Causaly: Backend Engineering Position Assignment

The purpose of this assignment is to evaluate a candidate's familiarity with specific problems and solutions often encountered in the domain, as well as to understand the knowledge of particular tools used to solve those problems.

First 4 questions are descriptive and open, they require you to elaborate on a problem/solution and potentially use charts and block diagrams to aid visualization. Last question is a practical one and aims to evaluate the knowledge of ElasticSearch. You will be expected to return a solution for this problem with a clear description of its execution for testing purposes.

Question 1.

From your experience using AWS, please describe how you would design an ecommerce website (i.e. Front End website for selling the goods, Back office website for managing orders and a DB for storing the data). Please provide a system diagram with any required reasoning/elaboration of your choices.

Question 2.

Continuing on from the ecommerce website, we would like to understand the customer habits and suggest products to them. Can you outline a data pipeline process that would extract the data and process the data to be able to suggest products?

Question 3.

From your experience, you have mentioned Git, Maven, Sonar and Jenkins which are tools widely used in CI/CD. Could you please explain a CD process related to the ecommerce website.

Question 4.

Please provide an example or two of how in your past experience you investigated and fixed an urgent issue in production.

Exercise.

You are provided with a following ElasticSearch schema representing an article index.

```
{
  "mappings": {
    "properties": {
      "uuid": {
        "type": "keyword",
        "ignore_above": 256,
        "normalizer": "to_lowercase"
      },
      "title": {
        "type": "text",
        "analyzer": "standard"
      },
      "abstract": {
        "type": "text",
        "analyzer": "standard"
      },
      "tags": {
        "type": "text",
        "fields": {
          "keyword": {
            "type": "keyword",
            "ignore_above": 256,
            "normalizer": "to_lowercase"
          }
        },
        "analyzer": "standard"
      },
      "relationships": {
        "type": "nested",
        "properties": {
          "cause_concept_name": {
            "type": "keyword",
            "ignore_above": 256,
            "normalizer": "to_lowercase"
          },
          "effect_concept_name": {
            "type": "keyword",
            "ignore_above": 256,
            "normalizer": "to_lowercase"
          }
        }
      }
    }
  },
  "settings": {
    "index": {
      "analysis": {
        "normalizer": {
          "to_lowercase": {
            "filter": ["lowercase"],
            "type": "custom"
          }
        }
      }
    }
  }
}
```

Each article has 5 attributes:

- **`uuid`** - article unique ID, e.g. "PMID15389210_20200420_0.4"
- **`title`** - article title
- **`abstract`** - article abstract (short description of what the article is about)
- **`tags`** - article tags
- **`relationships`** - causality data extracted from Causaly's ML algorithm, representing relationships between medical concepts found in the article

For the tasks below, please use Python.

Task 1.

Setup the above schema in an ElasticSearch index and load the mock data provided (**causaly_be_mock_data.ndjson** – feel free to change the format before loading).

Requirements

- In order to avoid stressing the ElasticSearch cluster, ensure that there will be no more than 3 in-flight upload requests in parallel.
- Provide simple instructions on how to execute the script.

Notes

- Mock data are provided in new-line delimited JSON format;
- In order to complete this task you will need to **setup the above schema in an ElasticSearch index**. To do so, you may use the docker-compose.yml file which will help you to quickly setup a local dev environment with ElasticSearch and Kibana.

  For the setup you will need Docker installed on your device. After you have it installed, you can run docker-compose up. Compose will download the official docker containers and start Elasticsearch and Kibana.

  When compose finishes, verify that ElasticSearch is running at: http://localhost:9200.
  You can also access Kibana at: http://localhost:5601/app/dev_tools#/console.

Skills assessment

- Logic and code structure;
- Attention to detail.

Time requirement

1 hour

Task 2.

Write an ElasticSearch query that searches for articles containing *effects of coffee* in the aforementioned index. By *effects of coffee* we mean articles that have "coffee" as cause_concept_name in **at least one (1)** of their relationships.

Boost the score of articles that contain the word "coffee" in their title, abstract or tags as follows:

Field Name	Boost Factor
title	1
tags	0.51
abstract	0.1

Sum all scores together to calculate the article's final score and **sort in DESC order**.

Requirements

- Deliver the ElasticSearch query in JSON format.

Notes

- It does not matter how many times you find the word "coffee" in the title, abstract or tags attributes. You should only score on the first match.

  For example, the following article title: "Is coffee healthy? A study on coffee." should yield a score of 1 instead of 2. Same for abstract and tags.

Skills assessment

- Familiarity with ElasticSearch;
- Attention to detail.

Time requirement

2-3 hours