RING8
SCALABLE STORAGE

splunk> SCALITY

WHITE PAPER

Using Scality RING to Store Splunk Data

August 2021

RING8
SCALABLE STORAGE

Splunk background

Splunk is a software solution for monitoring and searching machine-generated data via a web interface. Generally the way it works is data is slurped and indexed. Traditionally it's categorized into Hot, Warm, Cold, Frozen. Hot data is typically on SSD, Warm and Cold is somewhere between, most likely on a NAS. All of this is on mountable file systems, typically SSD or fast spinning drives. Frozen is historic and not indexed and typically resides on tape for ultra low cost, but also value is lost since it's not searchable and not easily retrievable. Our approach was to support an NFS based solution but the world has moved towards S3 and so has Splunk, thus enabling use of a scale-out on-prem object storage solution.

The feature set that enables S3 is called "SmartStore".

Splunk Value

Splunk license on amount of data ingested and using a on-premises object store has some clear advantages over a classic cloud object system where retrieval costs can mount up.

- Cheaper more easily accessed tiered storage could result in a customer keeping more data indexed, thus deriding more value from their splunk infrastructure.
- More space, and more perceived value, could mean customers will put more data into Splunk.
- A REST based scale-out approach is likely to be less operationally costly and more robust versus an NFS mount solution.
- Higher durability for data kept in an object store than traditional RAID based systems, meaning only one copy is kept.
- Seamless access to Frozen data with no need for multiple copies and tape bloat.
- Simpler tiering methodology with the removal of 'cold' tier.
- The new cache manager seamlessly moves data where it's needed.

What do Splunk say

The following is directly from the Splunk website:

https://docs.splunk.com/Documentation/Splunk/7.2.3/Indexer/AboutSmartStore

About SmartStore

SmartStore is an indexer capability that provides a way to use remote object stores, such as Amazon S3, to store indexed data.

As a deployment's data volume increases, demand for storage typically outpaces demand for compute resources. SmartStore allows you to manage your indexer storage and compute resources in a cost-effective manner by scaling those resources separately.

SmartStore introduces a remote storage tier and a cache manager. These features allow data to reside either locally on indexers or on the remote storage tier. Data movement between the indexer and the remote storage tier is managed by the cache manager, which resides on the indexer.

With SmartStore, you can reduce the indexer storage footprint to a minimum and choose I/O optimized compute resources. Most data resides on remote storage, while the indexer maintains a local cache that contains a minimal amount of data: hot buckets, copies of warm buckets participating in active or recent searches, and bucket metadata.

You can enable SmartStore for all indexes or for a subset of indexes.

SmartStore advantages

SmartStore offers several advantages to the deployment's indexing tier:

- Reduced storage cost. Your deployment can take advantage of the economy of remote object stores, instead of relying on costly local storage.
- Access to high availability and data resiliency features available through remote object stores.

- The ability to scale compute and storage resources separately, thus ensuring that you use resources efficiently.
- Simple and flexible configuration with per-index settings.

SmartStore offers additional advantages specific to deployments of indexer clusters:

- Fast recovery from peer failure and fast data rebalancing, requiring only metadata fixups for warm data.
- Lower overall storage requirements, as the system maintains only a single permanent copy of each warm bucket.
- Full recovery of warm buckets even when the number of peer nodes that goes down is greater than or equal to the replication factor.
- A bootstrapping capability that allows a new cluster to inherit the data from an old cluster.
- Global size-based data retention.
- Simplified upgrades.

An intelligent cache manager ensures that, for most search use cases, SmartStore provides similar performance to local storage configurations.

Choosing SmartStore

While SmartStore-enabled indexes can significantly decrease storage and management costs under the right circumstances, there are also times when you might find it preferable to continue to rely on local storage.

When to consider moving to SmartStore

SmartStore can help you to achieve significant costs savings for medium to large scale deployments. In particular, consider enabling SmartStore under these circumstances:

- As the amount of data in local storage continues to grow. While local storage costs might not be a significant issue for a small deployment, you should reconsider your use of local storage as your deployment

scales over time.

- If you are using indexer clusters to take advantage of features such as data recovery and disaster recovery. Through SmartStore, you can achieve these aims through the native capabilities of the remote store, without the need to store large amounts of redundant data on local storage.
- If you are using indexer clusters and you find that considerable amounts of your time and your compute resources are devoted to managing the cluster. Through SmartStore, you can eliminate much of the cluster management overhead. In particular, you can greatly reduce the scale of time-consuming activities such as offlining peer nodes, data rebalancing, and bucket fixup, because most of the data no longer resides on the peer nodes.
- When most searches are over recent data.

When not to move to SmartStore

There are a few situations where local storage might be a better choice:

- If you have a small deployment, with limited amounts of stored data, the advantages of SmartStore might not compensate for the costs of setting up and maintaining a remote store.
- If you have frequent need to run rare searches, SmartStore might not be appropriate for your purposes, as rare searches can require the indexer to copy large amounts of data from remote to local storage, causing a performance impact. This is particularly the case with searches that cover long timespans. If, however, the searches are across recent data and thus the necessary buckets are already in the cache, then there is no performance impact.
- If you run frequent long lookback searches, you might need to increase your cache size or continue to rely on local storage.

Current restrictions on SmartStore use

At this time, SmartStore support requires that your indexing tier conform to certain restrictions:

- No use of report acceleration or data model acceleration summaries. Because of this restriction, SmartStore is currently not compatible with any app, such as Splunk Enterprise Security, that uses these summaries.
- Available for indexer clusters only. SmartStore is not available for standalone indexers except for limited testing purposes.
- All nodes (peer nodes, master node, and search heads) in the indexer cluster must be running the same version of Splunk Enterprise, down to the maintenance level.
- Replication factor and search factor must be equal (for example, 3/3 or 2/2).
- The home path and cold path of each index must point to the same partition.
- Certain other indexes.conf settings are restricted with SmartStore.
- SmartStore-enabled index cannot be converted to non-SmartStore.

Splunk design

Customers will continue to maintain high speed storage for HOT data with Scality servicing Warm or Frozen buckets with S3 RING (and possibly Zenko). Note that “cold” is removed with SmartStore. The new cache manager feature will seamlessly move data back and forth between the high speed indexed environment and the longer term object storage.

As the Splunk indexer indexes Log data, it creates a number of files. These files contain two types of data:

- The raw data in compressed form (rawdata)
- Indexes that point to the raw data, plus some metadata files (index files)

Together, these files constitute the Splunk Enterprise index. The files reside in sets of directories organized by age. Some directories contain newly indexed data; others contain previously indexed data.

The number of such directories can grow quite large, depending on how much data you're indexing. Splunk takes care of tiering the data in and out of hot/warm/frozen buckets, and there are various settings to guide it into working the way you want.

How data ages

Each of the index directories is known as a bucket (not an S3 bucket).

To summarize so far:

- An "index" contains compressed raw data and associated index files.
- An index resides across many age-designated index directories.
- An index directory is a bucket.

A bucket moves through several stages as it ages:

- hot
- warm
- cold (not used with Smartstore)
- frozen

As buckets age, they "roll" from one stage to the next. Newly indexed data goes into a hot bucket, which is a bucket that's both searchable and actively being written to. After the hot bucket reaches a certain size, it becomes a warm bucket ("rolls to warm"), and a new hot bucket is created. Warm buckets are searchable, but are not actively written to. There are many warm buckets.

Traditionally once the indexer has created some maximum number of Warm buckets, it begins to roll the Warm buckets to Cold based on their age (unless it's Smartstore). The oldest warm bucket rolls to Cold. Buckets continue to roll to Cold as they age in this manner. After a set period of time, Cold buckets roll to Frozen, at which point they are either archived or deleted. With Smartstore, they roll from Warm to Frozen where they can be kept as desired. Splunk have realized with newer technologies it made sense to collapse Warm and Cold into the same space. Editing attributes in files like indexes.conf, allows you to specify the bucket aging policy, which determines when a bucket moves from one stage to the next.

Here are the stages that buckets age through:

Bucket Stage	Description	Searchable
Hot	Contains newly indexed data. Open for writing. One or more hot buckets for each index.	Yes
Warm	Data rolled from hot. There are many warm buckets.	Yes
Frozen	Data rolled from cold. The indexer deletes frozen data by default, but you can also archive it. Archived data can later be thawed.	No

The collection of buckets in a particular stage is sometimes referred to as a database or "db": the "hot db", the "warm db", etc. These are not to be confused with S3 buckets, although there is a loose correlation as a storage location.

Buckets and SmartStore - from Splunk

With SmartStore indexes, as with non-SmartStore indexes, hot buckets are built in the indexer's local storage cache. However, with SmartStore indexes, when a bucket rolls from hot to warm, a copy of the bucket is uploaded to remote storage. The remote copy then becomes the master copy of the bucket.

Eventually, the cache manager evicts the local bucket copy from the cache. When the indexer needs to search a warm bucket for which it doesn't have a local copy, the cache manager fetches a copy from remote storage and places it in the local cache.

The remote storage has a copy of every warm bucket.

Each indexer's local cache contains several types of data:

- Hot buckets. Hot buckets are created in local storage. They continue to reside solely on the indexer until they roll to warm.
- Copies of warm buckets that are currently participating in searches.
- Copies of recently created or recently searched warm buckets. The indexer maintains a cache of warm buckets, to minimize the need to fetch the same buckets from remote storage repeatedly.

- Metadata for remote buckets. The indexer maintains a small amount of information about each bucket in remote storage.

The buckets of SmartStore indexes ordinarily have just two active states: hot and warm. The cold state, which is used with non-SmartStore indexes to distinguish older data eligible for moving to cheap storage, is not necessary with SmartStore because warm buckets already reside on inexpensive remote storage. Buckets roll directly from warm to frozen. As with non-SmartStore indexes, archived frozen buckets can be thawed to local storage.

II. How to configure

Get the software:

Start Splunk

First thing to do is start splunk, which then forces you to accept the licenses and it will setup config files, which you can then edit below and restart splunk.

To set the home shell copy this into /root/bash_profile:

```
export SPLUNK_HOME=/opt/splunk
export PATH=$SPLUNK_HOME/bin:$PATH
```

Re-login to get the profile and run:

```
# splunk start
```

Config files

There's three main config files:

http://docs.splunk.com/Documentation/Splunk/7.2.0/Indexer/ConfigureSmartStore

Default files, which should not be edited directly are in /opt/splunk/etc/system/default/ and are overridden by those below.

```
/opt/splunk/etc/system/local/indexes.conf
/opt/splunk/etc/system/local/server.conf
/opt/splunk/etc/system/local/limits.conf
```

indexes.conf

Note this configuration will put ALL indexes into SmartStore. To be specific you create stanzas such as [scality] and for each one you can define specific overrides.

```
# cat /opt/splunk/etc/system/local/indexes.conf

[default]

frozenTimePeriodInSecs = 86400

maxGlobalDataSizeMB = 100

# Configure all indexes to use the SmartStore remote
volume called

# “remote_store”.

# Note: If you want only some of your indexes to use
SmartStore,

# place this setting under the individual stanzas for each
of the

# SmartStore indexes, rather than here.

remotePath = volume:remote_store/$_index_name

# Configure the remote volume

[volume:remote_store]

storageType = remote

# On the next line, the path attribute points to the
remote storage location

# where indexes reside. Each SmartStore index resides
directly below the location

# specified by the path attribute. The <scheme> identifies
a supported remote

# storage system type, such as S3. The <remote-location-
specifier> is a

# string specific to the remote storage system that
specifies the location

# of the indexes inside the remote system.

# This is an S3 example: “path = s3://mybucket/some/path”.

path = s3://splunk-db

remote.s3.access_key = access_key_here

remote.s3.secret_key = secret_key_here

remote.s3.list_objects_version = v1

remote.s3.signature_version = v2
```

```
remote.s3.auth_region =

remote.s3.supports_versioning = true

remote.s3.endpoint = http://10.200.2.35:8080
```

server.conf

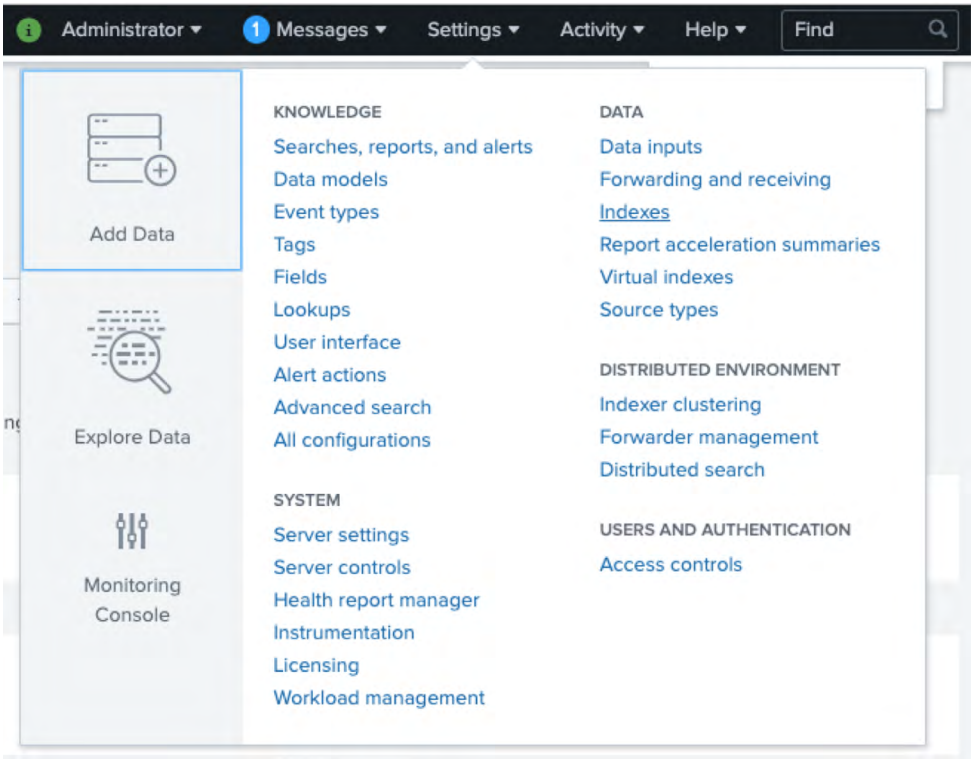
Changes I made to the default file are highlighted

```
# cat /opt/splunk/etc/system/local/server.conf

[general]

serverName = splunk-server

pass4SymmKey =
$7$5IcjPj9dPvxyQovHjq6B5qk96N1Rmh9w7cHnZ55ANDhw7vlTO/rElw==

sessionTimeout = 1d

[sslConfig]

sslPassword = $7$2lqIguVgzHUMzAoSMe5ZYQ6twduVp6W/
NRN5Yiiu9Z01tUBWDFqQLg==

[lmpool:auto_generated_pool_download-trial]

description = auto_generated_pool_download-trial

quota = MAX

slaves = *

stack_id = download-trial

[lmpool:auto_generated_pool_forwarder]

description = auto_generated_pool_forwarder

quota = MAX

slaves = *

stack_id = forwarder

[lmpool:auto_generated_pool_free]

description = auto_generated_pool_free

quota = MAX

slaves = *

stack_id = free

[cachemanager]

max_cache_size = 50
```

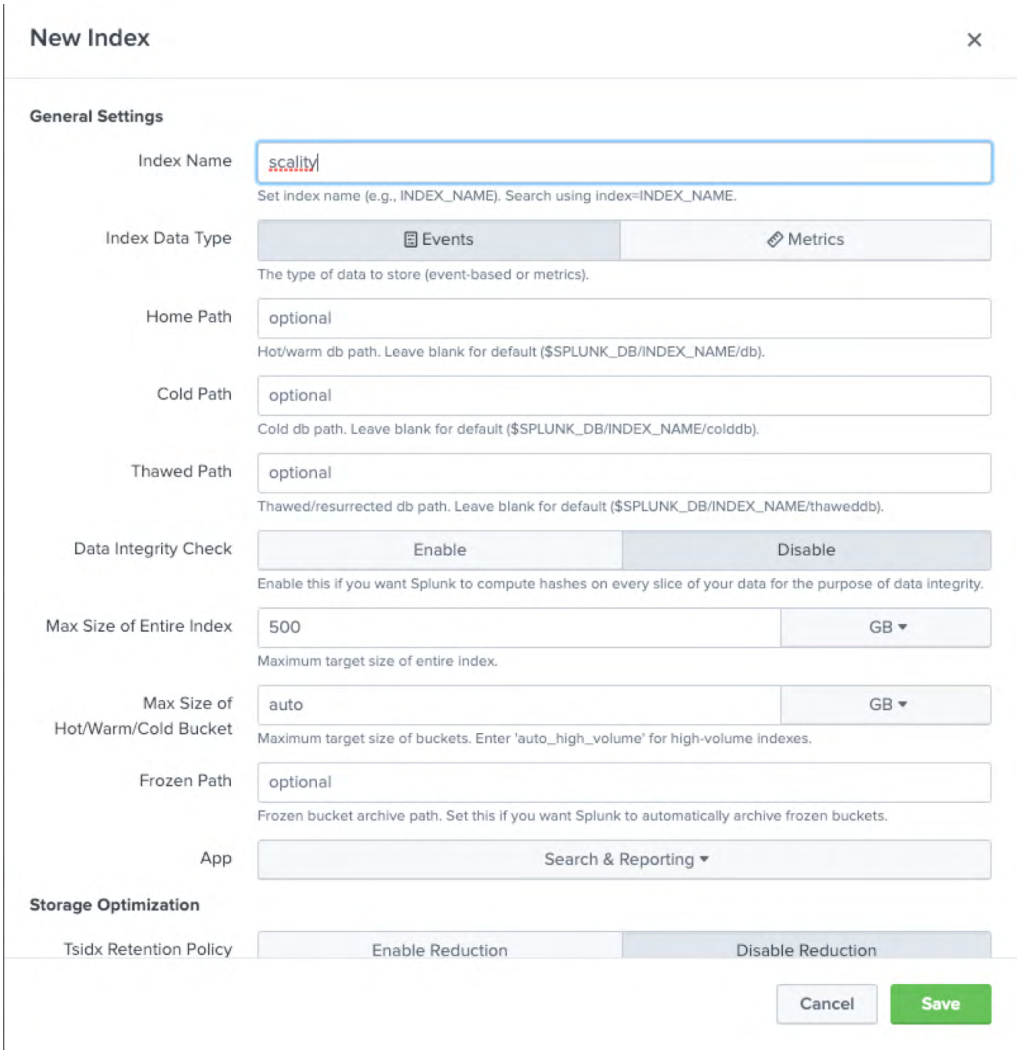
At RING 7.4.2 we can support v2 of list_objects_version. Note I also chose v2 signature to reduce overhead.

Default index

The default index is main, so we need to create a “scality” index on the splunk server. Under settings/indexes as below:



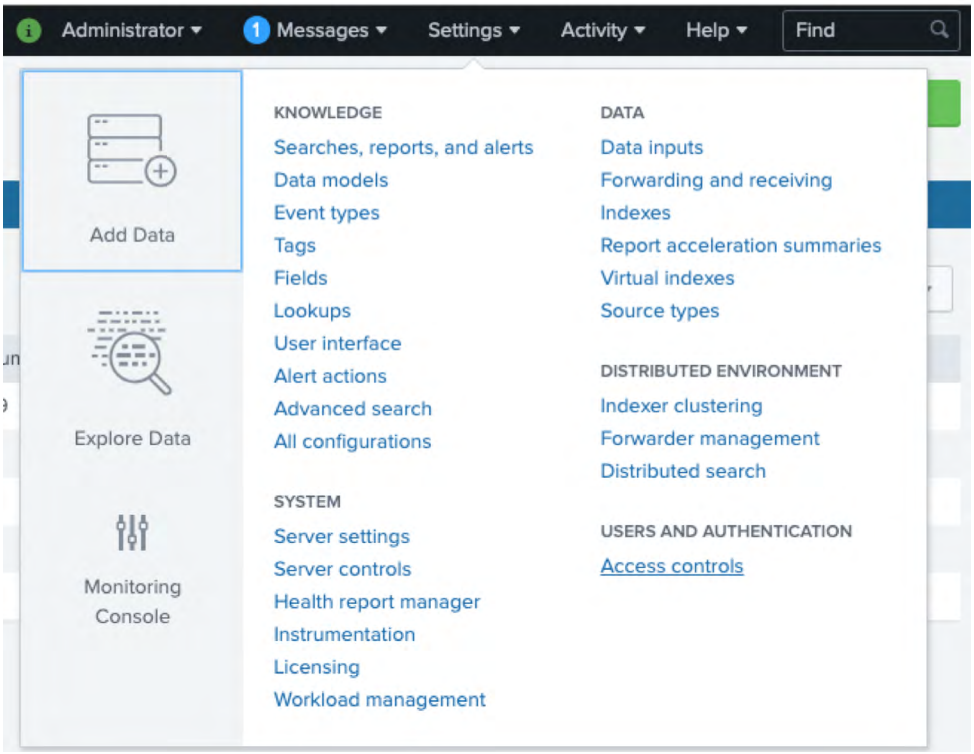
Create a new index



Leaving all the rest of the information default, or if needed reduce the sizes of Index and Max size.

This new index will not be the default index for admin, so we'll want to change that.

Go to "access controls", edit "roles" and then "admin" user. Further down the resulting page you can add the new 'scality' index and remove the 'main' index. This makes searching easier because you don't need to specify "index=scality" on the search line.



Sending data with the Universal Forwarder

To ingest some data into the Splunk indexes, install the universal installer on a source system and configure it to send the data into the spunk node.

Note: this KB article may prove useful:

https://answers.splunk.com/answers/95503/how-to-setup-universal-forwarder-on-linux.html

On the source system, download the universal forwarder, and configure:

```
/opt/splunkforwarder/bin/splunk add forward-server [splunk-server]:9997

/opt/splunkforwarder/bin/splunk list forward-server

Active forwards:

  [splunk-server]:9997

Configured but inactive forwards:

  None
```

input.conf

Sample configuration if the source system is a storage server.

```
# cat /opt/splunkforwarder/etc/system/local/inputs.conf
[default]
host = source-host
#[monitor:///var/log]
#disabled = false
#index = scality
#sourcetype = syslog
[monitor:///var/log/scality-sproxyd.log]
_TCP_ROUTING = *
index = scality
sourcetype = syslog
[monitor:///var/log/s3/scality-sproxyd/logs/sproxyd-*.log]
_TCP_ROUTING = *
index = scality
sourcetype = syslog
[monitor:///var/log/scality-biziod.log]
_TCP_ROUTING = *
index = scality
sourcetype = syslog
[monitor:////var/log/scality/node/chunkapi-node-*.log]
_TCP_ROUTING = *
index = scality
sourcetype = syslog
[monitor:////var/log/s3/scality-bucketd/logs/bucketd-0.log]
_TCP_ROUTING = *
index = scality
sourcetype = syslog
```

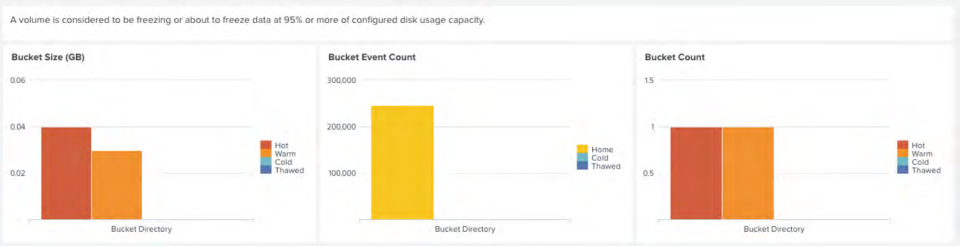
Then you need to restart the forwarder:

```
# cd /opt/splunkforwarder/bin
# splunk restart
Stopping splunkd...
```

```
Shutting down.  Please wait, as this may take a few
minutes.
                                                [  OK  ]
Stopping splunk helpers...
                                                [  OK  ]
Done.
Splunk> See your world.  Maybe wish you hadn't.
Checking prerequisites...
  Checking mgmt port [8099]: open
  Checking conf files for problems...
  Done
  Checking default conf files for edits...
  Validating installed files against hashes from '/opt/
splunkforwarder/splunkforwarder-7.2.0-8c86330ac18-linux-2.6-
x86_64-manifest'
File
All preliminary checks passed.
Starting splunk server daemon (splunkd)...
Done
                                                [  OK  ]
```

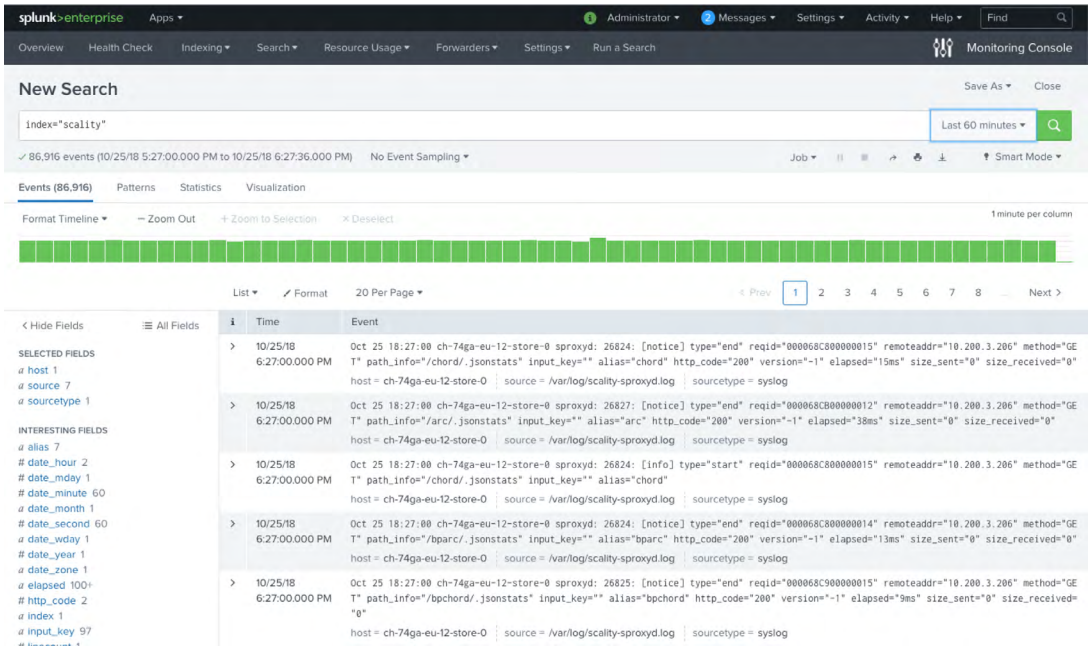
Example of indexing

As you can see below, this is an example of indexes moving from Hot to Warm



Search screen

Specifying the index of “scality” gives all results in the time-frame requested.



Debugging

Logs

One log to watch is the splunk log on the splunk server

```
/opt/splunk/var/log/splunk/splunkd.log
```

It can give some invaluable insight.