Day 1 Exercises: Cluster Exploration & CRUD Operations

Stack: `elk-single` | **Duration:** ~30 minutes | **Kibana Dev Tools:** `http://localhost:5601`

Setup

```
# Install Python dependencies (one-time)
pip install -r requirements.txt

# Start Elasticsearch (if not running)
docker compose -f docker/elk-single/docker-compose.yml --env-file .env up -d
```

Wait 1-2 minutes for Elasticsearch and Kibana to start, then open Kibana at **http://localhost:5601** (login: `elastic` / `elastic`).

Task 1: Cluster Inspection (Basic)

Goal: Get familiar with `_cat` APIs and understand your cluster state.

Run the following commands in Kibana Dev Tools and answer the questions:

```
GET _cluster/health

GET _cat/nodes?v

GET _cat/indices?v
```

Questions:

1. What is the cluster health status? Why is it not green?
2. How many nodes are in the cluster?
3. What system indices (starting with `.`) already exist?

Task 1: Cluster Inspection (continued)

> **Hint:** Single-node clusters are always **yellow** because replica shards cannot be allocated to the same node as their primary shard. This is by design for fault tolerance.

Task 2: Load Sample Data (Basic)

Goal: Load the movies dataset and Kibana sample data.

2a. Load movies dataset

```
python data/load_data.py --dataset movies --size small
```

Verify in Dev Tools:

```
GET movies/_count

GET movies/_search
{
  "size": 3
}
```

Task 2: Load Sample Data (continued)

2b. Load Kibana sample data

1. Open Kibana → Home → **Try sample data**
2. Load **Sample eCommerce orders**
3. Load **Sample flight data**

Verify:

```
GET _cat/indices?v&s=index
```

Questions:

1. How many documents are in the `movies` index?
2. What are the index names for the Kibana sample datasets?
3. How many shards does each index have?

Task 2: Load Sample Data (hint)

> **Hint:** The Kibana sample indices are typically named `kibana_sample_data_ecommerce` and `kibana_sample_data_flights`. Use `GET _cat/shards/movies?v` to see shard details for a specific index.

Task 3: CRUD Operations (Basic)

Goal: Practice creating, reading, updating, and deleting documents.

3a. Create a new movie with explicit ID

Create a document in the `movies` index with `_id` = `999`:

Field	Value
title	"My Favorite Movie"
overview	"An amazing story about..."
genres	["Drama", "Adventure"]
vote_average	9.5
release_date	"2024-01-15"

3b. Read it back

Retrieve the document you just created.

Task 3: CRUD Operations (continued)

3c. Update the vote_average to 9.0

Use a partial update (not a full replace).

3d. Delete the document

Remove the document and verify it's gone.

Hint:

- Create: `PUT /movies/_doc/999 { ... }`
- Read: `GET /movies/_doc/999`
- Update: `POST /movies/_update/999 { "doc": { "vote_average": 9.0 } }`
- Delete: `DELETE /movies/_doc/999`
- Verify deletion: `GET /movies/_doc/999` should return 404

Task 4: Understanding Mappings (Intermediate)

Goal: Explore the mapping (schema) that was automatically created for the movies index.

```
GET movies/_mapping
```

Questions:

1. What **type** is the `title` field? What analyzer does it use?
2. What **type** is the `genres` field? Why keyword instead of text?
3. What **type** is the `vote_average` field?
4. What would happen if you indexed a document with a new field `"director": "Nolan"`?

Task 4: Understanding Mappings (continued)

Hint:

- `text` fields are analyzed (tokenized) for full-text search
- `keyword` fields are stored as-is for exact matches, sorting, and aggregations
- ES uses **dynamic mapping**: new fields are automatically detected and mapped
- A string value would be mapped as both `text` AND `keyword` (multi-field) by default

Task 5: Shard Allocation (Intermediate)

Goal: Understand how shards are distributed and what affects cluster health.

```
GET _cat/shards/movies?v

GET _cluster/allocation/explain
{
  "index": "movies",
  "shard": 0,
  "primary": false
}
```

Questions:

1. Are there any UNASSIGNED shards? Why?
2. What does the allocation explain API tell you about why a replica can't be assigned?
3. If you added a second node, what would happen to cluster health?

Task 5: Shard Allocation (continued)

> **Hint:** In a single-node cluster, replica shards remain UNASSIGNED because Elasticsearch never places a replica on the same node as its primary (otherwise a node failure would lose both copies). Adding a second node would allow replicas to be assigned, turning the cluster **green**.

Task 6: Search Preview (Bonus)

Goal: Try a few search queries to get a taste of what's coming on Day 2.

6a. Find movies with "war" in the title

```
GET /movies/_search
{
  "query": {
    "match": {
      "title": "war"
    }
  }
}
```

Task 6: Search Preview (continued)

6b. Find highly-rated dramas

```
GET /movies/_search
{
  "query": {
    "bool": {
      "must": {
        "term": { "genres": "Drama" }
      },
      "filter": {
        "range": { "vote_average": { "gte": 8.5 } }
      }
    }
  }
}
```

6c. Experiment

Try modifying these queries:

- Search for your favorite genre
- Change the vote_average threshold
- Search in the `overview` field instead of `title`

Task 6: Search Preview (questions)

Questions:

1. Which query (6a or 6b) produces results with higher `_score` values?
2. Why does the `filter` clause in 6b not contribute to the `_score`?

> **Hint:** `filter` context does not calculate relevance scores -- it only includes/excludes documents (yes/no). This makes filters faster and cacheable. The `must` clause calculates BM25 scores, which is why it contributes to `_score`. We'll cover this in detail on Day 2.

Cleanup (Optional)

Only clean up if you're done for the day:

```
docker compose -f docker/elk-single/docker-compose.yml --env-file .env down -v
```

> Keep the stack running if you're continuing to Day 2!