**Notes: Preparing Kibana with Data for Elasticsearch**

**Introduction**

Kibana is not much fun without any data, so let’s add some. I have prepared two datasets for this course, which we will take a closer look at in the next lecture. The files containing the data are formatted to be compatible with Elasticsearch’s Bulk API, which you may be familiar with. We will be using the Bulk API through cURL.

In case you are not familiar with cURL, it’s the most popular command-line HTTP client. In fact, it’s preinstalled on most operating systems. I say “most” because it wasn’t installed on Windows by default prior to version 10. If it’s not available on your Windows installation, you can find a download link attached to this lecture. If you are on macOS or any Linux distribution, you should be all set.

**Downloading the Dataset**

1. **Accessing the Dataset**:
   * You can find the test data within the GitHub repository, which I have opened up here.
   * The link to the repository is attached to this lecture.
2. **Steps to Download**:
   * First, click on the file name.
   * Then click the download button to the right.
3. **Contents of the Zip Archive**:
   * The zip archive contains four files:
     + One for the **orders dataset**.
     + Three for the **access logs dataset**.
       - The latter covers three months: January through March of 2020.
4. **Extracting the Archive**:
   * Double-clicking the file should work on all operating systems as far as I know.

**Setting Up the Terminal**

1. **Opening the Terminal**:
   * If you are using Windows, open the Command Prompt instead.
2. **Setting the Working Directory**:
   * The working directory should be the directory in which the extracted files are placed.
   * Use the cd command to navigate to this directory.
3. **Reference Commands**:
   * You can find the commands shown in this lecture within the GitHub repository.

**Using cURL for Data Ingestion**

**Handling TLS Encryption**

Elasticsearch clusters are set up to encrypt network traffic with TLS by default. If your Elasticsearch cluster is running on your local machine, we need to handle this, since the cluster uses a self-signed certificate that HTTP clients do not trust by default.

1. **Options for Handling Certificates**:
   * **Option 1**: Instruct cURL to blindly trust the certificate and ignore that it’s self-signed.
   * **Option 2**: Provide a CA certificate.
2. **Implementation of Each Option**:
   * For the first approach, simply add -k to the command, and you are done.
   * For the second option, use the --cacert argument to specify a path to the CA certificate that is generated by Elasticsearch.
     + The path to it is relative to the Elasticsearch root directory and shown in the diagram attached to this lecture.



1. **Recommendation**:
   * Providing a CA certificate is technically the most correct thing to do.
   * However, since we are working with a development cluster, using -k to trust the self-signed certificate is completely fine (as long as you don’t do this for production clusters or cloud deployments).

**Authentication with cURL**

To communicate with our Elasticsearch cluster, we need to authenticate our requests. This is done by specifying a username and password.

1. **How to Get Credentials**:
   * For local deployments, the credentials are output within the terminal the first time you start up Elasticsearch.
   * For cloud deployments, credentials are provided during the setup process.
2. **Specifying Credentials in cURL**:
   * Use the -u option to specify credentials.
     + Example: -u username
   * When running the command, you will be prompted to enter the password before the request is sent.
   * Alternatively, you can append a colon followed by the password (e.g., -u username:password). However, this approach is less secure.

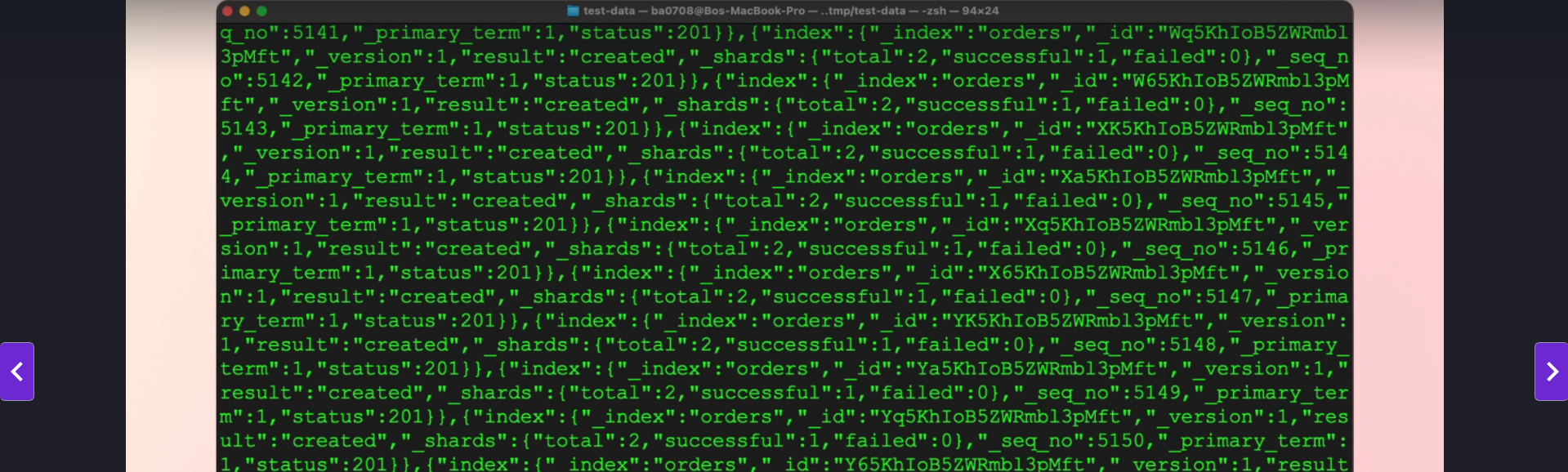
**Constructing the cURL Command**

1. **Specify the Content-Type Header**:
   * Use the -H argument to specify the NDJSON content type:
   * -H "Content-Type: application/x-ndjson"
2. **Set the HTTP Verb**:
   * Use -X POST to specify the HTTP verb (defaults to GET if omitted).
3. **Provide the Elasticsearch Endpoint**:
   * For a local cluster: Use http://localhost:9200/\_bulk.
   * For Elastic Cloud: Use the deployment’s endpoint and append the path to the Bulk API.
4. **Handling the Index Name**:
   * Notice that the bulk files already contain the name of the index that documents should be indexed into. Therefore, you don’t need to specify the index name within the endpoint.
5. **Send Data Along with the Request**:
   * Use --data-binary to send the file content as binary data:
   * --data-binary "@filename"
   * The @ symbol tells cURL that the value is a file name, not a path.
   * Binary data preserves newline characters, which is essential for NDJSON files whereas cUrl the newlines.

**Executing the cURL Command**

**Example Command:**

curl -k -u username -H "Content-Type: application/x-ndjson" -X POST http://localhost:9200/\_bulk --data-binary "@orders.json"

1. **Run the Command**:
   * After a short moment, Elasticsearch responds with information about the operations contained within the bulk file.  
     
2. **Inspect the Results**:
   * You can inspect the results if needed.
   * Generally, you can trust that the data was ingested correctly unless Elasticsearch reports an issue.

**Processing Multiple Files**

1. **Access Logs Dataset**:
   * This dataset contains three files (January, February, March 2020).
2. **Repeat the Command for Each File**:
   * Use the same cURL command.
   * Modify the file name to point to the respective month’s file:
   * curl -k -u username -H "Content-Type: application/x-ndjson" -X POST http://localhost:9200/\_bulk --data-binary "@access\_logs\_jan.json"
   * Use the **arrow up key** to recall the previous command, edit the file name, and rerun.

**Conclusion**

1. **All Data Ingested**:
   * Orders dataset.
   * Access logs for January, February, and March 2020.
2. **Ready for Kibana**:
   * The data has been ingested into Elasticsearch and is ready to be used within Kibana.
3. **Troubleshooting**:
   * If there were any issues, Elasticsearch’s response would provide details for debugging.