splunk

The Search Tutorial introduces you to the Search and Reporting app and guides you through adding data, searching your data, and building simple reports and dashboards.

//https://docs.splunk.com/Documentation/SplunkCloud/latest/Search/GetstartedwithSearch

**Basic Search app skills**

* [Navigating Splunk Web](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Search/NavigatingSplunkWeb)
* [Using the Search app](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Search/WhatsinSplunkSearch)
* [Types of searches](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Search/Aboutsearch)
* [Types of commands](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Search/Typesofcommands)

**Detailed Search information**

* [Retrieving events](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Search/Aboutretrievingevents)
* [Specifying time ranges](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Search/Aboutsearchtimeranges)
* [Optimizing searches](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Search/Quicktipsforoptimization)
* [Creating tables and charts](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Search/Aboutreportingcommands)
* [Evaluating and manipulating fields](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Search/Aboutevaluatingandmanipulatingfields)
* [Calculating statistics](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Search/Aboutcalculatingstatistics) and [advanced statistics](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Search/Aboutadvancedstatistics)
* [Grouping and correlating events](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Search/Abouteventcorrelation)
* [Managing search jobs](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Search/Aboutjobsandjobmanagement)

### Distributed Search

If you are using Splunk Enterprise, [**distributed search**](https://docs.splunk.com/Splexicon:Distributedsearch) provides a way to scale your deployment by separating the search management and presentation layer from the indexing and search retrieval layer. For an introduction to distributed search, see the [*Distributed Search Manual*](http://docs.splunk.com/Documentation/Splunk/8.1.2/DistSearch/Whatisdistributedsearch).

// https://docs.splunk.com/Documentation/SplunkCloud/latest/SearchReference/Search

**The implied search command**

The search command is implied at the beginning of every search.

When search is the first command in the search, you can use terms such as keywords, phrases, fields, boolean expressions, and comparison expressions to specify exactly which events you want to retrieve from Splunk indexes. If you don't specify a field, the search looks for the terms in the the \_raw field.

Some examples of search terms are:

* keywords: error login, which is the same as specifying for error AND login
* quoted phrases: "database error"
* boolean operators: login NOT (error OR fail)
* wildcards: fail\*
* field-value pairs: status=404, status!=404, or status>200

**Boolean expressions**

The order in which Boolean expressions are evaluated with the search is:

1. Expressions within parentheses
2. NOT clauses
3. OR clauses
4. AND clauses

This evaluation order is different than the order used with the where command. The where command evaluates AND clauses before OR clauses.

### Comparing two fields

To compare two fields, **do not** specify index=myindex fieldA=fieldB or index=myindex fieldA!=fieldB with the search command. When specifying a comparison\_expression, the search command expects a <field> compared with a <value>. The search command interprets fieldB as the value, and not as the name of a field.

Use the where command to compare two fields.

index=myindex | where fieldA=fieldB

For not equal comparisons, you can specify the criteria in several ways.

index=myindex | where fieldA!=fieldB

or

index=myindex | where NOT fieldA=fieldB

See [Difference between NOT and !=](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Search/NOTexpressions) in the *Search Manual*.

**Multiple field-value comparisons with the IN operator**

Use the IN operator when you want to determine if a field contains one of several values.

For example, use this syntax:

... error\_code IN (400, 402, 404, 406) | ...

Instead of this syntax:

... error\_code=400 OR error\_code=402 OR error\_code=404 OR error\_code=406 | ...

When used with the search command, you can use a wildcard character in the list of values for the IN operator. For example:

... error\_code IN (40\*) | ...

You can use the NOT operator with the IN operator. For example:

... NOT clientip IN (211.166.11.101, 182.236.164.11, 128.241.220.82) | ...

There is also an IN function that you can use with the eval and where commands. Wild card characters are not allowed in the values list when the IN function is used with the eval and where commands. See [Comparison and Conditional functions](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/SearchReference/ConditionalFunctions).

### Lexicographical order

Lexicographical order sorts items based on the values used to encode the items in computer memory. In Splunk software, this is almost always UTF-8 encoding, which is a superset of ASCII.

* Numbers are sorted before letters. Numbers are sorted based on the first digit. For example, the numbers 10, 9, 70, 100 are sorted lexicographically as 10, 100, 70, 9.
* Uppercase letters are sorted before lowercase letters.
* Symbols are not standard. Some symbols are sorted before numeric values. Other symbols are sorted before or after letters.

You can specify a custom sort order that overrides the lexicographical order. See the blog [Order Up! Custom Sort Orders](https://www.splunk.com/blog/2019/08/29/order-up-custom-sort-orders.html).

### Quotes and escaping characters

In general, you need quotation marks around phrases and field values that include white spaces, commas, pipes, quotations, and brackets. Quotation marks must be balanced. An opening quotation must be followed by an unescaped closing quotation. For example:

* A search such as error | stats count will find the number of events containing the string error.
* A search such as ... | search "error | stats count" would return the raw events containing error, a pipe, stats, and count, in that order.

Additionally, you want to use quotation marks around keywords and phrases if you do not want to search for their default meaning, such as Boolean operators and field/value pairs. For example:

* A search for the keyword AND without meaning the Boolean operator: error "AND"
* A search for this field/value phrase: error "startswith=foo"

The backslash character ( \ ) is used to escape quotes, pipes, and itself. Backslash escape sequences are still expanded inside quotation marks. For example:

* The sequence \| as part of a search will send a pipe character to the command, instead of having the pipe split between commands.
* The sequence \" will send a literal quotation mark to the command, for example for searching for a literal quotation mark or inserting a literal quotation mark into a field using rex.
* The \\ sequence will be available as a literal backslash in the command.

Unrecognized backslash sequences are not altered:

* For example \s in a search string will be available as \s to the command, because \s is not a known escape sequence.
* However, in the search string \\s will be available as \s to the command, because \\ is a known escape sequence that is converted to \.

### Search with TERM()

You can use the TERM() directive to force Splunk software to match whatever is inside the parentheses as a single term in the index. TERM is more useful when the term contains minor segmenters, such as periods, and is bounded by major segmenters, such as spaces or commas. In fact, TERM does not work for terms that are not bounded by major breakers.

See [Use CASE and TERM to match phrases](http://docs.splunk.com/Documentation/Splunk/8.1.2/Search/UseCASEandTERMtomatchphrases) in the *Search Manual*.

### Search with CASE()

You can use the CASE() directive to search for terms and field values that are case-sensitive.

See [Use CASE and TERM to match phrases](http://docs.splunk.com/Documentation/Splunk/8.1.2/Search/UseCASEandTERMtomatchphrases) in the *Search Manual*.

## Examples

These examples demonstrate how to use the search command. You can find more examples in the [Start Searching](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/SearchTutorial/Startsearching) topic of the [Search Tutorial](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/SearchTutorial/WelcometotheSearchTutorial).

### 1. Field-value pair matching

This example demonstrates field-value pair matching for specific values of source IP (src) and destination IP (dst).

src="10.9.165.\*" OR dst="10.9.165.8"

### 2. Using boolean and comparison operators

This example demonstrates field-value pair matching with boolean and comparison operators. Search for events with code values of either 10 or 29, and any host that isn't "localhost", and an xqp value that is greater than 5.

(code=10 OR code=29) host!="localhost" xqp>5

In this example you could also use the IN operator since you are specifying two field-value pairs on the same field. The revised search is:

code IN(10, 29) host!="localhost" xqp>5

### 3. Using wildcards

This example demonstrates field-value pair matching with wildcards. Search for events from all the web servers that have an HTTP client or server error status.

host=webserver\* (status=4\* OR status=5\*)

In this example you could also use the IN operator since you are specifying two field-value pairs on the same field. The revised search is:

host=webserver\* status IN(4\*, 5\*)

### 4. Using the IN operator

This example shows how to use the IN operator to specify a list of field-value pair matchings. In the events from an access.log file, search the action field for the values addtocart or purchase.

sourcetype=access\_combined\_wcookie action IN (addtocart, purchase)

### 5. Specifying a secondary search

This example uses the search command twice. The search command is implied at the beginning of every search with the criteria eventtype=web-traffic. The search command is used again later in the search pipeline to filter out the results. This search defines a web session using the [transaction](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/SearchReference/Transaction) command and searches for the user sessions that contain more than three events.

eventtype=web-traffic | transaction clientip startswith="login" endswith="logout" | search eventcount>3

### 6. Using the NOT or != comparisons

Searching with the boolean "NOT"comparison operator is not the same as using the "!=" comparison.

The following search returns everything except fieldA="value2", including all other fields.

NOT fieldA="value2"

The following search returns events where fieldA exists and does not have the value "value2".

fieldA!="value2"

If you use a wildcard for the value, NOT fieldA=\* returns events where fieldA is null or undefined, and fieldA!=\* never returns any events.

See [Difference between NOT and !=](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Search/Usethesearchcommand#Difference_between_NOT_and_.21.3D) in the *Search Manual*.

## Events, event data, and fields

The phrase *event data* refers to your data after it has been added to the Splunk index. Events are a single record of activity or instance of this event data. For example, an event might be a single log entry in a log file. Because the Splunk software separates individual events by their time information, an event is distinguished from other events by a timestamp.

Here is a sample event:

172.26.34.223 - - [01/Jul/2005:12:05:27 -0700] "GET /trade/app?action=logout HTTP/1.1" 200 2953

Events contain pairs of information, or fields. When you add data and it gets indexed, the Splunk software automatically extracts some useful fields for you, such as the host the event came from and the [type of data source](http://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Data/Whysourcetypesmatter) it is.

# Types of searches

https://docs.splunk.com/Documentation/SplunkCloud/8.1.2012/Search/Aboutsearch

## Raw event searches

## Transforming searches

## Information density

Whether you are retrieving raw events or building a report, you should also consider whether you are running a search for *sparse* or *dense* information:

* **Sparse searches** are searches that look for a single event or an event that occurs infrequently within a large set of data. You have probably heard these referred to as 'needle in a haystack' or "rare term" searches. Some examples of these searches include: searching for a specific and unique IP address or error code.
* **Dense searches** are searches that scan through and report on many events. Some examples of these searches include: counting the number of errors that occurred or finding all events from a specific host.

ELK

[**Search your data**](https://www.elastic.co/guide/en/elasticsearch/reference/current/search-your-data.html)

https://www.elastic.co/guide/en/elasticsearch/reference/current/search-your-data.html

* [Collapse search results](https://www.elastic.co/guide/en/elasticsearch/reference/current/collapse-search-results.html)
* [Filter search results](https://www.elastic.co/guide/en/elasticsearch/reference/current/filter-search-results.html)
* [Highlighting](https://www.elastic.co/guide/en/elasticsearch/reference/current/highlighting.html)
* [Long-running searches](https://www.elastic.co/guide/en/elasticsearch/reference/current/async-search-intro.html)
* [Near real-time search](https://www.elastic.co/guide/en/elasticsearch/reference/current/near-real-time.html)
* [Paginate search results](https://www.elastic.co/guide/en/elasticsearch/reference/current/paginate-search-results.html)
* [Retrieve inner hits](https://www.elastic.co/guide/en/elasticsearch/reference/current/inner-hits.html)
* [Retrieve selected fields](https://www.elastic.co/guide/en/elasticsearch/reference/current/search-fields.html)
* [Search across clusters](https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-cross-cluster-search.html)
* [Search multiple data streams and indices](https://www.elastic.co/guide/en/elasticsearch/reference/current/search-multiple-indices.html)
* [Search shard routing](https://www.elastic.co/guide/en/elasticsearch/reference/current/search-shard-routing.html)
* [Sort search results](https://www.elastic.co/guide/en/elasticsearch/reference/current/sort-search-results.html)

https://dzone.com/articles/23-useful-elasticsearch-example-queries

### Basic Match Query

### Boosting

Since we are searching across multiple fields, we may want to boost the scores in a certain field.

### Bool Query

### Fuzzy Queries

### Wildcard Query

### Regexp Query

### Match Phrase Query

### Match Phrase Prefix

### Query String

### Term/Terms Query

### Term Query - Sorted

### Range Query

kibana

https://www.elastic.co/guide/en/kibana/current/index.html

https://www.elastic.co/guide/en/kibana/current/kuery-query.html

[Discover](https://www.elastic.co/guide/en/kibana/current/discover.html)

* [Create an index pattern](https://www.elastic.co/guide/en/kibana/current/index-patterns.html)
* [Set the time filter](https://www.elastic.co/guide/en/kibana/current/set-time-filter.html)
* [Search data](https://www.elastic.co/guide/en/kibana/current/search.html)
  + [Kibana Query Language](https://www.elastic.co/guide/en/kibana/current/kuery-query.html)
  + [Lucene query syntax](https://www.elastic.co/guide/en/kibana/current/lucene-query.html)
  + [Save a search](https://www.elastic.co/guide/en/kibana/current/save-open-search.html)
  + [Save a query](https://www.elastic.co/guide/en/kibana/current/save-load-delete-query.html)