Hello! Congratulations on making it to this round of Cribl's interview process! The objective of this take home exercise is to demonstrate your design and implementation techniques for a real-world type of development problem. You can use any programming language or tech stack of your choice. However, please note that we are looking for solutions that demonstrate your unique code, so please limit external dependencies to a bare minimum (eg. frameworks, HTTP servers)

You should develop and submit code to Cribl via a GitHub project. Please commit and push code changes as you normally would - your thinking and working style is an important part for us to understand.

Of course, no great product is complete without clear documentation and testing. As part of your solution, please provide any design, usage, and testing documentation that you feel would be helpful to someone using your solution for the first time.

Problem statement:

A customer has asked you for a way to provide on-demand monitoring of various unix-based servers without having to log into each individual machine and opening up the log files found in /var/log. The customer has asked for the ability to issue a REST request to a machine in order to retrieve logs from /var/log on the machine receiving the REST request.

Acceptance criteria:

1. A README file describing how to run and use the service.
2. HTTP REST API should be the primary interface to make data requests
3. The lines returned must be presented with the newest log events first. It is safe to assume that log files will be written with newest events at the end of the file.
4. The REST API should support additional query parameters which include
    a. The ability to specify a filename within /var/log
    b. The ability to specify the latest N number of log entries to retrieve within the log
    c. The ability to filter results based on basic text/keyword matches
5. Must not use any pre-built log aggregation systems - this must be custom, purpose-built software.
6. The service should work and be reasonably performant when requesting files of>1GB
7. Minimize the number of external dependencies in the business logic code path (framework things like HTTP servers, etc are okay)

Bonus points:

There is potential to double the deal size with this customer if you can successfully implement "nice-to-have" features that will make your produce more valuable to them:

1. The ability to issue a REST request to one “primary” server in order to retrieve logs from a list of “secondary” servers. There aren’t any hard requirements for the protocol used between the primary and secondary servers.
2. A basic UI to demo the API

Hints

1. **Log Processing**
   - Implement efficient handling of logs, considering edge cases like empty files or unexpected formats.
   - Make thoughtful assumptions about file traversal and processing large files.
2. **Query and Filtering**
   - Design flexible query handling to accommodate optional parameters and basic text matching.
   - Consider how your solution could extend to support regex or advanced matching in the future.
3. **API Behavior**
   - Focus on clear and predictable API behavior, ensuring optional parameters are well-documented.
   - Return structured data (e.g., JSON) that clients can easily consume.
4. **Handling Secondary Servers**
   - Demonstrate asynchronous querying with fault tolerance.
   - Tag logs with source metadata and allow for sorting or aggregation.
5. **Error and Security Handling**
   - Include guardrails to prevent unauthorized access or harmful operations (e.g., directory traversal).
   - If time permits, document potential security measures like authentication.
6. **Performance Optimization**
   - Think about efficiency in searching and retrieving logs, especially for large files.
   - Document how your design could scale or improve with increased log file size or complexity.
7. **Enhancements**
   - Highlight additional features like pagination, UI integration, or a strategy for handling dynamic logs.
   - Provide brief notes on future improvements in comments or a README.