Course Project – Option 2

Deadline: April 2nd, 11:59pm ET/8:59pm PT

General Guidelines:

Difficulty Level: Moderate

Accept this project by accessing GitHub classroom via the following URL: https://classroom.github.com/a/Xjl5orOJ

You are required to submit your GitHub Repository URL on Gradescope. A penalty will be applied otherwise. If you are working in a team, a single submission from either member is sufficient.

The course project will include a lot of self-learning that is needed to complete the project. Students shouldn't expect any project support during office hours or on Piazza.

Students are highly encouraged to give themselves enough time to learn the skills they need to complete the project.

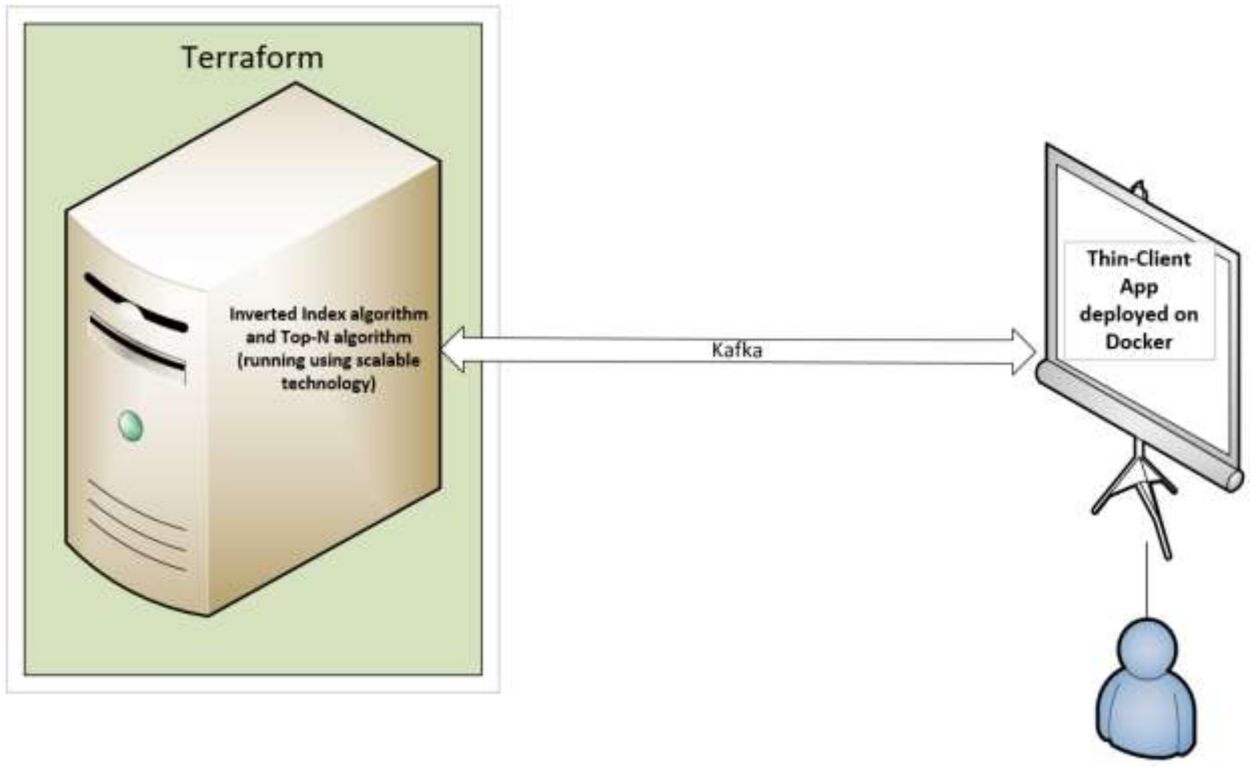
You may choose up to one peer to work with you on the project. You MUST list your peer's name at the top of your ReadMe file. Failure to list your peer's name will lead to grading penalty of 10 points to every team member. Having a peer is optional.

Your ReadMe file should include the exact steps that can be used to reproduce a functional version of your solution.

General Description:

You are asked to build a search engine that allows its users to search for specific keywords in large data files. Your search engine should be scalable to handle large datasets and respond to queries from multiple users simultaneously.

Mini Search Engine


Terraform
Inverted Index algorithm and Top-N algorithm (running using scalable technology)
Kafka
Thin-Client App deployed on Docker

In developing this system, you will build two applications:

1. A lightweight application deployed on the user machine where users can enter an input and view the search output. This application is deployed on a Docker container, and it shouldn't conduct any heavy processing. Rather, it should contact a cluster-based application to conduct any heavy-processing. **<u>You may create this application as a web application</u>**.
2. A cluster-based application that processes user requests received from the light-weight application.
    a. These requests include constructing inverted indices and searching these indices. Also, user requests may include inquiring for the TopN frequent terms in these indices. This application is deployed on a Dataproc Cluster, a Google Kubernetes Engine, or a similar AWS/Azure technology. It doesn't communicate with users directly and **<u>you send requests to the cluster via Kafka queue</u>**.

   b. This cluster should be coded using **Terraform**.

Example User Experience (UX) Flow: Check the Mockup PDF on Canvas. The mockup aims to provide general guidance on the program flow and you are welcome to innovate!

Important Notes:

- You may use the data files located in the Data folder on Canvas in testing and validating your applications.
- You can hardcode all Kafka attributes in an environment file.
    - You can run any configurations or manual Kafka commands as scripts from Terraform (e.g., topic creation). Don't run any Kafka commands outside of Terraform.
- Kafka can't be running on your local machine. You can deploy it to GKE, Dataproc or any other non-local option.
- Use ReadMe.md file on your repository to list any assumptions, steps and any important information to share.
- Keep any private keys for your GCP account outside of the code on GitHub and you don't need to submit them.

Submission Guidelines:

- Post URL for your GitHub repository to Gradescope and post your peer's name at the top of your ReadMe file.
- Your GitHub repository should have a ReadMe.md file that lists the "exact" steps on how to get this application to work.
- You should record a video demonstrating two elements:
    1. Code Walkthrough explaining your code changes.
        - Your code walkthrough video is expected to be at least 10 minutes long and should focus on the code you developed

2. Running application while navigating through every working functionality. This video will help with your grade assessment. You may lose points for any functionalities not demonstrated in the demo
    - Your demo video is expected to be at least 5 minutes long and should demonstrate the creation of resources in addition to a live demo of functionality

- Your video size may be large to be uploaded to GitHub. You may use Box to upload the video and add the URL to your ReadMe.md file.
    1. Make sure that your video is publicly shared. Private videos won't be visible to the instructor and TAs and therefore, your project grade will be impacted.

Grading Criteria:

- Lightweight App Implementation on Docker: 20% of the total project grade.
- Cluster deployment to the cloud using Terraform: 20% of the total project grade.
- Client to Cluster Communication via Kafka: 10% of the total project grade
- Inverted Indexing MapReduce Implementation and Execution on the Cluster (with stop-word list): 20% of the total project grade
- Top-N Search (including execution time): 20% of the total project grade
- **Submitting Your First Checkpoint Tasks by February 26th, 11:59PM ET/8:59PM PT. (10% of the total project grade)**
    - **You are supposed to submit the items highlighted in yellow below.**

Suggested Project Task Schedule (You may run ahead of schedule):

Week	Task
End of Week-4	Build the first web application that will interface with the user
End of Week-5	• Create the docker container and add the application to it. • Figure out how to communicate with GCP resources from inside the container. • Think about how to deploy Kafka for your final solution.

End of Week-8	• Write the Terraform script to deploy the cluster to the cloud and all required algorithms/jobs in addition to Kafka. • Manage bi-directional communication between Docker-deployed client application and the Cloud Cluster by configuring bidirectional communication using Kafka.
End of Week-9	Create the Inverted Indexing algorithm and Top-N Algorithm
End of Week-10	1. Complete the Cluster-based application (and jobs) and ensure they can be deployed using Terraform 2. Start working on your video recording for the demo and the code walkthrough
End of Week-11	1. Finish your ReadMe.md file to list all your steps, assumptions, and any information you find important to share.

Grading Notes:

- You have one grace day for course project submission.
- Unlike HW-assignment penalties, late project submissions on Gradescope or GitHub will receive 0 points (won't be graded).
- **Not submitting the GitHub video (<u>for both code walkthrough and functionality demo</u>): you will get up to 80% of the maximum grade.**
- Not providing clear details in the ReadMe file on how to run the application (or any variables that need to be updated/replaced): **you will get up to 90% of the maximum grade.**