Installation Guide

1 Download FSCrawler
2 Running as a Service on Windows
3 Upgrade FSCrawler
  3.1 Upgrade to 2.2
  3.2 Upgrade to 2.3
  3.3 Upgrade to 2.4
  3.4 Upgrade to 2.5
  3.5 Upgrade to 2.6
  3.6 Upgrade to 2.7
4 Getting Started
  4.1 Start FSCrawler
  4.2 Searching for docs
  4.3 Ignoring folders
5 Tutorial
  5.1 Prerequisites
  5.2 Install Elastic stack
  5.3 Start FSCrawler
  5.4 Create Index pattern
  5.5 Search for the CVs
  5.6 Adding new files
6 Crawler options
7 OCR integration
  7.1 OCR settings
  7.2 Disable/Enable OCR
  7.3 OCR Language
  7.4 OCR Path
  7.5 OCR Data Path
  7.6 OCR Output Type
  7.7 OCR PDF Strategy
8 Starting with a REST gateway
## REST service

- **20.1 FSCrawler status** ........................................... 75
- **20.2 Uploading a binary document** .......................... 76
- **20.3 Simulate Upload** ........................................... 78
- **20.4 Document ID** .............................................. 78
- **20.5 Additional tags** ............................................ 78
- **20.6 Specifying an elasticsearch index** ....................... 79
- **20.7 Enabling CORS** ............................................ 79
- **20.8 REST settings** ............................................. 80

## Building the project

- **21.1 Clone the project** ........................................ 81
- **21.2 Build the artifact** ....................................... 82
- **21.3 Integration tests** ........................................ 82
- **21.4 Check for vulnerabilities (CVE)** ....................... 84

## Writing documentation

**22 Release the project** ........................................ 87

## License

**23 Release the project** ........................................ 87

## Incompatible 3rd party library licenses

- **25 Incompatible 3rd party library licenses** ............... 91

## Special thanks

**26 Special thanks** ............................................. 93
Welcome to the FS Crawler for Elasticsearch.

This crawler helps to index binary documents such as PDF, Open Office, MS Office.

Main features:

- Local file system (or a mounted drive) crawling and index new files, update existing ones and removes old ones.
- Remote file system over SSH crawling.
- REST interface to let you “upload” your binary documents to elasticsearch.

Note: FS Crawler 2.7-SNAPSHOT is using Tika 1.24.1 and:

- Elasticsearch Rest Client 7.9.0 for Elasticsearch V7.
Download FSCrawler

Depending on your Elasticsearch cluster version, you can download FSCrawler 2.7 using the following links:

- fscrawler-es7-2.7-SNAPSHOT for Elasticsearch V7.
- fscrawler-es6-2.7-SNAPSHOT for Elasticsearch V6.

The filename ends with .zip.

**Warning:** This is a **SNAPSHOT** version. You can also download a **stable** version from Maven Central:

- fscrawler-es7-* for Elasticsearch V7.

The distribution contains:

```
$ tree
.
  LICENSE
  NOTICE
  README.md
  bin
    fscrawler
    fscrawler.bat
  lib
    ... All needed jars
```
Create a fscrawlerRunner.bat as:

```
set JAVA_HOME=c:\Program Files\Java\jdk1.8.0_144
set FS_JAVA_OPTS=-Xmx2g -Xms2g
/Elastic/fscrawler/bin/fscrawler.bat --config_dir /Elastic/fscrawler data >> /Elastic/logs/fscrawler.log 2>&1
```

Then use fscrawlerRunner.bat to create your windows service.
CHAPTER 3

Upgrade FSCrawler

It can happen that you need to upgrade a mapping or reindex an entire index before starting fscrawler after a version upgrade. Read carefully the following update instructions.

To update fscrawler, just download the new version, unzip it in another directory and launch it as usual. It will still pick up settings from the configuration directory. Of course, you need to stop first the existing running instances.

3.1 Upgrade to 2.2

- fscrawler comes with new default mappings for files. They have better defaults as they consume less disk space and CPU at index time. You should remove existing files in ~/.fscrawler/_default/_mappings before starting the new version so default mappings will be updated. If you modified manually mapping files, apply the modification you made on sample files.

- excludes is now set by default for new jobs to ["~*"]. In previous versions, any file or directory containing a ~ was excluded. Which means that if in your jobs, you are defining any exclusion rule, you need to add *~* if you want to get back the exact previous behavior.

- If you were indexing json or xml documents with the filename_as_id option set, we were previously removing the suffix of the file name, like indexing 1.json was indexed as 1. With this new version, we don’t remove anymore the suffix. So the _id for your document will be now 1.json.

3.2 Upgrade to 2.3

- fscrawler comes with new mapping for folders. The change is really tiny so you can skip this step if you wish. We basically removed name field in the folder mapping as it was unused.

- The way FSCrawler computes now path.virtual for docs has changed. It now includes the filename. Instead of /path/to you will now get /path/to/file.txt.

- The way FSCrawler computes now virtual for folders is now consistent with what you can see for folders.
• *path.encoded* in documents and *encoded* in folders have been removed as not needed by FSCrawler after all.

• *OCR integration* is now properly activated for PDF documents. This can be time, cpu and memory consuming though. You can disable explicitly it by setting *fs.pdf_ocr* to *false*.

All dates are now indexed in elasticsearch in UTC instead of without any time zone. For example, we were indexing previously a date like *2017-05-19T13:24:47.000* which was producing bad results when you were located in a time zone other than UTC. It’s now indexed as *2017-05-19T13:24:47.000+0000*.

In order to be compatible with the coming 6.0 elasticsearch version, we need to get rid of types as only one type per index is still supported. Which means that we now create index named *job_name* and *job_name_folder* instead of one index *job_name* with two types *doc* and *folder*. If you are upgrading from FSCrawler 2.2, it requires that you reindex your existing data either by deleting the old index and running again FSCrawler or by using the *reindex* API as follows:

```json
# Create folder index job_name_folder based on existing folder data
POST _reindex
{
   "source": {
      "index": "job_name",
      "type": "folder"
   },
   "dest": {
      "index": "job_name_folder"
   }
}
# Remove old folder data from job_name index
POST job_name/folder/_delete_by_query
{
   "query": {
      "match_all": {}
   }
}
```

Note that you will need first to create the right settings and mappings so you can then run the reindex job. You can do that by launching *bin/fscrawler job_name --loop 0*.

Better, you can run *bin/fscrawler job_name --upgrade* and let FSCrawler do all that for you. Note that this can take a loooong time.

Also please be aware that some APIs used by the upgrade action are only available from elasticsearch 2.3 (reindex) or elasticsearch 5.0 (delete by query). If you are running an older version than 5.0 you need first to upgrade to elasticsearch.

This procedure only applies if you did not set previously *elasticsearch.type* setting (default value was *doc*). If you did, then you also need to reindex the existing documents to the default *_doc* type as per elasticsearch 6.x (or *doc* for 5.x series):

```json
# Copy old type doc to the default doc type
POST _reindex
{
   "source": {
      "index": "job_name",
      "type": "your_type_here"
   },
   "dest": {
      "index": "job_name",
      "type": "_doc"
   }
}
```

(continues on next page)
# Remove old type data from job_name index
POST job_name/your_type_here/_delete_by_query
{
    "query": {
        "match_all": {}
    }
}

But note that this last step can take a very loooong time and will generate a lot of IO on your disk. It might be easier in such case to restart fscrawler from scratch.

- As seen in the previous point, we now have 2 indices instead of a single one. Which means that `elasticsearch.index` setting has been split to `elasticsearch.index` and `elasticsearch.index_folder`. By default, it’s set to the crawler name and the crawler name plus `_folder`. Note that the `upgrade` feature performs that change for you.
- `fscrawler` has removed now mapping files `doc.json` and `folder.json`. Mapping for doc is merged within `_settings.json` file and folder mapping is now part of `_settings_folder.json`. Which means you can remove old files to avoid confusion. You can simply remove existing files in `~/.fscrawler/_default` before starting the new version so default files will be created again.

## 3.3 Upgrade to 2.4

- No specific step needed. Just note that mapping changed as we support more metadata. Might be useful to run similar steps as for 2.2 upgrade.

## 3.4 Upgrade to 2.5

- A bug was causing a lot of data going over the wire each time FSCrawler was running. To fix this issue, we changed the default mapping and we set `store: true` on field `file.filename`. If this field is not stored and `remove_deleted` is `true` (default), FSCrawler will fail while crawling your documents. You need to create the new mapping accordingly and reindex your existing data either by deleting the old index and running again FSCrawler or by using the `reindex` API as follows:

  # Backup old index data
  POST _reindex
  {
    "source": {
      "index": "job_name"
    },
    "dest": {
      "index": "job_name_backup"
    }
  }

  # Remove job_name index
  DELETE job_name

  Restart FSCrawler with the following command. It will just create the right mapping again.

  $ bin/fscrawler job_name --loop 0

  Then restore old data:
The default mapping changed for FSCrawler for `meta.raw.*` fields. Might be better to reindex your data.

- The `excludes` parameter is also used for directory names. But this new implementation also brings a breaking change if you were using `excludes` previously. In the previous implementation, the regular expression was only applied to the filename. It’s now applied to the full virtual path name.

For example if you have a `/tmp` dir as follows:

```
/tmp
  └── folder
      ├── foo.txt
      └── bar.txt
```

Previously excluding `foo.txt` was excluding the virtual file `/folder/foo.txt`. If you still want to exclude any file named `foo.txt` whatever its directory you now need to specify `*/foo.txt`:

```
{
    "name" : "test",
    "fs": {
        "excludes": ["*/foo.txt"
    }
}
```

For more information, read *Includes and excludes*.

- For new indices, FSCrawler now uses `_doc` as the default type name for clusters running elasticsearch 6.x or superior.

### 3.5 Upgrade to 2.6

- FSCrawler comes now with multiple distributions, depending on the elasticsearch cluster you’re targeting to run.

- `elasticsearch.nodes` settings using `host`, `port` or `scheme` have been replaced by an easier notation using `url` setting like `http://127.0.0.1:9200`. You will need to modify your existing settings and use the new notation if warned.

### 3.6 Upgrade to 2.7

- FSCrawler comes now with an elasticsearch 7.x implementation.
• FSCrawler also supports YAML format for jobs (default).

• The Elasticsearch 6.x implementation does not support Elasticsearch versions prior to 6.7. If you are using an older version, it’s better to upgrade or you need to “hack” the distribution and replace all Elasticsearch/Lucene jars to the 6.6 version.

• FSCrawler does not follow symbolic links anymore. You need to set explicitly `fs.follow_symlink` to `true` if you wish revert to the previous behavior.

• The mapping for Elasticsearch 6.x can not contain anymore the type name.

• We removed the Elasticsearch V5 compatibility as it’s not maintained anymore by elastic.
Getting Started

You need to have at least **Java 1.8** and have properly configured `JAVA_HOME` to point to your Java installation directory. For example on MacOS you can define in your `~/.bash_profile` file:

```
export JAVA_HOME=`/usr/libexec/java_home -v 1.8`
```

### 4.1 Start FSCrawler

Start FSCrawler with:

```
$ bin/fscrawler job_name
```

FSCrawler will read a local file (default to `~/.fscrawler/{job_name}/_settings.yaml`). If the file does not exist, FSCrawler will propose to create your first job.

```
$ bin/fscrawler job_name
18:28:58,177 INFO  [f.p.e.c.f.FsCrawler] Do you want to create it (Y/N)?
y
18:29:05,711 INFO  [f.p.e.c.f.FsCrawler] Settings have been created in [~/.fscrawler/
˓→job_name/_settings.yaml]. Please review and edit before relaunch
```

Create a directory named `/tmp/es` or `c:\tmp\es`, add some files you want to index in it and start again:

```
$ bin/fscrawler --config_dir ./test job_name
18:30:34,330 INFO  [f.p.e.c.f.FsCrawlerImpl] Starting FS crawler
18:30:34,332 INFO  [f.p.e.c.f.FsCrawlerImpl] FS crawler started in watch mode. It
˓→will run unless you stop it with CTRL+C.
18:30:34,682 INFO  [f.p.e.c.f.FsCrawlerImpl] FS crawler started for [job_name] for [/‐
˓→tmp/es] every [15m]
```

If you did not create the directory, FSCrawler will complain until you fix it:
You can also run FSCrawler without arguments. It will give you the list of existing jobs and will allow you to choose one:

```
$ bin/fscrawler
```

### 4.2 Searching for docs

This is a common use case in elasticsearch, we want to search for something! ;-)  

```
GET docs/doc/_search
{
    "query" : {
        "query_string": {
            "query": "I am searching for something !"
        }
    }
}
```

See Search examples for more examples.

### 4.3 Ignoring folders

If you would like to ignore some folders to be scanned, just add a `.fscrawlerignore` file in it. The folder content and all sub folders will be ignored.

For more information, read Includes and excludes.
This tutorial use case is:
Search for the resumes (PDF or Word file which resides in One drive or local) and search for anything in the content using Kibana. For example location worked or the previous company, etc.

5.1 Prerequisites

- Java 1.8+ must be installed
- JAVA_HOME must be defined

5.2 Install Elastic stack

- Download Elasticsearch
- Download Kibana
- Start Elasticsearch server
- Start Kibana server
- Check that Kibana is running by opening http://localhost:5601

5.3 Start FSCrawler

- Download FSCrawler. See Download FSCrawler.
- Open a terminal and navigate to the fscrawler folder.
- Type:
# On Linux/Mac

bin/fscrawler resumes

# On Windows

.\bin\fscrawler resumes

- It will ask “Do you want to create it (Y/N)?”. Answer Y.

- Go to the FSCrawler configuration folder to edit the job configuration. The FSCrawler configuration folder named .fscrawler is by default in the user home directory, like C:\Users\myuser on Windows platform or ~ on Linux/MacOS. In this folder, you will find another folder named resumes. Enter this folder:

```bash
# On Linux/Mac
cd ~/.fscrawler/resumes
# On Windows
cd C:\Users\myuser\resumes
```

- Edit the _settings.yaml file which is in this folder and change the url value to your folder which contains the resumes you would like to index:

```yaml
---
nname: "resumes"
fs:
  # On Linux
  url: "/path/to/resumes"
  # On Windows
  url: "c:\path\to\resumes"
```

- Start again FSCrawler:

```bash
# On Linux/Mac
bin/fscrawler resumes
# On Windows
.\bin\fscrawler resumes
```

FSCrawler should index all the documents inside your directory.

**Note:** If you want to start again reindexing from scratch instead of monitoring the changes, stop FSCrawler, restart it with the --restart option:

```bash
# On Linux/Mac
bin/fscrawler resumes --restart
# On Windows
.\bin\fscrawler resumes --restart
```

## 5.4 Create Index pattern

- Open Kibana
- Go to the Management page
- Open the Index Patterns page under Kibana settings.
- Click on Create index pattern
• Type `resumes` in the input box. Don’t forget to remove the star * that is automatically added by default by Kibana.

Choose the date field you’d like to use if you want to be able to filter documents by date. Use `file.created` field if you want to filter by file creation date, `file.last_modified` to filter by last modification date or `file.indexing_date` if you want to filter by the date when the document has been indexed into Elasticsearch. You can also choose not to use the time filter (the last option).

• Click on “Create index pattern”. You should see something like:

5.4. Create Index pattern
5.5 Search for the CVs

- Open **Kibana**

- Go to the **Discover page**

- Depending on the date you selected in the **Create Index pattern** step, you should see something similar to the following image. If you don’t see it, you probably have to adjust the time picker to make sure you are looking at the right period of time.
You can select the fields you’d like to display in the result page, such as `content`, `file.filename`, `file.extension`, `file.url`, `file.filesize`, etc.
• Of course, you can search for content, like collaborateurs here and see the highlighted content.
5.6 Adding new files

Just copy new files in the resumes folder. It could take up to 15 minutes for FSCrawler to detect the change. This is the default value for update_rate option. You can also change this value. See Update rate.

**Note:** On some OS, moving files won’t touch the modified date and the “new” files won’t be detected. It’s then better probably to copy the files instead.

You might have to “touch” the files like:

```bash
touch /path/to/resumes/CV2.pdf
```

Just hit the Kibana refresh button and see the changes.
By default, FSCrawler will read your file from /tmp/es every 15 minutes. You can change those settings by modifying ~/.fscrawler/{job_name}/_settings.yaml file where {job_name} is the name of the job you just created.

```yaml
name: "job_name"
fs:
  url: "/path/to/data/dir"
  update_rate: "15m"
```

You can change also update_rate to watch more or less frequently for changes.

If you just want FSCrawler to run once and exit, run it with --loop option:

```bash
$ bin/fscrawler job_name --loop 1
```

18:47:37,487 INFO [f.p.e.c.f.FsCrawlerImpl] Starting FS crawler

... 18:47:37,855 INFO [f.p.e.c.f.FsCrawlerImpl] FS crawler is stopping after 1 run
18:47:37,959 INFO [f.p.e.c.f.FsCrawlerImpl] FS crawler [job_name] stopped

If you have already ran FSCrawler and want to restart (which means reindex existing documents), use the --restart option:

```bash
$ bin/fscrawler job_name --loop 1 --restart
```

You will find more information about settings in the following sections:

- **CLI options**
- **Local FS settings**
- **SSH settings**
- **Elasticsearch settings**
New in version 2.3.

To deal with images containing text, just install Tesseract. Tesseract will be auto-detected by Tika or you can explicitly set the path to tesseract binary. Then add an image (png, jpg, ...) into your Fscrawler Root directory. After the next index update, the text will be indexed and placed in “_source.content”.

### 7.1 OCR settings

Here is a list of OCR settings (under fs.ocr prefix):

<table>
<thead>
<tr>
<th>Name</th>
<th>Default value</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs.ocr.enabled</td>
<td>true</td>
<td>Disable/Enable OCR</td>
</tr>
<tr>
<td>fs.ocr.language</td>
<td>&quot;eng&quot;</td>
<td>OCR Language</td>
</tr>
<tr>
<td>fs.ocr.path</td>
<td>null</td>
<td>OCR Path</td>
</tr>
<tr>
<td>fs.ocr.data_path</td>
<td>null</td>
<td>OCR Data Path</td>
</tr>
<tr>
<td>fs.ocr.output_type</td>
<td>txt</td>
<td>OCR Output Type</td>
</tr>
<tr>
<td>fs.ocr.pdf_strategy</td>
<td>ocr_and_text</td>
<td>OCR PDF Strategy</td>
</tr>
</tbody>
</table>

### 7.2 Disable/Enable OCR

New in version 2.7.

You can completely disable using OCR by setting fs.ocr.enabled property in your ~/.fscrawler/test/_settings.yaml file:

```yaml
name: "test"
fs:
  url: "/path/to/data/dir"
```

(continues on next page)
By default, OCR is activated if tesseract can be found on your system.

### 7.3 OCR Language

If you have installed a Tesseract Language pack, you can use it when parsing your documents by setting `fs.ocr.language` property in your `~/.fscrawler/test/_settings.yaml` file:

```yaml
name: "test"
fs:
    url: "/path/to/data/dir"
ocr:
    language: "eng"
```

**Note:** You can define multiple languages by using + sign as a separator:

```yaml
name: "test"
fs:
    url: "/path/to/data/dir"
ocr:
    language: "eng+fas+fra"
```

### 7.4 OCR Path

If your Tesseract application is not available in default system PATH, you can define the path to use by setting `fs.ocr.path` property in your `~/.fscrawler/test/_settings.yaml` file:

```yaml
name: "test"
fs:
    url: "/path/to/data/dir"
ocr:
    path: "/path/to/tesseract/executable"
```

When you set it, it's highly recommended to set the **OCR Data Path**.

### 7.5 OCR Data Path

Set the path to the ‘tessdata’ folder, which contains language files and config files if Tesseract can not be automatically detected. You can define the path to use by setting `fs.ocr.data_path` property in your `~/.fscrawler/test/_settings.yaml` file:

```yaml
name: "test"
fs:
    url: "/path/to/data/dir"
ocr:
    data_path: "/path/to/tesseract"
7.6 OCR Output Type

New in version 2.5.

Set the output type from ocr process. `fs.ocr.output_type` property can be defined to `txt` or `hocr` in your `~/.fscrawler/test/_settings.yaml` file:

```yaml
name: "test"
fs:
  url: "/path/to/data/dir"
  ocr:
    output_type: "hocr"
```

**Note:** When omitted, `txt` value is used.

7.7 OCR PDF Strategy

By default, FSCrawler will also try to extract also images from your PDF documents and run OCR on them. This can be a CPU intensive operation. If you don’t mean to run OCR on PDF but only on images, you can set `fs.ocr.pdf_strategy` to "no_ocr" or to "auto":

```yaml
name: "test"
fs:
  ocr:
    pdf_strategy: "auto"
```

Supported strategies are:

- **auto**: No OCR is performed on PDF documents if there is more than 10 characters extracted. See PDFParser OCR Options.
- **no_ocr**: No OCR is performed on PDF documents. OCR might be performed on images though if OCR is not disabled. See Disable/Enable OCR.
- **ocr_only**: Only OCR is performed.
- **ocr_and_text**: OCR and text extraction is performed.

**Note:** When omitted, `ocr_and_text` value is used. If you have performance issues, it’s worth using the `auto` option instead as only documents with barely no text will go through the OCR process.
New in version 2.2.

FSCrawler can be a nice gateway to elasticsearch if you want to upload binary documents and index them into elasticsearch without writing by yourself all the code to extract data and communicate with elasticsearch.

To start FSCrawler with the REST service, use the --rest option. A good idea is also to combine it with --loop 0 so you won’t index local files but only listen to incoming REST requests:

```
$ bin/fscrawler job_name --loop 0 --rest
```

Check the service is working with:

```
curl http://127.0.0.1:8080/fscrawler/
```

It will give you back a JSON document.

Then you can start uploading your binary files:

```
echo "This is my text" > test.txt
curl -F "file=@test.txt" "http://127.0.0.1:8080/fscrawler/_upload"
```

It will index the file into elasticsearch and will give you back the elasticsearch URL for the created document, like:

```
{
  "ok" : true,
  "filename" : "test.txt",
  "url" : "http://127.0.0.1:9200/fscrawler-rest-tests_doc/doc/dd18bf3a8ea2a3e53e2661c7fb53534"
}
```

To enable CORS (Cross-Origin Request Sharing) functionality you will need to set `enable_cors: true` in your job settings.
Read the REST service chapter for more information.
FSCrawler supports all formats Tika supports, like:

- HTML
- Microsoft Office
- Open Office
- PDF
- Images
- MP3
- …
10.1 Moving files to a “watched” directory

When moving an existing file to the directory FSCrawler is watching, you need to explicitly touch all the files as when moved, the files are keeping their original date intact:

```
# single file
touch file_you_moved

# all files
find -type f -exec touch {} +

# all .txt files
find -type f -name "*.txt" -exec touch {} +
```

Or you need to restart from the beginning with the --restart option which will reindex everything.

10.2 Indexing from HDFS drive

There is no specific support for HDFS in FSCrawler. But you can mount your HDFS on your machine and run FS crawler on this mount point. You can also read details about HDFS NFS Gateway.

10.3 Using docker

To use FSCrawler with docker, check docker-fscrawler recipe.
10.4 Using docker-compose

To standup a full environment you can use docker-compose from the contrib directory. This environment will setup a node ElasticSearch cluster, a copy of Kibana for searching and FSCrawler as containers. No other installs are needed, aside form Docker and docker-compose.

Steps:

1. Download and install docker.
2. Download and install docker-compose.
3. Copy the contrib directory into your home directory.
4. Edit the docker-compose.yaml
   1. Edit the line (somewhere around 66) that points to the “files to be scanned”. This is the path on the host machine prior to the colon. (ex: /fs/resume)
   2. In the ./config/ directory exists the name of the index name that FSCrawler will use. By default, it’s set to ‘idx’. You can change it by renaming this directory, and changing the _settings.yaml file. Check the ./config/idx/_settings.yaml to update any changes you like. If you have multiple directories that you like to scan, I would suggest linking them under a single directory and changing the “follow_links” option.
5. Check the Dockerfile-fscrawler file. This is where the version of the package is determined. By default I have set to download the ‘master’ branch which is currently producing a es7-2.7-SNAPSHOT version but you can lock this into a specific version to make it more reliable. Update (DO NOT MOVE) the ENV variables to match what you want the build to be.
6. Issue docker-compose up -d in that directory and it’ll download and create the containers. It’ll also compile and build a custom container for fscrawler.
7. After the containers are up and running, wait about 30 seconds for everything to start syncing. You can now access Kibana and build your index (just need to do it once). After that the search will be available via Kibana.

TODO: Build a more robust link to a specific version in the Dockerfile so it’s a little more specific about what it downloads and builds.
Once the crawler is running, it will write status information and statistics in:

- `~/.fscrawler/(job_name)/_status.json`

It means that if you stop the job at some point, FSCrawler will restart it from where it stops.
CLI options

• --help displays help
• --silent runs in silent mode. No output is generated.
• --debug runs in debug mode.
• --trace runs in trace mode (more verbose than debug).
• --config_dir defines directory where jobs are stored instead of default ~/.fscrawler.
• --username defines the username to use when using an secured version of elasticsearch cluster. Read Using Credentials (X-Pack).
• --upgrade runs a reindex operation for indices created with an older version. See Upgrade.
• --loop x defines the number of runs we want before exiting. See Loop.
• --restart restart a job from scratch. See Restart.
• --rest starts the REST service. See Rest.

12.1 Upgrade

--upgrade runs a reindex operation for indices created with an older version which was using multiple types within the same index. More on this in Upgrade to 2.3 section.

12.2 Loop

New in version 2.2.
--loop x defines the number of runs we want before exiting:
• X where X is a negative value means infinite, like −1 (default)
• 0 means that we don’t run any crawling job (useful when used with rest).
• \( x \) where \( x \) is a positive value is the number of runs before it stops.

If you want to scan your hard drive only once, run with \(--\text{loop} 1\).

## 12.3 Restart

New in version 2.2.

You can tell FSCrawler that it must restart from the beginning by using \(--\text{restart} \) option:

```
bin/fscrawler job_name --restart
```

In that case, the \({\text{job_name}}/\_\text{status.json} \) file will be removed.

## 12.4 Rest

New in version 2.3.

If you want to run the \textit{REST service} without scanning your hard drive, launch with:

```
bin/fscrawler --rest --loop 0
```
If you want to provide JVM settings, like defining memory allocated to FSCrawler, you can define a system property named `FS_JAVA_OPTS`:

```
FS_JAVA_OPTS="-Xmx521m -Xms521m" bin/fscrawler
```
If you want to define an external log4j2.xml file, you can use the log4j.configurationFile JVM parameter which you can define in FS_JAVA_OPTS variable if you wish:

```
FS_JAVA_OPTS="-Dlog4j.configurationFile=path/to/log4j2.xml" bin/fscrawler
```

You can use the default log4j2.xml file as an example to start with.
Job file specification

The job file must comply to the following YAML specifications:

```yaml
name: "job_name"
fs:
  url: "/path/to/docs"
  update_rate: "5m"
  includes:
    - "*.doc"
    - "*.xls"
  excludes:
    - "resume.doc"
  json_support: false
  filename_as_id: true
  add_filesize: true
  remove_deleted: true
  add_as_inner_object: false
  store_source: true
  index_content: true
  indexed_chars: "10000.0"
  attributes_support: false
  raw_metadata: true
  xml_support: false
  index_folders: true
  lang_detect: false
  continue_on_error: false
  pdf_ocr: true
  ocr:
    language: "eng"
    path: "/path/to/tesseract/if/not/available/in/PATH"
    data_path: "/path/to/tesseract/tessdata/if/needed"
server:
  hostname: "localhost"
  port: 22
  username: "dadoonet"
```

(continues on next page)
password: "password"
protocol: "SSH"
pem_path: "/path/to/pemfile"
elasticsearch:
  nodes:
    # With Cloud ID
    - cloud_id: "CLOUD_ID"
    # With URL
    - url: "http://127.0.0.1:9200"
  index: "docs"
  bulk_size: 1000
  flush_interval: "5s"
  byte_size: "10mb"
  username: "elastic"
  password: "password"
rest:
  url: "https://127.0.0.1:8080/fscrawler"

Here is a list of existing top level settings:

<table>
<thead>
<tr>
<th>Name</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>name (mandatory field)</td>
<td>The most simple crawler</td>
</tr>
<tr>
<td>fs</td>
<td>Local FS settings</td>
</tr>
<tr>
<td>elasticsearch</td>
<td>Elasticsearch settings</td>
</tr>
<tr>
<td>server</td>
<td>SSH settings</td>
</tr>
<tr>
<td>rest</td>
<td>REST service</td>
</tr>
</tbody>
</table>

New in version 2.7.

You can define your job settings either in yaml (using .yaml extension) or in json (using .json extension).
You can define the most simple crawler job by writing a `~/.fscrawler/test/_settings.yaml` file as follow:

```yaml
name: "test"
```

This will scan every 15 minutes all documents available in `/tmp/es` dir and will index them into `test_doc` index. It will connect to an elasticsearch cluster running on `127.0.0.1:9200`.

**Note:** `name` is a mandatory field.
Local FS settings

Here is a list of Local FS settings (under `fs.` prefix):

<table>
<thead>
<tr>
<th>Name</th>
<th>Default value</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs.url</td>
<td>&quot;/tmp/es&quot;</td>
<td>Root directory</td>
</tr>
<tr>
<td>fs.update_rate</td>
<td>&quot;15m&quot;</td>
<td>Update Rate</td>
</tr>
<tr>
<td>fs.includes</td>
<td>null</td>
<td>Includes and excludes</td>
</tr>
<tr>
<td>fs.excludes</td>
<td>[&quot;<em>/~</em>&quot;]</td>
<td>Includes and excludes</td>
</tr>
<tr>
<td>fs.filters</td>
<td>null</td>
<td>Filter content</td>
</tr>
<tr>
<td>fs.json_support</td>
<td>false</td>
<td>Indexing JSON docs</td>
</tr>
<tr>
<td>fs.xml_support</td>
<td>false</td>
<td>Indexing XML docs</td>
</tr>
<tr>
<td>fs.add_as_inner_object</td>
<td>false</td>
<td>Add as Inner Object</td>
</tr>
<tr>
<td>fs.index_folders</td>
<td>true</td>
<td>Index folders</td>
</tr>
<tr>
<td>fs.attributes_support</td>
<td>false</td>
<td>Adding file attributes</td>
</tr>
<tr>
<td>fs.raw_metadata</td>
<td>false</td>
<td>Disabling raw metadata</td>
</tr>
<tr>
<td>fs.filename_as_id</td>
<td>false</td>
<td>Using filename as elasticsearch_id</td>
</tr>
<tr>
<td>fs.add_filesize</td>
<td>true</td>
<td>Disabling file size field</td>
</tr>
<tr>
<td>fs.remove_deleted</td>
<td>true</td>
<td>Ignore deleted files</td>
</tr>
<tr>
<td>fs.store_source</td>
<td>false</td>
<td>Storing binary source document</td>
</tr>
<tr>
<td>fs.index_content</td>
<td>true</td>
<td>Ignore content</td>
</tr>
<tr>
<td>fs.lang_detect</td>
<td>false</td>
<td>Language detection</td>
</tr>
<tr>
<td>fs.continue_on_error</td>
<td>false</td>
<td>Continue on Error</td>
</tr>
<tr>
<td>fs.pdf_ocr</td>
<td>true</td>
<td>OCR integration</td>
</tr>
<tr>
<td>fs.indexed_chars</td>
<td>100000.0</td>
<td>Extracted characters</td>
</tr>
<tr>
<td>fs.ignore_above</td>
<td>null</td>
<td>Ignore above</td>
</tr>
<tr>
<td>fs.checksum</td>
<td>null</td>
<td>File Checksum</td>
</tr>
<tr>
<td>fs.follow_symlinks</td>
<td>false</td>
<td>Follow Symlinks</td>
</tr>
</tbody>
</table>
17.1 Root directory

Define `fs.url` property in your `~/.fscrawler/test/_settings.yaml` file:

```yaml
name: "test"
fs:
  url: "/path/to/data/dir"
```

For Windows users, use a form like `c:/tmp` or `c:\tmp`.

17.2 Update rate

By default, `update_rate` is set to `15m`. You can modify this value using any compatible time unit.

For example, here is a 15 minutes update rate:

```yaml
name: "test"
fs:
  update_rate: "15m"
```

Or a 3 hours update rate:

```yaml
name: "test"
fs:
  update_rate: "3h"
```

`update_rate` is the pause duration between the last time we read the file system and another run. Which means that if you set it to `15m`, the next scan will happen on 15 minutes after the end of the current scan, whatever its duration.

17.3 Includes and excludes

Let's say you want to index only docs like `*.doc` and `*.pdf` but not `resume*`. So `resume_david.pdf` won't be indexed.

Define `fs.includes` and `fs.excludes` properties in your `~/.fscrawler/test/_settings.yaml` file:

```yaml
name: "test"
fs:
  includes:
  - "*/*.doc"
  - "*/*.pdf"
  excludes:
  - "*/resume*"
```

By default, FSCrawler will exclude files starting with `~`.

New in version 2.5.

It also applies to directory names. So if you want to ignore `ignore` dir, just add `.ignore` as an excluded name. Note that `includes` and `excludes` apply to directory names as well.

Let's take the following example with the root dir as `/tmp:`
If you define the following `fs.excludes` property in your `~/.fscrawler/test/_settings.yaml` file:

```yaml
name: "test"
fs:
excludes:
  - "/folderB/subfolder*"
```

Then all files but the ones in `/folderB/subfolderA`, `/folderB/subfolderB` and `/folderB/subfolderC` will be indexed.

Since the includes and excludes work on the entire *path of the file* you must consider that when using wildcards. Below are some includes and excludes pattern to help convey the idea better.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Includes</th>
<th>Excludes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>*.jpg</code></td>
<td>Include all jpg files</td>
<td>exclude all jpg files</td>
</tr>
<tr>
<td><code>/images/*.jpg</code></td>
<td>Include all jpg files in the images directory</td>
<td>Exclude all jpg files in the images directory</td>
</tr>
<tr>
<td><code>*/old-*.jpg</code></td>
<td>Include all jpg files that start with <code>old-</code></td>
<td>Exclude all jpg files that start with <code>old-</code></td>
</tr>
</tbody>
</table>

New in version 2.6.

If a folder contains a file named `.fscrawlerignore`, this folder and its subfolders will be entirely skipped.

### 17.4 Filter content

New in version 2.5.

You can filter out documents you would like to index by adding one or more regular expression that match the extracted content. Documents which are not matching will be simply ignored and not indexed.

If you define the following `fs.filters` property in your `~/.fscrawler/test/_settings.yaml` file:

```yaml
name: "test"
fs:
  filters:
    - "*.foo.*"
    - "^4\d{3}(\s[\s-]?)\d{4}\s\d{4}\s\d{4}\s\d{4}$"
```

With this example, only documents which contains the word `foo` and a VISA credit card number with the form like `4012888888888881, 4012 8888 8888 1881` or `4012-8888-8888-1881` will be indexed.
17.5 Indexing JSon docs

If you want to index JSon files directly without parsing with Tika, you can set `json_support` to `true`. JSon contents will be stored directly under `_source`. If you need to keep JSon documents synchronized to the index, set option `Add as Inner Object` which stores additional metadata and the JSon contents under field `object`.

```
name: "test"
fs:
  json_support: true
```

Of course, if you did not define a mapping before launching the crawler, Elasticsearch will auto guess the mapping.

17.6 Indexing XML docs

New in version 2.2.

If you want to index XML files and convert them to JSON, you can set `xml_support` to `true`. The content of XML files will be added directly under `_source`. If you need to keep XML documents synchronized to the index, set option `Add as Inner Object` which stores additional metadata and the XML contents under field `object`.

```
name: "test"
fs:
  xml_support: true
```

Of course, if you did not define a mapping before launching the crawler, Elasticsearch will auto guess the mapping.

17.7 Add as Inner Object

The default settings store the contents of json and xml documents directly onto the `_source` element of elasticsearch documents. Thereby, there is no metadata about file and path settings, which are necessary to determine if a document is deleted or updated. New files will however be added to the index, (determined by the file timestamp).

If you need to keep json or xml documents synchronized to elasticsearch, you should set this option.

```
name: "test"
fs:
  add_as_inner_object: true
```

17.8 Index folders

New in version 2.2.

By default FSCrawler will index folder names in the folder index. If you don’t want to index those folders, you can set `index_folders` to `false`.

Note that in that case, FSCrawler won’t be able to detect removed folders so any document has been indexed in elasticsearch, it won’t be removed when you remove or move the folder away.

```
name: "test"
fs:
  index_folders: false
```
17.9 Dealing with multiple types and multiple dirs

If you have more than one type, create as many crawlers as types:

```
~/.fscrawler/test_type1/_settings.yaml:

name: "test_type1"
fs:
  url: "/tmp/type1"
  json_support: true
elasticsearch:
  index: "mydocs1"
  index_folder: "myfolders1"

~/.fscrawler/test_type2/_settings.yaml:

name: "test_type2"
fs:
  url: "/tmp/type2"
  json_support: true
elasticsearch:
  index: "mydocs2"
  index_folder: "myfolders2"

~/.fscrawler/test_type3/_settings.yaml:

name: "test_type3"
fs:
  url: "/tmp/type3"
  xml_support: true
elasticsearch:
  index: "mydocs3"
  index_folder: "myfolders3"
```

17.10 Dealing with multiple types within the same dir

You can also index many types from one single dir using two crawlers scanning the same dir and by setting `includes` parameter:

```
~/.fscrawler/test_type1.yaml:

name: "test_type1"
fs:
  url: "/tmp"
  includes:
    - "type1*.json"
  json_support: true
elasticsearch:
  index: "mydocs1"
  index_folder: "myfolders1"

~/.fscrawler/test_type2.yaml:

name: "test_type2"
fs:
```

(continues on next page)
url: "/tmp"
includes:
- "type2*.json"
json_support: true
elasticsearch:
  index: "mydocs2"
  index_folder: "myfolders2"

~/.fscrawler/test_type3.yaml:

name: "test_type3"
fs:
  url: "/tmp"
  includes:
    - "+xml"
  xml_support: true
elasticsearch:
  index: "mydocs3"
  index_folder: "myfolders3"

17.11 Using filename as elasticsearch _id

Please note that the document _id is always generated (hash value) from the filename to avoid issues with special characters in filename. You can force to use the _id to be the filename using filename_as_id attribute:

name: "test"
fs:
  filename_as_id: true

17.12 Adding file attributes

If you want to add file attributes such as attributes.owner, attributes.group and attributes.permissions, you can set attributes_support to true.

name: "test"
fs:
  attributes_support: true

Note: On Windows systems, attributes.group and attributes.permissions are not generated.

17.13 Disabling raw metadata

FSCrawler can extract all found metadata within a meta.raw object in addition to the standard metadata fields. If you want to enable this feature, you can set raw_metadata to true.
Generated raw metadata depends on the file format itself.

For example, a PDF document could generate:

```json
{
    "date" : "2016-07-07T08:37:42Z",
    "pdf:PDFVersion" : "1.5",
    "xmp:CreatorTool" : "Microsoft Word",
    "Keywords" : "keyword1, keyword2",
    "access_permission:modify_annotations" : "true",
    "access_permission:can_print_degraded" : "true",
    "subject" : "Test Tika Object",
    "dc:creator" : "David Pilato",
    "dcterms:created" : "2016-07-07T08:37:42Z",
    "last-Modified" : "2016-07-07T08:37:42Z",
    "dcterms:modified" : "2016-07-07T08:37:42Z",
    "dc:format" : "application/pdf; version=1.5",
    "title" : "Test Tika title",
    "last-Save-Date" : "2016-07-07T08:37:42Z",
    "access_permission:fill_in_form" : "true",
    "meta:save-date" : "2016-07-07T08:37:42Z",
    "pdf:encrypted" : "false",
    "dc:title" : "Test Tika title",
    "modified" : "2016-07-07T08:37:42Z",
    "cp:subject" : "Test Tika Object",
    "Content-Type" : "application/pdf",
    "X-Parsed-By" : "org.apache.tika.parser.DefaultParser",
    "creator" : "David Pilato",
    "meta:author" : "David Pilato",
    "dc:subject" : "keyword1, keyword2",
    "dc:creator" : "David Pilato",
    "dc:creation-date" : "2016-07-07T08:37:42Z",
    "created" : "Thu Jul 07 10:37:42 CEST 2016",
    "access_permission:extract_for_accessibility" : "true",
    "access_permission:assemble_document" : "true",
    "xmpTPg:NPages" : "2",
    "Creation-Date" : "2016-07-07T08:37:42Z",
    "access_permission:extract_content" : "true",
    "access_permission:can_print" : "true",
    "meta:keyword" : "keyword1, keyword2",
    "Author" : "David Pilato",
    "access_permission:can_modify" : "true"
}
```

Where a MP3 file would generate:

```json
{
    "xmpDM:genre" : "Vocal",
    "X-Parsed-By" : "org.apache.tika.parser.DefaultParser",
    "creator" : "David Pilato",
    "xmpDM:album" : "FS Crawler",
    "xmpDM:trackNumber" : "1",
    "xmpDM:releaseDate" : "2016",
    "meta:author" : "David Pilato",
    "xmpDM:artist" : "David Pilato",
}
```

(continues on next page)
"dc:creator" : "David Pilato",
"xmpDM:audioCompressor" : "MP3",
"title" : "Test Tika",
"xmpDM:audioChannelType" : "Stereo",
"version" : "MPEG 3 Layer III Version 1",
"xmpDM:logComment" : "Hello but reverted",
"xmpDM:audioSampleRate" : "44100",
"channels" : "2",
"dc:title" : "Test Tika",
"Author" : "David Pilato",
"xmpDM:duration" : "1018.775146484375",
"Content-Type" : "audio/mpeg",
"samplerate" : "44100"
}

**Note:** All fields are generated as text even though they can be valid booleans or numbers.

The `meta.raw.*` fields have a default mapping applied:

```
{
    "type": "text",
    "fields": {
        "keyword": {
            "type": "keyword",
            "ignore_above": 256
        }
    }
}
```

If you want specifically tell elasticsearch to use a date type or a numeric type for some fields, you need to modify the default template provided by FSCrawler.

**Note:** Note that dots in metadata names will be replaced by a :. For example `PTEX.Fullbanner` will be indexed as `PTEX:Fullbanner`.

**Note:** Note that if you have a lot of different type of files, that can generate a lot of raw metadata which can make you hit the total number of field limit in elasticsearch mappings. In which case you will need to change the index settings `foo`.

See elasticsearch documentation

### 17.14 Disabling file size field

By default, FSCrawler will create a field to store the original file size in octets. You can disable it using ‘add_filesize’ option:

```
name: "test"
fs:
    add_filesize: false
```
17.15 Ignore deleted files

If you don’t want to remove indexed documents when you remove a file or a directory, you can set `remove_deleted` to `false` (default to `true`):

```json
name: "test"
fs:
  remove_deleted: false
```

17.16 Ignore content

If you don’t want to extract file content but only index filesystem metadata such as filename, date, size and path, you can set `index_content` to `false` (default to `true`):

```json
name: "test"
fs:
  index_content: false
```

17.17 Continue on Error

New in version 2.3.

By default FSCrawler will immediately stop indexing if he hits a Permission denied exception. If you want to just skip this File and continue with the rest of the directory tree you can set `continue_on_error` to `true` (default to `false`):

```json
name: "test"
fs:
  continue_on_error: true
```

17.18 Language detection

New in version 2.2.

You can ask for language detection using `lang_detect` option:

```json
name: "test"
fs:
  lang_detect: true
```

In that case, a new field named `meta.language` is added to the generated JSON document.

If you are using elasticsearch 5.0 or superior, you can use this value to send your document to a specific index using a `Node Ingest pipeline`.

For example, you can define a pipeline named `langdetect` with:

```json
PUT _ingest/pipeline/langdetect
{
  "description" : "langdetect pipeline",
}
```

(continues on next page)
In FSCrawler settings, set both `fs.lang_detect` and `elasticsearch.pipeline` options:

```json
name: "test"
fs:
  lang_detct: true
elasticsearch:
  pipeline: "langdetect"
```

And then, a document containing french text will be sent to `myindex-fr`. A document containing english text will be sent to `myindex-en`.

You can also imagine changing the field name from `content` to `content-fr` or `content-en`. That will help you to define the correct analyzer to use.

Language detection might detect more than one language in a given text but only the most accurate will be set. Which means that if you have a document containing 80% of french and 20% of english, the document will be marked as `fr`.

Note that language detection is CPU and time consuming.

### 17.19 Storing binary source document

You can store in Elasticsearch itself the binary document (BASE64 encoded) using `store_source` option:

```json
name: "test"
fs:
  store_source: true
```

In that case, a new field named `attachment` is added to the generated JSON document. This field is not indexed. Default mapping for `attachment` field is:

```json
{  
  "_doc" : {  
    "properties" : {  
      "attachment" : {  
        "type" : "binary",
        "doc_values" : false
      }
    }
  }
}
```
17.20 Extracted characters

By default FSCrawler will extract only the first 100 000 characters. But, you can set `indexed_chars` to 5000 in FSCrawler settings in order to overwrite this default settings.

```
name: "test"
fs:
  indexed_chars: "5000"
```

This number can be either a fixed size, number of characters that is, or a percent using % sign. The percentage value will be applied to the filesize to determine the number of character the crawler needs to extract.

If you want to index only 80% of filesize, define `indexed_chars` to "80%". Of course, if you want to index the full document, you can set this property to "100%". Double values are also supported so "0.01%" is also a correct value.

**Compressed files:** If your file is compressed, you might need to increase `indexed_chars` to more than "100%". For example, "150%".

If you want to extract the full content, define `indexed_chars` to "-1".

**Note:** Tika requires to allocate in memory a data structure to extract text. Setting `indexed_chars` to a high number will require more memory!

17.21 Ignore Above

New in version 2.5.

By default FSCrawler will send to Tika every single file, whatever its size. But some files on your file system might be a way too big to be parsed.

Set `ignore_above` to the desired value of the limit.

```
name: "test"
fs:
  ignore_above: "5mb"
```

17.22 File checksum

If you want FSCrawler to generate a checksum for each file, set `checksum` to the algorithm you wish to use to compute the checksum, such as MD5 or SHA-1.

```
name: "test"
fs:
  checksum: "MD5"
```

17.23 Follow Symlinks

New in version 2.7.
If you want FSCrawler to follow the symbolic links, you need to be explicit about it and set `follow_symlink` to `true`. Starting from version 2.7, symbolic links are not followed anymore.

```yaml
name: "test"
fs:
  follow_symlink: true
```
You can index files remotely using SSH.

Here is a list of SSH settings (under `server` prefix):

<table>
<thead>
<tr>
<th>Name</th>
<th>Default value</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>server.hostname</td>
<td>null</td>
<td>Hostname</td>
</tr>
<tr>
<td>server.port</td>
<td>22</td>
<td>Port</td>
</tr>
<tr>
<td>server.username</td>
<td>null</td>
<td>Username / Password</td>
</tr>
<tr>
<td>server.password</td>
<td>null</td>
<td>Username / Password</td>
</tr>
<tr>
<td>server.protocol</td>
<td>&quot;local&quot;</td>
<td>Set it to ssh</td>
</tr>
<tr>
<td>server.pem_path</td>
<td>null</td>
<td>Using Username / PEM file</td>
</tr>
</tbody>
</table>

### 18.1 Username / Password

Let’s say you want to index from a remote server using SSH:

- **FS URL**: /path/to/data/dir/on/server
- **Server**: mynode.mydomain.com
- **Username**: username
- **Password**: password
- **Protocol**: ssh (default to local)
- **Port**: 22 (default to 22)

```yaml
name: "test"
fs:
  url: "/path/to/data/dir/on/server"
server:
  hostname: "mynode.mydomain.com"
```
18.2 Using Username / PEM file

Let’s say you want to index from a remote server using SSH:

- **FS URL:** /path/to/data/dir/on/server
- **Server:** mynode.mydomain.com
- **Username:** username
- **PEM File:** /path/to/private_key.pem
- **Protocol:** ssh (default to local)
- **Port:** 22 (default to 22)

```
name: "test"
fs:
    url: "/path/to/data/dir/on/server"
server:
    hostname: "mynode.mydomain.com"
    port: 22
    username: "username"
    password: "password"
    protocol: "ssh"
pem_path: "/path/to/private_key.pem"
```

18.3 Windows drives

When using Windows, you might want to index documents coming from another drive than C:. To specify the drive, you need to use the following format:

```
name: "test"
fs:
    url: "/D:/path/to/data/dir/on/server"
server:
    hostname: "mynode.mydomain.com"
    port: 22
    username: "username"
    password: "password"
    protocol: "ssh"
```
Here is a list of Elasticsearch settings (under `elasticsearch.prefix`):

<table>
<thead>
<tr>
<th>Name</th>
<th>Default value</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>elasticsearch.index</code></td>
<td><code>job name</code></td>
<td>Index settings for documents</td>
</tr>
<tr>
<td><code>elasticsearch.index_folder</code></td>
<td><code>job name + _folder</code></td>
<td>Index settings for folders</td>
</tr>
<tr>
<td><code>elasticsearch.bulk_size</code></td>
<td><code>100</code></td>
<td>Bulk settings</td>
</tr>
<tr>
<td><code>elasticsearch.flush_interval</code></td>
<td>&quot;5s&quot;</td>
<td>Bulk settings</td>
</tr>
<tr>
<td><code>elasticsearch.byte_size</code></td>
<td>&quot;10mb&quot;</td>
<td>Bulk settings</td>
</tr>
<tr>
<td><code>elasticsearch.pipeline</code></td>
<td><code>null</code></td>
<td>Using Ingest Node Pipeline</td>
</tr>
<tr>
<td><code>elasticsearch.nodes</code></td>
<td><code>http://127.0.0.1:9200</code></td>
<td>Node settings</td>
</tr>
<tr>
<td><code>elasticsearch.path_prefix</code></td>
<td><code>null</code></td>
<td>Path prefix</td>
</tr>
<tr>
<td><code>elasticsearch.username</code></td>
<td><code>null</code></td>
<td>Using Credentials (X-Pack)</td>
</tr>
<tr>
<td><code>elasticsearch.password</code></td>
<td><code>null</code></td>
<td>Using Credentials (X-Pack)</td>
</tr>
</tbody>
</table>

### 19.1 Index settings

#### 19.1.1 Index settings for documents

By default, FSCrawler will index your data in an index which name is the same as the crawler name (name property) plus _doc suffix, like `test_doc`. You can change it by setting index field:

```json
name: "test"
elasticsearch:
  index: "docs"
```

#### 19.1.2 Index settings for folders

FSCrawler will also index folders in an index which name is the same as the crawler name (name property) plus _folder suffix, like `test_folder`. You can change it by setting index_folder field:
19.1.3 Mappings

When FSCrawler needs to create the doc index, it applies some default settings and mappings which are read from ~/.fscrawler/_default/7/_settings.json. You can read its content from the source.

Settings define an analyzer named fscrawler_path which uses a path hierarchy tokenizer.

FSCrawler applies as well a mapping automatically for the folders which can also be read from the source.

You can also display the index mapping being used with Kibana:

```
GET docs/_mapping
GET docs_folder/_mapping
```

Or fall back to the command line:

```
curl 'http://localhost:9200/docs/_mapping?pretty'
curl 'http://localhost:9200/docs_folder/_mapping?pretty'
```

**Note:** FSCrawler is actually applying default index settings depending on the elasticsearch version it is connected to. The default settings definitions are stored in ~/.fscrawler/_default/_mappings:

- 2/_settings.json: for elasticsearch 2.x series document index settings
- 2/_settings_folder.json: for elasticsearch 2.x series folder index settings
- 5/_settings.json: for elasticsearch 5.x series document index settings
- 5/_settings_folder.json: for elasticsearch 5.x series folder index settings
- 6/_settings.json: for elasticsearch 6.x series document index settings
- 6/_settings_folder.json: for elasticsearch 6.x series folder index settings
- 7/_settings.json: for elasticsearch 7.x series document index settings
- 7/_settings_folder.json: for elasticsearch 7.x series folder index settings

**Note:** For versions before 6.x series, the type of the document is doc. From 6.x, the type of the document is _doc.

Creating your own mapping (analyzers)

If you want to define your own index settings and mapping to set analyzers for example, you can either create the index and push the mapping or define a ~/.fscrawler/_default/7/_settings.json document which contains the index settings and mappings you wish before starting the FSCrawler.

The following example uses a french analyzer to index the content field.


```json
{
  "settings": {
    "number_of_shards": 1,
    "index.mapping.total_fields.limit": 2000,
    "analysis": {
      "analyzer": {
        "fscrawler_path": {
          "tokenizer": "fscrawler_path"
        },
        "tokenizer": {
          "fscrawler_path": {
            "type": "path_hierarchy"
          }
        }
      },
      "tokenizer": {
        "fscrawler_path": {
          "type": "path_hierarchy"
        }
      }
    },
    "mappings": {
      "_doc": {
        "dynamic_templates": [
          {
            "raw_as_text": {
              "path_match": "meta.raw.*",
              "mapping": {
                "type": "text",
                "fields": {
                  "keyword": {
                    "type": "keyword",
                    "ignore_above": 256
                  }
                }
              }
            }
          }
        ],
        "properties": {
          "attachment": {
            "type": "binary",
            "doc_values": false
          },
          "attributes": {
            "properties": {
              "group": {
                "type": "keyword"
              },
              "owner": {
                "type": "keyword"
              }
            }
          },
          "content": {
            "type": "text",
            "analyzer": "french"
          },
          "file": {
            "properties": {
              "content_type": {
                ...
              }
            }
          }
        }
      }
    }
  }
}
```

(continues on next page)
"type": "keyword"
},
"filename": {
  "type": "keyword",
  "store": true
},
"extension": {
  "type": "keyword"
},
"filesize": {
  "type": "long"
},
"indexed_chars": {
  "type": "long"
},
"indexing_date": {
  "type": "date",
  "format": "dateOptionalTime"
},
"created": {
  "type": "date",
  "format": "dateOptionalTime"
},
"last_modified": {
  "type": "date",
  "format": "dateOptionalTime"
},
"last_accessed": {
  "type": "date",
  "format": "dateOptionalTime"
},
"checksum": {
  "type": "keyword"
},
"url": {
  "type": "keyword",
  "index": false
}
},
"meta": {
  "properties": {
    "author": {
      "type": "text"
    },
    "date": {
      "type": "date",
      "format": "dateOptionalTime"
    },
    "keywords": {
      "type": "text"
    },
    "title": {
      "type": "text"
    },
    "language": {
      "type": "keyword"
    }
  }
}
Note that if you want to push manually the mapping to Elasticsearch, you can use the classic REST calls:

```bash
# Create index (don't forget to add the fscrawler_path analyzer)
PUT docs
{
  // Same index settings as previously seen
}
```
Define explicit mapping/settings per job

Let’s say you created a job named `job_name` and you are sending documents against an elasticsearch cluster running version 6.x.

If you create the following files, they will be picked up at job start time instead of the default ones:

- `~/.fscrawler/{job_name}/_mappings/7/_settings.json`
- `~/.fscrawler/{job_name}/_mappings/7/_settings_folder.json`

**Tip:** You can do the same for other elasticsearch versions with:

- `~/.fscrawler/{job_name}/_mappings/2/_settings.json` for 2.x series (deprecated)
- `~/.fscrawler/{job_name}/_mappings/2/_settings_folder.json` for 2.x series (deprecated)
- `~/.fscrawler/{job_name}/_mappings/5/_settings.json` for 5.x series
- `~/.fscrawler/{job_name}/_mappings/5/_settings_folder.json` for 5.x series
- `~/.fscrawler/{job_name}/_mappings/6/_settings.json` for 6.x series
- `~/.fscrawler/{job_name}/_mappings/6/_settings_folder.json` for 6.x series

Replace existing mapping

Unfortunately you can not change the mapping on existing data. Therefore, you’ll need first to remove existing index, which means remove all existing data, and then restart FSCrawler with the new mapping.

You might to try elasticsearch Reindex API though.

### 19.2 Bulk settings

FSCrawler is using bulks to send data to elasticsearch. By default the bulk is executed every 100 operations or every 5 seconds or every 10 megabytes. You can change default settings using `bulk_size`, `byte_size` and `flush_interval`:

```json
name: "test"
elasticsearch:
  bulk_size: 1000
  byte_size: "500kb"
  flush_interval: "2s"
```

**Tip:** Elasticsearch has a default limit of 100mb per HTTP request as per elasticsearch HTTP Module documentation.

Which means that if you are indexing a massive bulk of documents, you might hit that limit and FSCrawler will throw an error like `entity content is too long [xxx] for the configured buffer limit [104857600]`.

You can either change this limit on elasticsearch side by setting `http.max_content_length` to a higher value but please be aware that this will consume much more memory on elasticsearch side.

Or you can decrease the `bulk_size` or `byte_size` setting to a smaller value.
19.3 Using Ingest Node Pipeline

New in version 2.2.

If you are using an elasticsearch cluster running a 5.0 or superior version, you can use an Ingest Node pipeline to transform documents sent by FSCrawler before they are actually indexed.

For example, if you have the following pipeline:

```
PUT _ingest/pipeline/fscrawler
{
  "description" : "fscrawler pipeline",
  "processors" : [
    {
      "set" : {
        "field": "foo",
        "value": "bar"
      }
    }
  ]
}
```

In FSCrawler settings, set the `elasticsearch.pipeline` option:

```
name: "test"
elasticsearch:
  pipeline: "fscrawler"
```

**Note:** Folder objects are not sent through the pipeline as they are more internal objects.

19.4 Node settings

FSCrawler is using elasticsearch REST layer to send data to your running cluster. By default, it connects to `http://127.0.0.1:9200` which is the default when running a local node on your machine.

Of course, in production, you would probably change this and connect to a production cluster:

```
name: "test"
elasticsearch:
  nodes:
    - url: "http://mynode1.mycompany.com:9200"
```

If you are using Elasticsearch service by Elastic, you can just use the Cloud ID which is available in the Cloud Console and paste it:

```
name: "test"
elasticsearch:
  nodes:
    - cloud_id: "fscrawler:ZXVyb3BlLXdlc3QxLmdjcC5jbG91ZC5lcy5pbyQxZDFlYTk5Njg4Nzc0NWE2YTJiN2NiNzkzMTUzNDhhMyQyOTk1MDI3MzZmZGQ0OTI5OTE5NzdlOTk3ZmU3Nw=="
```

This ID will be used to automatically generate the right host, port and scheme.
Hint: In the context of Elasticsearch service by Elastic, you will most likely need to provide as well the username and the password. See Using Credentials (X-Pack).

You can define multiple nodes:

```yaml
name: "test"
elasticsearch:
  nodes:
    - url: "http://mynode1.mycompany.com:9200"
    - url: "http://mynode2.mycompany.com:9200"
```

Note: New in version 2.2: you can use HTTPS instead of default HTTP.

```yaml
name: "test"
elasticsearch:
  nodes:
    - url: "https://CLUSTERID.eu-west-1.aws.found.io:9243"
```

For more information, read SSL Configuration.

19.5 Path prefix

New in version 2.7: If your elasticsearch is running behind a proxy with url rewriting, you might have to specify a path prefix. This can be done with path_prefix setting:

```yaml
name: "test"
elasticsearch:
  nodes:
    - url: "http://mynode1.mycompany.com:9200"
  path_prefix: "/path/to/elasticsearch"
```

Note: The same path_prefix applies to all nodes.

19.6 Using Credentials (X-Pack)

New in version 2.2.

If you secured your elasticsearch cluster with X-Pack, you can provide username and password to FSCrawler:

```yaml
name: "test"
elasticsearch:
  username: "elastic"
  password: "changeme"
```
**Warning**: For the current version, the elasticsearch password is stored in plain text in your job setting file. A better practice is to only set the username or pass it with `--username elastic` option when starting FSCrawler. If the password is not defined, you will be prompted when starting the job:

```
22:46:42,528 INFO [f.p.e.c.f.FsCrawler] Password for elastic:
```

### 19.7 SSL Configuration

In order to ingest documents to Elasticsearch over HTTPS based connection, you need to perform additional configuration steps:

**Important**: Prerequisite: you need to have root CA chain certificate or Elasticsearch server certificate in DER format. DER format files have a `.cer` extension.

1. Logon to server (or client machine) where FSCrawler is running
2. Run:

   ```
   keytool -import -alias <alias name> -keystore "<JAVA_HOME>libsecuritycacerts" -file <Path of Elasticsearch Server certificate or Root certificate>
   ```

   It will prompt you for the password. Enter the certificate password like `changeit`.

3. Make changes to FSCrawler _settings.json file to connect to your Elasticsearch server over HTTPS:

   ```
   name: "test"
   elasticsearch:
     nodes:
       - url: "https://localhost:9243"
   ```

**Tip**: If you can not find `keytool`, it probably means that you did not add your `JAVA_HOME/bin` directory to your path.

### 19.8 Generated fields

FSCrawler may create the following fields depending on configuration and available data:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>content</td>
<td>Extracted content</td>
<td>&quot;This is my text!&quot;</td>
</tr>
<tr>
<td>attachment</td>
<td>BASE64 encoded binary file</td>
<td>BASE64 Encoded document</td>
</tr>
<tr>
<td>meta.author</td>
<td>Author if any in</td>
<td>&quot;David Pilato&quot;</td>
</tr>
<tr>
<td>meta.title</td>
<td>Title if any in document metadata</td>
<td>&quot;My document title&quot;</td>
</tr>
<tr>
<td>meta.date</td>
<td>Last modified date</td>
<td>&quot;2013-04-04T15:21:35&quot;</td>
</tr>
<tr>
<td>meta.keywords</td>
<td>Keywords if any in document metadata</td>
<td>[&quot;fs&quot;,&quot;elasticsearch&quot;]</td>
</tr>
</tbody>
</table>
### Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>meta.format</td>
<td>Format of the media</td>
<td>&quot;application/pdf; version=1.6&quot;</td>
</tr>
<tr>
<td>meta.identifier</td>
<td>URL/DOI/ISBN for example</td>
<td>&quot;FOOBAR&quot;</td>
</tr>
<tr>
<td>meta.contributor</td>
<td>Contributor</td>
<td>&quot;foo bar&quot;</td>
</tr>
<tr>
<td>meta.coverage</td>
<td>Coverage</td>
<td>&quot;FOOBAR&quot;</td>
</tr>
<tr>
<td>meta.modifier</td>
<td>Last author</td>
<td>&quot;David Pilato&quot;</td>
</tr>
<tr>
<td>meta.creator_tool</td>
<td>Tool used to create the resource</td>
<td>&quot;HTML2PDF-TCPDF&quot;</td>
</tr>
<tr>
<td>meta.publisher</td>
<td>Publisher: person, organisation, service</td>
<td>&quot;elastic&quot;</td>
</tr>
<tr>
<td>meta.relation</td>
<td>Related resource</td>
<td>&quot;FOOBAR&quot;</td>
</tr>
<tr>
<td>meta.rights</td>
<td>Information about rights</td>
<td>&quot;CC-BY-ND&quot;</td>
</tr>
<tr>
<td>meta.source</td>
<td>Source for the current document (derivated)</td>
<td>&quot;FOOBAR&quot;</td>
</tr>
<tr>
<td>meta.type</td>
<td>Nature or genre of the content</td>
<td>&quot;Image&quot;</td>
</tr>
<tr>
<td>meta.description</td>
<td>An account of the content</td>
<td>&quot;This is a description&quot;</td>
</tr>
<tr>
<td>meta.created</td>
<td>Date of creation</td>
<td>&quot;2013-04-04T15:21:35&quot;</td>
</tr>
<tr>
<td>meta.print_date</td>
<td>When was the doc last printed?</td>
<td>&quot;2013-04-04T15:21:35&quot;</td>
</tr>
<tr>
<td>meta.metadata_date</td>
<td>Last modification of metadata</td>
<td>&quot;2013-04-04T15:21:35&quot;</td>
</tr>
<tr>
<td>meta.latitude</td>
<td>The WGS84 Latitude of the Point</td>
<td>&quot;N 48° 51' 45.81''&quot;</td>
</tr>
<tr>
<td>meta.longitude</td>
<td>The WGS84 Longitude of the Point</td>
<td>&quot;E 2° 17'15.331''&quot;</td>
</tr>
<tr>
<td>meta.altitude</td>
<td>The WGS84 Altitude of the Point</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>meta.rating</td>
<td>A user-assigned rating -1, [0..5]</td>
<td>0</td>
</tr>
<tr>
<td>meta.comments</td>
<td>Comments</td>
<td>&quot;Comments&quot;</td>
</tr>
<tr>
<td>meta.raw</td>
<td>An object with all raw metadata</td>
<td>&quot;meta.raw.channels&quot;: &quot;2&quot;</td>
</tr>
<tr>
<td>file.content_type</td>
<td>Content Type</td>
<td>&quot;application/vnd.oasis.opendocument.text&quot;</td>
</tr>
<tr>
<td>file.created</td>
<td>Creation date</td>
<td>&quot;2018-07-30T11:19:23.000+0000&quot;</td>
</tr>
<tr>
<td>file.last_modified</td>
<td>Last modification date</td>
<td>&quot;2018-07-30T11:19:23.000+0000&quot;</td>
</tr>
<tr>
<td>file.last_accessed</td>
<td>Last accessed date</td>
<td>&quot;2018-07-30T11:19:23.000+0000&quot;</td>
</tr>
<tr>
<td>file.indexing_date</td>
<td>Indexing date</td>
<td>&quot;2018-07-30T11:19:30.703+0000&quot;</td>
</tr>
<tr>
<td>file.filesize</td>
<td>File size in bytes</td>
<td>1256362</td>
</tr>
<tr>
<td>file.indexed_chars</td>
<td>Extracted chars</td>
<td>100000</td>
</tr>
<tr>
<td>file.filename</td>
<td>Original file name</td>
<td>&quot;mydocument.pdf&quot;</td>
</tr>
<tr>
<td>file.extension</td>
<td>Original file name extension</td>
<td>&quot;pdf&quot;</td>
</tr>
<tr>
<td>file.url</td>
<td>Original file url</td>
<td>&quot;file://tmp/otherdir/mydocument.pdf&quot;</td>
</tr>
<tr>
<td>file.checksum</td>
<td>Checksum</td>
<td>&quot;c32eafae2587beff4b3b32f73743c3c61&quot;</td>
</tr>
<tr>
<td>path.virtual</td>
<td>Relative path from</td>
<td>&quot;/otherdir/mydocument.pdf&quot;</td>
</tr>
<tr>
<td>path.root</td>
<td>MD5 encoded parent path (internal use)</td>
<td>&quot;112aed383738239dbfe4485f024cd4ce1&quot;</td>
</tr>
<tr>
<td>attributes.owner</td>
<td>Owner name</td>
<td>&quot;david&quot;</td>
</tr>
<tr>
<td>attributes.group</td>
<td>Group name</td>
<td>&quot;staff&quot;</td>
</tr>
<tr>
<td>attributes.permissions</td>
<td>Permissions</td>
<td>764</td>
</tr>
<tr>
<td>external</td>
<td>Additional tags</td>
<td>(&quot;tenantId&quot;: 22, &quot;projectId&quot;: 3)</td>
</tr>
</tbody>
</table>

For more information about meta data, please read the TikaCoreProperties.

Here is a typical JSON document generated by the crawler:

```json
{
  "content": "This is a sample text available in page 1

This second part of the text is in Page 2

",   
  "meta": {
    "author": "David Pilato",
    "title": "Test Tika title",
    "date": "2016-07-07T16:37:00.000+0000"
  }
}
```

(continues on next page)
19.9 Search examples

You can use the content field to perform full-text search on

```
GET docs/_search
{
  "query" : {
    "match" : {
      "content" : "the quick brown fox"
    }
  }
}
```

You can use meta fields to perform search on.

```
GET docs/_search
{
  "query" : {
    "term" : {
      "file.filename" : "mydocument.pdf"
    }
  }
}
```

Or run some aggregations on top of them, like:

```
GET docs/_search
{
}
```
"size": 0,
"aggs": {
  "by_extension": {
    "terms": {
      "field": "file.extension"
    }
  }
}

New in version 2.2.
FSCrawler can expose a REST service running at http://127.0.0.1:8080/fscrawler. To activate it, launch FSCrawler with --rest option.

### 20.1 FSCrawler status

To get an overview of the running service, you can call GET / endpoint:

```bash
curl http://127.0.0.1:8080/fscrawler
```

It will give you a response similar to:

```json
{
   "ok" : true,
   "version" : "2.2",
   "elasticsearch" : "5.1.1",
   "settings" : {
      "name" : "fscrawler-rest-tests",
      "fs" : {
         "url" : "/tmp/es",
         "update_rate" : "15m",
         "json_support" : false,
         "filename_as_id" : false,
         "add_filesize" : true,
         "remove_deleted" : true,
         "store_source" : false,
         "index_content" : true,
         "attributes_support" : false,
         "raw_metadata" : true,
         "xml_support" : false,
         "index_folders" : true,
      }
   }
}
```
20.2 Uploading a binary document

To upload a binary, you can call POST `/_upload` endpoint:

```bash
echo "This is my text" > test.txt
curl -F "file=@test.txt" "http://127.0.0.1:8080/fscrawler/_upload"
```

It will give you a response similar to:

```
{
   "ok" : true,
   "filename" : "test.txt",
   "url" : "http://127.0.0.1:9200/fscrawler-rest-tests_doc/doc/dd18bf3a8ea2a3e53e2661c7fb53534"
}
```

The `url` represents the elasticsearch address of the indexed document. If you call:

```bash
curl http://127.0.0.1:9200/fscrawler-rest-tests_doc/doc/dd18bf3a8ea2a3e53e2661c7fb53534?pretty
```

You will get back your document as it has been stored by elasticsearch:

```
{
   "_index" : "fscrawler-rest-tests_doc",
   "_type" : "_doc",
   "_id" : "dd18bf3a8ea2a3e53e2661c7fb53534",
   "_version" : 1,
   "found" : true,
   "source" : {
      "content" : "This file contains some words.
      "meta" : {
          "raw" : {
              "X-Parsed-By" : "org.apache.tika.parser.DefaultParser",
          }
      }
   }
}
```
If you started FSCrawler in debug mode with --debug or if you pass debug=true query parameter, then the response will be much more complete:

```bash
echo "This is my text" > test.txt
curl -F "file=@test.txt" "http://127.0.0.1:8080/fscrawler/_upload?debug=true"
```

will give

```json
{
  "ok" : true,
  "filename" : "test.txt",
  "url" : "http://127.0.0.1:9200/fscrawler-rest-tests_doc/doc/dd18bf3a8ea2a3e53e2661c7fb53534",
  "doc" : {
    "content" : "This file contains some words.\n",
    "meta" : {
      "raw" : {
        "X-Parsed-By" : "org.apache.tika.parser.DefaultParser",
        "Content-Encoding" : "ISO-8859-1",
        "Content-Type" : "text/plain; charset=ISO-8859-1"
      }
    },
    "file" : {
      "extension" : "txt",
      "content_type" : "text/plain; charset=ISO-8859-1",
      "indexing_date" : "2017-01-04T14:05:10.325",
      "filename" : "test.txt"
    },
    "path" : {
      "virtual" : "test.txt",
      "real" : "test.txt"
    }
  }
}
```
20.3 Simulate Upload

If you want to get back the extracted content and its metadata but without indexing into elasticsearch you can use simulate=true query parameter:

```
echo "This is my text" > test.txt
curl -F "file@test.txt" "http://127.0.0.1:8080/fscrawler/_upload?debug=true&simulate=true"
```

20.4 Document ID

By default, FSCrawler encodes the filename to generate an id. Which means that if you send 2 files with the same filename test.txt, the second one will overwrite the first one because they will both share the same ID.

You can force any id you wish by adding id=YOUR_ID in the form data:

```
echo "This is my text" > test.txt
curl -F "file@test.txt" -F "id=my-test" "http://127.0.0.1:8080/fscrawler/_upload"
```

There is a specific id named _auto_ where the ID will be autogenerated by elasticsearch. It means that sending twice the same file will result in 2 different documents indexed.

20.5 Additional tags

Add custom tags to the document. In case you want to do filtering on those tags (examples are projectId or tenantId). These tags can be assigned to an external object field. As you can see in the json, you are able to overwrite the content field. meta, file and path fields can be overwritten as well. To upload a binary with additional tags, you can call POST /_upload endpoint:

```
{
  "content": "OVERWRITE CONTENT",
  "external": {
    "tenantId": 23,
    "projectId": 34,
    "description": "these are additional tags"
  }
}
```

```
echo "This is my text" > test.txt
echo "{"content":"OVERWRITE CONTENT","external":{"tenantId": 23,"projectId": 34,"description":"these are additional tags"}}" > tags.txt
curl -F "file@test.txt" -F "tags=tags.txt" "http://127.0.0.1:8080/fscrawler/_upload"
```

The field external doesn’t necessarily be a flat structure. This is a more advanced example:

```
{
  "external": {
    "tenantId": 23,
    "company": "shoe company",
    "projectId": 34,
    "project": "business development",
    "daysOpen": [
    ...
    
    ]
}
```
Attention: Only standard FSCrawler fields can be set outside external field name.

20.6 Specifying an elasticsearch index

By default, fscrawler creates document in the index defined in the _settings.yaml file. However, using the REST service, it is possible to require fscrawler to use different indexes, by adding index=YOUR_INDEX in the form data:

```bash
echo "This is my text" > test.txt
curl -F "file=@test.txt" -F "index=my-index" "http://127.0.0.1:8080/fscrawler/_upload"
```

20.7 Enabling CORS

To enable Cross-Origin Request Sharing you will need to set enable_cors: true under rest in your job settings. Doing so will enable the relevant access headers on all REST service resource responses (for example /fscrawler and /fscrawler/_upload).

You can check if CORS is enabled with:

```bash
curl -I http://127.0.0.1:8080/fscrawler/
```

The response header should contain Access-Control-Allow-* parameters like:

- Allow-Origin: *
- Access-Control-Allow-Headers: origin, content-type, accept, authorization
- Access-Control-Allow-Credentials: true
- Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE, OPTIONS, HEAD
20.8 REST settings

Here is a list of REST service settings (under `rest.` prefix):

<table>
<thead>
<tr>
<th>Name</th>
<th>Default value</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rest.url</code></td>
<td><a href="http://127.0.0.1:8080/fscrawler">http://127.0.0.1:8080/fscrawler</a></td>
<td>Rest Service URL</td>
</tr>
<tr>
<td><code>rest.enable_cors</code></td>
<td>false</td>
<td>Enables or disables Cross-Origin Resource Sharing globally for all resources</td>
</tr>
</tbody>
</table>

**Tip:** Most *Local FS settings* (under `fs.*` in the settings file) also affect the REST service, e.g. `fs.indexed_chars`. Local FS settings that do **not** affect the REST service are those such as `url`, `update_rate`, `includes`, `excludes`.

REST service is running at http://127.0.0.1:8080/fscrawler by default.

You can change it using `rest` settings:

```yaml
name: "test"
rest:
  url: "http://192.168.0.1:8180/my_fscrawler"
```

It also means that if you are running more than one instance of FS crawler locally, you can (must) change the port as it will conflict.
CHAPTER 21

Building the project

This project is built with Maven. Source code is available on GitHub. Thanks to JetBrains for the IntelliJ IDEA License!

21.1 Clone the project

Use git to clone the project locally:

```
$ git clone git@github.com:dadoonet/fscrawler.git
$ cd fscrawler
```
21.2 Build the artifact

To build the project, run:

```
mvn clean package
```

The final artifacts are available in `distribution/esX/target` directory where `X` is the elasticsearch major version target.

**Tip:** To build it faster (without tests), run:

```
mvn clean package -DskipTests
```

21.3 Integration tests

When running from the command line with `mvn` integration tests are ran against all supported versions. This is done by running a Docker instance of elasticsearch using the expected version.

A HTTP server is also started on port 8080 during the integration tests, alternatively the assigned port can be set with `-Dtests.rest.port=8090` argument.

21.3.1 Run tests from your IDE

To run integration tests from your IDE, you need to start tests in `fscrawler-it-common` module. But you need first to specify the Maven profile to use and rebuild the project.

- `es-7x` for Elasticsearch 7.x
- `es-6x` for Elasticsearch 6.x

21.3.2 Run tests with an external cluster

To run the test suite against an elasticsearch instance running locally, just run:

```
```

**Tip:** If you want to run against a version 6, run:

```
```

**Hint:** If you are using a secured instance, use `tests.cluster.user`, `tests.cluster.pass` and `tests.cluster.url`:
mvn verify -pl fr.pilato.elasticsearch.crawler:fscrawler-it-v7 \
-Dtests.cluster.user=elastic \n-Dtests.cluster.pass=changeme \n-Dtests.cluster.url=https://127.0.0.1:9200 \

**Hint:** To run tests against another instance (ie. running on Elasticsearch service by Elastic, you can also use `tests.cluster.url` to set where elasticsearch is running:

mvn verify -pl fr.pilato.elasticsearch.crawler:fscrawler-it-v7 \
-Dtests.cluster.user=elastic \n-Dtests.cluster.pass=changeme \n-Dtests.cluster.url=https://XYZ.es.io:9243

Or even easier, you can use the Cloud ID available on your Cloud Console:

mvn verify -pl fr.pilato.elasticsearch.crawler:fscrawler-it-v7 \
-Dtests.cluster.user=elastic \n-Dtests.cluster.pass=changeme \n-Dtests.cluster.cloud_id=fscrawler:ZXVyb3BlLXdlc3QxLmdjcC5jbG91ZC5lcy5pbyQxZDFlYTk5Njg4Nzc0NWE2YTJiN2NiNzkzMTUzNDhhMMyOTk1MDI3MzZmZGQ0OTI5OTE5NzJlOTk3ZmU3Nw==

### 21.3.3 Using security feature

Integration tests are run by default against a standard Elasticsearch cluster, which means with no security feature activated.

New in version 2.7.

You can run all the integration tests against a secured cluster by using the `security` profile:

```sh
mvn verify -Psecurity
```

Note that secured tests are using by default `changeme` as the password. You can change this by using `tests.cluster.pass` option:

```sh
mvn verify -Psecurity -Dtests.cluster.pass=mystrongpassword
```

### 21.3.4 Tests options

Some options are available from the command line when running the tests:

- `tests.leaveTemporary` leaves temporary files after tests. `false` by default.
- `tests.parallelism` how many JVM to launch in parallel for tests. Set to `auto` by default which means that it depends on the number of processors you have.
- `tests.output` what should be displayed to the console while running tests. By default it is set to `onError` but can be set to `always`
- `tests.verbose` `false` by default
- `tests.seed` if you need to reproduce a specific failure using the exact same random seed
- `tests.timeoutSuite` how long a single can run. It’s set by default to `600000` which means 5 minutes.

### 21.3. Integration tests
• **tests.locale** by default it’s set to **random** but you can force the locale to use.
• **tests.timezone** by default it’s set to **random** but you can force the timezone to use.

For example:

```bash
mvn install -rf :fscrawler-it -Dtests.output=always
```

## 21.4 Check for vulnerabilities (CVE)

The project is using **OSS Sonatype service** to check for known vulnerabilities. This is ran during the **verify** phase.

Sonatype provides this service but with a anonymous account, you might be limited by the number of tests you can run during a given period.

If you have an existing account, you can use it to bypass this limit for anonymous users by setting **sonatype.username** and **sonatype.password**:

```bash
mvn verify -DskipTests \
  -Dsonatype.username=youremail@domain.com \
  -Dsonatype.password=yourverysecuredpassword
```

If you want to skip the check, you can run with **-Dossindex.fail=false**:

```bash
mvn clean install -Dossindex.fail=false
```
This project uses ReadTheDocs to build and serve the documentation.

If you want to run the generation of documentation (recommended!), you need to have Python installed. Then install `sphinx`:

```bash
$ pip install sphinx sphinx-autobuild
```

Assuming you have Python already, install Sphinx:

```bash
$ pip install sphinx sphinx-autobuild
```

Go to the `docs` directory and build the html documentation:

```bash
$ cd docs
$ make html
```

Just open then `target/html/index.html` page in your browser.

**Hint:** You can hot reload your changes by using `sphinx-autobuild`:

```bash
$ sphinx-autobuild source target/html
```

Then just edit the documentation and look for your changes at `http://127.0.0.1:8000`

To learn more about the reStructuredText format, please look at the [basic guide](#).
To release the project, run:

$ release.sh

And follow the instructions.

**Note:** Only developers with write rights to the sonatype repository under `fr.pilato` space can perform the release.
Important: This software is licensed under the Apache 2 license, quoted below.

Copyright 2011-2020 David Pilato

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.
Incompatible 3rd party library licenses

Some libraries are not Apache2 compatible. Therefore they are not packaged with FSCrawler so you need to download and add manually them to the lib directory:

- for JBIG2 images, you need to add levigo-jbig2-imageio:2.0 library
- for TIFF images, you need to add jai-imageio-core:1.4.0 library
- for JPEG 2000 (JPX) images, you need to add jai-imageio-jpeg2000:1.3.0 library

See pdfbox documentation for more details.
Thanks to JetBrains for the IntelliJ IDEA License!