

Development of a Full Stack Web Application to Manage Monte Carlo Methods

Summer Student Project Note

Pablo Apausa

Supervisors: Sandro Wenzel, Benedikt Volkel

ALICE Collaboration
CERN

Geneva, Switzerland
September, 2023

Keywords

Software Development • Interface Design • Web Application • Dashboard • ALICE Collaboration • Worldwide LHC Computing Grid • TypeScript • Bash

Abstract

This Summer Student Project Note documents the development of a full stack application from scratch in Next.js with TypeScript, for researchers within the ALICE Collaboration at CERN to configure Monte Carlo methods, run them either on a local server or the worldwide LHC Computing Grid, and keep track of their status. Also, to filter them and visualize their workflow. The idea was inspired by ALICE Hyperloop and Detector Control System web applications.

Contents

- 1. Introduction: Background
- 2. Description: Problem Statement
- 3. Objectives
- 4. Methodology
- 5. Next.js Application
 - 5.1. Front-end
 - 5.1.1. Root segment
 - 5.1.2. Configuration segment
 - 5.1.3. Details segment
 - 5.1.4. Visualise workflow leaf segment
 - 5.2. Back-end
- 6. Conclusion: Future Work
- Appendix: Screenshots
- References

1. Introduction: Background

Monte Carlo methods play an important role in heavy-ion physics experiments. These mathematical techniques are used to estimate the possible outcomes of an uncertain event (IBM, 2023). Like the behaviour of particles under specific conditions, enabling physicists to model and understand particle interactions. While the Worldwide LHC Computing Grid provides the global computing resources to analyse these simulations and store their respective outcomes.

2. Description: Problem Statement

Simulations management within the ALICE Collaboration is not efficient. Setting up Monte Carlo methods, executing them and monitor their status are separate tasks. Therefore, a user-friendly dashboard to streamline this process and implement more features is necessary to improve research work.

First, configuration stage is complex and error-prone due to the nature of these simulations. Researchers must ensure parameters are properly written and their values correctly set, while remembering their respective intended function and potential incompatibilities. Which may lead to failed runs.

Also, running Monte Carlo Methods either on a local server or the Worldwide LHC Computing Grid is intricate and time-consuming, as it involves many initial steps like getting a Grid Certificate or loading the O2sim package. Which can present different authentication or compatibility issues.

Keeping track stage is optimised thanks to the `grid_submit.sh` file. This script not only simplifies sending simulations to the WLCG but also facilitates accessing their results through terminal outputs. However, users may find themselves struggling to manage several processes running at the same time.

3. Objectives

The main objective is to support physicists within the ALICE Collaboration manage multiple Monte Carlo simulations through the development of a full stack application in Next.js with TypeScript: streamlining the process of configuring these methods, running them and keeping track of their status; while implementing features like filter or workflow visualisation. This is a prototype, since it is not yet deployed nor is a data base or a remote server implemented. So at the moment it can only be run locally.

4. Methodology

The web application has been developed with TypeScript and Bash languages, following a procedural programming methodology. Main technologies used to carry out the project have been: Next.js and Tailwind CSS frameworks; Node.js environment (server side); React.js (client side), UUID, Next UI and D3.js libraries; and ESLint and Figma tools.

All technologies can be found on the internet: Figma through its own website, while the others through the NPM registry. Bash is installed by default on Linux and MacOS.

5. Next.js Application

5.1. Front-end

There are three main segments: '/', '/configuration' and '/simulation/[id]'.

5.1.1. Root segment

Path: '/'

Root segment page allows to keep track of configured simulations through a table, showcasing their respective name, sub-jobs number, creation date, and local or WLCG status ('staged', 'completed', 'error' or 'running'). It also shows the total number of computing jobs and implements pagination.

Monte Carlo Simulations can be sorted by creation date, when clicking on its respective header column, and filtered by WLCG status or name (query). Users can navigate to `/configuration` segment page by clicking on the 'add job' button or to details (`/simulation/[id]`) segment page by clicking on a configured job.

5.1.2. Configuration segment

Path: '/configuration'

Configuration segment page allows to interactively set up Monte Carlo Simulations through a form, providing a text input to write a name, a calendar input to select a version, a number input to set an amount of sub-jobs and the option to switch between a default and an advanced mode tab.

Default mode tab renders checkbox groups showcasing all possible arguments for 'create workflow' and 'run workflow' commands; every parameter has its own predefined value and placeholder. It also shows the number of selected arguments for each. In contrast, advanced mode tab displays a text field with the parsed selection from all checkbox groups as its initial value. Users can set the form back to its original state by clicking on the 'Reset Form' button.

Additionally, navigation to root segment page can be achieved either by clicking on the arrow back button or by staging the configuration.

5.1.3. Details segment

Path: '/simulation/[id]'

Details segment page allows to seamlessly execute and review any configured job, providing the option to switch between the script for local and WLCG run tabs. Each one shows a read only text input with its respective script path, a read only text field with its

content, an accordion with its 'stdout' and 'stderr' streams and a 'run script' button; whose color is based on its status ('staged' in blue, 'running' in yellow, 'error' in red and 'completed' in green).

'Local run' tab also shows a 'visualise workflow' button that opens visualise workflow leaf segment page. Users can navigate to root segment page either by clicking on the '←' button or by removing the computing job, and to configuration segment page by recreating the setup. If simulation is not found it redirects the user to 'not found' page.

5.1.4. Visualise workflow leaf segment

Path: `'/simulation/[id]/visualizeWorkflow'`

Visualise workflow leaf segment page allows to graphically see the configured simulation workflow by rendering generated workflow.gv file content.

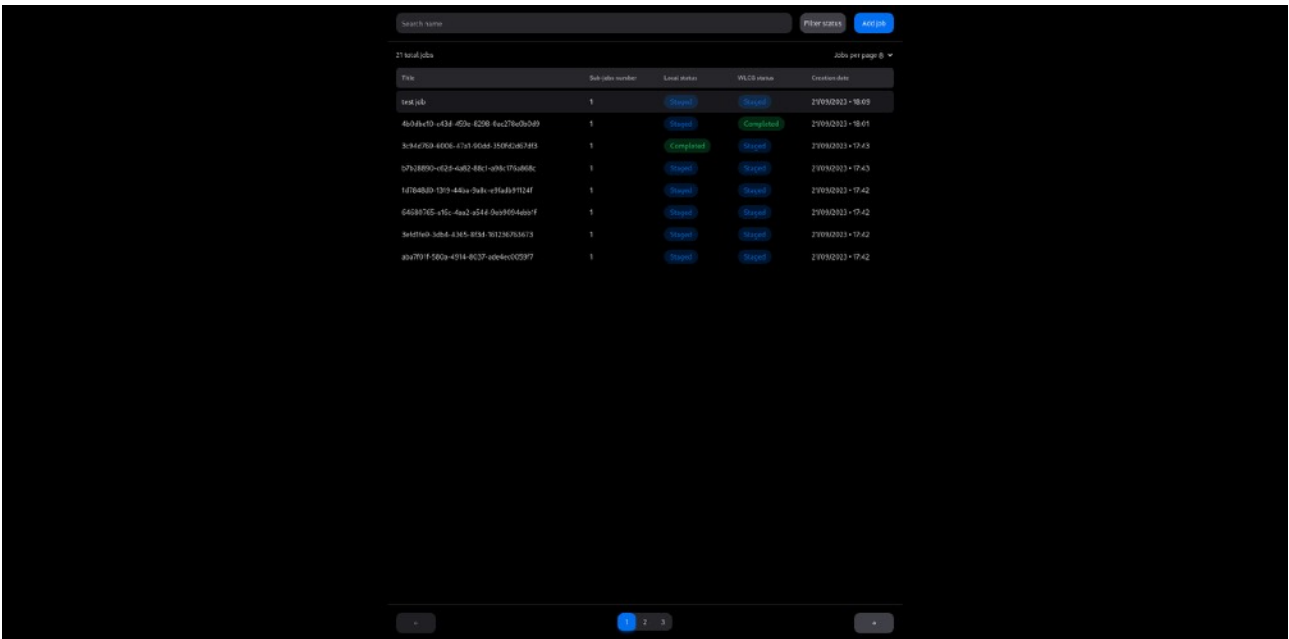
5.2. Back-end

There are three api routes: `'/api/localCreateWorkflow'`, `'/api/localRunWorkflow'` and `'/api/gridRunWorkflow'`. First two run the job on a local server, while the third one on the WLCG. All three of them return the updated Monte Carlo method status, 'stderr' and 'stdout' streams. Only difference between `'/api/localCreateWorkflow'` and `'/api/localRunWorkflow'`, is that first one returns a 'graphvizData' property within 'stdout' stream with the generated workflow.gv file content, so that it is rendered on the details segment page.

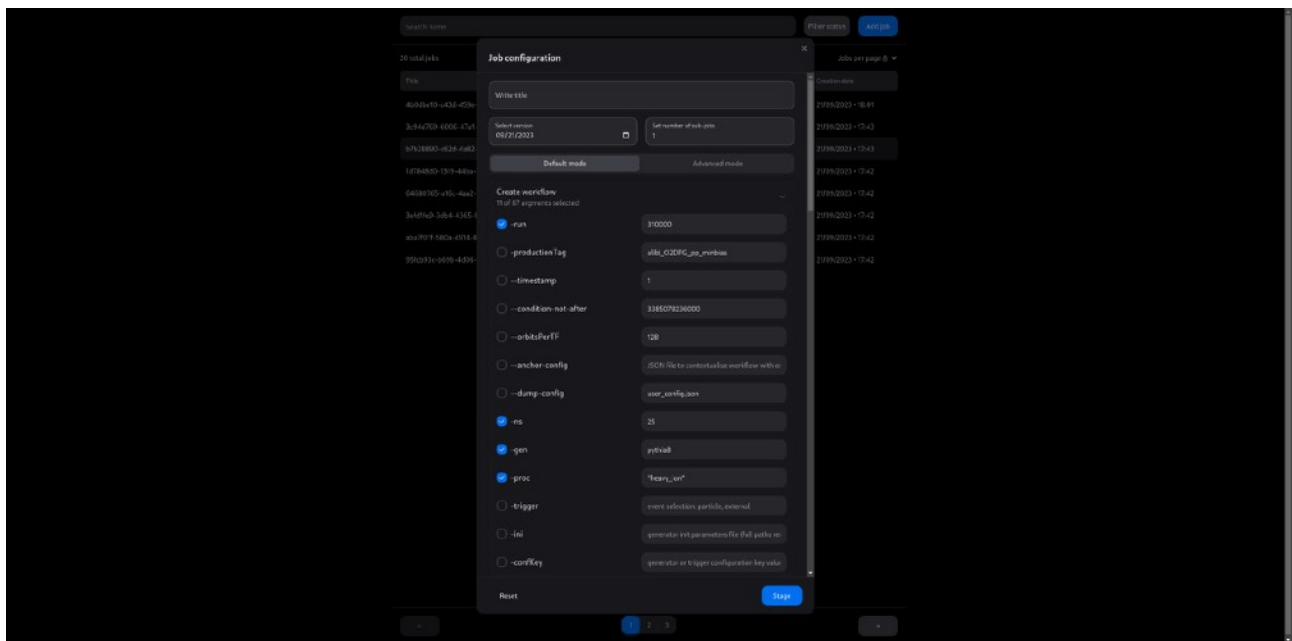
6. Conclusion: Future Work

Next step of the project would be to develop authentication, implementing a database in which to introduce users and admins as well as storing each of their staged Monte Carlo simulations. Next, link remote servers so that jobs are not executed locally. Then, unit test the code with Jest and React Testing Library. And deploy the application.

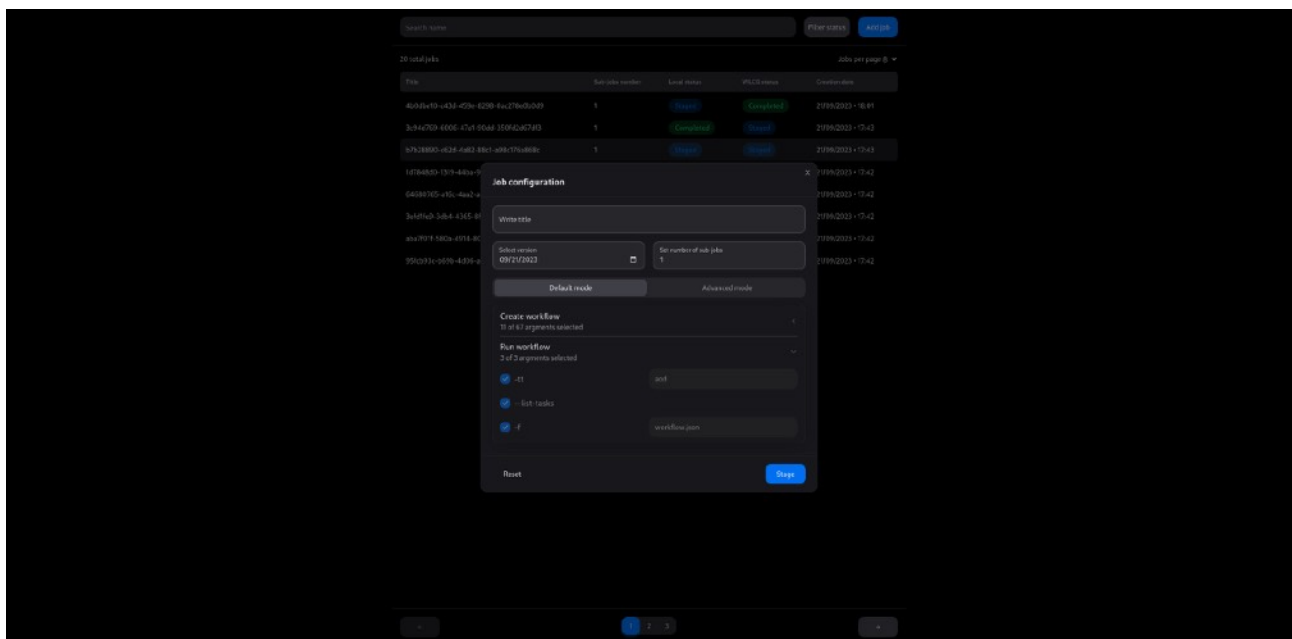
Appendix: Screenshots



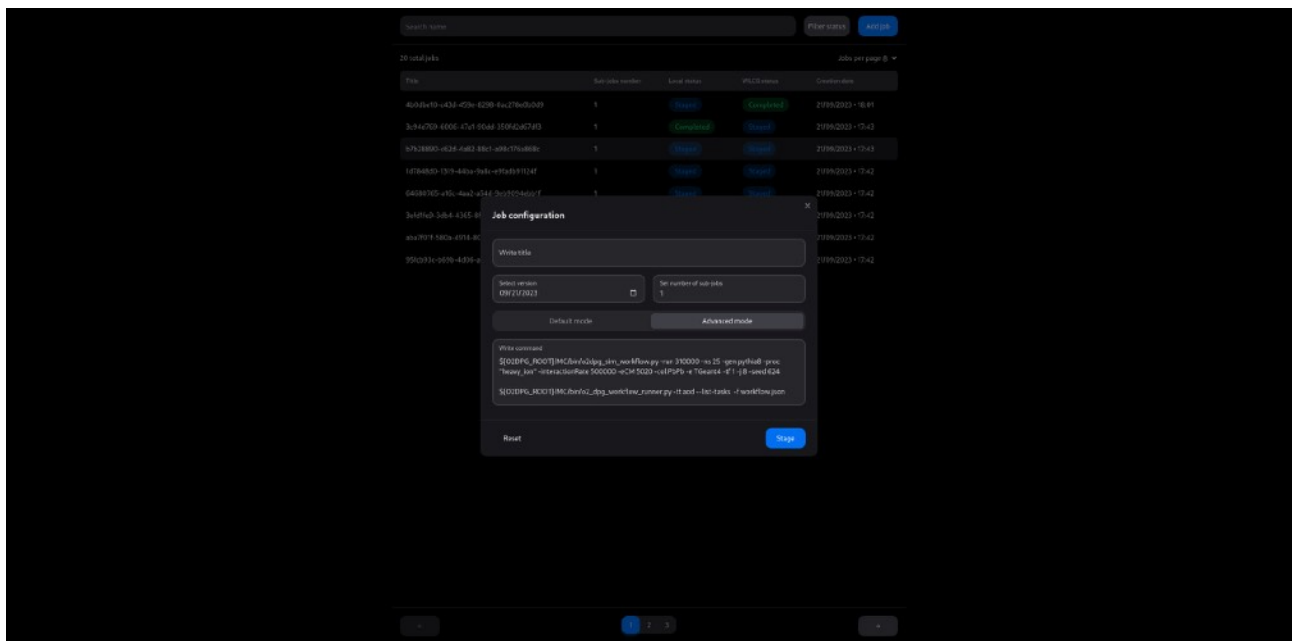
Root page.



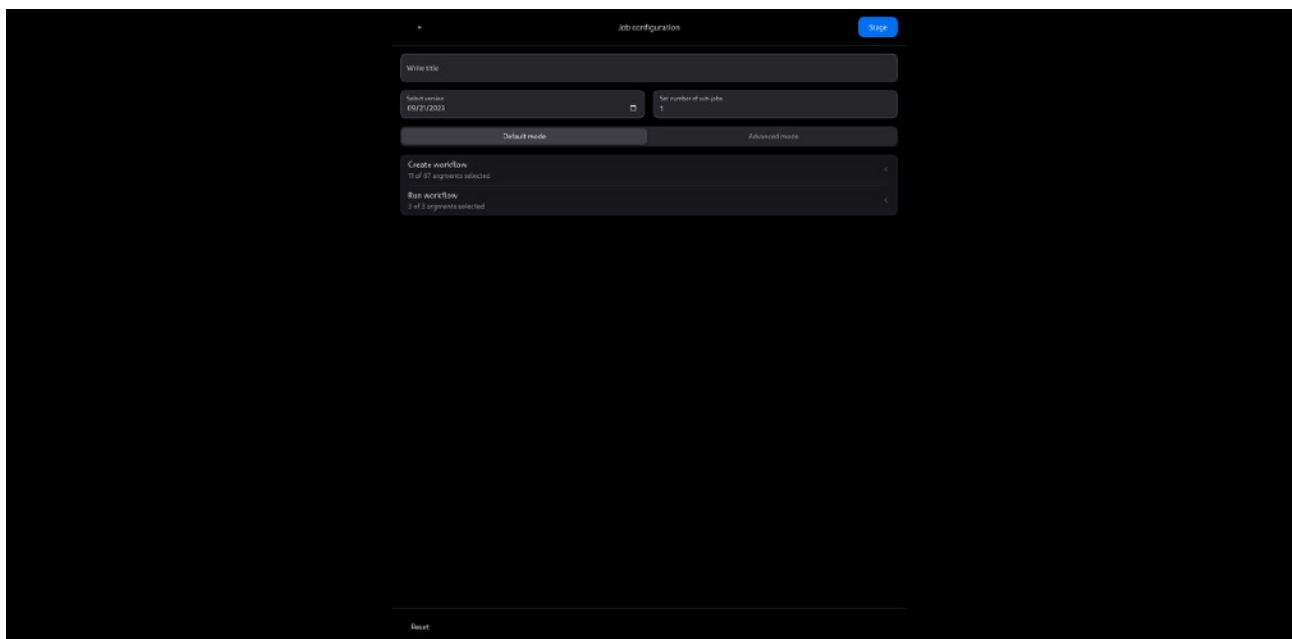
Configuration page, parallel route.
 'Default mode' tab, 'Create workflow' accordion.



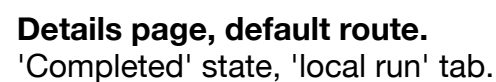
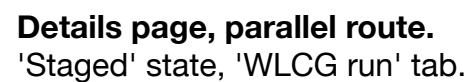
Configuration page, parallel route.
 'Default mode' tab, 'Run workflow' accordion.

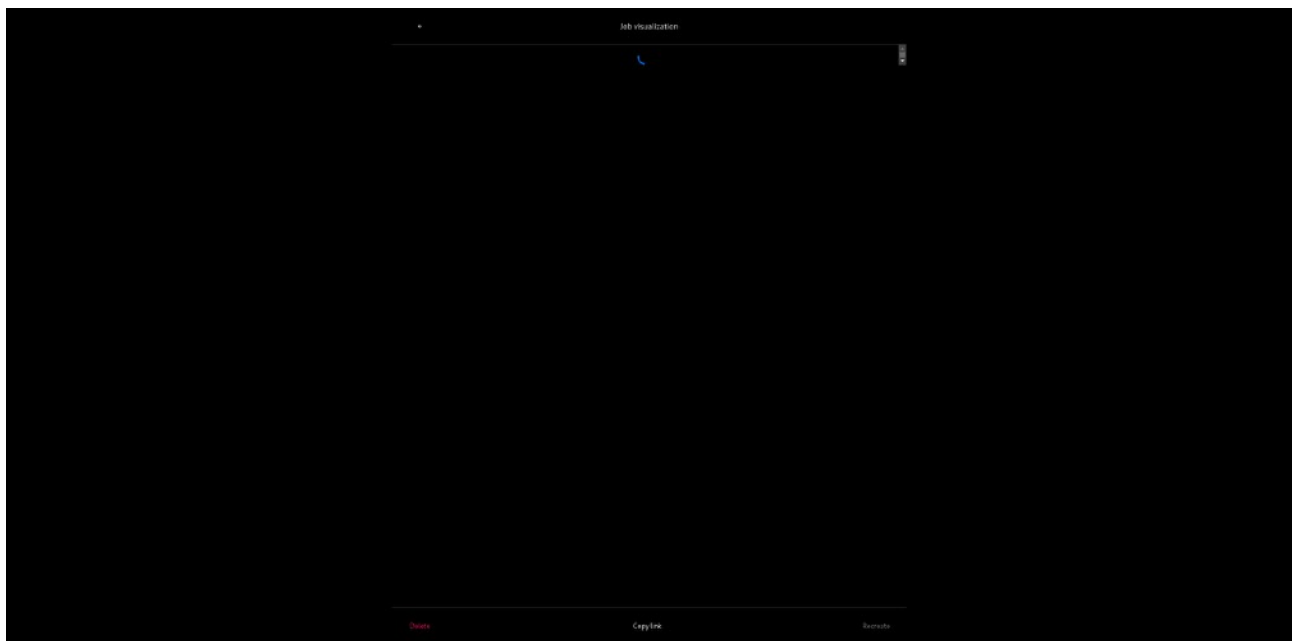


Configuration page, parallel route.
 'Advanced mode' tab.

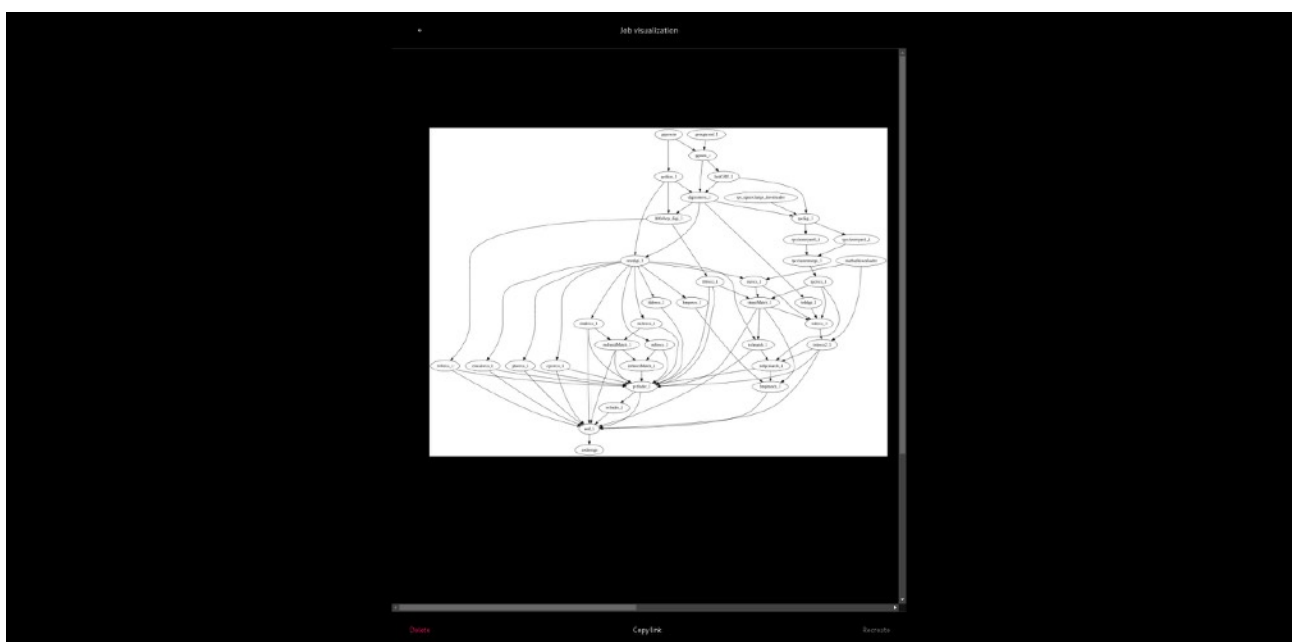


Configuration page, default route.
 'Default mode' tab.





Visualise workflow page.
Loading state.



Visualise workflow page.
Loaded state.

References

CERN. (2023). *ALICE*. Retrieved from <https://home.cern/science/experiments/alice>

ALICE Collaboration. (2023). *ALICE*. Retrieved from <https://alice-collaboration.web.cern.ch/>

IBM. (2023). *What is Monte Carlo Simulation?*. Retrieved from <https://www.ibm.com/topics/monte-carlo-simulation>

GitHub. (2023). *Welcome to the ALICE Analysis Tutorial documentation*. Retrieved from <https://alice-doc.github.io/alice-analysis-tutorial/>