalSales

| fieldformat TotalSales = "$" + tostring(TotalSales, "commas")

**5. Visual Formatting**

Visual formatting improves readability of charts and dashboards.

* **Column order** → determined by the order of fields in the table command.
* **Default limits** → chart command shows 10 columns by default (configurable with limit).
* **Chart overlay** → compare multiple data series in the same chart.
* **Single value visualization formatting** → add currency symbols, commas, or percentages.
* **Fieldformat vs eval**:
  + fieldformat → changes display only.
  + eval → changes actual values.

**Splunk - Working with Time (02-09-2025)**

**1. Working with Time**

**The \_time Field**

* Every event in Splunk has the special field \_time.
* It represents the event timestamp Splunk uses for searching, charting, and time-based calculations.
* If Splunk cannot parse a timestamp in the raw data, \_time defaults to the time the event was indexed.

**Raw Timestamp vs \_time vs \_indextime**

* Raw timestamp → String inside the log (2025-09-02 10:15:30).
* \_time → Splunk’s interpreted timestamp (epoch UTC).
* \_indextime → The exact time when Splunk wrote the event into the index (can differ if delayed ingestion/backfill).

**If logs arrive late, \_time may be hours/days earlier than \_indextime.**

**🔹 Default Time Fields (auto-extracted from \_time)**

**Splunk creates extra fields for easy time slicing:**

* **date\_year →** e.g. 2025
* **date\_month →** 9
* **date\_mday →** day of month (e.g. 2)
* **date\_hour →** hour (00–23)
* **date\_minute →** 00–59
* **date\_second →** 00–59
* **date\_wday →** weekday (0 = Sunday … 6 = Saturday)
* **date\_time →** human-readable timestamp in your local session time zone

**date\_time = \_time formatted for your Splunk Web time zone setting.**

**2. Searching with Time**

**🔹 Time Picker**

* Top-right in Splunk search.
* Provides relative ranges like *Last 15m, Last 24h, Yesterday, Today, Week-to-date*.

**🔹 Overriding Time Picker with Modifiers**

You can add earliest and latest directly in the search.

**Common Time Units**

* **s →** seconds
* **m →** minutes
* **h →** hours
* **d →** days
* **w →** weeks
* **mon →** months
* **q →** quarters
* **y →** years

**Examples**

* earliest=-15m → last 15 minutes.
* earliest=-24h latest=now → last 24 hours.
* earliest=-1d@d latest=@d → yesterday midnight to midnight.
* earliest=-30d@d latest=@d → last 30 calendar days, from midnight to midnight.

**🔹 The @ Symbol =** Snap to Boundary

* **@d →** start of day
* **@h →** start of hour
* **@m →** start of minute
* **@w0 →** start of week (Sunday)
* **@w1 →** start of week (Monday)

**Examples**

* earliest=@d+3h → today 03:00 AM to now.
* earliest=-7d@d → 7 days ago at midnight → now.

**Gotcha: @ rounds down to the boundary, never forward.**

**3. Formatting Time**

strftime() → Epoch → String

Converts \_time (or any epoch) into a human-readable string.

**Common codes:**

* %Y → Year
* %m → Month number
* %b → Short month (Sep)
* %d → Day
* %H → Hour (24h)
* %I → Hour (12h)
* %M → Minute
* %S → Second
* %p → AM/PM

**Example:**

| eval formatted=strftime(\_time, "%b %d, %I:%M %p")

If \_time=2025-09-01 14:30:00 → Sep 01, 02:30 PM.

**strptime() → String → Epoch**

Useful when you need to parse custom timestamps.

| eval epoch=strptime("2025-09-01 14:30", "%Y-%m-%d %H:%M")

**4. Time Commands**

bin

Groups events into equal time buckets.

* | bin \_time span=1m → 1-minute buckets.
* | bin \_time span=1h → 1-hour buckets.

**timechart**

Builds time-series charts.

| timechart count span=1h

**→ returns event counts in 1-hour chunks.**

**timewrap**

Compares different time ranges on one chart.

... | timechart count span=1d | timewrap 1w

**→ daily counts, aligned across weeks.**

**5. Working with Time Zones**

* \_time is always stored in UTC.
* Displayed as local time zone set in user preferences.
* strftime() and date\_time both reflect the session’s time zone.
* \_indextime never changes—it’s absolute.