splunk> .conf2017

# Revealing the Magic

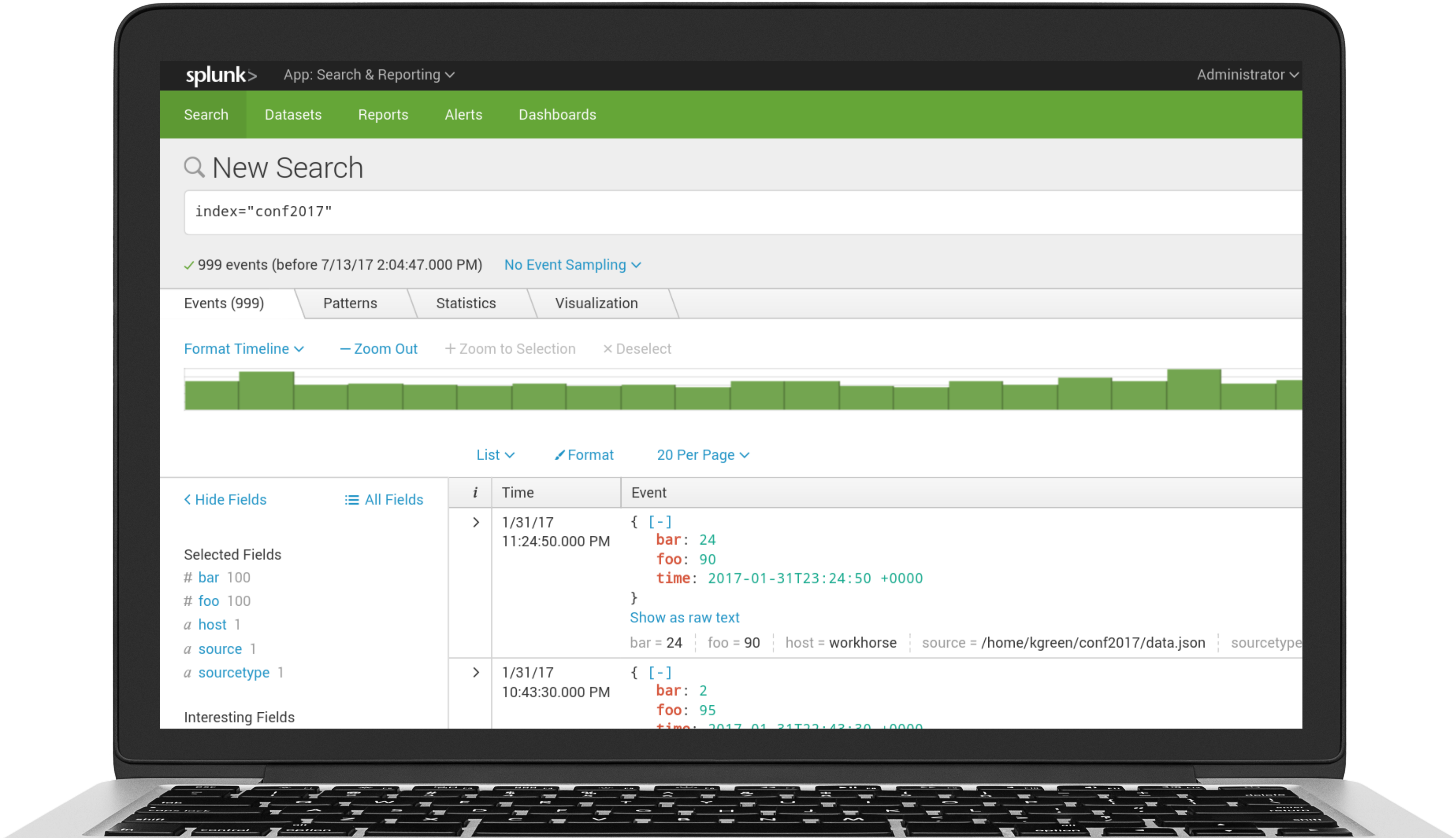
## The Lifecycle of a Splunk Search

Kellen Green | Senior Software Engineer

September 27th, 2017 | Washington, DC

# About Myself

web developer


splunk>
App: Search & Reporting
Administrator
Search
Datasets
Reports
Alerts
Dashboards
New Search
index="conf2017"
999 events (before 7/13/17 2:04:47.000 PM)
No Event Sampling
Events (999)
Patterns
Statistics
Visualization
Format Timeline
Zoom Out
Zoom to Selection
Deselect
List
Format
20 Per Page
Hide Fields
All Fields
Selected Fields
bar 100
foo 100
host 1
source 1
sourcetype 1
Interesting Fields
i
Time
Event
1/31/17
11:24:50.000 PM
{ [-]
bar: 24
foo: 90
time: 2017-01-31T23:24:50 +0000
}
Show as raw text
bar = 24
foo = 90
host = workhorse
source = /home/kgreen/conf2017/data.json
sourcetype
1/31/17
10:43:30.000 PM
{ [-]
bar: 2
foo: 95

# "OUR DEVELOPERS HAVE PRODUCTIVITY SUPERPOWERS."

Kris Wehner, VP of Engineering, Yelp Reservations

yelp

**IT'S NOT MAGIC. IT'S SPLUNK.™**

splunk> .conf2017

# Magic?

Let's debunk that!

1. Develop a deeper understanding of the core components that make up a Splunk search.
2. Increase performance of your searches through more efficient queries.
3. Obtain stronger grasp of which deployment types are better suited for specific workloads.

splunk> .conf2017

# Data Set

## 26 event CSV file

```
time,foo,bar
2017-09-01T16:00:00 +0000,a,z
2017-09-02T02:00:00 +0000,b,y
2017-09-03T12:00:00 +0000,c,x
2017-09-04T18:00:00 +0000,d,w
2017-09-05T03:00:00 +0000,e,v
2017-09-06T08:00:00 +0000,f,u
2017-09-07T22:00:00 +0000,g,t
...
```

- One event per day from Sept. 1 - 26
  - Random hour of the day
- Indexed field `foo`
  - Descending A - Z
- Unindexed field `bar`
  - Ascending Z - A

# Search #1

## Indexer Workflow

```
index="conf2017" foo="0"
```

Sept. 1st to the 27th

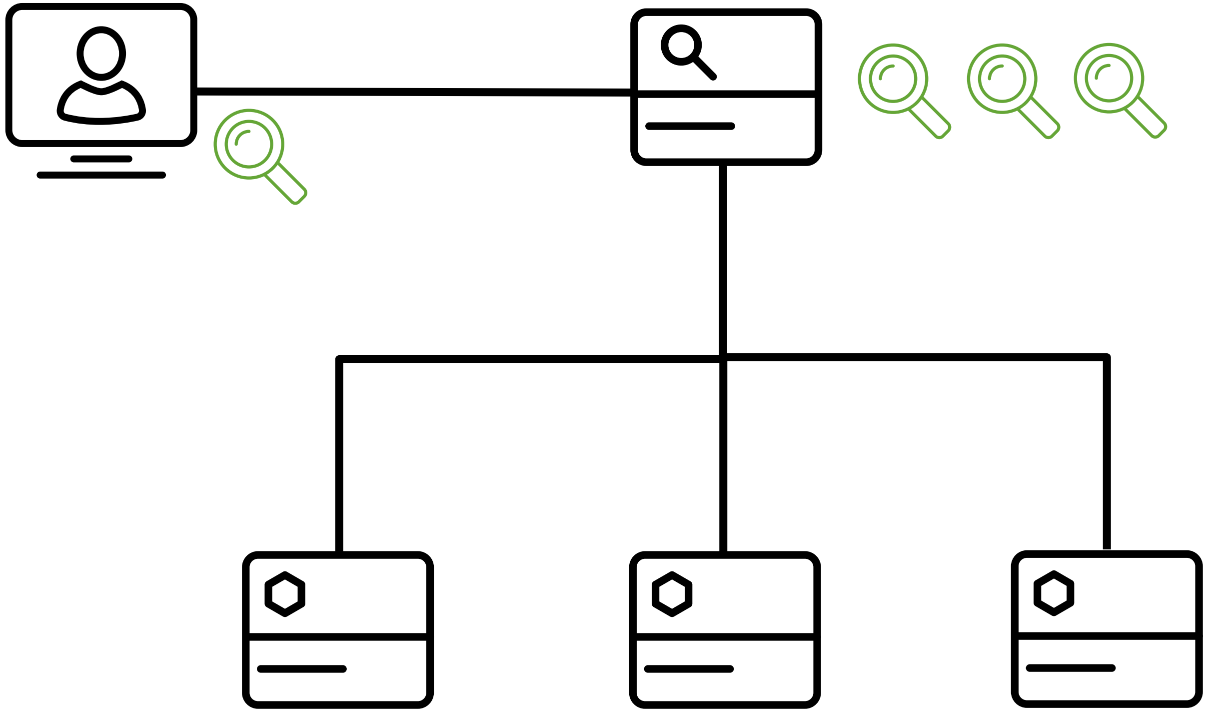
# No Results?

index="conf2017" foo="0"

Yep, but I promise it's interesting!

# Client to Indexer

index="conf2017" foo="0"



# Indexes Directory

index="conf2017" foo="0"

```
$ cd $SPK_IDX/var/lib/splunk/
$ ls -l
    audit
    authDb
    conf2015
    conf2016
    conf2017
    defaultdb
    historydb
    Kvstore
```

- Root directory for indexes.
- Check if queried index directory exists.
- Specify an index to improve improve search performance.

# Index Directory

index="conf2017" foo="0"

```
$ cd conf2017/
$ ls -l
colddb
datamodel_summary
db
thaweddb
```

- `colddb` houses older searchable data.
  - Implement cheaper storage solutions.
- `db` directory for fresh data in high demand.
- Configurable in `indexes.conf`.

# Buckets Directory

index="conf2017" foo="0"

```
$ cd db/
$ ls -l
    .bucketManifest
    CreationTime
→   db_1468867200_1471545599_0
→   db_1485388800_1493228720_1
→   hot_v1_2
    GlobalMetaData
```

- Hot buckets are still being written to.
- Warm buckets are event immutable.
  - Named by time range.
- Specify strict time range to boost Performance.

# Bloom Filter

index="conf2017" foo="0"

```
$ cd db_1485388800_1493228720_1/
$ ls -l
    1485388800-1483228800.tsidx
    bloomfilter
    bucket_info.csv
    Hosts.data
    optimize.result
    rawdata
    Sources.data
    SourceTypes.data
    Strings.data
```

- Scanning buckets can be expensive.
- Bloom filter provides us with a fast way to determine if a term is **NOT** in a bucket.

# Bloom Filter Hashing

index="conf2017" foo="0"

```
hash_1("foo=0") == 0
hash_2("foo=0") == 7
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

- Second hash result points to `false`, so bucket will not contain matching events.

# Bloom Filter Performance

index="conf2017" foo="0"

- For search terms that are common, the bloom filter will do nothing to improve search performance.

- Huge performance boost for rare and nonexistent events.
    - Speed up on the order of 100x (1-2s to 10ms).

# Search #2

## Indexing

```
index="conf2017" foo="a"
```

**vs**

```
index="conf2017" bar="z"
```

# Both Give Same Result

foo="a" vs bar="z"

```
2017-09-01T16:00:00 +0000,a,z
```

# TSIDX File

foo="a" vs bar="z"

```
$ pwd
    db_1485388800_1493228720_1
$ ls -l
──→ 1485388800-1483228800.tsidx
    bloomfilter
    bucket_info.csv
    Hosts.data
    optimize.result
    rawdata
    Sources.data
    SourceTypes.data
    Strings.data
```

- Index file used to reduce the number of matching events.
- Lexigraphically sorted array of all terms within the bucket.
- The flag for `|delete` is also set here.

splunk> .conf2017

# Lispy Query

## foo="a" vs bar="z"

- The Lispy query is used to when searching through TSIDX files.
- Created by the Search Head at search time.
- `foo="a"` becomes `[foo::a]` in Lispy.
- This will match all events where `foo` equals exactly `a`.

# Lispy for Unindexed Fields

foo="a" vs bar="z"

- `bar="z"` becomes `[z]` in Lispy.
- This will match all events that contain `z` anywhere within the event.
- This might seem counter intuitive, but there is a good reason for this behavior.

# Post TSIDX Results

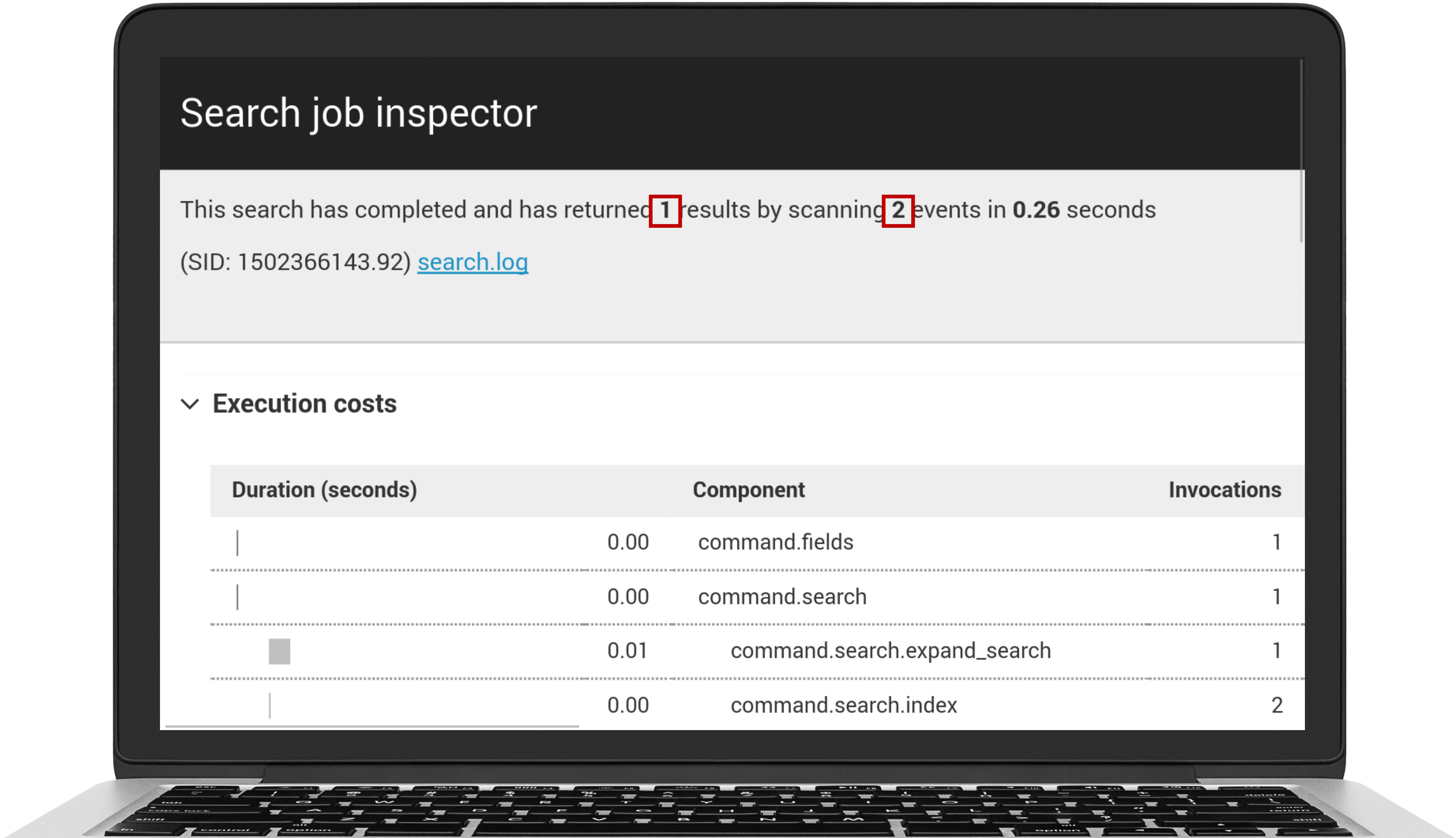
foo="a" vs bar="z"

[foo::a]

| Time | Foo | Bar |
|---|---|---|
| 2017-09-01T16:00:00 +0000 | **a** | z |

[z]

| Time | Foo | Bar |
|---|---|---|
| 2017-09-01T16:00:00 +0000 | a | **z** |
| 2017-09-26T07:00:00 +0000 | **z** | a |



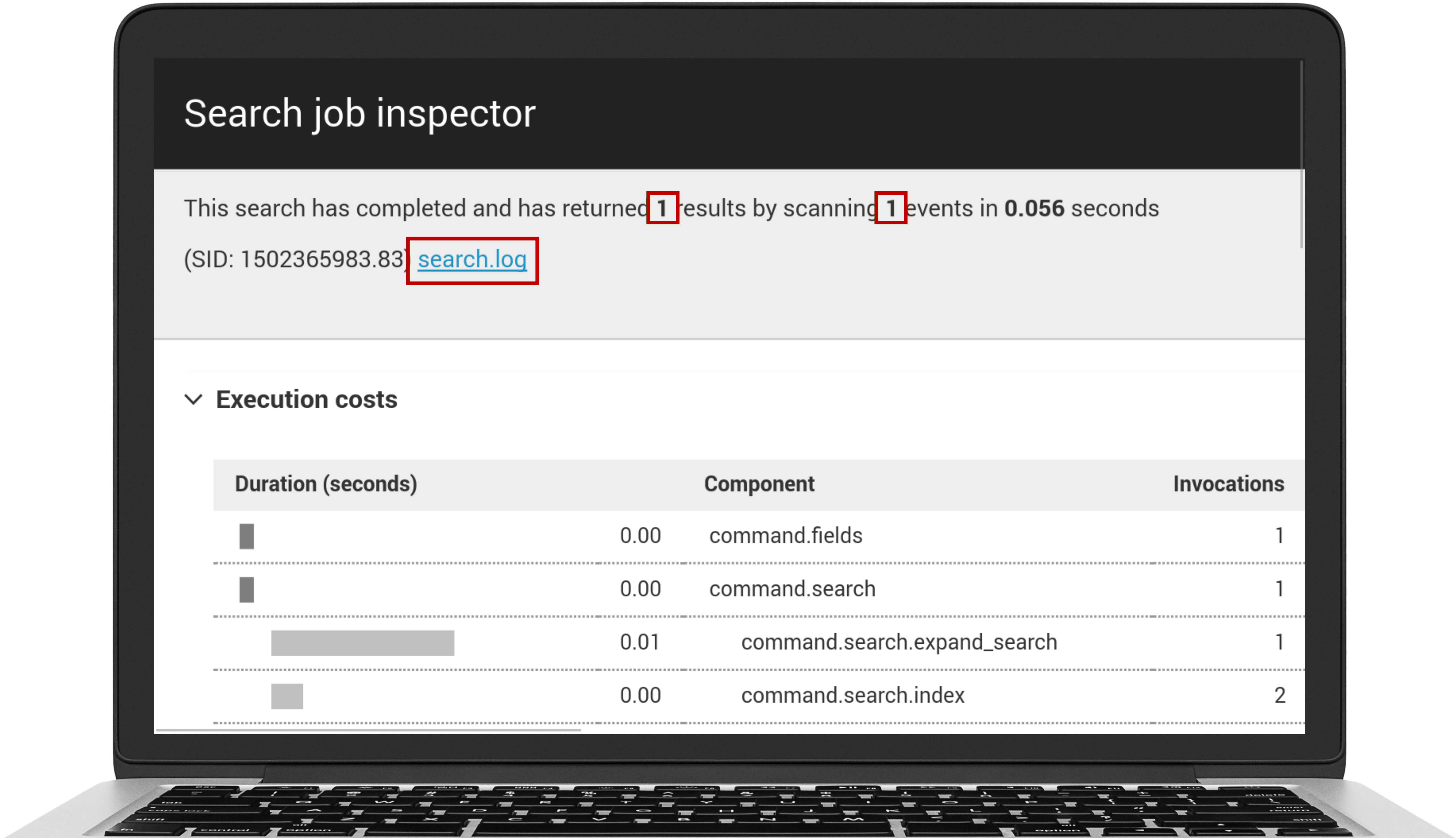
# Search job inspector

This search has completed and has returned **1** results by scanning **2** events in **0.26** seconds

(SID: 1502366143.92) search.log

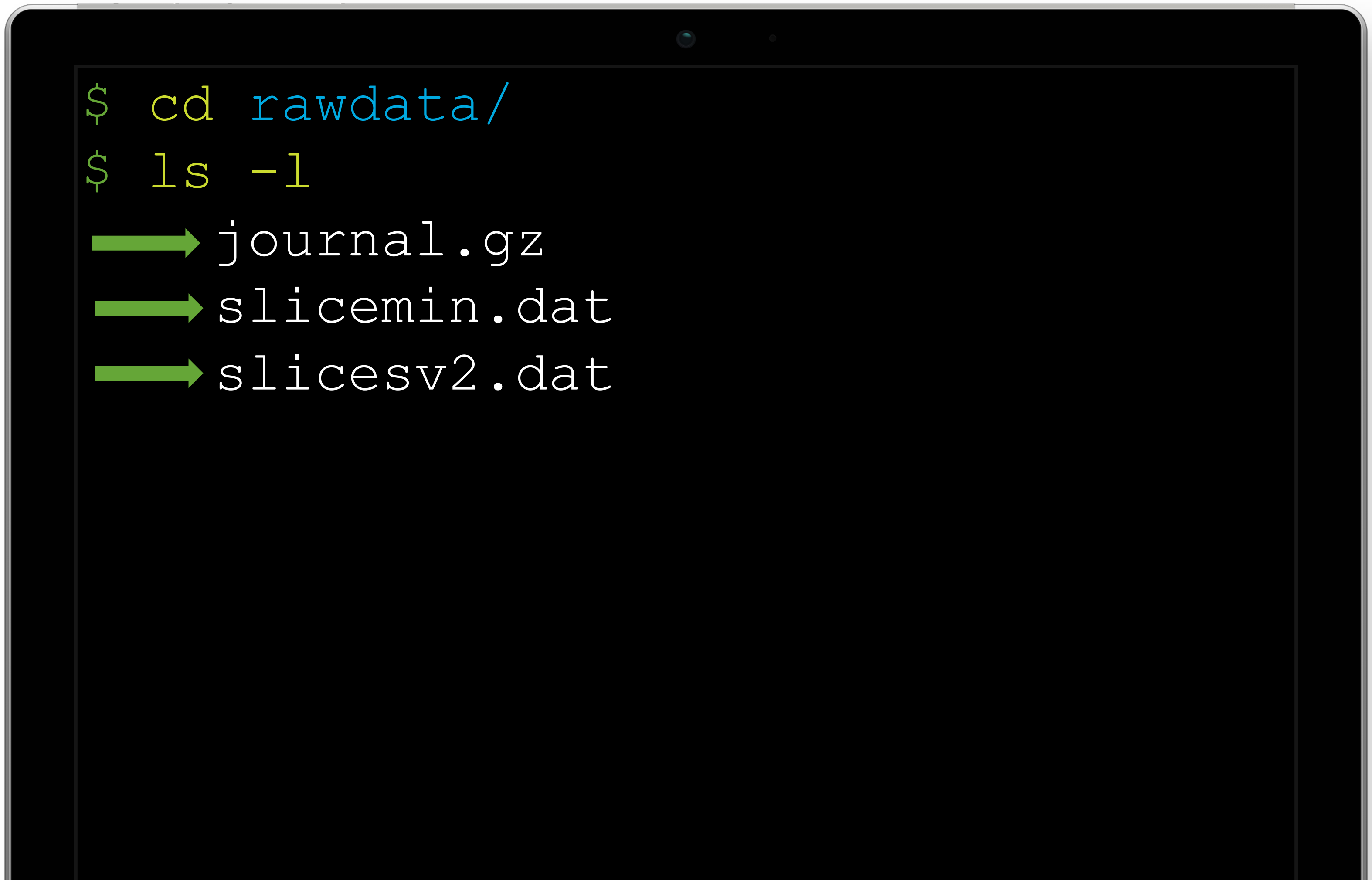
## Execution costs

| Duration (seconds) | Component | Invocations |
|---|---|---|
| 0.00 | command.fields | 1 |
| 0.00 | command.search | 1 |
| 0.01 | command.search.expand_search | 1 |
| 0.00 | command.search.index | 2 |


Search job inspector
This search has completed and has returned 1 results by scanning 1 events in 0.056 seconds
(SID: 1502365983.83) search.log
Execution costs
Duration (seconds)
Component
Invocations
0.00
command.fields
1
0.00
command.search
1
0.01
command.search.expand_search
1
0.00
command.search.index
2

# Raw Data Extraction

foo="a" vs bar="z"




- `journal.gz` compressed slices of raw events.
- `slices.dat` map from TSIDX to slice.
- Remaining unwanted events will be filtered during extraction.

# Cons of Unindexed Fields

foo="a" vs bar="z"

- Increased number of potential matching events coming out of TSIDX.
- This list is kept in memory, leading to increased memory usage.
- More events, leads to more CPU needed for Journal decompression.

# Index Everything?

foo="a" vs bar="z"

- This can quickly explode the size of your TSIDX files.
  - Leading to slow queries across the board.

- Only recommended for fields who's key-val pair is important, AND has a value which frequently occurs in other fields.
  - For example the pair `foo="a"` is important and often searched.
  - But `bar="a"`, `baz="a"`, and `biz="a"` are also common occurrences.
  - Then `foo` might make for a good index candidate.

# Walklex Command

foo="a" vs bar="z"

```
$ walklex my.tsidx "foo::a"
0035130149.tsidx "foo::a"
my needle: foo::a
209 1 foo::a

$ walklex my.tsidx "z"
0035130149.tsidx "z"
my needle: z
287 2 z
```

- Shows us the number of matching TSIDX events for a given Lispy query.
- Useful for hunting down field indexing candidates.

# Search #3

## Wildcards

```
index="conf2017" foo="*a"
```

**vs**

```
index="conf2017" foo="a*"
```

# Again Same Result

foo="*a" vs foo="a*"

```
2017-09-01T16:00:00 +0000,a,z
```

# Search job inspector

This search has completed and has returned **1** results by scanning **1** events in **0.056** seconds

(SID: 1502365983.83) search.log

## Execution costs

| Duration (seconds) | Component | Invocations |
|---|---|---|
| 0.00 | command.fields | 1 |
| 0.00 | command.search | 1 |
| 0.01 | command.search.expand_search | 1 |
| 0.00 | command.search.index | 2 |

# Trailing Wildcard

foo="*a" vs foo="a*"

| Term |
|---|
| e |
| f |
| foo::a |
| foo::b |
| foo::c |
| foo::d |
| foo::e |
| foo::f |
| foo::g |

- Terms are sorted lexicographically within the TSIDX file.
- Binary search the index for the first matching term.
- For `foo="a*"`, continue downward until we come to the first none matching term.

# Leading Wildcard

foo="*a" vs foo="a*"

| Term |
| --- |
| f |
| foo::a |
| foo::b |
| foo::c |
| ... |
| foo::x |
| foo::y |
| foo::z |
| g |

- Same as trailing wildcard, start with the first matching term.
- However this time we must check all events that match our field name.
- Only when we get to "g", can we stop the search.

# Trailing Wildcard + Unindexed

foo="*a" vs foo="a*"

| Term |
|---|
| f |
| [illegible] |
| [illegible] |
| [illegible] |
| [illegible] |
| [illegible] |
| [illegible] |
| foo::z |
| g |

- What if we searched for `bar="*z"`?
- Lispy is for this search is `"[]"`.
- Skips TSIDX reducing altogether, relying completely on Journal extraction.

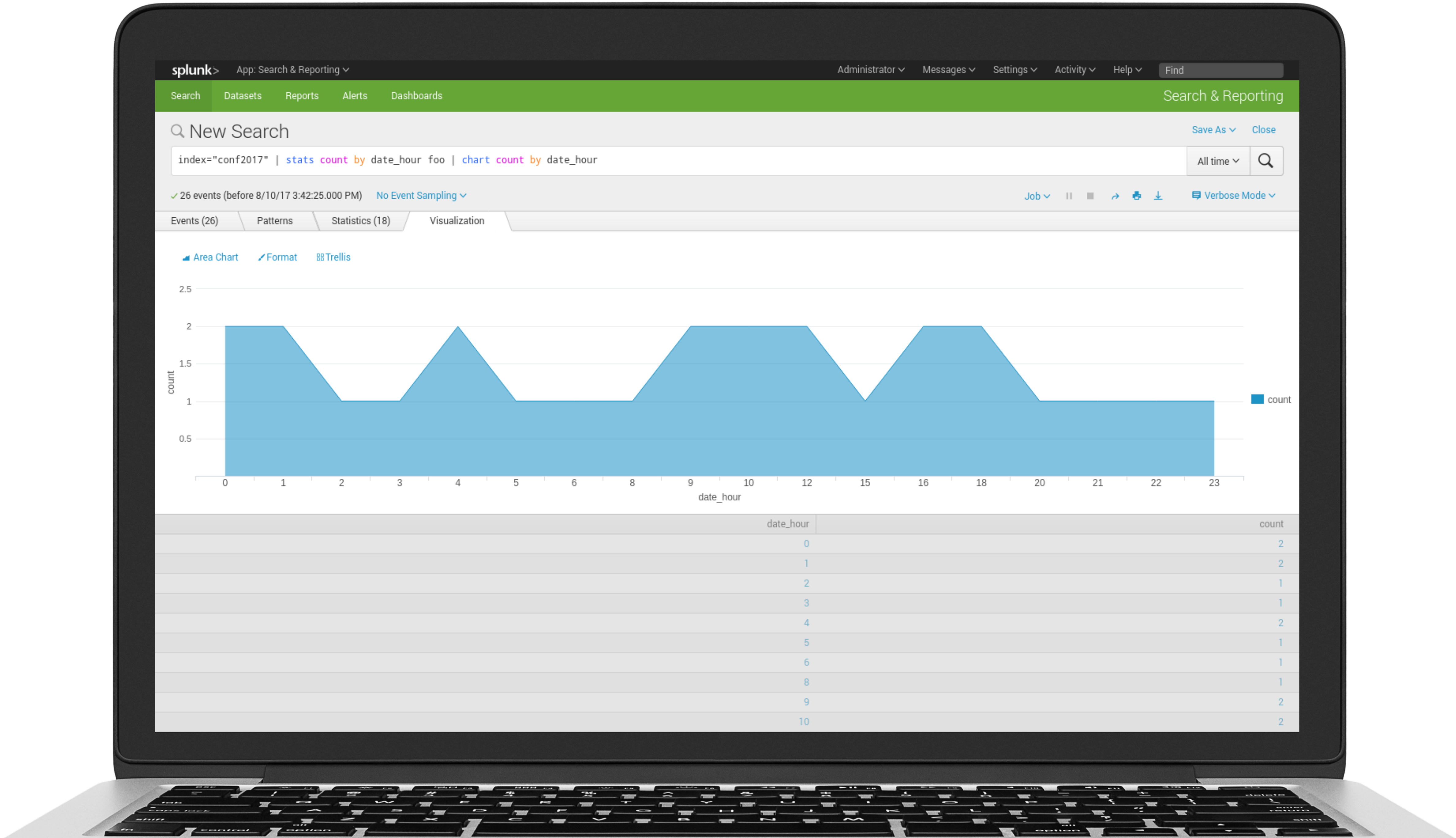
# Search #4

Transactions

```
index="conf2017"
| transaction date_hour
```

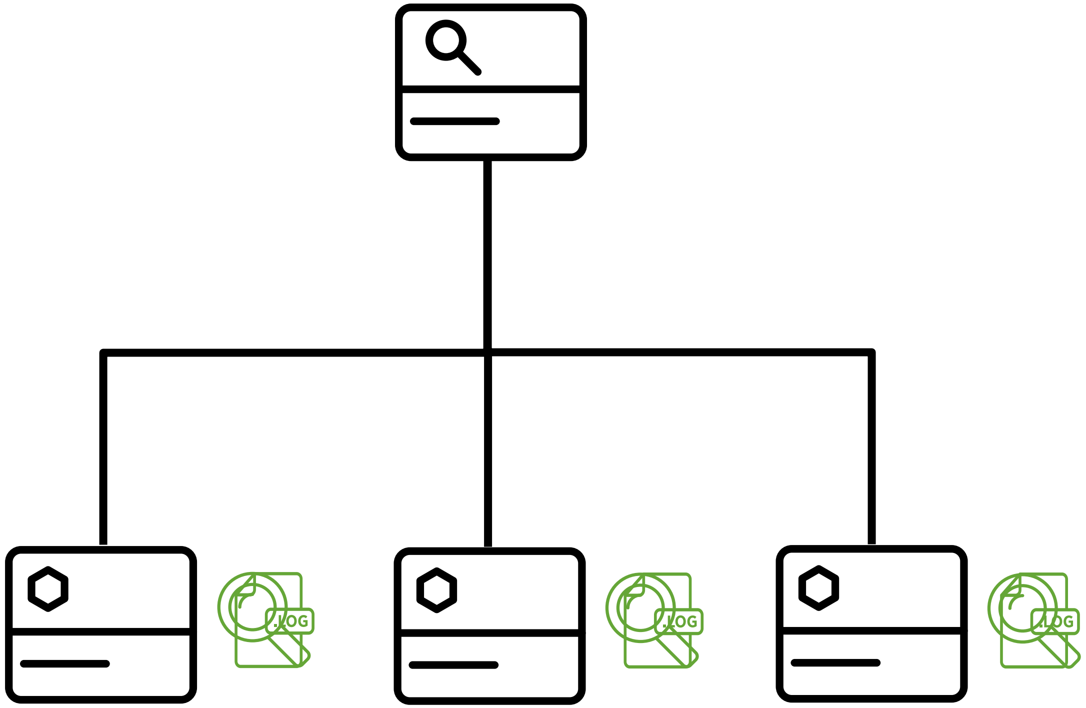
vs

```
index="conf2017"
| stats count by date_hour
```


splunk>
App: Search & Reporting
Administrator
Messages
Settings
Activity
Help
Find
Search
Datasets
Reports
Alerts
Dashboards
Search & Reporting
New Search
Save As
Close
index="conf2017" | stats count by date_hour foo | chart count by date_hour
All time
26 events (before 8/10/17 3:42:25.000 PM)
No Event Sampling
Job
Verbose Mode
Events (26)
Patterns
Statistics (18)
Visualization
Area Chart
Format
Trellis
count
2.5
2
1.5
1
0.5
0 1 2 3 4 5 6 8 9 10 12 15 16 18 20 21 22 23
date_hour
count
date_hour count
0 2
1 2
2 1
3 1
4 2
5 1
6 1
8 1
9 2
10 2

# Back to the Search Head

## transaction vs stats



# Dispatch Folder

## transaction vs stats

```
$ cd $SPK_SH/var/run/splunk/dispatch
$ ls -l
    1501601198.142
    1501601202.143
    1501601739.144
    1501601740.145
    1501601741.146
    1501601742.147
```

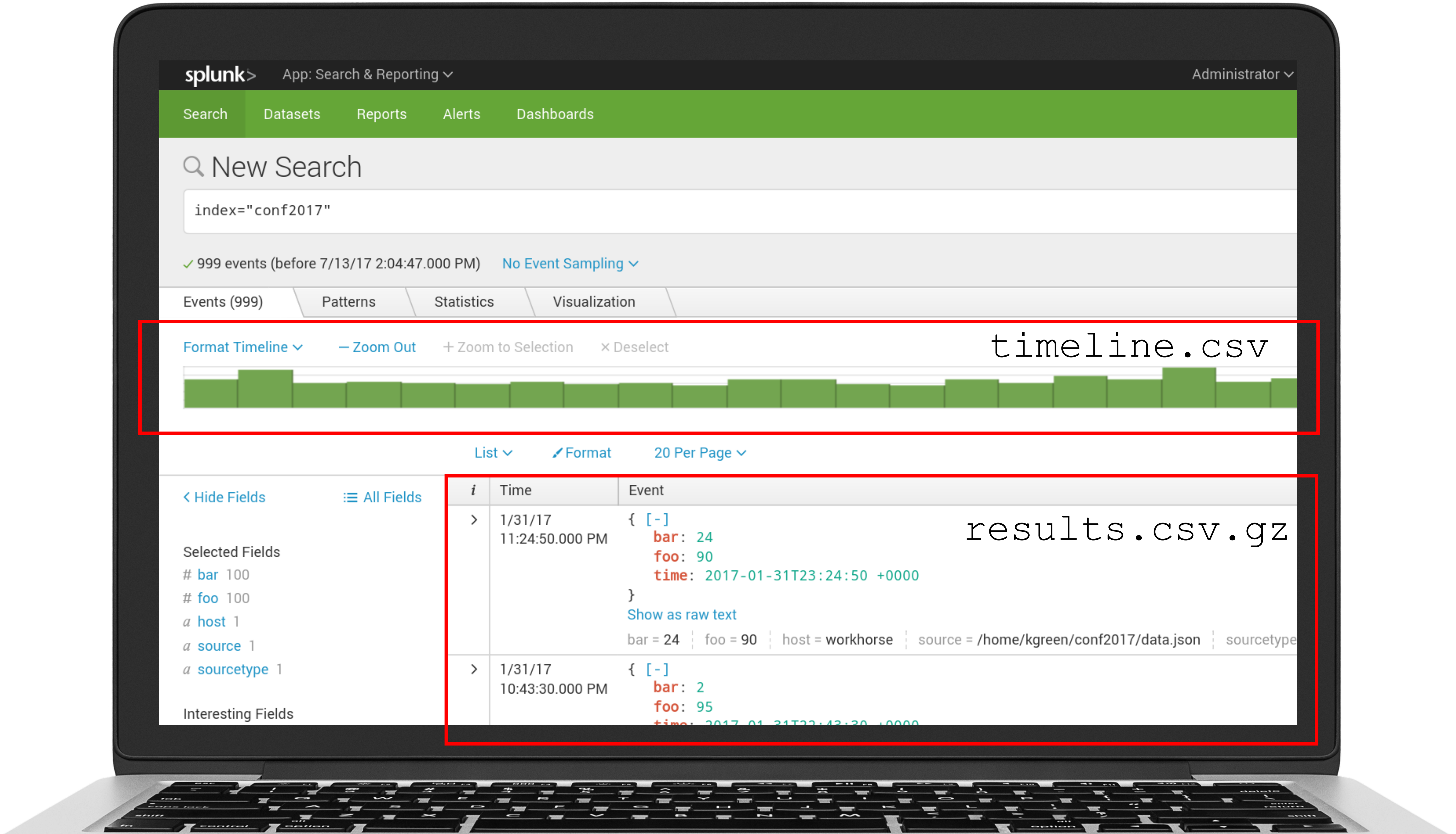
- Directory of all saved and running searches on the Search Head.
- `sid` can be obtained in the Job Inspector.

# Search Folder

## transaction vs stats

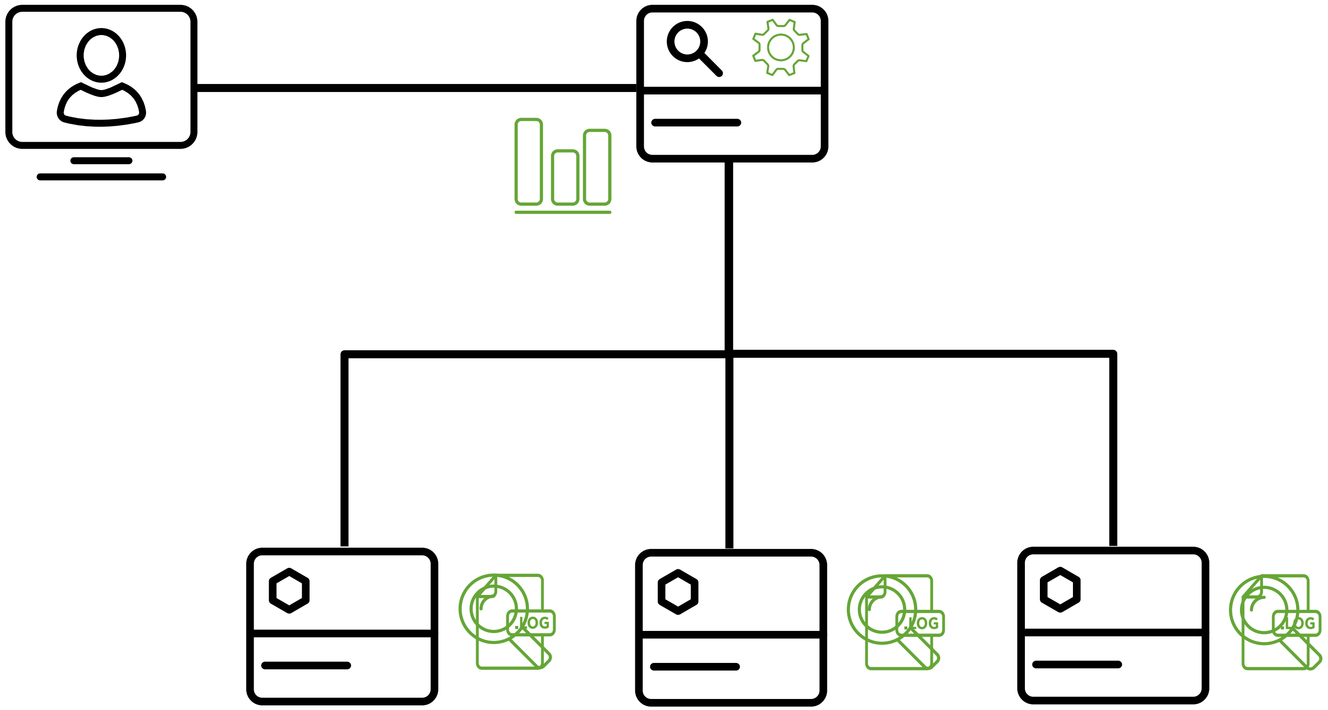
```
$ cd 1501601741.146/
$ ls -l
    args.txt
    buckets
    custom_prop.csv
    events
    timeline.csv
    info.csv
    peers.csv
    results.csv.gz
    search.log
```

- Collection of all data being returned from the indexers.
- `results.csv.gz` compressed events.
- `timeline.csv` UI timeline numbers.


splunk>
App: Search & Reporting
Administrator
Search
Datasets
Reports
Alerts
Dashboards
New Search
index="conf2017"
999 events (before 7/13/17 2:04:47.000 PM)
No Event Sampling
Events (999)
Patterns
Statistics
Visualization
Format Timeline
Zoom Out
Zoom to Selection
Deselect
timeline.csv
List
Format
20 Per Page
Hide Fields
All Fields
Selected Fields
bar 100
foo 100
host 1
source 1
sourcetype 1
Interesting Fields
i
Time
Event
1/31/17 11:24:50.000 PM
{ [-]
bar: 24
foo: 90
time: 2017-01-31T23:24:50 +0000
}
Show as raw text
bar = 24
foo = 90
host = workhorse
source = /home/kgreen/conf2017/data.json
sourcetype
1/31/17 10:43:30.000 PM
{ [-]
bar: 2
foo: 95
results.csv.gz

# Transaction Workflow

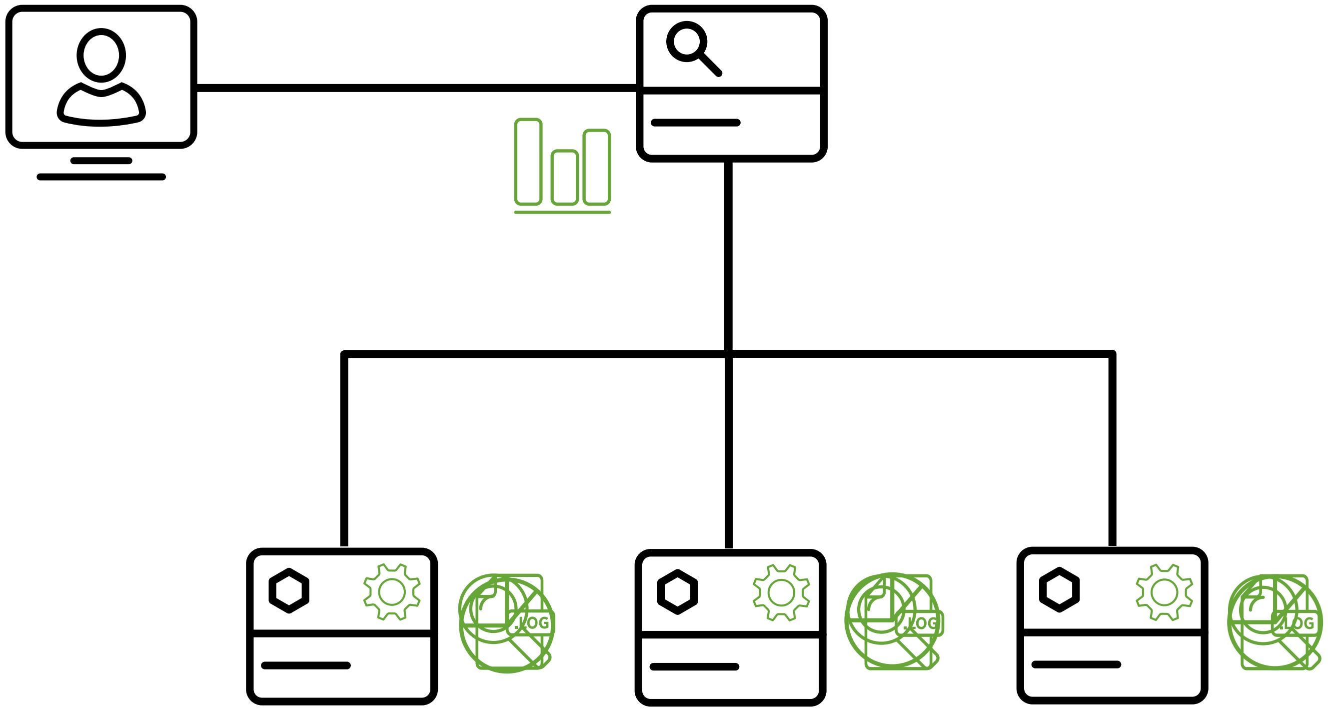
## transaction vs stats




splunk> .conf2017

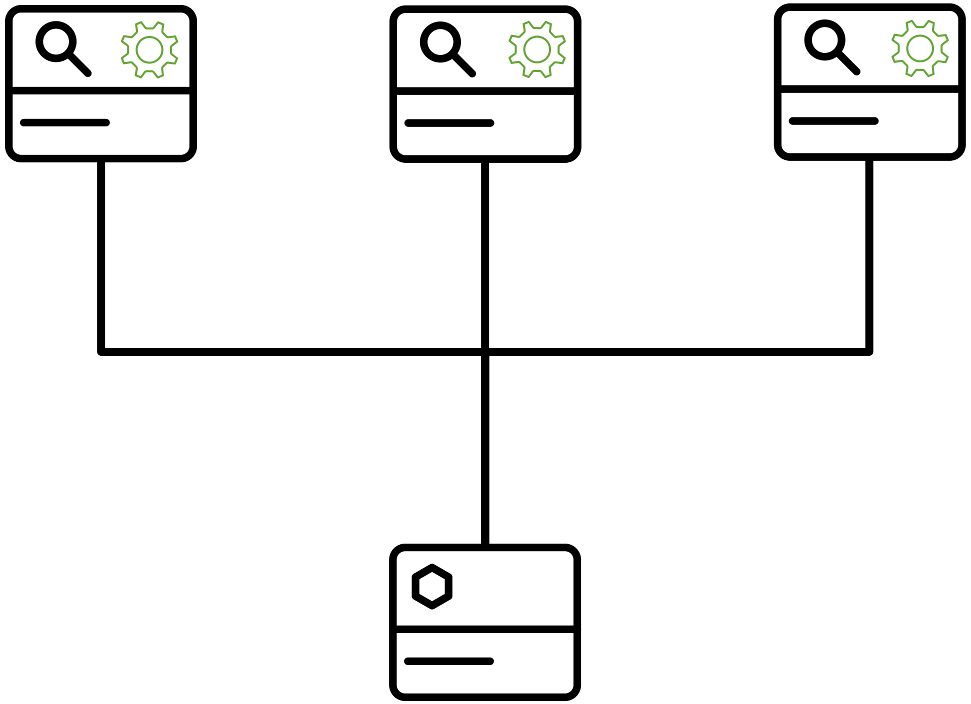
# Stats Workflow

transaction vs stats



# Search Head Cluster

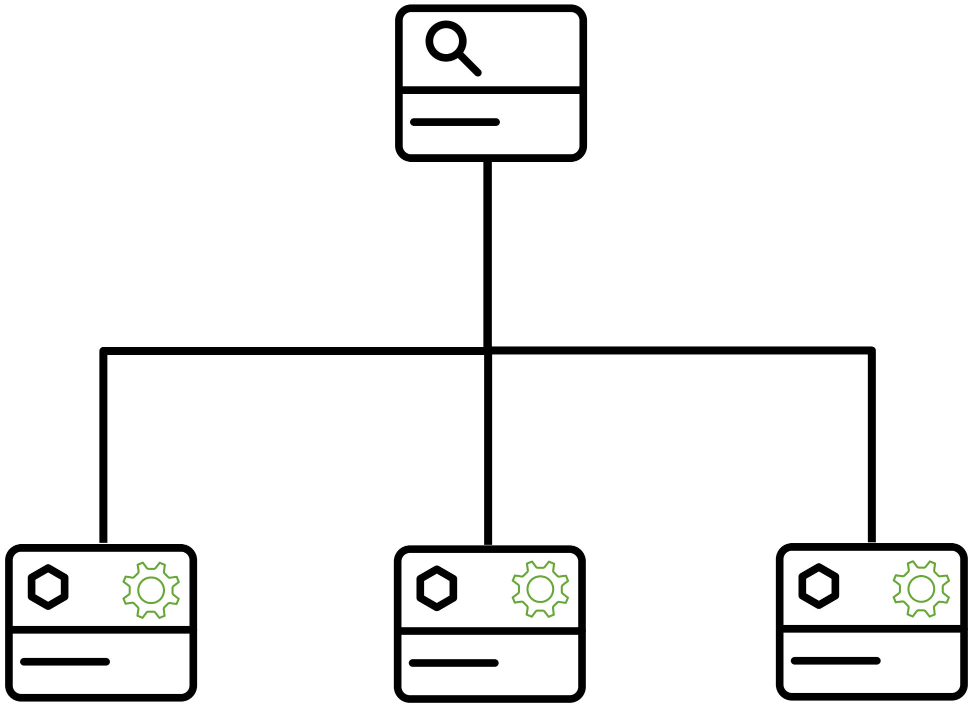
## transaction vs stats



Performance boost for `transactions` running in parallel.

# Distributed Search & Index Cluster

## transaction vs stats



Scalable performance boost to `stats` and `eval`.

# Stats Computation

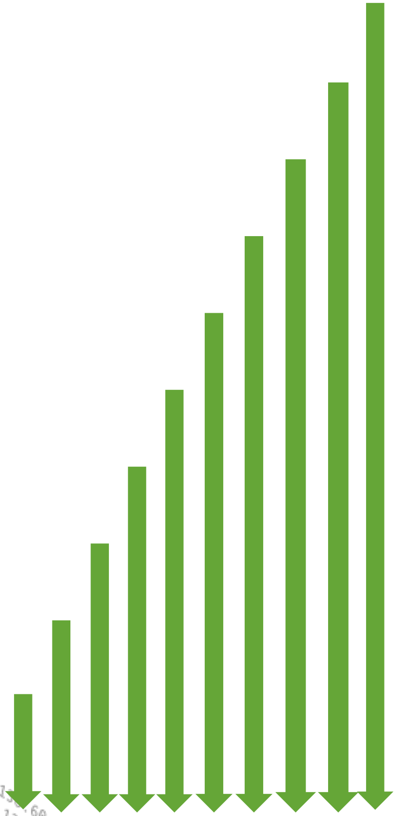
## transaction vs stats

| Time | Foo | Bar |
|---|---|---|
| 2017-09-01T**09**:00:00 | a | z |
| 2017-09-02T**07**:00:00 | b | y |
| 2017-09-03T**02**:00:00 | c | x |
| 2017-09-04T**13**:00:00 | d | w |
| 2017-09-05T**23**:00:00 | e | v |
| ... | | |
| 2017-09-21T**15**:00:00 | v | e |
| 2017-09-23T**16**:00:00 | w | d |
| 2017-09-24T**09**:00:00 | x | c |
| 2017-09-25T**06**:00:00 | y | b |
| 2017-09-26T**10**:00:00 | z | a |

- `stats` only concerns itself with a single event at once.
- Requires only one pass to complete the computation.
- For `stats count` Splunk returns value plus event occurrence count.
  - For example: hour `"09"` has 2 events.

# Transaction Discovery

## transaction vs stats



| Time | Foo | Bar |
|---|---|---|
| 2017-09-01T**09**:00:00 | a | z |
| 2017-09-02T**07**:00:00 | b | y |
| 2017-09-03T**02**:00:00 | c | x |
| 2017-09-04T**13**:00:00 | d | w |
| 2017-09-05T**23**:00:00 | e | v |
| ... | | |
| 2017-09-21T**15**:00:00 | v | e |
| 2017-09-23T**16**:00:00 | w | d |
| 2017-09-24T**09**:00:00 | x | c |
| 2017-09-25T**06**:00:00 | y | b |
| 2017-09-26T**10**:00:00 | z | a |

- Splunk must iterate over each event for every transaction window.
- Looking at a time complexity difference between n and $n^2$.
- Running only on a single Search Head doesn't help the situation.

# Search #5

## transaction plus stats

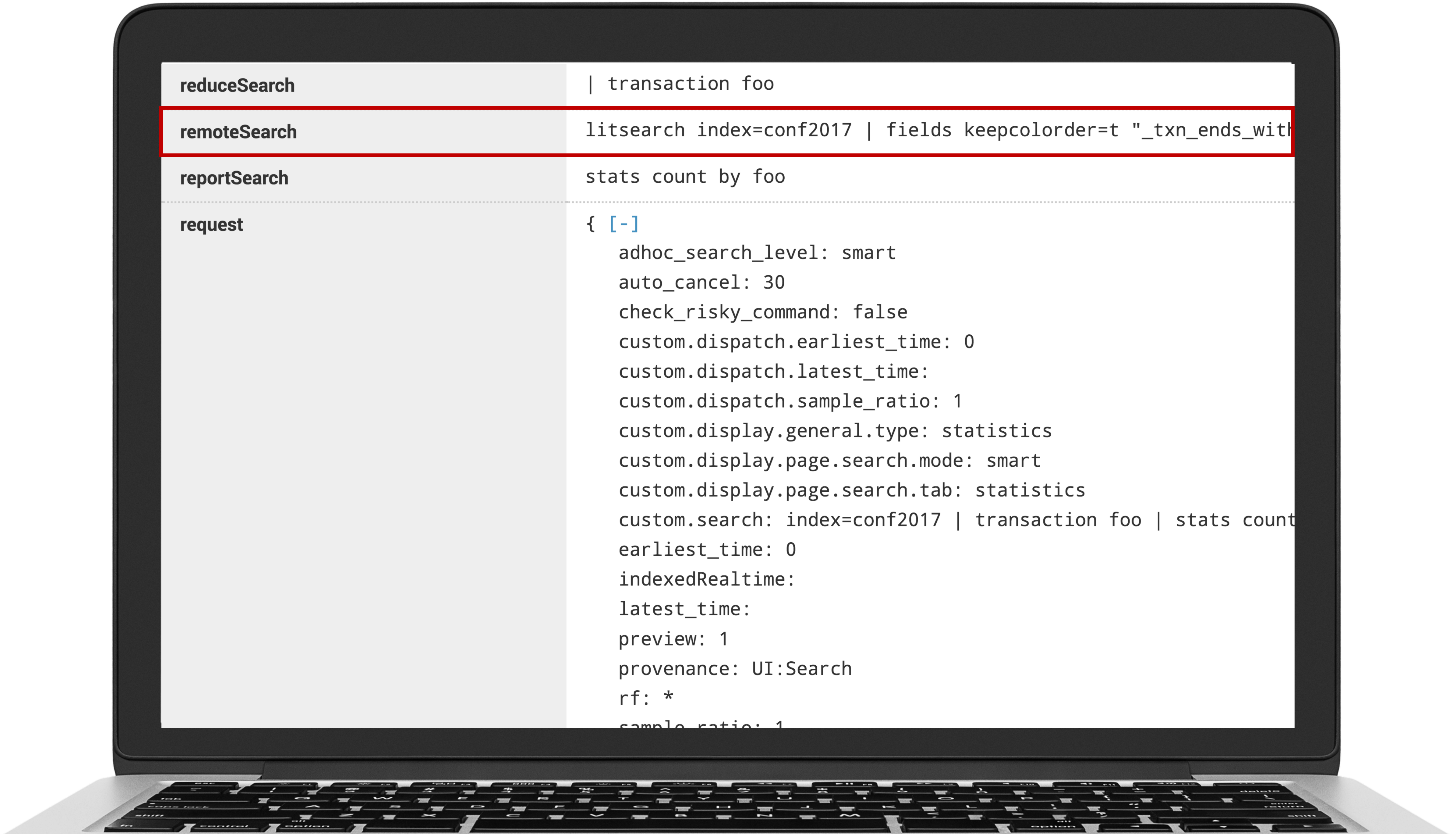
```
index=conf2017
| transaction foo
| stats count by foo
```

# Where Does it Run?

## transaction plus stats

```
index=conf2017
| transaction foo
| stats count by foo
```

- Splunk runs everything on the Indexer, until the first "slow" command forces otherwise.
- Everything trailing that command, will be forced to run on the Search Head.
- `transactions` and `joins` are examples of commands which would trigger this behavior.


reduceSearch
| transaction foo
remoteSearch
litsearch index=conf2017 | fields keepcolorder=t "_txn_ends_wit
reportSearch
stats count by foo
request
{ [-]
adhoc_search_level: smart
auto_cancel: 30
check_risky_command: false
custom.dispatch.earliest_time: 0
custom.dispatch.latest_time:
custom.dispatch.sample_ratio: 1
custom.display.general.type: statistics
custom.display.page.search.mode: smart
custom.display.page.search.tab: statistics
custom.search: index=conf2017 | transaction foo | stats count
earliest_time: 0
indexedRealtime:
latest_time:
preview: 1
provenance: UI:Search
rf: *

# Takeaways

You're all wizards now!

1. Leverage `stats` and `eval` over `transactions` whenever possible.
2. Choose trailing wildcards over leading in queries that require such functionality.
3. Look into indexing important fields who shares values with other fields.
4. Move slow commands as far right into the query string as possible.

# No Magic

Just Splunk

Don't forget to rate this session in the .conf2017 mobile app

splunk> .conf2017

# Q&A