Log Viewer

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. Please note that we are looking for solutions that **demonstrate your unique development skills** with React and JavaScript or TypeScript. With that in mind, here are some constraints:

- Limit external dependencies to a bare minimum
- Avoid UI frameworks (e.g. MaterialUI, Ant Design, Tailwind), but feel free to use the CSS you know (CSS Preprocessors, CSS Modules, Styled Components, etc.). Our hope is to see your HTML/CSS skills shine with the tools you're familiar with.
- For state management, stick with statement management that the React API provides and avoid external libraries such as Redux, Zustand, etc.

We know that taking on a code submission assignment like this one is not a trivial request, and we really appreciate the time you put into this - it helps us get to know you and how you work! We don't want you spending an inordinate amount of your time on it, though, so if you find that it's taking more than ~4 hours to complete, please reach out to us and we can help make sure you're pointed in the right direction.

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. In your submission, please include a link to the public GitHub repo containing your code.

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 to provide an UI for viewing the contents of a log file downloaded from an URL. They would like to be able to quickly scroll through the log entries, as well as be able to expand an entry to see the full log event. Due to the fact that log files can be huge, they would like to see the log events as soon as they are transmitted to the browser over the wire, without waiting for the entire file to download.

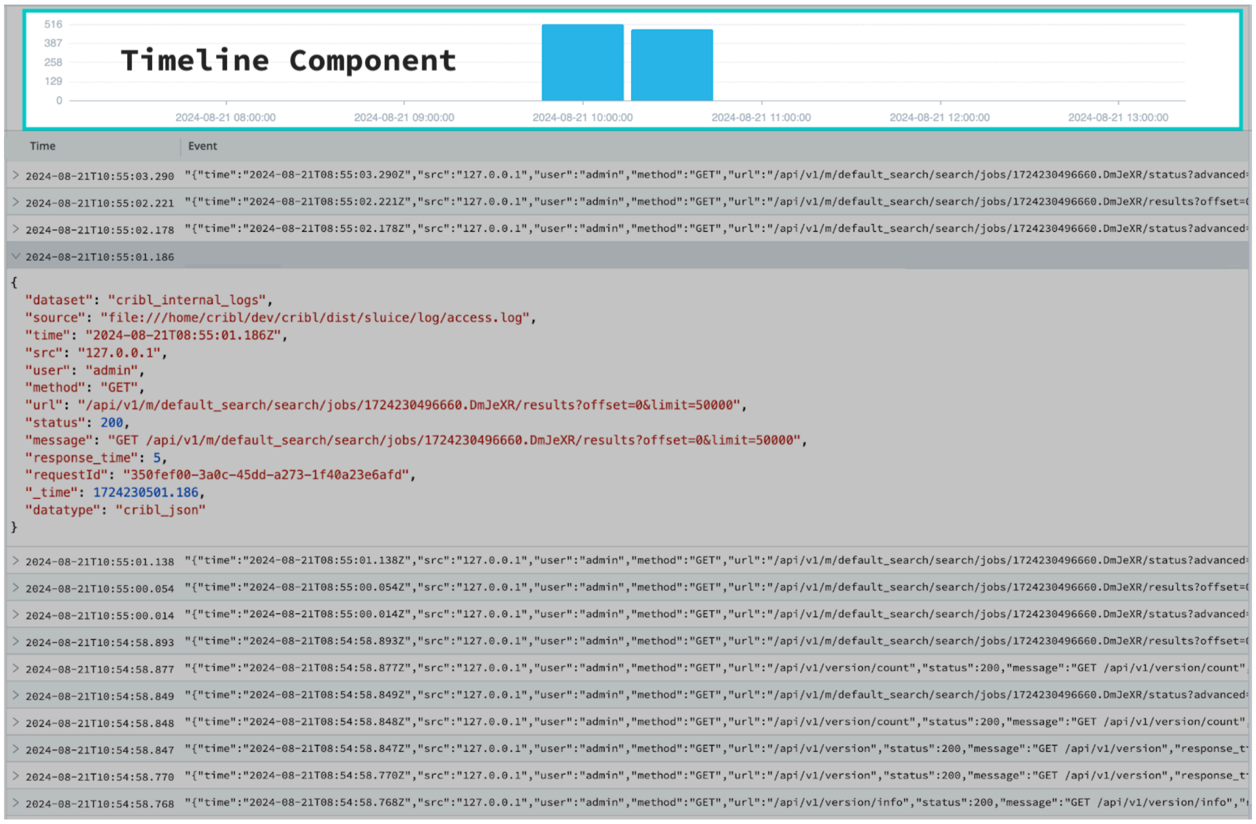
Here's a sample URL that can be used for downloading the log file:
https://s3.amazonaws.com/io.cribl.c021.takehome/cribl.log

Acceptance criteria

1. Your submission should include Unit Tests to show your ability to test your code. Full coverage is not required but you should have some working tests and a short write up of what else you would test given additional time.
2. The component should render the list of log entries as a table with two columns. The first column should display the time (the value of the `_time` property), formatted as ISO 8601. The second column should present the entire event formatted as single line JSON. Each log entry should be rendered as a separate row.
3. It should be possible to expand/collapse rows. When expanded, the row should display the entire event formatted as multiline JSON, each property in a new line.
4. The component should pull data from the given URL and update the view while individual log entries are being downloaded from the server. The data is provided in the NDJSON format (new line delimited JSON). The UX should be optimized for time to first byte, i.e. the component should render the events as soon as they are downloaded from the server, without waiting for the entire file to download.

Bonus points

Implement a simple timeline component, similar to the one presented on the mockup below, that will show the distribution of log events over time. For this portion of the exercise, feel free to use whatever charting library you want (or even better, no charting library at all).



```
> 2024-08-21T10:55:03.290  "{"time":"2024-08-21T08:55:03.290Z","src":"127.0.0.1","user":"admin","method":"GET","url":"/api/v1/m/default_search/search/jobs/1724230496660.DmJeXR/status?advanced=
> 2024-08-21T10:55:02.221  "{"time":"2024-08-21T08:55:02.221Z","src":"127.0.0.1","user":"admin","method":"GET","url":"/api/v1/m/default_search/search/jobs/1724230496660.DmJeXR/results?offset=
> 2024-08-21T10:55:02.178  "{"time":"2024-08-21T08:55:02.178Z","src":"127.0.0.1","user":"admin","method":"GET","url":"/api/v1/m/default_search/search/jobs/1724230496660.DmJeXR/status?advanced=
v 2024-08-21T10:55:01.186
{
  "dataset": "cribl_internal_logs",
  "source": "file:///home/cribl/dev/cribl/dist/sluice/log/access.log",
  "time": "2024-08-21T08:55:01.186Z",
  "src": "127.0.0.1",
  "user": "admin",
  "method": "GET",
  "url": "/api/v1/m/default_search/search/jobs/1724230496660.DmJeXR/results?offset=0&limit=50000",
  "status": 200,
  "message": "GET /api/v1/m/default_search/search/jobs/1724230496660.DmJeXR/results?offset=0&limit=50000",
  "response_time": 5,
  "requestId": "350fef00-3a0c-45dd-a273-1f40a23e6afd",
  "_time": 1724230501.186,
  "datatype": "cribl_json"
}
> 2024-08-21T10:55:01.138  "{"time":"2024-08-21T08:55:01.138Z","src":"127.0.0.1","user":"admin","method":"GET","url":"/api/v1/m/default_search/search/jobs/1724230496660.DmJeXR/status?advanced=
> 2024-08-21T10:55:00.054  "{"time":"2024-08-21T08:55:00.054Z","src":"127.0.0.1","user":"admin","method":"GET","url":"/api/v1/m/default_search/search/jobs/1724230496660.DmJeXR/results?offset=
> 2024-08-21T10:55:00.014  "{"time":"2024-08-21T08:55:00.014Z","src":"127.0.0.1","user":"admin","method":"GET","url":"/api/v1/m/default_search/search/jobs/1724230496660.DmJeXR/status?advanced=
> 2024-08-21T10:54:58.893  "{"time":"2024-08-21T08:54:58.893Z","src":"127.0.0.1","user":"admin","method":"GET","url":"/api/v1/m/default_search/search/jobs/1724230496660.DmJeXR/results?offset=
> 2024-08-21T10:54:58.877  "{"time":"2024-08-21T08:54:58.877Z","src":"127.0.0.1","user":"admin","method":"GET","url":"/api/v1/version/count","status":200,"message":"GET /api/v1/version/count",
> 2024-08-21T10:54:58.849  "{"time":"2024-08-21T08:54:58.849Z","src":"127.0.0.1","user":"admin","method":"GET","url":"/api/v1/m/default_search/search/jobs/1724230496660.DmJeXR/status?advanced=
> 2024-08-21T10:54:58.848  "{"time":"2024-08-21T08:54:58.848Z","src":"127.0.0.1","user":"admin","method":"GET","url":"/api/v1/version/count","status":200,"message":"GET /api/v1/version/count",
> 2024-08-21T10:54:58.847  "{"time":"2024-08-21T08:54:58.847Z","src":"127.0.0.1","user":"admin","method":"GET","url":"/api/v1/version","status":200,"message":"GET /api/v1/version","response_t
> 2024-08-21T10:54:58.770  "{"time":"2024-08-21T08:54:58.770Z","src":"127.0.0.1","user":"admin","method":"GET","url":"/api/v1/version","status":200,"message":"GET /api/v1/version","response_t
> 2024-08-21T10:54:58.768  "{"time":"2024-08-21T08:54:58.768Z","src":"127.0.0.1","user":"admin","method":"GET","url":"/api/v1/version/info","status":200,"message":"GET /api/v1/version/info","
```

Delivery

Our goal is to have a submission within **one week**. You may use any tools available to you to complete the challenge, including using AI-assisted development. However the application should work, bug free, when first visited by our reviewing team.

For ease of testing your solution, please include a link to CodeSandbox. You can use this template as a starting point or simply import your project from GitHub.

When you are ready to submit, please send the following to your hiring contact at Cribl:

- Link to a public GitHub repository
- Link to CodeSandbox
- (optional) zip of source code

Once the application has been verified as working, your code will be passed around the team for a blind code review. We'll inform you of next steps after the code review has been completed.

That's it! We appreciate your interest in working at Cribl and for taking the time to perform this coding challenge.