**Implementing an Elasticsearch Backed Microservice**

**Introduction**

In this lab, we will create a Lambda function which will connect the provided web interface to an Elasticsearch domain to provide search functionality.

**Note:** This lab can have extended launch times (10 to 30 minutes).

**Additional Information and Resources**

The FrankenWords frontend web development team has completed their version 1.0 interface, which is intended to provide statistical information about the appearance of words and phrases in the novel Frankenstein by Mary Shelley. The backend team has stood up an Elasticsearch domain in Amazon Elasticsearch Service. It is your job to complete the connection between the two teams by writing a Lambda function which will forward search entries to the Elasticsearch domain as properly formatted queries, and return a appropriately formatted response.

You've been provided with the following Kibana login information for the Elasticsearch domain:

* **Username:** cloud\_user
* **Password:** Strongpass1!

**Lab Resources**

* [Lab Solution Code](https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/tree/master/Lab_Assets/implementing_an_elasticsearch_backed_search_microservice)
* [Elasticsearch Full Text Query Reference](https://www.elastic.co/guide/en/elasticsearch/reference/current/full-text-queries.html)
* [Python Elasticsearch Module Documentation](https://elasticsearch-py.readthedocs.io/en/6.8.2/)

**Solution**

Log in to the AWS Management Console using the credentials provided for the lab. Then, open the FrankenWords website in a separate browser tab. The site is hosted in an S3 bucket in your lab account.

**Investigate the Lab Environment**

**Review the Elasticsearch Query in Kibana**

1. Navigate to OpenSearch Service (The Elasticsearch Service has been renamed} using the *Services* menu or the unified search bar. In the *My Elasticsearch domains* section, you should see a *frankensearch* domain.
2. Select the **frankensearch** domain.
3. On the *Overview* tab, open the **Kibana** URL in a separate browser.
4. Log in to Kibana using the credentials provided for the lab:
   * **Username:** cloud\_user
   * **Password:** Strongpass1!
5. In the *Let's get started* window, select **Explore on my own**. Choose Private Tenant.
6. Click the menu icon in the top left corner, then select **Dev Tools**.
7. Copy the search query and paste it into the query editor on the left.
8. GET \_search
9. {
10. "query": {
11. "simple\_query\_string": {
12. "query": "Horse~20",
13. "default\_operator": "and"
14. }
15. }

}

1. Click the Play icon in the top right corner of the query editor to send the request and display the output on the right. The output shows 29 hits.
2. In the query editor on the left, delete the ~20 parameter from the query line to remove any fuzziness.
3. GET \_search
4. {
5. "query": {
6. "simple\_query\_string": {
7. "query": "Horse",
8. "default\_operator": "and"
9. }
10. }

}

1. Click the Play icon in the top right corner of the query editor, then review the new output on the right. You now have 0 hits.

**Review the Lambda Function and FrankenWords Website Code**

1. Navigate back to the AWS Management Console and open Lambda using the *Services* menu or the unified search bar. You should have 3 functions provided.
2. Select the **Franken\_Search** function and scroll down to the *Code source* section.
3. Open function.py and review the function code, which currently has just a basic function.
4. Navigate to S3 using the *Services* menu or the unified search bar.
5. Select the franken-search-<ACCOUNT\_NUMBER> bucket.
6. Check the checkbox to the left of *frankenwords.js*, then use the *Actions* dropdown to select **Download**.
7. Review the code from the FrankenWords website. For a detailed walkthrough of how this code works, check out our *Solution: Planning* video.

**Update the Lambda Function**

1. Navigate back to the *Franken\_Search* function in Lambda, then copy the [Lambda function](https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/tree/master/Lab_Assets/implementing_an_elasticsearch_backed_search_microservice) provided for the lab and paste it into the query editor. For a detailed walkthrough of how this code works, check out our *Solution: Execution* video.
2. Copy the ES endpoint provided for the lab and paste it into the <ES ENDPOINT> placeholder in the code, and choose "Deploy"
3. Navigate back to the Dev Tools tab in Kibana and re-enter the ~20 parameter to the query line.
4. GET \_search
5. {
6. "query": {
7. "simple\_query\_string": {
8. "query": "Horse~20",
9. "default\_operator": "and"
10. }
11. }

}

1. Click the Play icon in the top right corner of the query editor, then review the output on the right. You should have 29 hits again.
2. Navigate back to Lambda and click **Deploy** to update the function.

**Test FrankenWords!**

1. Navigate to the FrankenWords website.
2. Enter Horse~20 into the search bar and click **Search**. The search should return 29 hits. You can also see how many chapters contained the term "Horse" and the term's total score.
3. Enter Food~20 into the search bar and click **Search**, then compare the results to your previous results. Based on the total score, horses are described more often than food in Frankenstein.
4. (Optional) Continue changing the search term to learn how often other words and phrases are used in Frankenstein.

**Conclusion**

Congratulations — you've completed this hands-on lab!