# HowToReindex

**Contents moving**

Please see the Solr Reference Guide, the "Reindexing" page. It is being re-worked to avoid trying to keep multiple copies of this process in sync. Solr 8.8 and later will contain the up-to-date information. Meanwhile, this is a summary.

## Reindexing in Solr

### Terminology

If you use Solr for any length of time, someone will eventually tell you that you have to reindex after making a change. It comes up over and over ... but what does that actually mean?

Changes to the schema will require a reindex, unless you only change query-time behavior. A very small subset of changes to solrconfig.xml also require a reindex, and for some changes, a reindex is recommended even when it's not required. For example when upgrading a major version, e.g. Solr7 to Solr8. Changing the schema is also a virtual certainty as your application changes, so you should be prepared to reindex.

⚠ In this context, "reindex" means **you must completely delete your old index**, removing all segments and index again from your system-of-record. **Si mply adding all your documents again to an existing index is not sufficient. Nor is using the UpgradeIndexTool.** There are technical reasons having to do with segment merging that make trying anything except starting with an empty index when the schema changes impractical.

⚠ Indexing (and reindexing) is not something that happens automatically, you must initiate the process.

It's reasonable to wonder why deleting the existing data and building it again is necessary. Here's why: *A Lucene index is a lossy abstraction designed for efficient search.* Your schema  definition for a field defines a transformation function y=f(x) where "y" is what's stored in the index and "x" is the original input and "f" is the transformation function. Lucene has never guaranteed to preserve "x", so if you change "f" (i.e. change your schema), calculating "y" cannot be guaranteed since there is no way to reconstruct "x" from "y".

Note: With SolrCloud, you can define a new schema and index the data to a *new* collection then, once the new collection satisfies your QA process, use the Collections API CREATEALIAS command to seamlessly switch the users to the new collection without interruption.

### Using Solr as a data source

Don't do this unless you have no other option. It is always preferable to preserve your original data in a system that's designed for that purpose. SolrCloud goes a long way towards high availability, but absolute data reliability in the face of any problem is difficult to achieve for any software, which is why it's always important to have backups.

⚠ Using Solr as a data source to build a new index is only possible if the index meets the requirements for Atomic Update.  Please see the newest reference guide for full details on exactly what that means.

If you have no other choice but to use a Solr index as the data source for another index, and you have stored every field except perhaps copyField destinations, you have a few possible options:

1. Use the REINDEXCOLLECTION if it is available in your version of Solr.
2. Use the export capability. This has some restrictions.
3. Use the dataimport handler with SolrEntityProcessor. NOTE: dataimport handler will be moved to a community-supported package in Solr 9.
4. Export the data using Solr queries, then reimport it after making sure it's in the correct format. This is not a trivial process. There is no process or program available from the Solr project for doing this. Here are some ideas:
   a. Write a custom script/program that fetches from one Solr collection and indexes to another. Please make sure you use CursorMark for fetching.
   b. Recent versions of Solr have added a new export capability – the /export handler. This might prove useful, but has some restrictions.
   c. Recent versions of Solr can use streaming expressions to accomplish this in some situations.
   d. Write a program that writes the data to the filesystem, then another program to index it. Solr maintains an API called SolrJ that can be used to to this in Java.

### Alternatives when a traditional reindex isn't possible

Sometimes the option of "do your indexing again from the system of record" is difficult. Perhaps the original data is very slow to access, or it may be difficult to get in the first place. There are creative ways to use Solr, one is outlined below. If you find yourself in a similar situation, you can always ask on the user's list for thoughts on how to proceed.

Above we said "don't use Solr itself as a datasource" ... but one way to deal with data availability problems is to set up a completely separate Solr install with a config designed for ONLY data storage, not searching.  The schema would have stored="true", indexed="false", and docValues="false" for all fields, and would only use basic types like int, long, date, uuid, and string. It would not have any copyFields. You would need to make sure that the separate Solr install receives all new documents and changes that are applied to your primary Solr installation.

This is the approach used by a large and very well-known library organization for their Solr installation, because getting access to the source databases for the individual entities within the organization is very difficult. This way they can reindex the online Solr at any time without having to get special permission from all those entities. When they index new content, it goes into both their primary installation as well as the installation configured for storage only.

In this scenario you may have to re-index *twice* in order to utilize that method. Once to your intermediate Solr server(s), then from there to your server(s) that you're using for search.

We emphasize again that this kind of setup should only be considered if you cannot re-index from your system-of-record since it will be an additional maintenance cost.

## How long does it take?

Under any circumstance, reindexing will take *at least* as long as the original indexing took.