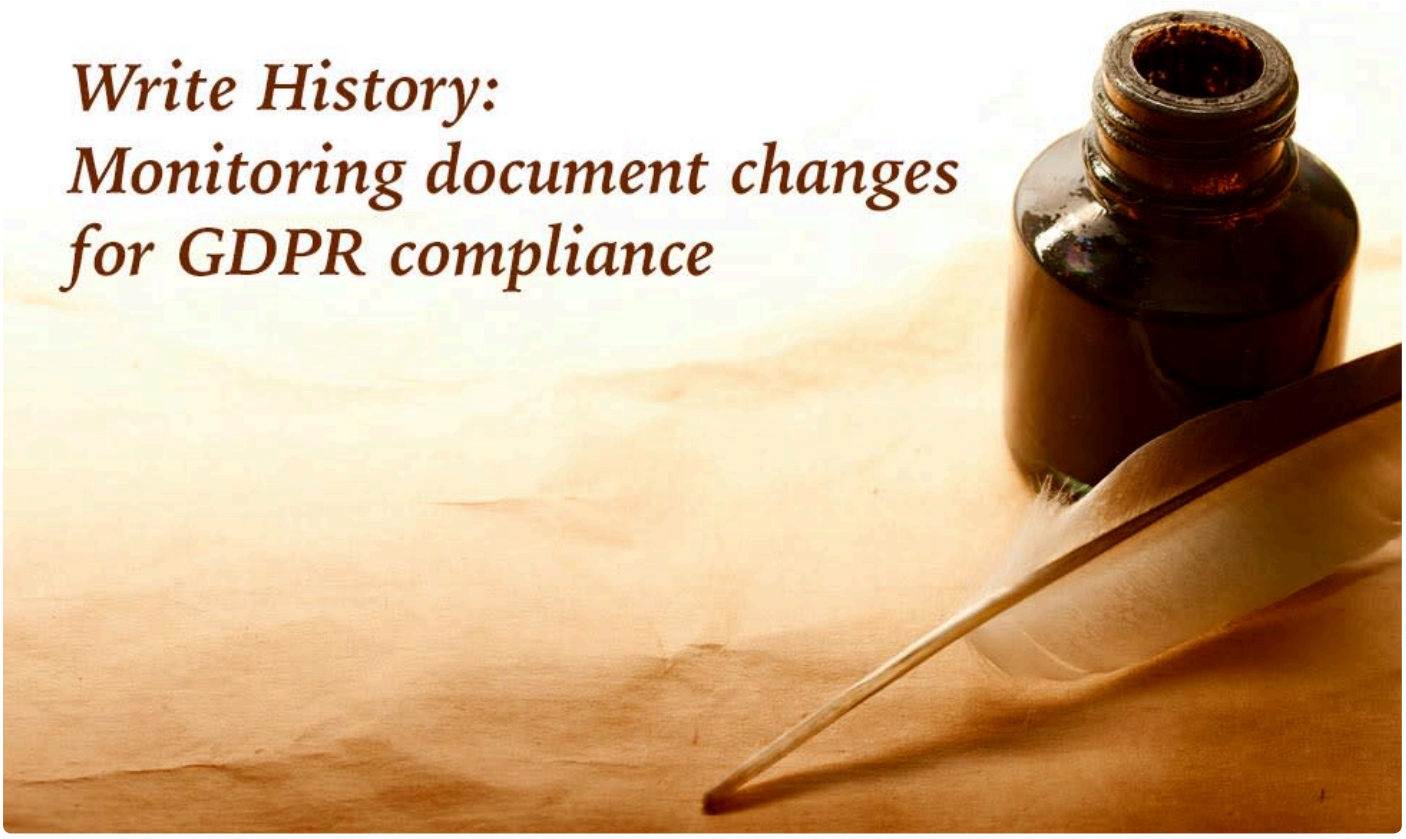
Write History: Monitoring document changes for GDPR compliance

Jochen Kressin · 2018-04-21



Many, if not all compliance regulations place a heavy burden on anyone storing sensitive user information. GDPR in particular gives the user extensive rights regarding any personal data you store. You need to know exactly when personal data has been created, what data it is, for what purpose it is used, and when and by whom it has been accessed. In addition, the user can demand changes and deletion of the data at any time (*right to be forgotten*). Failures to comply with these regulations can result in very hefty fines. The Search Guard Compliance Edition provides you with an extensive set of features to handle these requirements, like the Write History audit trail.

To stay compliant with GDPR you need to be able to answer these questions regarding personal data of your users or customers:

- When has the data been created, and by whom?
- How did it change over time, and who made the changes?
- Who has access to the data, and what exact information did this person access?
- When has the data been deleted, and who deleted it?

All these events are required for staying compliant with GDPR: At any time, any customer can request information about what personal information has been stored, who has accessed to it and for what

purpose. Likewise, the customer can demand changes to his or her data, and request that all personal information has to be deleted ("right to be forgotten").

Out of the box Elasticsearch does not have any means to monitor read or write access to data stored in your cluster. Even with additional plugins this is not possible at the time of writing. Meet the Search Guard Read- and Write History audit trails, which brings exactly this feature to Elasticsearch. A huge step on your way to GDPR compliance.

Track the lifetime of your documents

The Write History makes it possible to monitor which user has created, changed or deleted documents, what the changes were and when they occurred. By using the Write History you can keep an audit trail over the complete lifetime of your documents, and store it safely in an Immutable Elasticsearch Index.

To activate the Write History, first configure for what indices you would like to enable it. You simply list all watched indices in *elasticsearch.yml* like:

```
searchguard.compliance.history.write.watched_indices:
 - customers
```

Next, tell Search Guard where you want to store the generated audit trail. You can use the powerful audit event routing to configure one or more storage endpoints for each event category. For the sake of simplicity, we will store all events in an index called *auditlog* on the same Elasticsearch cluster. This just requires minimal configuration in *elasticsearch.yml*:

```
searchguard:
 audit:
   type: internal_elasticsearch
   config:
     index: "auditlog"
```

You can configure if you want to record and store metadata only (here: field names), or also the actual values. We want to track how data has changed over time, so we will include the field values in the audit logs:

```
searchguard.compliance.history.metadata_only: false
```

Document creation

Now we're all set and can start our nodes. Let's create a new document:

```
curl -u admin:password \
  -H 'Content-Type: application/json' \
  -XPOST "https://example.com:9200/customers/_doc/1?pretty" -d \
  '{
    "firstname": "John",
    "lastname": "Doe",
    "email": "john.doe@example.com",
    "address":"327 West Orchard Road, FL 33056",
    "passportnumber": "311955855"
  }'
```

The creation of the document creates a new compliance event which, according to the audit log settings in *elasticsearch.yml*, is stored in the *auditlog* index:

```
curl -u admin:password "https://example.com:9200/auditlog/_search?
pretty&sort=audit_utc_timestamp:desc"
```

Which give us:

```
{
 "audit_category": "COMPLIANCE_DOC_WRITE",
 "audit_compliance_operation": "CREATE",
 ...
 "audit_request_body": "{
     "firstname": "John",
     "lastname": "Doe",
     "email": "john.doe@example.com",
     "address": "327 West Orchard Road, FL 33056",
     "passportnumber":"311955855"
 }",
 ...
 "audit_utc_timestamp" : "2018-04-20T20:23:04.161+00:00",
 "audit_trace_doc_id": "1",
 "audit_request_remote_address" : "172.16.0.1",
 "audit_request_effective_user": "admin",
 "audit_trace_indices": [
   "customers"
 ]
 ...
}
```

Apart from common information like the date (*audit_utc_timestamp*) or the remote IP address (*audit_request_remote_address*) Search Guard also stores the name of the user that inserted the document (*audit_request_effective_user*), the index name (*audit_trace_indices*) and the document id (*audit_trace_doc_id*).

Since we decided to log both the field names and the values, Search Guard logs the complete content of the inserted document in the *audit_request_body* field. Thus, we have stored what our document looked like when it had been first created.

Changing a document

Next, let's make some changes by issuing an index request. In this request we will remove the field *passportnumber*, change the value of the field *address* and add a new field *ssn*:

```
curl -u admin:password \
 -H 'Content-Type: application/json' \
 -XPUT "https://example.com:9200/customers/_doc/1?pretty" -d \
 '{
   "firstname": "John",
   "lastname": "Doe",
   "email": "john.doe@example.com",
   "address":"9718 Sutor Rd., NY 11040",
   "ssn": "536689991"
 }'
```

The generated event resembles exactly the additions, changes and deletions we have made:

```
{
 "audit_category": "COMPLIANCE_DOC_WRITE",
 "audit_compliance_operation": "UPDATE",
 ...
 "audit_compliance_diff_content": "[
   {"op":"replace","path":"/address","value":"9718 Sutor Rd., NY 11040"},
   {"op":"add","path":"/ssn","value":"536689991"},
   {"op":"remove","path":"/passportnumber"}
 ]",
 "audit_utc_timestamp": "2018-04-21T03:01:09.244+00:00",
 "audit_trace_doc_id": "1",
 "audit_request_remote_address" : "172.16.0.1",
 "audit_request_effective_user": "admin",
 "audit_trace_resolved_indices": [
   "customers"
 ],
 ...
}
```

All changes to the document are stored in JSON patch format. The JSON patch format

" [...] defines a JSON document structure for expressing a sequence of operations to apply to a JavaScript Object Notation (JSON) document."

(https://tools.ietf.org/html/rfc6902)

JSON patch acts like a diff between objects and defines

- the operation that was applied (add, replace, remove)
- the path to the affected field in the JSON object
- the new value (optional)

So far we are able to monitor and store when a document is created, what the initial values are and how it changed over time. If a customer demands information about what personal information you store or requires you to change any data, you can easily fulfil this request by providing the write history of his or

her data directly. This will help you tremendously on your way to GDPR compliance for your Elasticsearch cluster.

Deleting documents: Right to be forgotten

Let's assume a customer executes the *right to be forgotten*: In this case, all data that can be associated with the user has to be deleted. You cannot just state "Ok, we've deleted everything in question", rather, you need to prove to the customer that the data has really been deleted.

So, let's just try to delete our newly created document and see how the Write History can help:

```
curl -u admin:password  \
 -XDELETE "https://example.com:9200/customers/_doc/1"
```

As expected, this results in an DELETE event in the audit log index:

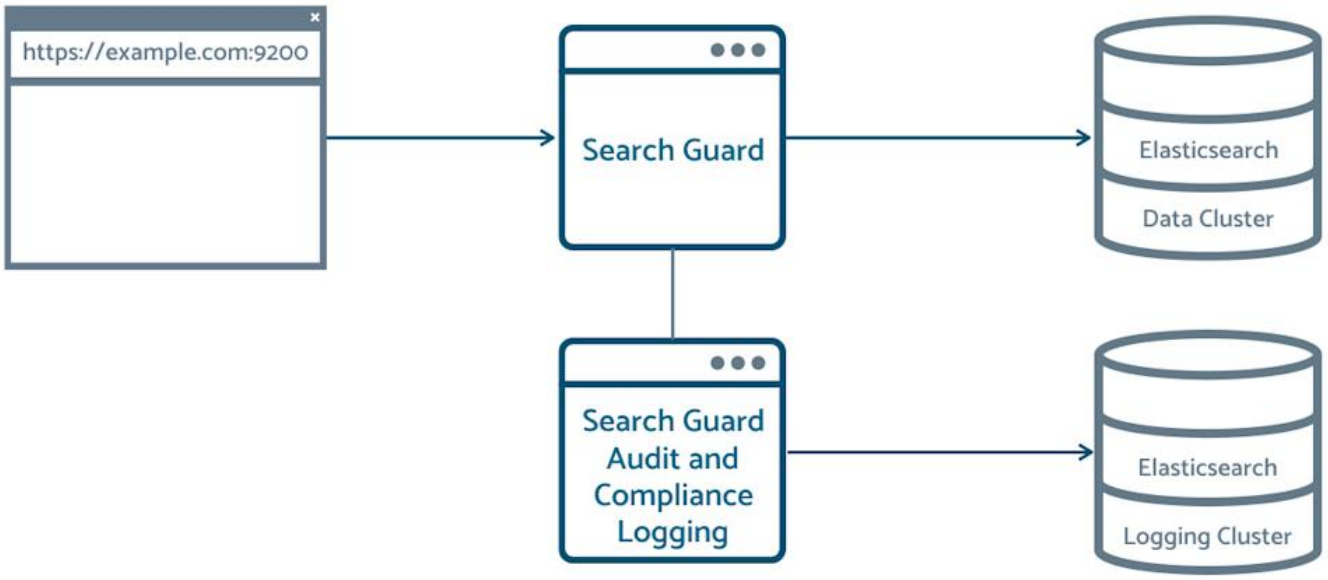
```
{
 "audit_category": "COMPLIANCE_DOC_WRITE",
 "audit_compliance_operation": "DELETE",
 ...
 "audit_trace_doc_id": "1",
 "audit_request_effective_user": "admin",
 "audit_trace_indices": [
   "customers"
 ],
 ...
}
```

Storage considerations

It is always good practice to store audit and compliance events in a separate, external system. This is to decouple the actual data from the events generated about it. By using the webhook storage endpoint you can ship compliance event to any system that supports HTTP(S) webhooks. You can also use the log4j endpoint to store the events in systems like Kafka or Cassandra. But why not store the events in a system you already have in place - Elasticsearch.

Elasticsearch Logging cluster

To store audit and compliance events in Elasticsearch, we recommend to set up an own, dedicated logging cluster and to secure it with Search Guard.



In addition, to prevent any changes to the audit events once the have been stored, we recommend to mark the respective index as immutable: An Immutable Index is a write-once index: Once a document has been created, Search Guard automatically prevents any changes to it. This ensures the integrity of the data and makes Immutable Indices a perfect candidate for storing compliance and security related audit events.

Summary

GDPR requires you to provide a lot of information about the storage and usage of personal information of your users upon request. Failing to do so can lead to very hefty fines. The Search Guard Write History tracks the complete lifetime of documents in your cluster, so you can prove to your customers at any time what data has been stored, when it has been created and what changes have been made. By tracking the deletion of documents you can prove that you complied with the *"right to be forgotten"*. The Write History is one of the building blocks to make your Elasticsearch cluster GDPR compliant.

Where to go next

- Download the Search Guard Compliance Edition and try the WRITE history for yourself
- Learn about the Search Guard Compliance features in our official documentation
- Combine the Write History with the Immutable Indices feature to ensure data integrity

Image: shutterstock / garsya

This article was published on the Search Guard blog: https://search-guard.com/blog/gdpr-write-history/
Search Guard is Security and Alerting for Elasticsearch and Kibana — search-guard.com · Start a free trial