Riak Search Settings

Contents

Enabling Riak Search

Riak Search is enabled in the app.config file. Simply change the setting to "true" in Riak Search Config section (shown below).

Erlang

```
%% Riak Search Config
{riak_search, [
                %% To enable Search functionality set this
                {enabled, false}
               ]},
```

You will have to make this change on every node in your Riak cluster, and it will require you to shut down and restart the node for the changes to take effect. (You can use Riaknostic to check if Search is enabled on all your nodes.)

After you have made these changes, Riak Search will automatically start up when Riak is started.

Default Ports

By default, Riak Search uses the following ports:

- `8098` - Solr Interface
- `8099` - Riak Handoff
- `8087` - Protocol Buffers interface

Be sure to take the necessary security precautions to prevent exposing these ports to the outside world.

Merge Index Settings

These settings can be found in the Riak Search app.config file under the "merge_index" section.

- `data_root` - Set to the location where data files are written,relative to the Riak Search root directory.
- `buffer_rollover_size` - Maximum size of the in-memory buffer before it is transformed to a segment and written to disk. Higher numbers will result in faster indexing but more memory usage.
- `buffer_delayed_write_size` - Bytes to accumulate in the write-ahead log before flushing to disk.
- `buffer_delayed_write_ms` - Interval to flush write-ahead log to disk.
- `max_compact_segments` - The maximum number of segments to compact during a compaction. Smaller values will result in quicker compactions and a more balanced number of files in each partition, at the expense of more frequent compactions, and a higher likelihood of compacting the same data multiple times.
- `segment_query_read_ahead_size` - Size of the file read-ahead buffer, in bytes, to use when looking up results in a query.
- `segment_compact_read_ahead_size` - Size of the file read-ahead buffer, in bytes, to use when reading a segment for compaction.
- `segment_file_buffer_size` - Amount of segment compaction data to batch, in bytes, before writing to the file handle. This should be less than or equal to segment_delayed_write_size, otherwise that setting will have no effect.

- `segment_delayed_write_size` - Size of the delayed write buffer in bytes. Once this is exceeded, the compaction buffer is flushed to disk.
- `segment_delayed_write_ms` - Interval at which data will be written to a file during compaction.
- `segment_full_read_size` - Segment files below this size will be read into memory during a compaction for higher performance at the cost of more RAM usage. This setting plus*max_compact_segments* directly affects the maximum amount of RAM that a compaction can take.
- `segment_block_size` - Determines the block size across which a segment will calculate offsets and lookup information. Setting to a lower value will increase query performance, but will also lead to more RAM and disk usage.
- `segment_values_staging_size` - Maximum number of values to hold in memory before compressing the batch and adding to the output buffer.
- `segment_values_compression_threshold` - Since compression is more effective with a larger number of values, this is the number of values that must be present in a batch before the system compresses the batch.
- `segment_values_compression_level` - zlib compression level to use when compressing a batch of values.

These May Also Interest You

- Searching and Accessing
- Advanced Search Schema
- Advanced Search
- Search Indexing Reference
- Taste of Riak: Object Modeling with Erlang
- Taste of Riak: Object Modeling with Java