Development of the Automatic Verification and Validation System for the CUPID Code Quality Assurance

J. S. Suh [a*], I. K. Park [b], S. J. Do [b], H. Y. Yoon [b]
[a]*SENTECH Co., 105, Sinildong-ro, Daedeok-gu, Daejeon, Korea, 34324*
[b]*KAERI, 111-989, Daedeok-daro, Yuseong-gu, Daejeon, Korea, 34057*

[*]*Corresponding author: jssuh@s2ntech.com*

Keywords : GitLab, CI/CD, Variation Report, CUPID Code, Quality Assurance

1. Introduction

In recent years, the adoption of Continuous Integration and Continuous Deployment (CI/CD)[1] practices has become increasingly prevalent in software development workflows (Fig. 1). CI/CD offers a streamlined approach to software development, enabling developers to automate the process of integrating code changes into a shared repository and deploying applications to production environments swiftly and reliably.

GitLab, a widely used platform for version control and collaboration, provides robust features for implementing CI/CD pipelines seamlessly within the development workflow. By leveraging GitLab's CI/CD capabilities, teams can automate the building, testing, and deployment of their software applications, thereby enhancing productivity and ensuring the consistency and quality of code releases.

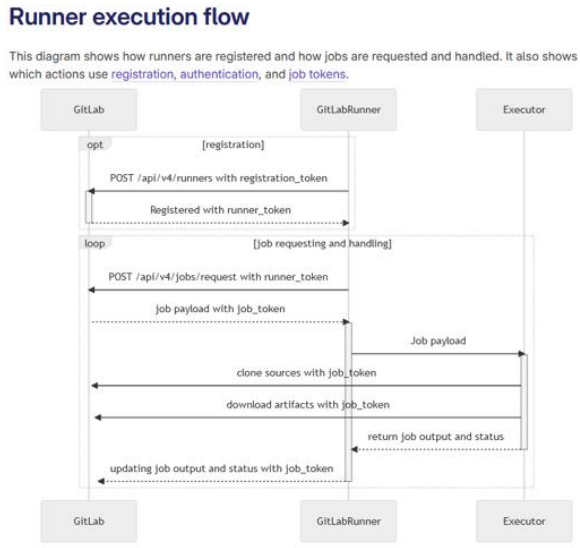

Fig. 1. GitLab/GitLab-Runner execution flow.[2]

The advantages of CI/CD extend beyond automation and efficiency. Some key benefits include:

(1) Improved Collaboration:

CI/CD encourages collaboration among team members by facilitating the integration of code changes from multiple developers into a central repository. This fosters a culture of transparency and accountability, as developers can easily track changes, review code, and provide feedback within the development pipeline.

(2) Faster Time-to-Market:

By automating the build, test, and deployment processes, CI/CD reduces the time required to deliver new features or updates to end-users. Rapid feedback loops enable developers to identify and address issues early in the development cycle, leading to shorter release cycles and faster time-to-market for software products.

(3) Enhanced Quality Assurance:

CI/CD promotes a culture of continuous testing, allowing developers to run automated tests on each code commit to identify regressions or bugs promptly. By integrating testing into the development pipeline, teams can maintain high code quality and reliability throughout the software development lifecycle.

(4) Increased Reliability:

With CI/CD, the process of deploying applications to production environments becomes more reliable and predictable. Automated deployment pipelines ensure consistent deployment practices across different environments, reducing the likelihood of deployment errors or configuration drifts.

(5) Scalability and Flexibility:

CI/CD pipelines are highly scalable and adaptable to the needs of diverse development projects. Whether developing small-scale applications or large-scale enterprise systems, teams can customize CI/CD pipelines to accommodate various workflows, technologies, and deployment targets.

In this paper, we explore the utilization of GitLab CI/CD functionality in the context of scientific computing and numerical analysis. By harnessing the power of CI/CD, researchers can streamline the execution of complex computational tasks, enhance collaboration, and accelerate the pace of scientific discovery.

2. Methods and Results

2.1 Migration of DOS Batch Operations to GitLab CI/CD

The migration of legacy DOS batch operations to GitLab CI/CD represents a significant step towards modernizing the computational workflow and improving efficiency in scientific computing environments. In this section, we delineate the process of transforming the existing DOS batch scripts into GitLab CI/CD jobs[3], enabling the automation of the computational workflow described earlier.

(1) Directory Management:

In the DOS batch environment, the task of organizing input files into separate directories for each case is achieved through manual scripting. This involves creating directories and copying relevant input files for each case individually.

In the GitLab CI/CD pipeline, directory management is automated using YAML configurations. By defining stages and jobs within the pipeline, we can dynamically create directories for each case and populate them with the necessary input files.

```
stages:
  - directory_creation
  - cupid_analysis
  - graph_generation
  - report_generation
  - pdf_generation
```

The YAML script specifies the sequence of actions to be performed, ensuring the orderly execution of the computational workflow.

(2) CUPID code Execution:

The execution of CUPID codes[4] within the DOS batch environment involves manually invoking the executable files for each case, providing input parameters, and capturing the output results.

With GitLab CI/CD, CUPID code execution is automated through the definition of job scripts within the pipeline configuration. Each job corresponds to a specific case, wherein the CUPID code is invoked with the requisite input files.

```
cupid_analysis:
  script:
    - ./run_cupid_analysis.sh $CASE_INPUT_DIR
```

By parallelizing the execution of CUPID code analysis jobs across multiple runners, we can expedite the computation process and enhance resource utilization.

(3) Graph Generation:

After the completion of CUPID code analysis for all cases, the next step involves generating graphical representations of the computational results using Matplotlib.

GitLab CI/CD facilitates the integration of graph generation tasks into the pipeline, wherein Python scripts utilizing Matplotlib are executed to produce graphs based on the output data.

```
graph_generation:
  script:
    - python generate_graphs.py $RESULTS_DIR
```

By leveraging Docker containers with pre-installed dependencies, we ensure the reproducibility of graph generation across different computing environments.

(4) Report Generation:

The final stage of the computational workflow entails the generation of a validation report incorporating the computational results and corresponding graphs.

Using LaTeX templates, we automate the creation of the validation report within the GitLab CI/CD pipeline, integrating the generated graphs and textual analyses.

```
report_generation:
  script:
    - pdflatex generate_report.tex
```

GitLab artifacts are utilized to capture the intermediate files generated during the report generation process, facilitating traceability and review.

(5) PDF Generation:

The culmination of the workflow involves the compilation of the validation report and associated graphs into a comprehensive PDF document for dissemination.

GitLab CI/CD orchestrates the compilation of the LaTeX document and associated resources, resulting in the generation of a PDF file ready for distribution.

```
pdf_generation:
  script:
    - pdflatex main_pdf.tex
```

The automated PDF generation process ensures consistency and reproducibility in the dissemination of research findings.

2.2 Empowering Scientific Computing with GitLab CI/CD

The migration of DOS batch operations to GitLab CI/CD represents a paradigm shift in the automation and management of computational workflows in scientific computing. By harnessing the power of GitLab CI/CD, researchers can streamline the execution of complex simulations, facilitate collaboration, and enhance the reproducibility of computational analyses.

Through the integration of GitLab CI/CD into our computational workflow, we have demonstrated the feasibility of automating repetitive tasks, such as directory management, code execution, graph generation, and report generation. This automation not only accelerates the pace of scientific discovery but also improves the reliability and traceability of computational results.

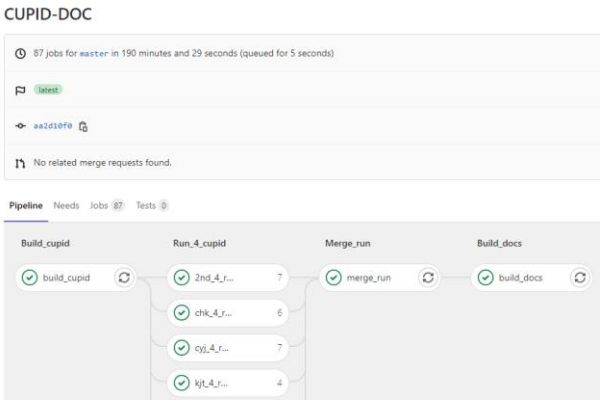

Fig. 2. Output of the CUPID Code Quality Assurance with GitLab CI/CD and GitLab-Runner.

3. Conclusions

Moving forward, the adoption of GitLab CI/CD in scientific computing environments holds immense potential for advancing research methodologies and accelerating the pace of innovation. By embracing modern software development practices within the realm of scientific computation, we can unlock new avenues for collaboration, reproducibility, and discovery.

Acknowledgments

This work was supported by Korea Atomic Energy Research Institute (KAERI, 524510-23).

REFERENCES

[1] GitLab. 2024. *Use CI/CD to build your application*. https://docs.gitlab.com/ee/topics/build_your_application.html.
[2] GitLab. 2024. *Runner execution flow*. https://docs.gitlab.com/runner/.
[3] GitLab. 2024. *Configure GitLab Runner*. https://docs.gitlab.com/runner/configuration/.
[4] I. K. Park, S. J. Lee, H. Y. Yoon, Development of Regulatory Verification Technology Element for Nuclear Reactor Based on Multi-scale Thermal Hydraulics Analysis, KAERI, KAERI/RR-4069, 2015.