Introduction of the Automatic Verification and Validation System for the System Code Quality Assurance

D. H. Kang [a*], M. J. Lee [a], Y. S. Cho [a], J. S. Suh [a], T. Kim [b]
[a]*SENTECH Co. Ltd., 105, Sinildong-ro, Daedeok-gu, Daejeon, Korea, 34324*
[b]*Incheon National University, 119 Academy-ro, Yeonsu-gu, Incheon, Korea, 22012*

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

Keywords : GitLab, CI/CD, Quality Assurance, SMR, System Code

1. Introduction

GitLab Continuous Integration and Continuous Deployment (CI/CD) [1] is a continuous method of software development, where you continuously build, test, deploy, and monitor iterative code changes. In recent years, the adoption of GitLab CI/CD practices has become increasingly prevalent in software development workflows (Fig. 1). GitLab 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. GitLab is a powerful tool that automates and integrates software development and deployment processes, as it includes not only a code repository but also CI/CD features.

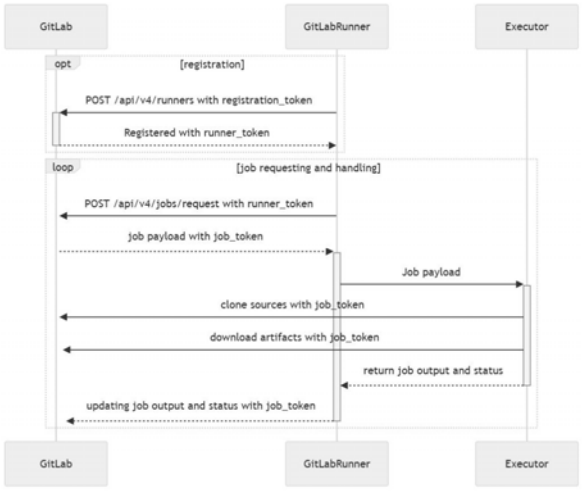

Fig. 1. GitLab/GitLab-Runner execution flow [2]

GitLab CI/CD is an integrated continuous delivery service provided by GitLab. It includes the following key features:

- Set up a CI/CD pipeline: You can set up a separate CI/CD pipeline for each project via the .gitlab-ci.yml file.
- Automated builds and tests: You can automatically run builds and tests when code is pushed.
- Container-based execution environment: You can use a container-based execution environment such as Docker to maintain environment consistency and simplify dependency management.
- Continuous deployment: When builds and tests are successful, you can automatically move to the deployment stage.

The advantages of CI/CD extend beyond automation and efficiency. Some key benefits include: [3]

- Improved Collaboration: CI/CD fosters a culture of transparency and accountability by encouraging collaboration among team members by integrating code changes from multiple developers into a central repository, making it easy for developers to track changes, review code, and provide feedback within the development pipeline.
- Faster Time-to-Market: CI/CD automates the build, test, and deployment processes, reducing the time required to deliver new features or updates to end users.
- Enhanced Quality Assurance: Developers can run automated tests on each code commit to immediately identify regressions or bugs. By integrating tests into the development pipeline, teams can maintain high code quality and stability throughout the software development life cycle.
- Increased Reliability: With CI/CD, the process of deploying applications to production environments becomes more reliable and predictable.
- Scalability and Flexibility: CI/CD pipelines are highly scalable and adaptable to the needs of diverse development projects.

In this paper, we would like to introduce the process of building a verification calculation environment based on GitLab for system code evaluation in the project "Development of base technology for regulatory verification of light-water SMR accident analysis and core design" as part of the small and medium-sized reactor safety regulation foundation technology development project being carried out as a new project by the Nuclear Safety and Security Commission (NSSC) and the Regulatory Research Management Agency for SMRs (RMAS) [4].

2. Methods and Results

2.1 Building a GitLab-Based Verification and Validation Calculation Environment for System Code Evaluation

GitLab will serve as a very effective platform for building a verification and validation computation environment for code evaluation and will perform quality assurance on system code and computation results.

(1) Continuous Integration and Deployment

CI/CD pipelines provide excellent capabilities for continuously integrating and testing code changes to improve quality and automatically deploying them. CI/CD capabilities enable developers to quickly check the quality of code changes and efficiently automate the deployment process.

(2) Automated Quality Assurance

GitLab improves your QA process with automated testing and verification & validation of code changes. GitLab can help you efficiently carry out your QA process and increase the stability and reliability of your system code. Automating the QA process helps reduce human errors and increase test coverage.

(3) Basic Configuration for Building a GitLab Environment

Build a stage and job environment for CI/CD for verification calculations based on accident scenarios. Store the system code input deck for each accident scenario in the "Input directory" and program a YML (.gitlab-ci.yml)file for GitLab to perform verification calculations in the "Run directory". Create an Excel file with a list of graphs for each accident scenario and the information required to create the graphs using Gnuplot.

(4) Basic a Code Environment based on GitLab/GitLab-Runner

Configure the project with a GitLab server on the LINUX operating system and a code development environment (using FORTRAN compiler) on the Windows operating system. Configure a system code verification and validation calculation example project using GitLab-Runner (as a submodule of the verification calculation project) linked to the code development environment. Create graphs using pre-written Gnuplot script files using the results of system code verification calculations for each accident scenario.

(5) Compare Feature

To ensure the reproducibility and reliability of the verification results, we plan to add a feature to compare previous results and with the latest results. With this feature added, you can compare previous results graphs and latest results graphs in one graph to see the differences.

2.2 Enabling Scientific Computing with GitLab CI/CD

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.

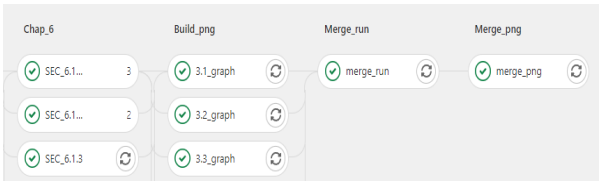

Fig. 2. Display of the System Code Quality Assurance with GitLab CI/CD and GitLab-Runner.

3. Conclusions

GitLab offers numerous benefits, especially for development teams and organizations that prioritize collaboration, continuous integration/continuous delivery, and development operations (DevOps) practices. Overall, GitLab's comprehensive features, flexibility, and strong community support make it a powerful tool for modern DevOps practices. We have built an environment for verification calculations based on accident scenarios using powerful tools and have enabled quality assurance processes for verification calculations to be performed efficiently, thereby increasing the safety and reliability of the system code.

Acknowledgments

This work was supported by the Nuclear Safety Research Program through the Regulatory Research Management Agency for SMRs (RMAS) and the Nuclear Safety and Security Commission (NSSC) of the Republic of Korea (No. 1500-1501-409).

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] J. S. Suh, et al., Development of the Automatic Verification and Validation System for the CUPID Code Quality Assurance, KNS Spring Meeting Jeju, Korea, May 9-10, 2024.
[4] RMAS and NSSC, Development of Base Technology for Light Water SMR Accident Analysis and Core Design Regulation Verification, New project for the 2024 Small and Medium-Sized Nuclear Reactor Safety Regulation Base Technology Development Project, 2024.