# Elasticsearch Data Management

**Learning Outcomes**

- Create and configure indices (shards, replicas, mappings) for optimal search/performance.
- Automate index setup with templates (static/dynamic) for consistent field mappings.
- Enforce lifecycle policies (ILM) to auto-manage indices (hot/warm/delete phases).
- Streamline time-series data using data streams and templates for auto-rollover/retention.
- Visualize/import data in Kibana and apply end-to-end best practices for scalability.

Provided By:  Mohamed AbdelGawwad Gaber
https://www.linkedin.com/in/moaelgawad/

## Introduction to Elasticsearch Data Management

Elasticsearch Data Management empowers users to efficiently organize, store, and optimize data for high performance and scalability. By mastering index creation and configuration - including shards, replicas, and mappings - users can enhance search speed and resource utilization. Automation through index templates ensures consistent field mappings, whether using static or dynamic approaches. Index Lifecycle Management (ILM) further simplifies operations by automating index transitions across hot, warm, and delete phases based on predefined policies. For time-series data, data streams combined with rollover templates enable seamless retention and auto-rollover management. Additionally, Kibana provides powerful tools for data visualization and import, supporting end-to-end best practices. This lesson equips you with the skills to implement these strategies for robust, scalable data management in Elasticsearch.

## Defining Indices

An index in Elasticsearch is similar to a database in relational databases - it's where your documents are stored and indexed.

**Example: Creating a Basic Index**

```
PUT my_index
{
  "settings": {
    "number_of_shards": 5,
    "number_of_replicas": 1
  },
  "mappings": {
    "properties": {
      "title": {
        "type": "text"
      },
      "timestamp": {
        "type": "date"
      },
      "category": {
        "type": "keyword"
      }
    }
  }
}
```

**Key Parameters:**

- number_of_shards: Number of primary shards (5 in this case)
- number_of_replicas: Number of replica shards (1 in this case)
- mappings: Defines how documents and their fields are stored/indexed

**Field Types:**

- text: For full-text search (analyzed)
- keyword: For exact matching (not analyzed)
- date: For date/time values

## Index Templates

Index templates automatically apply settings/mappings when new indices are created that match a pattern.

**Example: Basic Index Template**

```
PUT _index_template/my_template
{
 "index_patterns": ["logs-*"],
 "template": {
  "settings": {
   "number_of_shards": 1,
   "number_of_replicas": 0
  },
  "mappings": {
   "properties": {
    "message": {
     "type": "text"
    },
    "timestamp": {
     "type": "date"
    }
   }
  }
 }
}
```

**Key Features:**

- Applies to any index matching "logs-*"
- Defines shard/replica settings
- Pre-defines field mappings

## Dynamic Templates

Dynamic templates allow custom rules for field mapping based on field names or datatypes.

**Example: Dynamic Template in Index**

```
PUT /my_index2
{
  "mappings": {
    "dynamic_templates": [
      {
        "custom_fields": {
          "match": "custom_*",
          "mapping": {
            "type": "keyword"
          }
        }
      },
      {
        "date_field": {
          "match": "*_date",
          "mapping": {
            "type": "date"
          }
        }
      }
    ]
  }
}
```

**Example: Dynamic Template in Component Template**

```
PUT _component_template/my_dynamic_template
{
 "template": {
  "mappings": {
   "dynamic_templates": [
    {
     "custom_keyword_fields": {
      "match_mapping_type": "string",
      "match": "custom_*",
      "mapping": {
       "type": "keyword"
      }
     }
    },
    {
     "date_fields": {
      "match_mapping_type": "string",
      "match": "*_date",
      "mapping": {
       "type": "date",
       "format": "yyyy-MM-dd HH:mm:ss"
      }
     }
    }
   ]
  }
 }
}
```

**Rules Applied:**

- Any field starting with "custom_" becomes a keyword
- Any field ending with "_date" becomes a date with specified format

## Index Lifecycle Management (ILM)

ILM automates managing indices through their lifecycle (hot, warm, cold, delete phases).

**Example: Time-Series ILM Policy**

```
PUT _ilm/policy/my_ilm
{
  "policy": {
    "phases": {
      "hot": {
        "actions": {
          "rollover": {
            "max_age": "1d"
          }
        }
      },
      "warm": {
        "min_age": "7d",
        "actions": {
          "forcemerge": {
            "max_num_segments": 1
          }
        }
      },
      "delete": {
        "min_age": "30d",
        "actions": {
          "delete": {}
        }
      }
    }
  }
}
```

**Policy Details:**

- Hot Phase
  - Active index receiving writes
  - Rolls over after 1 day
- Warm Phase:
  - Starts 7 days after rollover
  - Force merges segments to optimize storage
- Delete Phase:
  - Deletes index after 30 days

## Data Streams

Data streams provide a single writeable endpoint for time-series data that automatically manages backing indices.

**Example: Creating a Data Stream Template**

```
PUT _component_template/shards
{
  "template": {
    "settings": {
      "number_of_shards": 1,
      "number_of_replicas": 1
    },
    "mappings": {
      "properties": {
        "message": {
          "type": "text"
        },
        "timestamp": {
          "type": "date"
        }
      }
    }
  }
}
```

```
PUT _index_template/myindex_template2
{
  "index_patterns": ["my_data_stream"],
  "data_stream": {},
  "composed_of": ["shards"]
}
```

**Key Points:**

- First create component templates with settings/mappings
- Then create index template that:
    - Targets the data stream name pattern
    - Includes `"data_stream": {}` to enable data stream behavior
    - Composes the component templates

## Practical Usage Examples

**Question 1: Index Lifecycle Management (ILM)**

You need to implement an ILM policy for log data that meets the following requirements:

- Hot phase for 7 days with 1 primary shard and 1 replica
- Warm phase for 30 days with 1 primary shard and 1 replica (read-only)
- Delete phase after 37 days
- Force merge to 1 segment in warm phase
- Shrink to 1 shard in warm phase

**Answer:**

```
PUT _ilm/policy/logs_policy
{
 "policy": {
  "phases": {
   "hot": {
    "min_age": "0ms",
    "actions": {
     "rollover": {
      "max_age": "7d"
     },
     "set_priority": {
      "priority": 100
     }
    }
   },
   "warm": {
    "min_age": "7d",
    "actions": {
     "forcemerge": {
      "max_num_segments": 1
     },
     "shrink": {
      "number_of_shards": 1
     },
     "readonly": {},
     "set_priority": {
      "priority": 50
     }
    }
    },
    "delete": {
      "min_age": "37d",
      "actions": {
        "delete": {}
      }
    }
  }
 }
}
```

**Explanation:**

1. The hot phase starts immediately (0ms) and rolls over after 7 days (max_age: "7d")
2. The warm phase begins after 7 days (min_age: "7d") and performs:
   - Force merge to 1 segment (max_num_segments: 1)
   - Shrink to 1 shard (number_of_shards: 1)
   - Makes index read-only (readonly: {})
3. The delete phase triggers after 37 days (7+30) and deletes the index
4. Priorities are set appropriately (higher for hot, lower for warm)
5. Replica count is managed at index creation/template level, not in ILM

**Question 2: Index Templates and Data Streams**

You need to create an index template for logs data that:

- Uses data streams
- Has 1 primary shard and 1 replica
- Sets index.lifecycle.name to "logs_policy"
- Adds a @timestamp field as a date field in the mapping
- Creates a custom field called log_level with keyword type
- Adds a company name as a constant keyword field

**Answer:**

```
PUT _index_template/logs_template
{
 "index_patterns": ["logs-*"],
 "data_stream": {},
 "template": {
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 1,
    "index.lifecycle.name": "logs_policy"
  },
  "mappings": {
    "properties": {
      "@timestamp": {
        "type": "date"
      },
      "log_level": {
        "type": "keyword"
      },
      "company": {
        "type": "constant_keyword",
        "value": "your_company"
      }
    }
  }
 }
}
```
```

To create the first data stream:

```
PUT _data_stream/logs-app1
```

**Explanation:**

1. The template targets indices matching logs-* pattern
2. data_stream: {} enables data stream functionality
3. Settings configure the required shard/replica count and ILM policy
    - Mappings define:
    - @timestamp as required for data streams
    - log_level as keyword for efficient filtering
4. company as constant keyword (saves storage for repeated values)
5. Data streams are created with a simple PUT request to _data_stream
6. The first backing index will be created automatically when data is indexed
```