SPLUNK MASTERY:
MASTER THE SPLUNK GUIDELINE

Created By : Farhath Nathvi
LinkedIn

www.linkedin.com/in/farhathnathvi

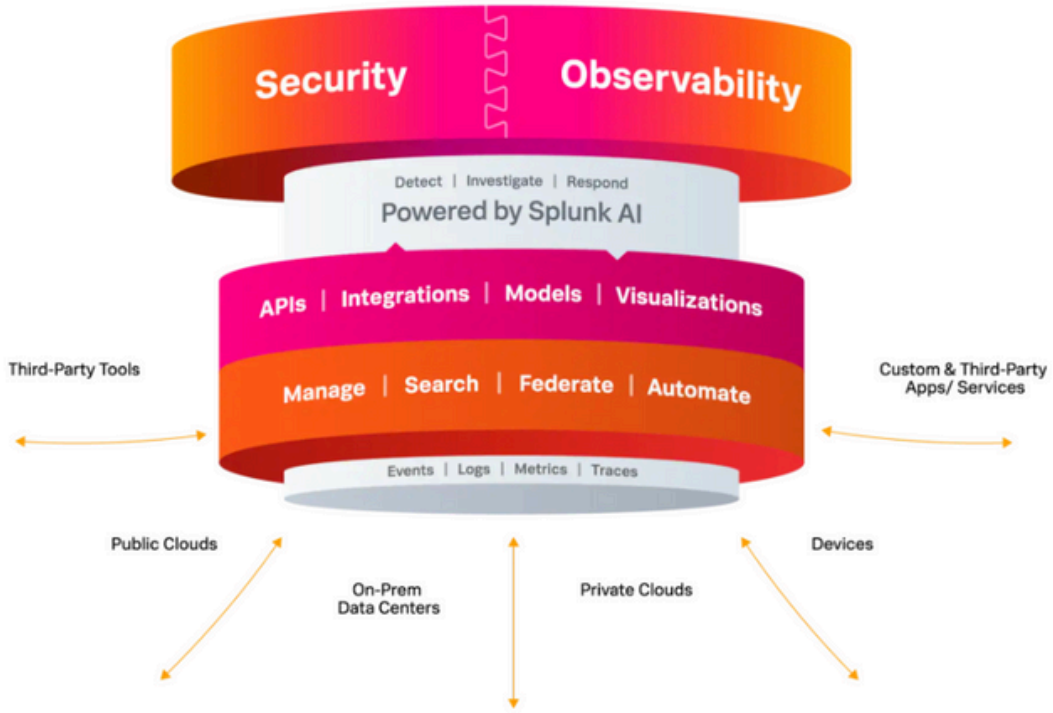
Table of Contents

What Is Splunk?

In today's data-driven cyber landscape, organizations across the globe are faced with an ever-increasing volume of data from various assets and network infrastructure. To harness the power of this data and enable cyber resilience, they need tools and technologies that can help them collect, analyze, and visualize the logs and events effectively to detect and prevent cyber security threats.

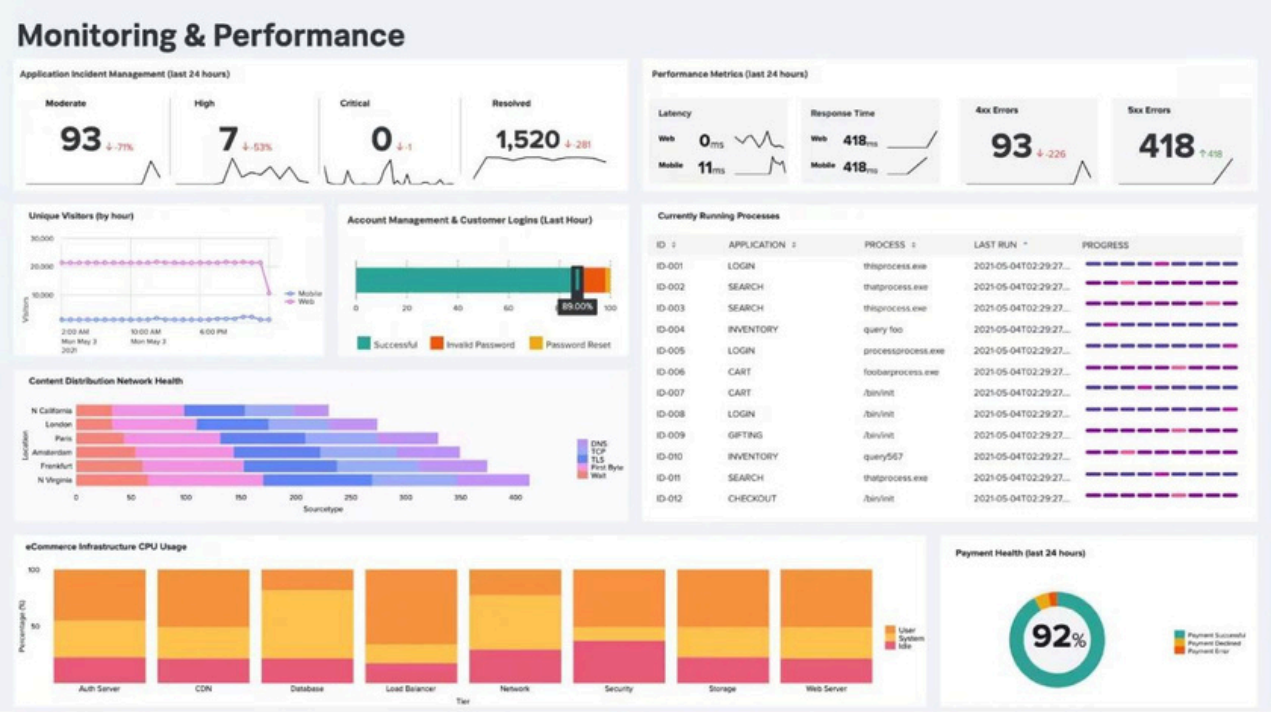
Splunk is a powerful SIEM (Security Information and Event Management) tool that is widely used to solve this purpose. It offers a comprehensive platform for collecting, analyzing, and visualizing machine-generated data to gain valuable insights and detect potential security threats.

Though Splunk is usually considered a SIEM tool, it has been recently rebranded as a Unified Security and Observability Platform, and currently, Splunk is offered as Splunk Cloud, Splunk Enterprise, and Splunk Observability Cloud platforms.



So, what is Splunk used for? Splunk is designed to ingest and index large volumes of data from various sources, including logs, sensors, devices, applications, and systems. It provides real-time monitoring, analysis, security, and observability capabilities, allowing organizations to identify and respond to security incidents proactively.

One of the key features of Splunk is its ability to correlate and aggregate data from different sources like servers, firewalls, load balancers, network devices, etc., enabling security analysts to investigate and identify patterns, anomalies, and potential threats. Its advanced search and query functionalities allow users to perform complex searches and create custom reports and dashboards.



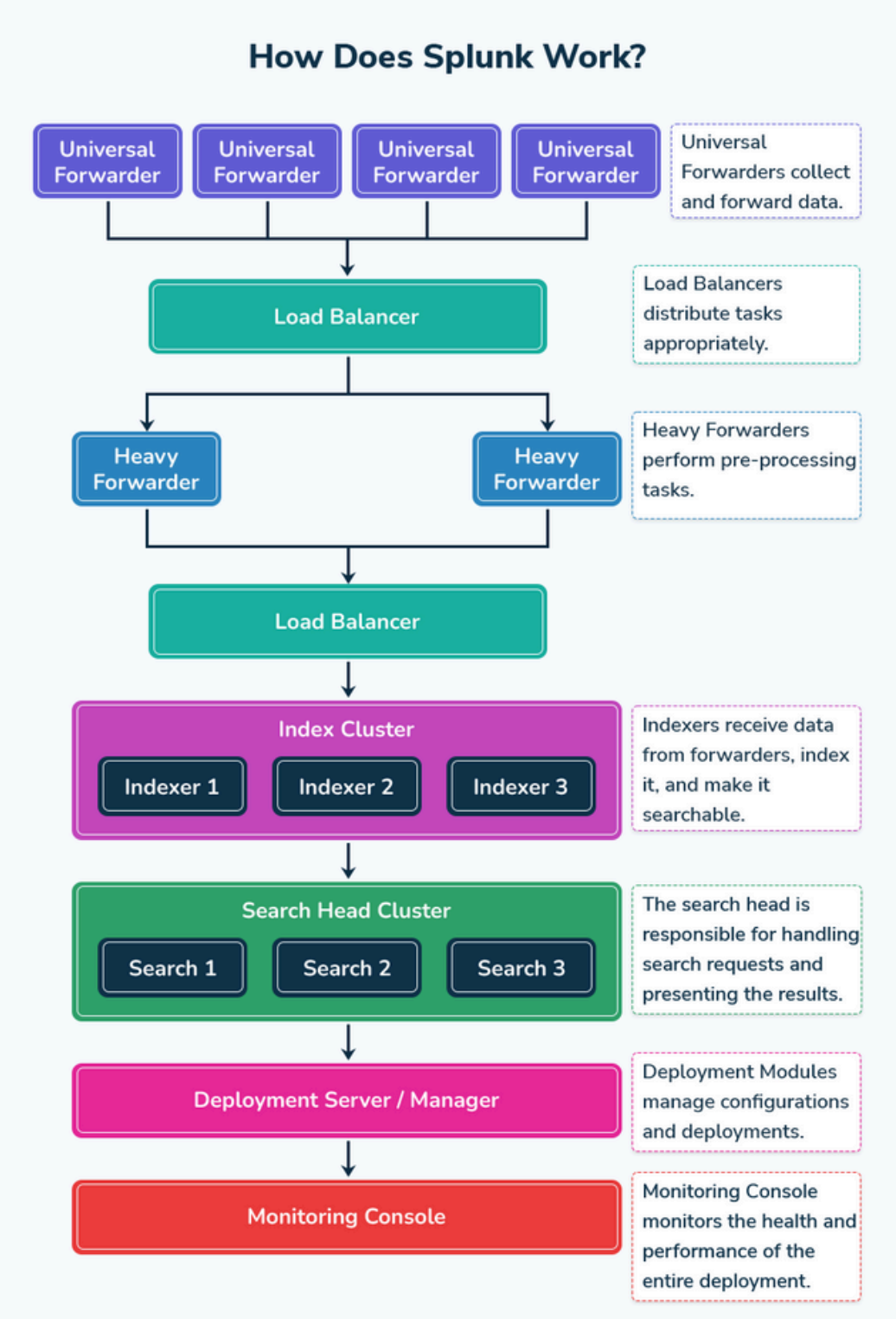
Splunk also offers a wide range of security-specific applications and add-ons that provide additional functionality and help automate various security tasks. These include threat intelligence, incident response, compliance monitoring, observability, and user behavior analytics, among others.

By analyzing and visualizing data in real-time, Splunk helps organizations improve their security posture by identifying and mitigating vulnerabilities, detecting and responding to security incidents, and ensuring compliance with industry regulations and best practices.

In addition to its security applications, Splunk is also widely used for other purposes, such as IT operations monitoring, application performance monitoring, business analytics, and log management. Its versatility and scalability make it a popular choice for organizations of all sizes and across various industries.

How Does Splunk Work?

Splunk's architecture consists of various components that work together to enable data ingestion, indexing, searching, and visualization. Here is a typical Splunk architecture diagram and the corresponding key components of Splunk architecture:



1. Forwarders:

Universal Forwarder: A lightweight component installed on data sources to collect and forward data to the Splunk indexer. It has minimal resource requirements and is suitable for high-volume data sources.

Heavy Forwarder: A more feature-rich version of the universal forwarder that allows data preprocessing before indexing. It is suitable for environments requiring additional data manipulation.

2. Load Balancer (LB):

A load balancer in Splunk helps distribute incoming network traffic evenly across multiple Splunk instances or servers. It acts as a mediator between clients and the backend Splunk instances, ensuring that the workload is evenly distributed and efficiently managed.

3. HTTP Event Collector (HEC):

Allows the submission of events to Splunk over HTTP. It allows external sources to send data to Splunk for indexing and analysis.

4. Indexer:

Indexer Cluster: Multiple indexers can be configured in a cluster to ensure high availability and fault tolerance. Indexers receive data from forwarders, index it, and make it searchable.

5. Search Head:

Search Head Cluster: The search head is responsible for handling search requests and presenting the results. A cluster of search heads can be configured for load balancing and redundancy.

Search Head Pooling: Distributes search requests across a pool of search heads, optimizing performance and providing fault tolerance.

6. Deployment Modules:

Deployment Server: Manages configurations for forwarders, ensuring consistency across the environment. It simplifies the process of deploying and managing Splunk components.

Deployment Manager: Facilitates the management of configurations across multiple Splunk instances. It ensures consistency and simplifies the deployment process.

7. License Master:

Manages licenses for all Splunk components in the environment. It ensures that the usage complies with licensing agreements.

8. Monitoring Console:

Provides a centralized interface for monitoring the health and performance of the Splunk deployment. It helps administrators track the status of components and troubleshoot issues.

9. Data Inputs:

Various mechanisms for ingesting data into Splunk, including file monitoring, scripted inputs, scripted modular inputs, and various protocol-based inputs.

Core Features of Splunk

Splunk is a powerful SIEM software platform that offers a wide range of features that help businesses gain valuable insights from their data and ensure cyber resilience.

1. Enormous Amounts of Data Collection and Ingestion

Splunk excels in collecting and ingesting diverse data sources crucial for cyber security. Its versatility, from logs to events and metrics, ensures comprehensive coverage, enabling real-time threat detection.

2. Lightning Fast Real-Time Indexing

The heartbeat of Splunk's SIEM capabilities lies in real-time indexing. Immediate visibility into security events allows for swift responses, minimizing the impact of cyber incidents.

3. Powerful Analytical Search and Investigation

In the cyber security realm, quick and precise investigations are essential. Splunk's search and investigation features, powered by the Splunk Query Language (SPL), enable security professionals to identify and analyze threats quickly and accurately.

4. Appealing Data Visualizations and Dashboards

Splunk's intuitive data visualization tools play a pivotal role in cyber security. Interactive dashboards facilitate monitoring security metrics, threat landscapes, and incident trends at a glance.

5. Real-Time Alerts and Notifications

Proactivity is key in cyber security. Splunk enables the creation of alerts and notifications, ensuring that security teams are promptly informed of potential threats or anomalous activities.

Primary Use Cases for Splunk

Splunk's application spans various critical areas. As we embark on this exploration, we'll discover how Splunk's versatility addresses critical operational challenges across various domains, making it a cornerstone for organizations seeking holistic IT, security, and business intelligence solutions.

1. IT Operations Management

In the cyber security domain, IT operations management is synonymous with threat detection, incident response, and system integrity. Splunk's role extends beyond IT operations, ensuring a holistic security posture.

2. Security and Compliance (SIEM)

As a SIEM tool, Splunk shines in real-time security monitoring, threat detection, and compliance management. It aids organizations in staying ahead of cyber threats and adhering to regulatory requirements.

3. Application Performance Monitoring (APM)

Applications are prime targets for cyber attacks. Splunk's APM capabilities enhance cyber security by monitoring application performance, detecting anomalies, and mitigating potential security risks.

4. Business Analytics and Intelligence

Splunk's application in cyber security extends to business intelligence. By deriving insights from security data, organizations can make informed decisions, ensuring a proactive cyber security strategy.

Advantages of Using Splunk

Splunk stands as the paramount choice in the realm of cyber security and data analysis, offering a comprehensive solution that outshines its competitors. Through a meticulous exploration of its core features, primary use cases, and advantages, it becomes evident that Splunk's robust capabilities empower organizations to navigate the intricate landscape of cyber security and derive actionable insights from their data. Splunk's adoption in cyber security is underpinned by several advantages:

1. Scalability and Flexibility

Cyber security landscapes are dynamic and diverse. Splunk's scalability ensures it can adapt to organizations' evolving data and security needs, from startups to large enterprises.

2. Speed and Efficiency in Threat Detection

Real-time indexing and search capabilities position Splunk as a frontline defender. Its speed and efficiency in processing data enable rapid threat detection and response, minimizing dwell time. The Splunk Query Language (SPL) provides a powerful and flexible way to query and analyze data, enabling more sophisticated searches compared to some other platforms.

3. Machine Learning Capabilities

Splunk incorporates machine learning for advanced analytics and anomaly detection, enhancing its capabilities for proactive threat detection.

4. Intuitive User Interface and Visualization Capabilities

In the high-stakes environment of cyber security, simplicity is powerful. Splunk's user-friendly interface and robust visualization capabilities empower security professionals with actionable insights.

5. Seamless Cloud Integration

Splunk seamlessly integrates with cloud environments and offers native cloud support, providing flexibility and scalability for organizations adopting cloud technologies.

Comparing Splunk to Other Data Analysis Tools

Splunk's cyber security and data analysis prowess is further highlighted through a comprehensive comparison with other leading solutions. Here, we compare Splunk with other leading tools, providing detailed insights into their features, strengths, and unique offerings:

Splunk vs. ELK (Elasticsearch, Logstash, Kibana)

Comparison Highlights

- Cost: ELK is open-source, making it cost-effective. Splunk offers free versions, but enterprise solutions have licensing fees.
- Ease of Use: Splunk has a more user-friendly interface and search language (SPL). ELK, being open-source, may require more technical expertise.
- Scalability: Both are scalable, but Splunk offers commercial support for demanding cyber security needs.
- Community and Ecosystem: ELK gets most of its support from a large open-source community. Splunk has its community and Splunkbase marketplace.

Splunk vs. Datadog

Comparison Highlights

- Focus: Datadog emphasizes infrastructure and application monitoring. Splunk's versatility extends to broader cyber security use cases.
- Ease of Use: Datadog offers a user-friendly interface. Splunk may require more configuration for specific cyber security use cases.
- Pricing: Datadog follows a subscription-based model. Splunk's pricing varies based on data volume and cyber security deployment needs.

Splunk vs. New Relic

Comparison Highlights

- Focus: New Relic specializes in APM. Splunk's versatility makes it suitable for a broader spectrum of cyber security and data analysis.
- Pricing: New Relic follows a subscription model. Splunk's pricing varies based on cyber security needs and data volumes.
- Versatility: Splunk's adaptability makes it a better choice for organizations with diverse cyber security requirements.

Splunk vs. IBM QRadar

Comparison Highlights

- Focus: Splunk offers a broader focus on data analysis and cyber security. IBM QRadar specializes in security information and event management (SIEM).
- Ease of Use: Splunk is known for its intuitive interface. IBM QRadar may have a steeper learning curve.
- Scalability: Both are scalable, but Splunk's commercial support enhances scalability for demanding cyber security environments.
- Community and Ecosystem: Splunk's active community and Splunkbase Marketplace provide a robust ecosystem. IBM QRadar also has a community but may have fewer community-driven resources.

Splunk vs. ArcSight

Comparison Highlights

- Focus: Splunk offers a broader focus on data analysis and cyber security. ArcSight specializes in security information and event management (SIEM).
- Ease of Use: Splunk is known for its intuitive interface. ArcSight may have a steeper learning curve.
- Scalability: Both are scalable, but Splunk's commercial support enhances scalability for demanding cybersecurity environments.
- Community and Ecosystem: Splunk's active community and Splunkbase Marketplace provide a robust ecosystem. ArcSight also has a community but may have fewer community-driven resources.

Search Language in Splunk

Splunk uses what's called Search Processing Language (SPL), which consists of keywords, quoted phrases, Boolean expressions, wildcards (*), parameter/value pairs, and comparison expressions. Unless you're joining two explicit Boolean expressions, omit the AND operator because Splunk assumes the space between any two search terms to be AND.

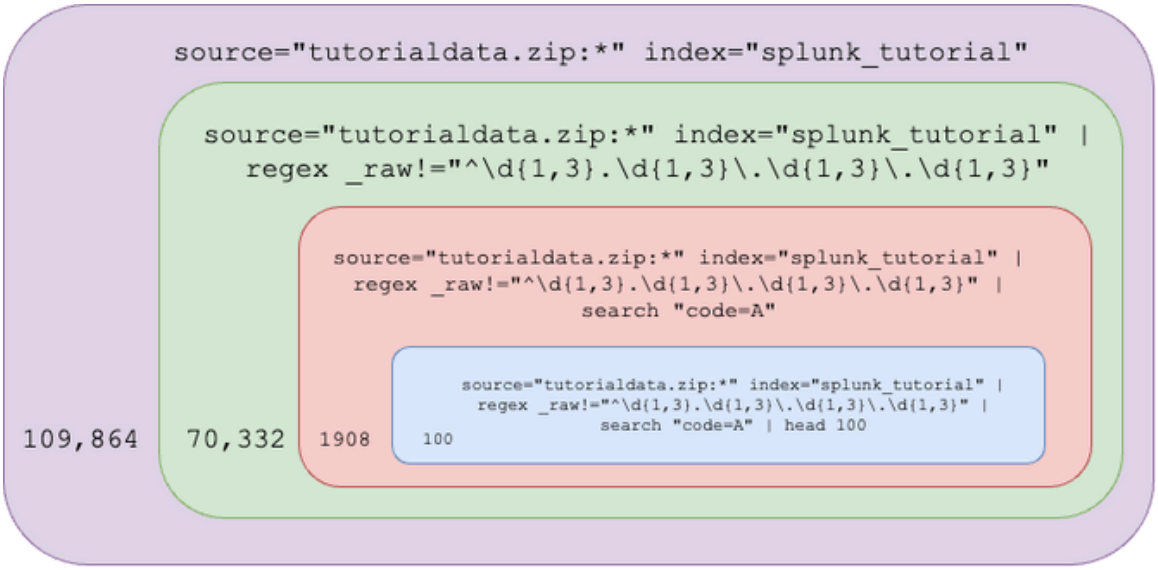
Basic Search offers a shorthand for simple keyword searches in a body of indexed data myIndex without further processing:

```
index=myIndex keyword
```

An event is an entry of data representing a set of values associated with a timestamp. It can be a text document, configuration file, or entire stack trace. Here is an example of an event in a web activity log:

```
[10/Aug/2022:18:23:46] userID=176 country=US paymentID=30495
```

Search commands help filter unwanted events, extract additional information, calculate values, transform data, and statistically analyze the indexed data. It is a process of narrowing the data down to your focus. Note the decreasing number of results below:



Common Search Commands

Command	Description
`chart, timechart`	Returns results in a tabular output for (time-series) charting
`dedup X`	Removes duplicate results on a field X
`eval`	Calculates an expression (see Calculations)
`fields`	Removes fields from search results
`head/tail N`	Returns the first/last N results, where N is a positive integer
`lookup`	Adds field values from an external source
`rename`	Renames a field. Use wildcards (*) to specify multiple fields.
`rex`	Extract fields according to specified regular expression(s)
`search`	Filters results to those that match the search expression
`sort X`	Sorts the search results by the specified fields X
`stats`	Provides statistics, grouped optionally by fields
`mstats`	Similar to stats but used on metrics instead of events
`table`	Displays data fields in table format.
`top/rare`	Displays the most/least common values of a field
`transaction`	Groups search results into transactions
`where`	Filters search results using eval expressions. For comparing two different fields.

SPL Syntax

Begin by specifying the data using the parameter `index`, he equal sign =, and the data index of your choice:

Complex queries involve the pipe character |, which feeds the output of the previous query into the next.

Basic Search

This is the shorthand query to find the word hacker in an index called cybersecurity:

```
index=cybersecurity hacker
```

SPL search terms	**Description**
Full Text Search	
`Cybersecurity`	Find the word “Cybersecurity” irrespective of capitalization
`White Black Hat`	Find those three words in any order irrespective of capitalization
`"White Black+Hat"`	Find the exact phrase with the given special characters, irrespective of capitalization
Filter by fields	
`source="/var/log/myapp/access .log" status=404`	All lines where the field status has value 404 from the file /var/log/myapp/access.log
`source="bigdata.rar:*" index="data_tutorial" Code=RED`	All entries where the field Code has value RED in the archive bigdata.rar indexed as data_tutorial
`index="customer_feedback" _raw="*excellent*"`	All entries whose text contains the keyword “excellent” in the indexed data set customer_feedback
`Filter by host`	

SPL search terms	Description
`source="/var/log/myapp/access.log" status=404`	All lines where the field status has value 404 from the file /var/log/myapp/access.log
`source="bigdata.rar:*" index="data_tutorial" Code=RED`	All entries where the field Code has value RED in the archive bigdata.rar indexed as data_tutorial
`index="customer_feedback" _raw="*excellent*"`	All entries whose text contains the keyword “excellent” in the indexed data set customer_feedback
`Filter by host`	
`host="myblog" source="/var/log/syslog" Fatal`	Show all Fatal entries from /var/log/syslog belonging to the blog host myblog
`Selecting an index`	
`index="myIndex" password`	Access the index called myIndex and text matching password.
`source="test_data.zip:*"`	Access the data archive called test_data.zip and parse all its entries (*).
`sourcetype="datasource01"`	(Optional) Search data sources whose type is datasource01.

This syntax also applies to the arguments following the search keyword. Here is an example of a longer SPL search string:

```
index=* OR index=_* sourcetype=generic_logs | search Cybersecurity | head 10000
```

In this example, `index=* OR index=_* sourcetype=generic_logs` is the data body on which Splunk performs `search Cybersecurity`, and then `head 10000` causes Splunk to show only the first (up to) 10,000 entries.

Basic Filtering

You can filter your data using regular expressions and the Splunk keywords rex and regex. An example of finding deprecation warnings in the logs of an app would be:

```
index="app_logs" | regex error="Deprecation Warning"
```

SPL filters	Description	Examples
`search`	Find keywords and/or fields with given values	• `index=names | search Chris` • `index=emails | searchemailAddr="*mysite.com"`
`regex`	Find expressions matching a given regular expression	Find logs not containing IPv4 addresses:`index=syslogs | regex!="^\d{1,3}.\d{1,3}\.\d{1,3}\.\d{1,3}"`
`rex`	Extract fields according to specified regular expression(s) into a new field for further processing	Extract email addresses:`source="email_dump.txt" | rexfield=_raw "From: <(?<from>.*)> To: <(?<to>.*)>"`

The biggest difference between search and regex is that you can only exclude query strings with regex. These two are equivalent:

```
source="access.log" Fatal
source="access.log" | regex _raw=".*Fatal.*"
```

But you can only use regex to find events that do not include your desired search term:

```
source="access.log" | regex _raw!=".*Fatal.*"
```

Calculations

Combine the following with eval to do computations on your data, such as finding the mean, longest and shortest comments in the following example:

```
index=comments | eval cmt_len=len(comment) | stats

avg(cmt_len), max(cmt_len), min(cmt_len) by index
```

Function	Return value / Action	Usage:eval foo=...
abs(X)	absolute value of X	`abs(number)`
`case(X,"Y",…)`	Takes pairs of arguments X and Y, where X arguments are Boolean expressions. When evaluated to TRUE, the arguments return the corresponding Y argument	`case(id == 0, "Amy", id == 1,"Brad", id == 2, "Chris")`
`ceil(X)`	Ceiling of a number X	`ceil(1.9)`
`cidrmatch("X",Y)`	Identifies IP addresses that belong to a particular subnet	`cidrmatch("123.132.32.0/25",ip)`
`coalesce(X,…)`	The first value that is not NULL	`coalesce(null(), "Returned val", null())`
`cos(X)`	Cosine of X	`n=cos(60) #1/2`

Function	Return value / Action	Usage:eval foo=...
`exact(X)`	Evaluates an expression X using double precision floating point arithmetic	`exact(3.14*num)`
`exp(X)`	e (natural number) to the power X (eX)	`exp(3)`
`if(X,Y,Z)`	If X evaluates to TRUE, the result is the second argument Y. If X evaluates to FALSE, the result evaluates to the third argument Z	`if(error==200, "OK", "Error")`
`in(field,valuelist)`	TRUE if a value in valuelist matches a value in field. You must use the in() function embedded inside the if() function	`if(in(status, "404","500","503"),"true","false")`
`isbool(X)`	TRUE if X is Boolean	`isbool(field)`
`isint(X)`	TRUE if X is an integer	`isint(field)`
`isnull(X)`	TRUE if X is NULL	`isnull(field)`
`isstr(X)`	TRUE if X is a string	`isstr(field)`
`len(X)`	Character length of string X	`len(field)`
`like(X,"Y")`	TRUE if and only if X is like the SQLite pattern in Y	`like(field, "addr%")`
`log(X,Y)`	Logarithm of the first argument X where the second argument Y is the base. Y defaults to 10 (base-10 logarithm)	`log(number,2)`
`lower(X)`	Lowercase of string X	`lower(username)`
`ltrim(X,Y)`	X with the characters in Y trimmed from the left side. Y defaults to spaces and tabs	`ltrim(" ZZZabcZZ ", " Z")`
`match(X,Y)`	TRUE if X matches the regular expression pattern Y	`match(field, "^\d{1,3}\.\d$")`
`max(X,…)`	The maximum value in a series of data X,...	`max(delay, mydelay)`
`md5(X)`	MD5 hash of a string value X	`md5(field)`
`min(X,…)`	The minimum value in a series of data X,...	`min(delay, mydelay)`
`mvcount(X)`	Number of values of X	`mvcount(multifield)`

Function	Return value / Action	Usage:eval foo=...
`mvfilter(X)`	Filters a multi-valued field based on the Boolean expression X	`mvfilter(match(email, "net$"))`
`mvindex(X,Y,Z)`	Returns a subset of the multi-valued field X from start position (zero-based) Y to Z (optional)	`mvindex(multifield, 2)`
`mvjoin(X,Y)`	Joins the individual values of a multi-valued field X using string delimiter Y	`mvjoin(address, ";")`
`now()`	Current time as Unix timestamp	`now()`
`null()`	NULL value. This function takes no arguments.	`null()`
`nullif(X,Y)`	X if the two arguments, fields X and Y, are different. Otherwise returns NULL.	`nullif(fieldX, fieldY)`
`random()`	Pseudo-random number ranging from 0 to 2147483647	`random()`
`relative_time (X,Y)`	Unix timestamp value of relative time specifier Y applied to Unix timestamp X	`relative_time(now()," -1d@d")`
`replace(X,Y,Z)`	A string formed by substituting string Z for every occurrence of regex string Y in string XThe example swaps the month and day numbers of a date.	`replace(date, "^(\d{1,2})/(\d{1,2}) /", "\2/\1/")`
`round(X,Y)`	X rounded to the number of decimal places specified by Y, or to an integer for omitted Y	`round(3.5)`
`rtrim(X,Y)`	X with the characters in (optional) Y trimmed from the right side. Trim spaces and tabs for unspecified Y	`rtrim(" ZZZZabcZZ ", " Z")`
`split(X,"Y")`	X as a multi-valued field, split by delimiter Y	`split(address, ";")`
`sqrt(X)`	Square root of X	`sqrt(9) # 3`
`strftime(X,Y)`	Unix timestamp value X rendered using the format specified by Y	`strftime(time, "%H:%M")`
`strptime(X,Y)`	Value of Unix timestamp X as a string parsed from format Y	`strptime(timeStr, "%H:%M")`
`substr(X,Y,Z)`	Substring of X from start position (1-based) Y for (optional) Z characters	`substr("string", 1, 3) #str`
`time()`	Current time to the microsecond.	`time()`
`tonumber(X,Y)`	Converts input string X to a number of numerical base Y (optional, defaults to 10)	`tonumber("FF",16)`

Function	Return value / Action	Usage:eval foo=...
tostring(X,Y)	Field value of X as a string.If X is a number, it reformats it as a string. If X is a Boolean value, it reformats to "True" or "False" strings.If X is a number, the optional second argument Y is one of:"hex": convert X to hexadecimal,"commas": formats X with commas and two decimal places, or"duration": converts seconds X to readable time format HH:MM:SS.	This example returns bar=00:08:20:| makeresults | eval bar = tostring(500, "duration")
typeof(X)	String representation of the field type	This example returns "NumberBool":| makeresults | eval n=typeof(12) + typeof(1==2)
urldecode(X)	URL X, decoded.	urldecode("http%3A%2F%2Fwww.site.com%2Fview%3Fr%3Dabout")
validate(X,Y,...)	For pairs of Boolean expressions X and strings Y, returns the string Y corresponding to the first expression X which evaluates to False, and defaults to NULL if all X are True.	validate(isint(N), "Not an integer", N>0, "Not positive")

Statistical and Graphing Functions

Common statistical functions used with the chart, stats, and timechart commands. Field names can contain wildcards (*), so avg(*delay) might calculate the average of the delay and *delay fields

Function	Return valueUsage: stats foo=... / chart bar=... / timechart t=...
`avg(X)`	average of the values of field X
`count(X)`	number of occurrences of the field X. To indicate a specific field value to match, format X as eval(field="desired_value").
`dc(X)`	count of distinct values of the field X
`earliest(X)` `latest(X)`	chronologically earliest/latest seen value of X
`max(X)`	maximum value of the field X. For non-numeric values of X, compute the max using alphabetical ordering.
`median(X)`	middle-most value of the field X
`min(X)`	minimum value of the field X. For non-numeric values of X, compute the min using alphabetical ordering.
`mode(X)`	most frequent value of the field X
`percN(Y)`	N-th percentile value of the field Y. N is a non-negative integer < 100.Example: perc50(total) = 50th percentile value of the field total.
`range(X)`	difference between the max and min values of the field X
`stdev(X)`	sample standard deviation of the field X
`stdevp(X)`	population standard deviation of the field X
`sum(X)`	sum of the values of the field X
`sumsq(X)`	sum of the squares of the values of the field X
`values(X)`	list of all distinct values of the field X as a multi-value entry. The order of the values is alphabetical
`var(X)`	sample variance of the field X

Index Statistics

Compute index-related statistics.

From this point onward, splunk refers to the partial or full path of the Splunk app on your device $SPLUNK_HOME/bin/splunk, such as /Applications/Splunk/bin/splunk on macOS, or, if you have performed cd and entered /Applications/Splunk/bin/, simply ./splunk.

Function	Description
`| eventcount summarize=false index=* | dedup index | fields index`	List all indexes on your Splunk instance. On the command line, use this instead:splunk list index
`| eventcount summarize=false report_size=true index=* | eval size_MB = round(size_bytes/1024/1024,2)`	Show the number of events in your indexes and their sizes in MB and bytes
`| REST /services/data/indexes | table title currentDBSizeMB`	List the titles and current database sizes in MB of the indexes on your Indexers
`index=_internal source=*metrics.log group=per_index_thruput series=* | eval MB = round(kb/1024,2) | timechart sum(MB) as MB by series`	Query write amount in MB per index from metrics.log
`index=_internal metrics kb series!=_* "group=per_host_thruput" | timechart fixedrange=t span=1d sum(kb) by series`	Query write amount in KB per day per Indexer by each host
`index=_internal metrics kb series!=_* "group=per_index_thruput" | timechart fixedrange=t span=1d sum(kb) by series`	Query write amount in KB per day per Indexer by each index

Reload apps

To reload Splunk, enter the following in the address bar or command line interface.

Address bar	Description
`http://localhost:8000/debug/refresh`	Reload Splunk. Replace localhost:8000 with the base URL of your Splunk Web server if you're not running it on your local machine.
Command line	**Description**
`splunk _internal call /data/inputs/monitor/_reload`	Reload Splunk file input configuration
`splunk stopsplunk enable webserversplunk start`	These three lines in succession restart Splunk.

Debug Traces

You can enable traces listed in
`$SPLUNK_HOME/var/log/splunk/splunkd.log.`

To change trace topics permanently, go to $SPLUNK_HOME/bin/splunk/etc/log.cfg and change the trace level, for example, from INFO to DEBUG: category.TcpInputProc=DEBUG

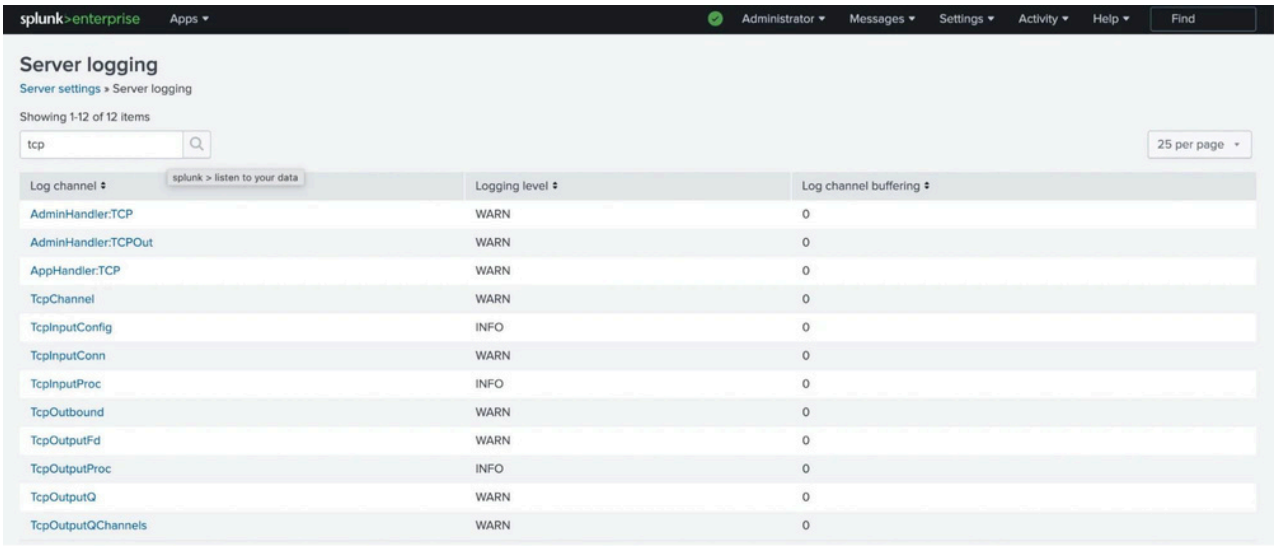
Then

```
08-10-2022 05:20:18.653 -0400 INFO  ServerConfig [0 MainThread] - Will
generate GUID, as none found on this server.
```

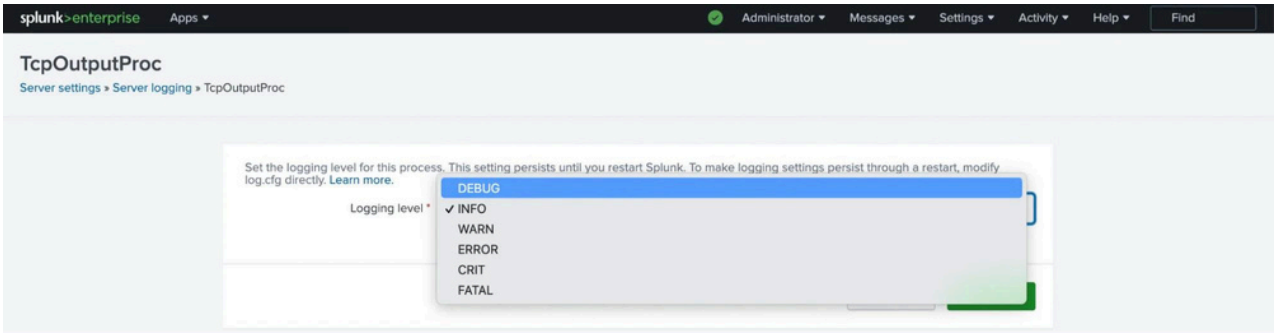
becomes

```
08-10-2022 05:20:18.653 -0400 DEBUG  ServerConfig [0 MainThread] - Will
generate GUID, as none found on this server.
```

To change the trace settings only for the current instance of Splunk, go to Settings > Server Settings > Server Logging:



Filter the log channels as above.



Select your new log trace topic and click Save. This persists until you stop the server.

Configuration

The following changes Splunk settings. Where necessary, append -auth user:pass to the end of your command to authenticate with your Splunk web server credentials.

Command line	Description
Troubleshooting	
`splunk btool inputs list`	List Splunk configurations
`splunk btool check`	Check Splunk configuration syntax
Input management	
`splunk _internal call /data/inputs/tcp/raw`	List TCP inputs
`splunk _internal call /data/inputs/tcp/raw -get:search sourcetype=foo`	Restrict listing of TCP inputs to only those with a source type of foo
`License details of your current Splunk instance`	
`splunk list licenses`	Show your current license
`User management`	
`splunk _internal call /authentication/providers/services/_reload`	Reload authentication configurations for Splunk 6.x
`splunk _internal call /services/authentication/users -get:search admin`	Search for all users who are admins
`splunk _internal call /services/authentication/users -get:search indexes_edit`	See which users could edit indexes
`splunk _internal call /services/authentication/users/helpdesk -method DELETE`	Use the remove link in the returned XML output to delete the user helpdesk

Capacity Planning

Importing large volumes of data takes much time. If you're using Splunk in-house, the software installation of Splunk Enterprise alone requires ~2GB of disk space. You can find an excellent online calculator

The essential factors to consider are:

Input data

- Specify the amount of data concerned. The more data you send to Splunk Enterprise, the more time Splunk needs to index it into results that you can search, report and generate alerts on.

Data Retention

- Specify how long you want to keep the data. You can only keep your imported data for a maximum length of 90 days or approximately three months.
- Hot/Warm: short-term, in days.
- Cold: mid-term, in weeks.
- Archived (Frozen): long-term, in months.

Architecture

-
- Specify the number of nodes required. The more data to ingest, the greater the number of nodes required. Adding more nodes will improve indexing throughput and search performance.

Storage Required

- Specify how much space you need for hot/warm, cold, and archived data storage.

Storage Configuration

- Specify the location of the storage configuration. If possible, spread each type of data across separate volumes to improve performance: hot/warm data on the fastest disk, cold data on a slower disk, and archived data on the slowest.

Thank you

Farhath Nathvi