### 1. **Bulk API Introduction**

The **Bulk API** is used to perform **multiple indexing, update, delete** operations in a single request. This reduces round-trip overhead and improves performance.

Common Use Case:

* Load 100,000+ documents from logs or CSV
* Batch updates from a data lake or microservice

Supported operations:

* index – Adds/overwrites
* create – Fails if doc exists
* update – Modifies existing doc
* delete – Removes doc

### 🔷 2. **Bulk API Demo**

📌 Format: **NDJSON (newline-delimited JSON)**

POST /\_bulk

Content-Type: application/x-ndjson

{ "index": { "\_index": "books", "\_id": 1 } }

{ "title": "Elasticsearch in Action", "author": "Smith" }

{ "create": { "\_index": "books", "\_id": 2 } }

{ "title": "Distributed Systems", "author": "Tanenbaum" }

{ "update": { "\_index": "books", "\_id": 1 } }

{ "doc": { "author": "John Smith" } }

{ "delete": { "\_index": "books", "\_id": 2 } }

✅ Use this in Kibana Dev Tools (remove Content-Type if using Dev Tools).

### 🔷 3. **Granular Error Handling**

Each item in a bulk request may succeed or fail independently.

📌 Example of a response with partial errors:

{

"errors": true,

"items": [

{ "index": { "status": 201 } },

{ "create": { "status": 409, "error": { "type": "version\_conflict\_engine\_exception" } } }

]

}

📌 Best Practice: Always check .errors == true and inspect each item.

### 🔷 4. **Optimistic Concurrency Control in Bulk**

Use \_seq\_no and \_primary\_term to ensure data consistency across updates.

📌 OCC Bulk Example:

POST /\_bulk

{ "update": { "\_index": "books", "\_id": "1", "if\_seq\_no": 5, "if\_primary\_term": 2 } }

{ "doc": { "views": 25 } }

⚠️ If versions don’t match, the update fails gracefully.

### 🔷 5. **Multiple Indices in Bulk**

You can bulk-load documents across different indices:

POST /\_bulk

{ "index": { "\_index": "logs-2025-07-01", "\_id": "a1" } }

{ "event": "login", "user": "alice" }

{ "index": { "\_index": "logs-2025-07-02", "\_id": "a2" } }

{ "event": "logout", "user": "bob" }

📘 Use case: Time-based indices in logging systems.

### 🔷 6. **File Upload (Bulk Index via File)**

📌 Example bulk\_data.json file (NDJSON):

{ "index": { "\_index": "movies", "\_id": "1" } }

{ "title": "Inception", "year": 2010 }

{ "index": { "\_index": "movies", "\_id": "2" } }

{ "title": "Interstellar", "year": 2014 }

📌 Upload using curl:

curl -XPOST "http://localhost:9200/\_bulk" -H "Content-Type: application/x-ndjson" --data-binary @bulk\_data.json

✅ Useful for offline loading or during migration.

### 🔷 7. **Reindex API**

Reindex allows copying documents from one index to another (even transform during copy).

📌 Basic reindex:

POST /\_reindex

{

"source": {

"index": "old-index"

},

"dest": {

"index": "new-index"

}

}

📌 Reindex with query filter:

POST /\_reindex

{

"source": {

"index": "old-index",

"query": {

"term": {

"status": "active"

}

}

},

"dest": {

"index": "active-only"

}

}

📘 Real-World:

Migrating data after changing shard counts or mapping.

## ✅ Summary

| Concept | Purpose/Example |
| --- | --- |
| Bulk API | Multiple create/update/delete/index ops in one request |
| Granular Error Handling | Each op in bulk may succeed/fail individually |
| OCC in Bulk | Ensures safe update using \_seq\_no and \_primary\_term |
| Multiple Indices in Bulk | Write to logs-2025-07-01 and logs-2025-07-02 in same request |
| File Upload | Upload NDJSON via curl or Logstash |
| Reindex | Copy docs across indices, optionally filter/transform |

## 📝 QUIZ – Bulk API and Reindexing

**1. What is the default format of the Bulk API?**

a. XML  
b. JSON Array  
c. NDJSON  
d. CSV

✅ **Correct:** c. NDJSON

**2. Can one operation fail in a Bulk API while others succeed?**

a. No  
b. Yes  
c. Only if the index doesn’t exist  
d. Only for deletes

✅ **Correct:** b. Yes

**3. How do you prevent overwriting documents unintentionally in bulk updates?**

a. Use version: "safe"  
b. Use \_source: false  
c. Use \_seq\_no and \_primary\_term  
d. Use doc\_as\_upsert: true

✅ **Correct:** c. Use \_seq\_no and \_primary\_term

**4. What API is used to copy data from one index to another?**

a. \_bulk\_index  
b. \_clone  
c. \_reindex  
d. \_snapshot

✅ **Correct:** c. \_reindex

**5. What is the content type for the bulk API when uploading via curl?**

a. application/json  
b. application/x-ndjson  
c. text/plain  
d. application/binary

✅ **Correct:** b. application/x-ndjson

Would you like this in PDF/Markdown format? Or want me to generate 50 sample NDJSON lines for bulk insert practice?