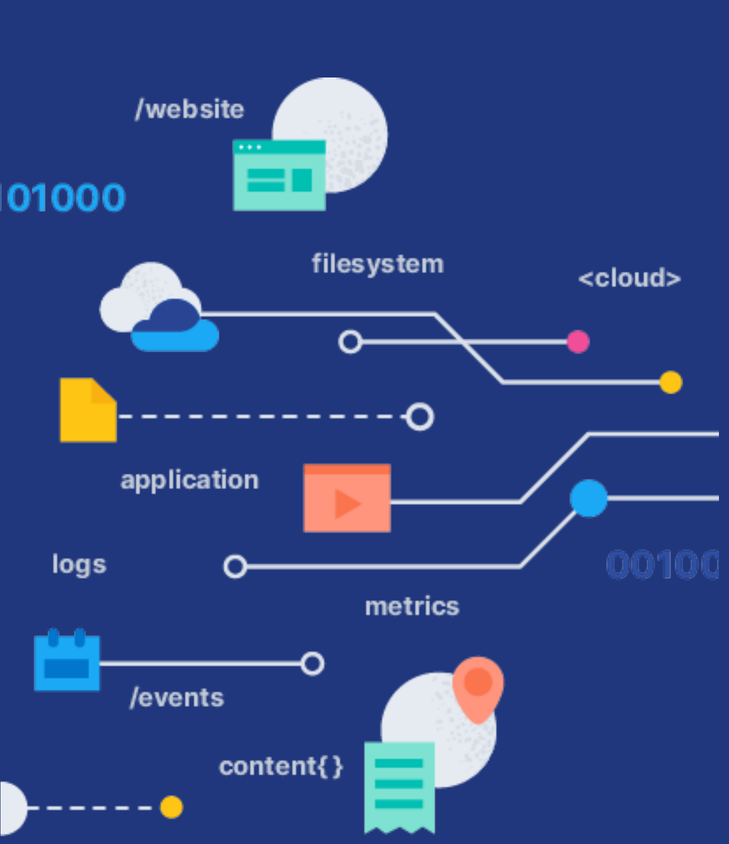
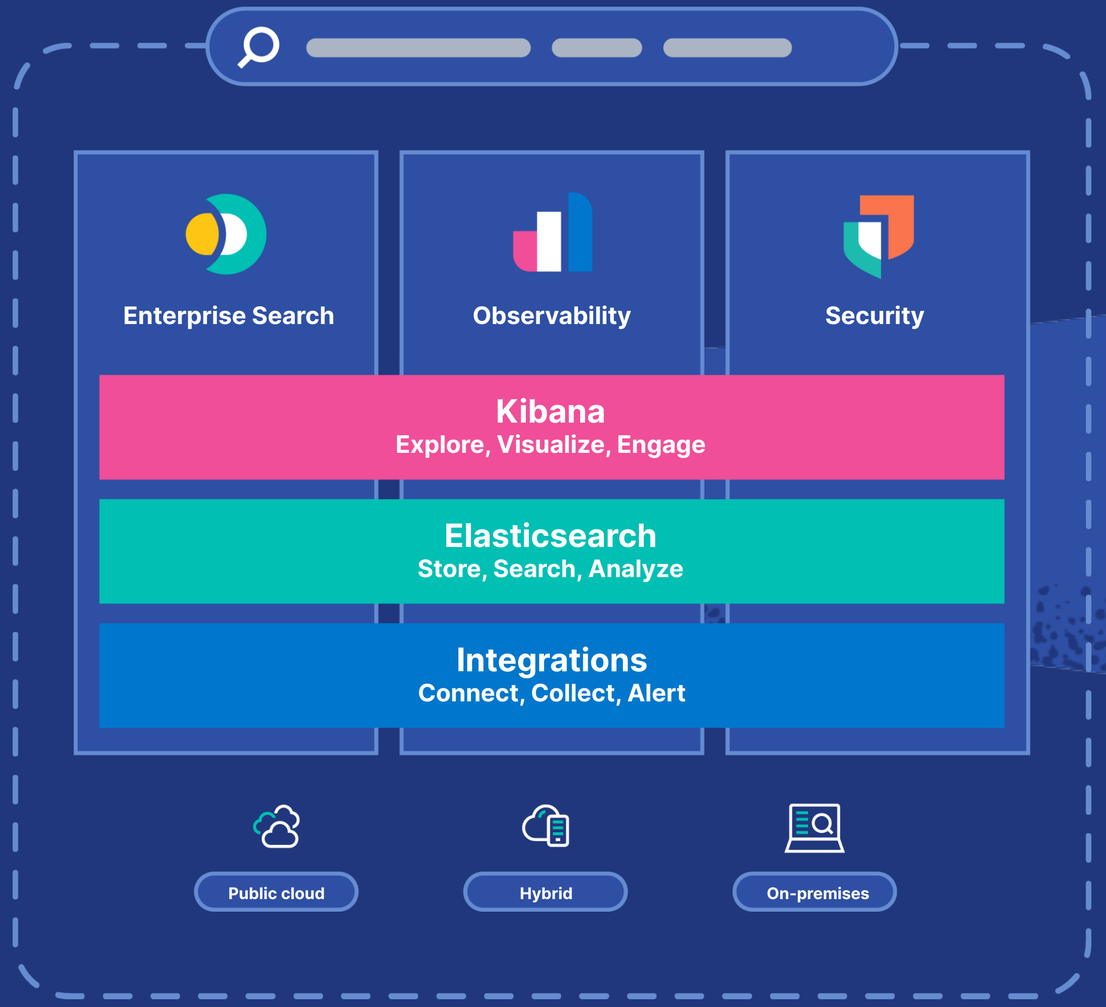
Indexing your office documents
with Elastic and FSCrawler

David Pilato
Developer I Evangelist, Community
@dadoonet

SNOWCAMP

The Elastic Search Platform





elastic

The Apache Software Foundation

Apache Tika - a content analysis toolkit

The Apache Tika™ toolkit detects and extracts metadata and text from over a thousand different file types (such as PPT, XLS, and PDF). All of these file types can be parsed through a single interface, making Tika useful for search engine indexing, content analysis, translation, and much more. You can find the latest release on the download page. Please see the Getting Started page for more information on how to start using Tika.

The Parser and Detector pages describe the main interfaces of Tika and how they work.

For more in-depth documentation, see our wiki, especially for tika-server.

If you're interested in contributing to Tika, please see the Contributing page or send an email to the Tika development list.

Tika is a project of the Apache Software Foundation, and was formerly a subproject of Apache Lucene.

Latest News

2 May 2022: Apache Tika Release

Apache Tika 2.4.0 has been released! This release includes new mime detection for http-responses, frictionless data packages, DGN files and others. Added basic parsers for WARC and WACZ. Added configuration for metadata write filters, custom content handler decorators and

elastic

Please note that Apache Tika is able to detect a much wider range of formats than those listed below, this page only documents those formats from which Tika is able to extract metadata and/or textual content.

Parsing a stream

and getting content and metadata

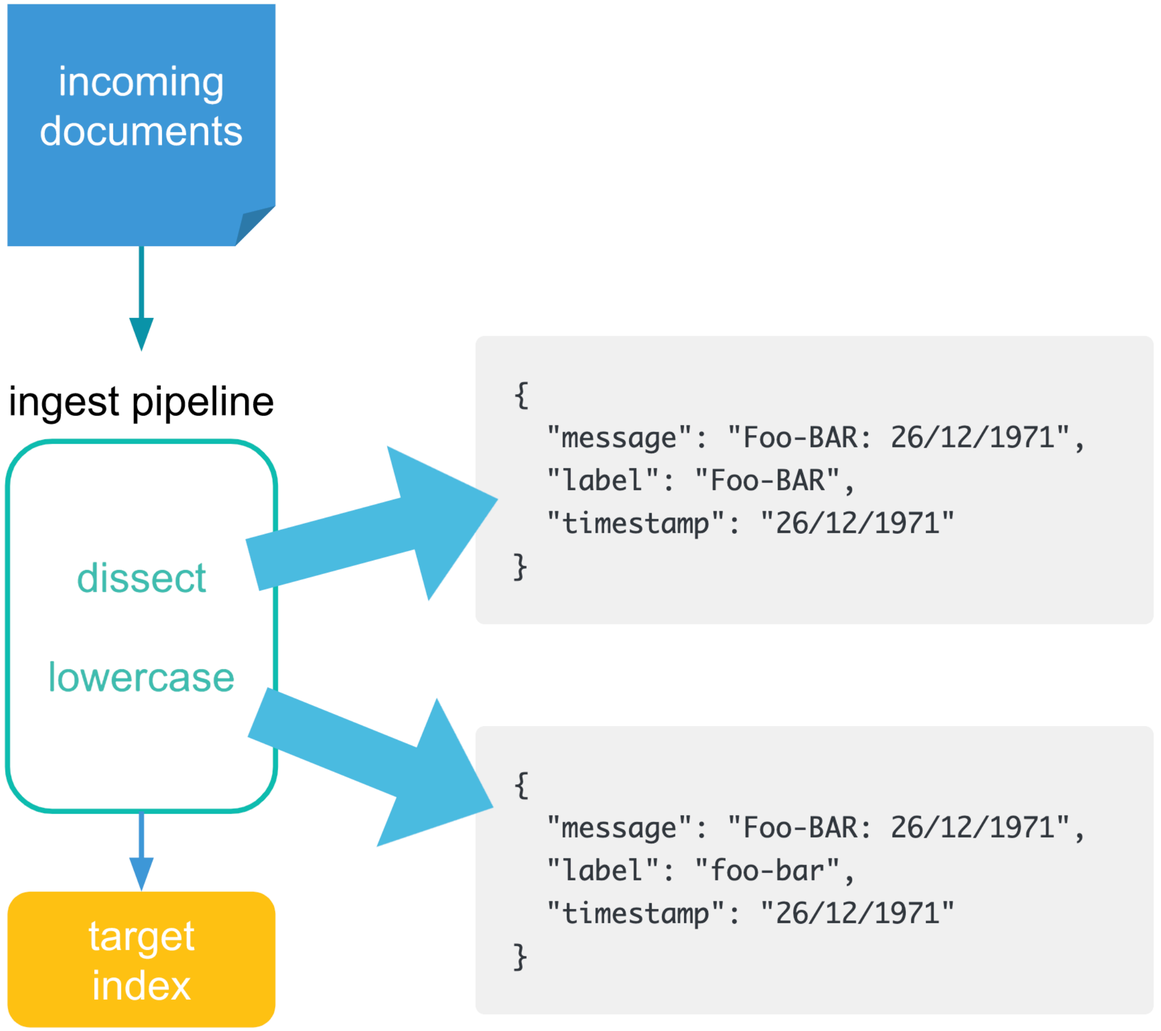
```
static void extractTextAndMetadata(InputStream stream) throws Exception {
    BodyContentHandler handler = new BodyContentHandler();
    Metadata metadata = new Metadata();
    try (stream) {
        new DefaultParser().parse(stream, handler, metadata, new ParseContext());
        String extractedText = handler.toString();
        String title = metadata.get(TikaCoreProperties.TITLE);
        String keywords = metadata.get(TikaCoreProperties.KEYWORDS);
        String author = metadata.get(TikaCoreProperties.CREATOR);
    }
}
```

Demo

ingest-attachment processor
extracting from BASE64 or CBOR

An ingest pipeline

```
{
  "message": "Foo-BAR: 26/12/1971"
}
```



elastic

ingest-attachment processor

using Tika behind the scene

Attachment processor

edit

The attachment processor lets Elasticsearch extract file attachments in common formats (such as PPT, XLS, and PDF) by using the Apache text extraction library Tika.

The source field must be a base64 encoded binary. If you do not want to incur the overhead of converting back and forth between base64, you can use the CBOR format instead of JSON and specify the field as a bytes array instead of a string representation. The processor will skip the base64 decoding then.

Using the attachment processor in a pipeline

edit

Table 4. Attachment options

Name	Required	Default	Description
`field`	yes	-	The field to get the base64 encoded field from
`target_field`	no	attachment	The field that will hold the attachment information
`indexed_chars`	no	100000	The number of chars being used for extraction to prevent huge fields. Use `-1` for no limit.
`indexed_chars_field`	no	`null`	Field name from which you can overwrite the number of chars being used for extraction. See `indexed_chars`.
`properties`	no	all properties	Array of properties to select to be stored. Can be `content`, `title`, `name`, `author`, `keywords`, `date`, `content_type`, `content_length`, `language`

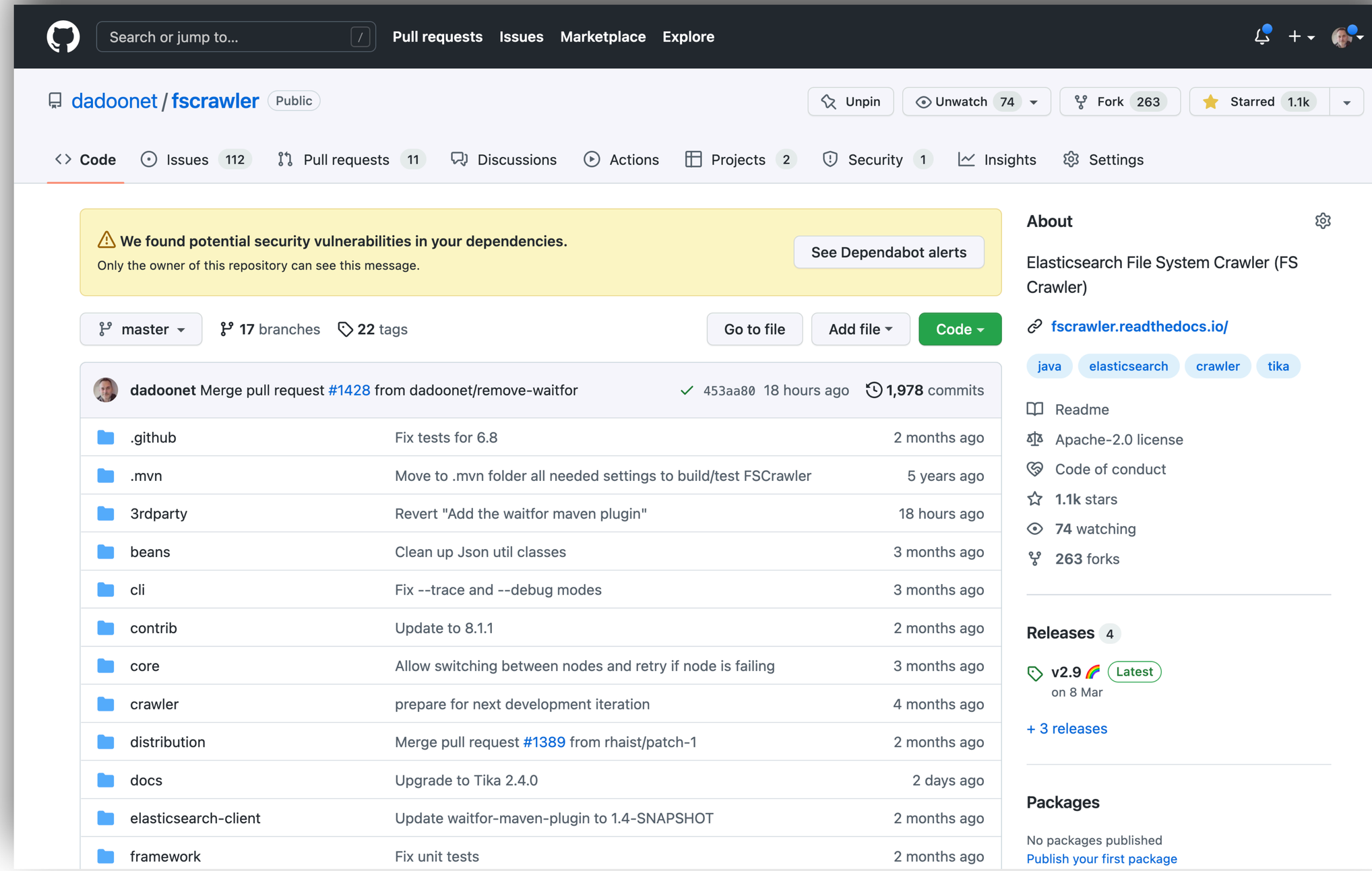
elastic

Demo

https://cloud.elastic.co

FSCrawler

You know, for files…


Search or jump to...
Pull requests
Issues
Marketplace
Explore
dadoonet / fscrawler
Public
Unpin
Unwatch 74
Fork 263
Starred 1.1k
Code
Issues 112
Pull requests 11
Discussions
Actions
Projects 2
Security 1
Insights
Settings
We found potential security vulnerabilities in your dependencies.
Only the owner of this repository can see this message.
See Dependabot alerts
master
17 branches
22 tags
Go to file
Add file
Code
dadoonet Merge pull request #1428 from dadoonet/remove-waitfor
453aa80 18 hours ago
1,978 commits
.github Fix tests for 6.8 2 months ago
.mvn Move to .mvn folder all needed settings to build/test FSCrawler 5 years ago
3rdparty Revert "Add the waitfor maven plugin" 18 hours ago
beans Clean up Json util classes 3 months ago
cli Fix --trace and --debug modes 3 months ago
contrib Update to 8.1.1 2 months ago
core Allow switching between nodes and retry if node is failing 3 months ago
crawler prepare for next development iteration 4 months ago
distribution Merge pull request #1389 from rhaist/patch-1 2 months ago
docs Upgrade to Tika 2.4.0 2 days ago
elasticsearch-client Update waitfor-maven-plugin to 1.4-SNAPSHOT 2 months ago
framework Fix unit tests 2 months ago
About
Elasticsearch File System Crawler (FS Crawler)
fscrawler.readthedocs.io/
java
elasticsearch
crawler
tika
Readme
Apache-2.0 license
Code of conduct
1.1k stars
74 watching
263 forks
Releases 4
v2.9
Latest
on 8 Mar
+ 3 releases
Packages
No packages published
Publish your first package

Disclaimer

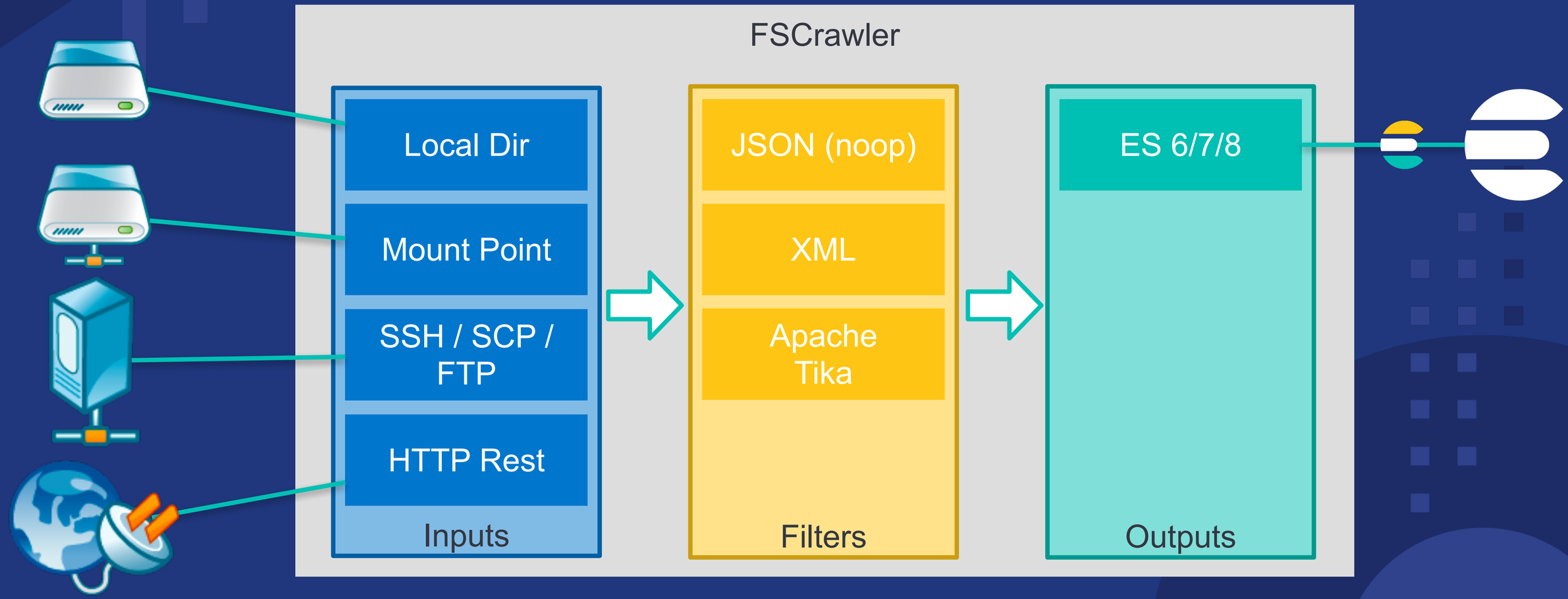
This project is a community project.
It is not officially supported by Elastic.
Support is only provided by FSCrawler community
on discuss and stackoverflow.

http://discuss.elastic.co/
https://stackoverflow.com/questions/tagged/fscrawler

elastic

FSCrawler

Architecture



FSCrawler

Key Features

- Much more formats than ingest attachment processor
- OCR (Tesseract)
- ~~Much more metadata than ingest attachment processor~~
  (See https://fscrawler.readthedocs.io/en/latest/admin/fs/elasticsearch.html#generated-fields)
- Extraction of non standard metadata

elastic

Documentation

- https://fscrawler.readthedocs.io/
- https://fscrawler.readthedocs.io/en/latest/user/tutorial.html
- https://fscrawler.readthedocs.io/en/latest/user/formats.html
- https://fscrawler.readthedocs.io/en/latest/admin/fs/index.html

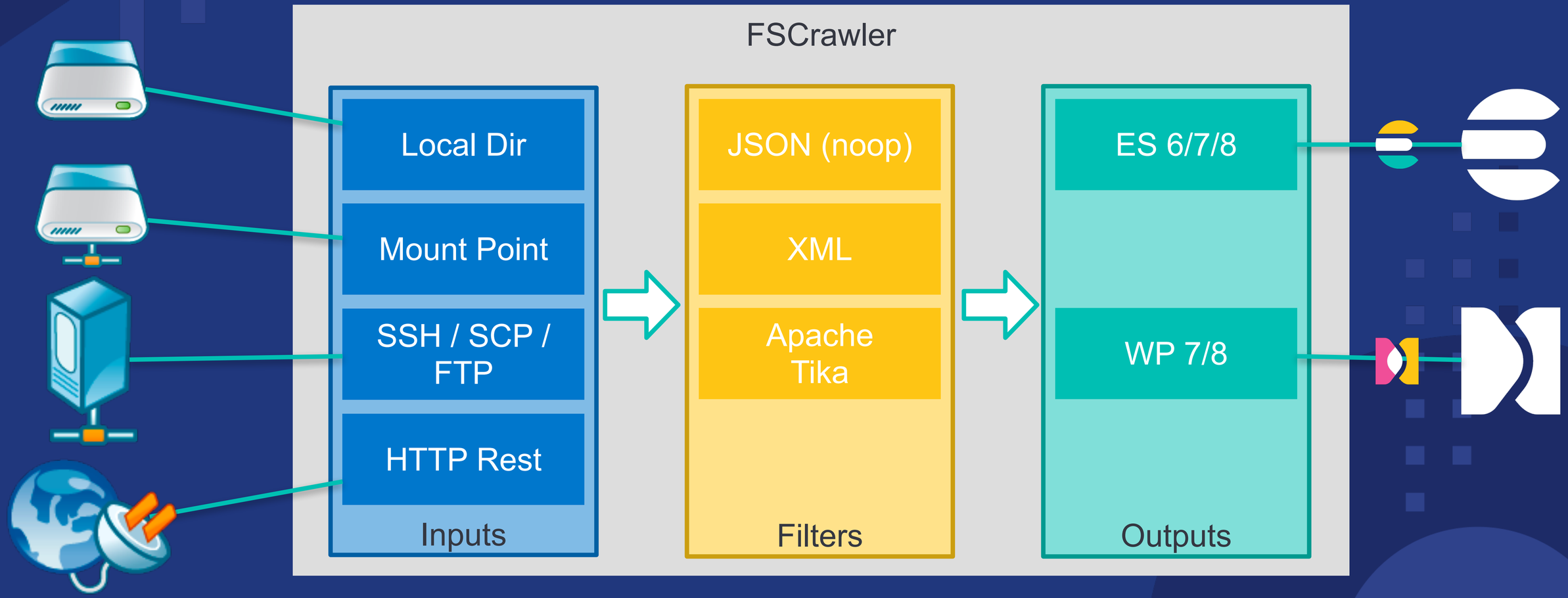
Demo

https://cloud.elastic.co

FSCrawler
even better with a UI

FSCrawler

Architecture



Demo

https://cloud.elastic.co

Beta
8.2

Network drives connector package

for Enterprise Search

https://github.com/elastic/enterprise-search-network-drives-connector/

elastic

FSCrawler v3

Roadmap ("It depends")

Extended CLI parameters

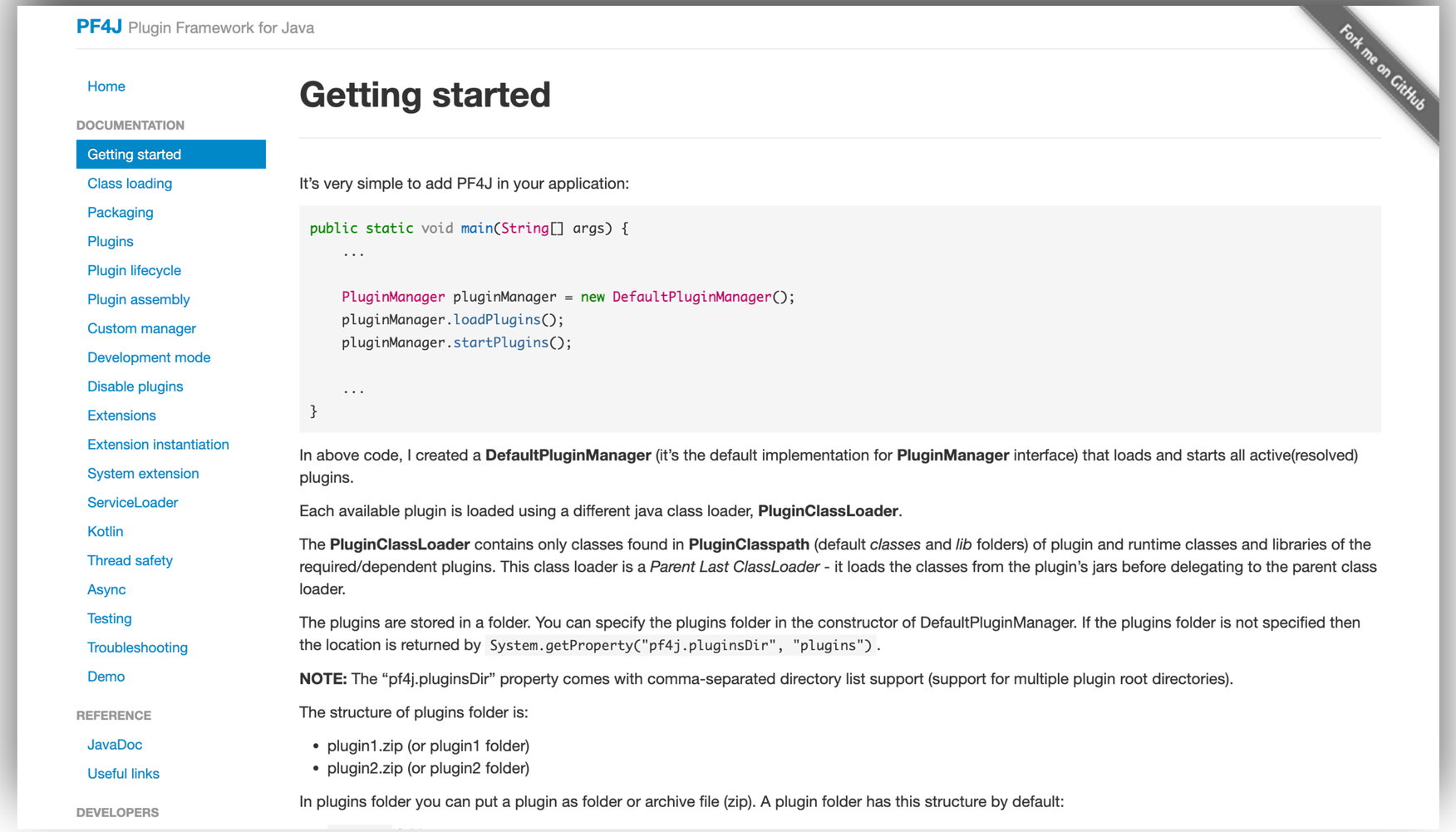
https://github.com/dadoonet/fscrawler/issues/857

```
$ bin/fscrawler --input.fs.dir=/path/to/files \
                --filter.tika.indexed_chars=100% \
                --output.elasticsearch=https://localhost:9200


$ bin/fscrawler --input.fs.dir=/path/to/files \
                --filter.tika.lang_detect=true \
                --output.wpsearch=https://localhost:3002
```

Add support for plugins (inputs, filters and outputs) with pf4j

https://github.com/dadoonet/fscrawler/issues/1114



PF4J Plugin Framework for Java

Fork me on GitHub

Home

DOCUMENTATION

- Getting started
- Class loading
- Packaging
- Plugins
- Plugin lifecycle
- Plugin assembly
- Custom manager
- Development mode
- Disable plugins
- Extensions
- Extension instantiation
- System extension
- ServiceLoader
- Kotlin
- Thread safety
- Async
- Testing
- Troubleshooting
- Demo

REFERENCE

- JavaDoc
- Useful links

DEVELOPERS

Getting started

It's very simple to add PF4J in your application:

```
public static void main(String[] args) {
    ...

    PluginManager pluginManager = new DefaultPluginManager();
    pluginManager.loadPlugins();
    pluginManager.startPlugins();

    ...
}
```

In above code, I created a **DefaultPluginManager** (it's the default implementation for **PluginManager** interface) that loads and starts all active(resolved) plugins.

Each available plugin is loaded using a different java class loader, **PluginClassLoader**.

The **PluginClassLoader** contains only classes found in **PluginClasspath** (default *classes* and *lib* folders) of plugin and runtime classes and libraries of the required/dependent plugins. This class loader is a *Parent Last ClassLoader* - it loads the classes from the plugin's jars before delegating to the parent class loader.

The plugins are stored in a folder. You can specify the plugins folder in the constructor of DefaultPluginManager. If the plugins folder is not specified then the location is returned by `System.getProperty("pf4j.pluginsDir", "plugins")`.

NOTE: The "pf4j.pluginsDir" property comes with comma-separated directory list support (support for multiple plugin root directories).

The structure of plugins folder is:

- plugin1.zip (or plugin1 folder)
- plugin2.zip (or plugin2 folder)

In plugins folder you can put a plugin as folder or archive file (zip). A plugin folder has this structure by default:

elastic

Add rsync input

https://github.com/dadoonet/fscrawler/issues/377

```
$ bin/fscrawler --input.rsync.port=14415

$ rsync --port=14415 -r example localhost::Uploads
```

Add S3 input

https://github.com/dadoonet/fscrawler/issues/377

```
$ bin/fscrawler --input.s3.object=s3://foo/bar.txt

$ bin/fscrawler --input.s3.bucket=s3://foo
```

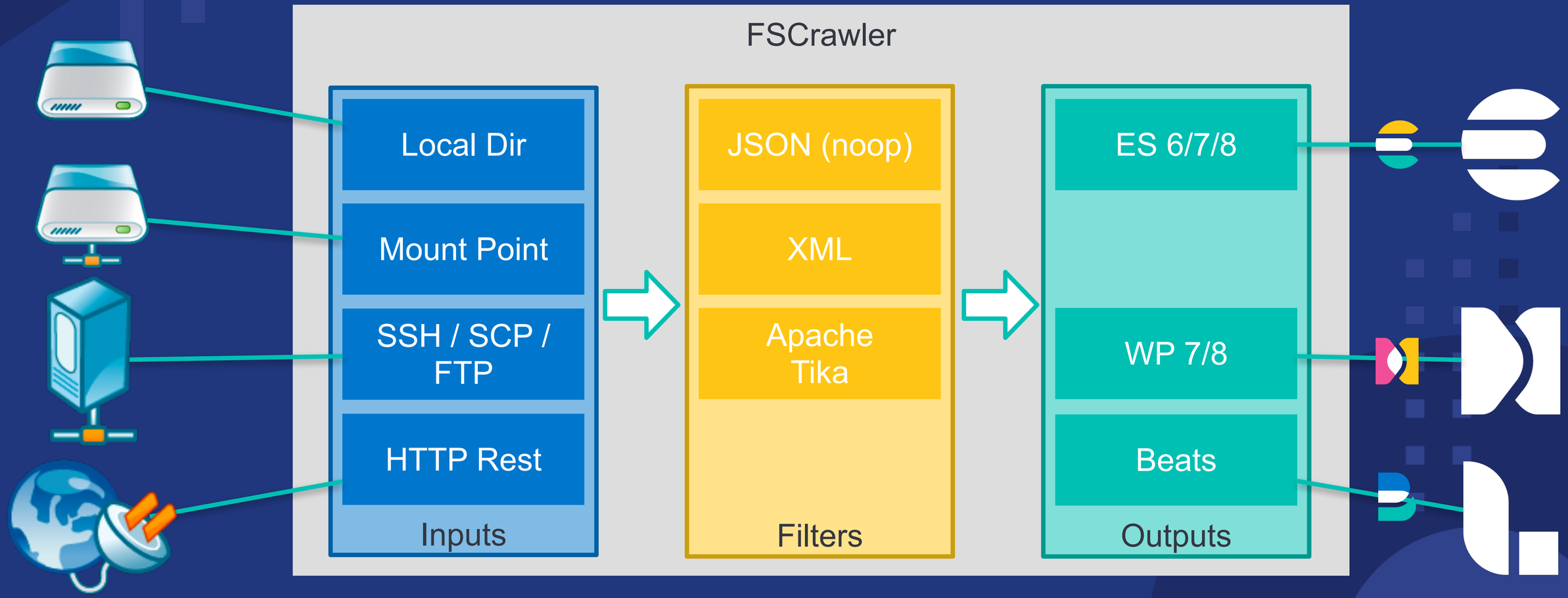
Add Dropbox input

https://github.com/dadoonet/fscrawler/issues/264

```
$ bin/fscrawler --input.dropbox.access_token=XYZ \
                --input.dropbox.dir=/path/to/files
```

Add Beats output

https://github.com/dadoonet/fscrawler/issues/682



Add Beats output

https://github.com/dadoonet/fscrawler/issues/682

```
$ bin/logstash -e '
input {
  beats {
    port => 5044
  }
}

output {
  elasticsearch {
    hosts => ["https://localhost:9200"]
  }
}'

$ bin/fscrawler --output.beats.url=https://localhost:5044
```

Manage jobs from the REST Service

https://github.com/dadoonet/fscrawler/issues/1549

```
# Create
curl -XPUT http://127.0.0.1:8080/_jobs/my_job -d '{
  "type": "fs",
  "fs": {
    "url": "file://foo/bar.txt"
  }
}
# Start / Stop
curl -XPOST http://127.0.0.1:8080/_jobs/my_job/_start
curl -XPOST http://127.0.0.1:8080/_jobs/my_job/_stop
# Job info and status
curl -XGET http://127.0.0.1:8080/_jobs/my_job
# Remove the job
curl -XDELETE http://127.0.0.1:8080/_jobs/my_job
```

Read from any FS Provider using the REST Service

https://github.com/dadoonet/fscrawler/issues/1247

```
curl -XPOST http://127.0.0.1:8080/_upload -d '{
  "type": "fs",
  "fs": {
    "url": "file://foo/bar.txt"
  }
}

curl -XPOST http://127.0.0.1:8080/_upload -d '{
  "type": "s3",
  "s3": {
    "url": "s3://foo/bar.txt"
  }
}
```

Other ideas

- New local file crawling implementation (WatchService): #399
- Store jobs, configurations, status in Elasticsearch: #717
- Switch to ECS format for the most common fields: #677
- Extract ACL informations: #464

Thanks!

PR are warmly welcomed!

https://github.com/dadoonet/fscrawler

SNOWCAMP