Migrating Elasticsearch Indexes from Version 6 to 8: A Real-World Approach

Jochen Kressin · 2025-10-23

Migrating Elasticsearch Indexes from Version 6 to 8: A Real-World Approach

Introduction

Upgrading Elasticsearch across major versions is one of those tasks that can keep you up at night. When you're jumping from version 6 to version 8, you're not just dealing with a simple version bump, you're crossing significant architectural changes that affect how indexes are created, managed, and accessed. If you've been putting off this migration, you're not alone, and this post will walk you through a practical approach that actually works.

Problem Description

Picture this: you're running Elasticsearch 6.8 in production, and you need to migrate to version 8. You've got indexes with data you can't afford to lose, and you're facing the reality that Elasticsearch 8 won't directly read indexes created in version 6. The migration path isn't straightforward, and you're weighing your options between reindexing everything or finding a more efficient approach.

The core challenge here is that Elasticsearch 8 has dropped support for indexes created in version 6. This means you can't simply upgrade your cluster and expect everything to work. You need a strategy that preserves your data while ensuring compatibility with the new version.

Root Cause Analysis

The incompatibility between Elasticsearch 6 and 8 stems from fundamental changes in how Elasticsearch handles index formats and mappings. Over the years, Elastic has introduced breaking changes to improve performance, security, and functionality. These changes include:

- **Index format evolution**: The underlying Lucene index format has evolved significantly
- **Mapping changes**: Field types and mapping structures have been refined
- **Settings modifications**: Many index settings have been deprecated or changed
- **Security enhancements**: New security features require different approaches to authentication and authorization

Elasticsearch follows a policy where each major version supports indexes created in the current version and the previous major version. Since version 8 can only read indexes from versions 7 and 8, your version 6 indexes are incompatible.

Solution

The most reliable approach to this migration involves using Elasticsearch 7 as an intermediate step. Here's how to tackle this migration:

Step 1: Set Up Elasticsearch 7 as an Intermediate Cluster

First, create a new Elasticsearch 7 cluster that will serve as your migration bridge:

```
# Download and install Elasticsearch 7
wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.17.10-linux-
x86_64.tar.gz
tar -xzf elasticsearch-7.17.10-linux-x86_64.tar.gz
cd elasticsearch-7.17.10
```

Configure your Elasticsearch 7 cluster with appropriate settings for your environment. Make sure to set up proper networking and security configurations.

Step 2: Migrate Data from Version 6 to 7

Use the `_reindex` API to copy your data from the version 6 cluster to the version 7 cluster:

```
# Reindex from remote cluster (version 6) to local cluster (version 7)
curl -X POST "localhost:9200/_reindex" -H 'Content-Type: application/json' -d'
{
  "source": {
    "remote": {
      "host": "http://your-es6-cluster:9200"
    },
    "index": "your-source-index"
  },
  "dest": {
    "index": "your-dest-index"
  }
}'
```

You can also use tools like `elasticdump` for more granular control:

```
# Install elasticdump
npm install elasticdump -g

# Copy mapping
elasticdump --input=http://es6-cluster:9200/source-index --output=http://es7-cluster:9200/dest-index --type=mapping

# Copy data
elasticdump --input=http://es6-cluster:9200/source-index --output=http://es7-cluster:9200/dest-index --type=data
```

Step 3: Verify the Migration

After reindexing, verify that your data has been successfully migrated:

```
# Check document count
curl -X GET "localhost:9200/your-dest-index/_count"

# Compare with original
curl -X GET "your-es6-cluster:9200/your-source-index/_count"

# Verify mappings
curl -X GET "localhost:9200/your-dest-index/_mapping"
```

Step 4: Migrate from Version 7 to 8

Once your data is safely in Elasticsearch 7, you can proceed with the migration to version 8. This process is more straightforward since version 8 can read version 7 indexes:

```
# You can either:
# 1. Perform a rolling upgrade (recommended for production)
# 2. Use snapshot and restore
# 3. Use reindex again for additional safety
```

For production environments, consider using the snapshot and restore method:

```
# Create snapshot in ES 7
curl -X PUT "es7-cluster:9200/_snapshot/my_backup/snapshot_1?wait_for_completion=true"

# Restore in ES 8
curl -X POST "es8-cluster:9200/_snapshot/my_backup/snapshot_1/_restore"
```

Alternative Approach: Direct Migration Tools

If you prefer a more streamlined approach, you can use specialized migration tools:

Using Elasticsearch Migration Assistant

The Migration Assistant can help identify compatibility issues and provide guidance:

```
# Install the migration assistant
curl -X GET "localhost:9200/_migration/assistance"
```

Using Third-Party Tools

Tools like `elasticsearch-dump` or `reindex-from-remote` can simplify the process, but they still require the intermediate version 7 step for version 6 to 8 migrations.

Takeaways

Migrating from Elasticsearch 6 to 8 requires patience and planning, but it's absolutely achievable with the right approach. Here are the key lessons:

Plan for intermediate steps: Don't try to jump directly from version 6 to 8. Using version 7 as an intermediate step is not just recommended, it's often necessary.

Test thoroughly: Always validate your migration in a non-production environment first. Compare document counts, mappings, and query results to ensure data integrity.

Consider your options: While reindexing is the most common approach, evaluate whether snapshot/restore or other migration methods better suit your specific use case.

Monitor performance: Large reindexing operations can impact cluster performance. Plan for appropriate timing and resource allocation.

Keep backups: Always maintain backups of your original data until you're confident the migration is successful and stable.

The Elasticsearch documentation provides detailed guidance on upgrade procedures and breaking changes between versions. These resources are invaluable for understanding what changes might affect your specific setup.

Remember, this type of migration is a significant undertaking, but with proper planning and testing, you can successfully modernize your Elasticsearch infrastructure while preserving your valuable data.

This article was published on the Search Guard blog: https://search-guard.com/blog/migrating-elasticsearch-indexes-from-version-6-to-8-a-real-world-approach/