CI for ASAP Applications: Using Github Actions for Rapid Development

Haniye Kashgarani[1,2], Supreeth Suresh[1], and Cena Brown[1]
[1]National Center for Atmospheric Research, [2]University of Wyoming

ABSTRACT

This project uses Continuous Integration (CI) to enhance the reproducibility, productivity, and efficiency of the software development process, specifically for parallel GPU and CPU applications. This method streamlines code management and helps resolve potential issues early when multiple community developers contribute to a project.

- The project involved creating a CI pipeline using GitHub Actions
- This tool allowed us to create self-hosted runners customized to our specific hardware and software needs.
- We applied CI to GPU-accelerated CLUBB, MPAS-A, and MURaM models, requiring a special hardware-software setup.
- The self-hosted runners were set up on machines equipped with GPUs (CASPER).

As an essential part of the project, all processes have been documented to provide a comprehensive guide for future implementation. Additionally, we have provided example workflows for building and testing MPI, CUDA, Fortran, and C programs with OpenACC support with commonly used build systems like Make, Autoconf, and CMake. These resources support both current project needs and expected future development.

WHAT IS CI/CD?

CI (Continuous Integration) is a method that automates the integration process each time team members commit code, ensuring seamless teamwork and early problem detection.

CD (Continuous Delivery) means we're always ready to show our work. It's like having a project that's always polished and ready for a presentation and deployment, with minimal extra effort needed.

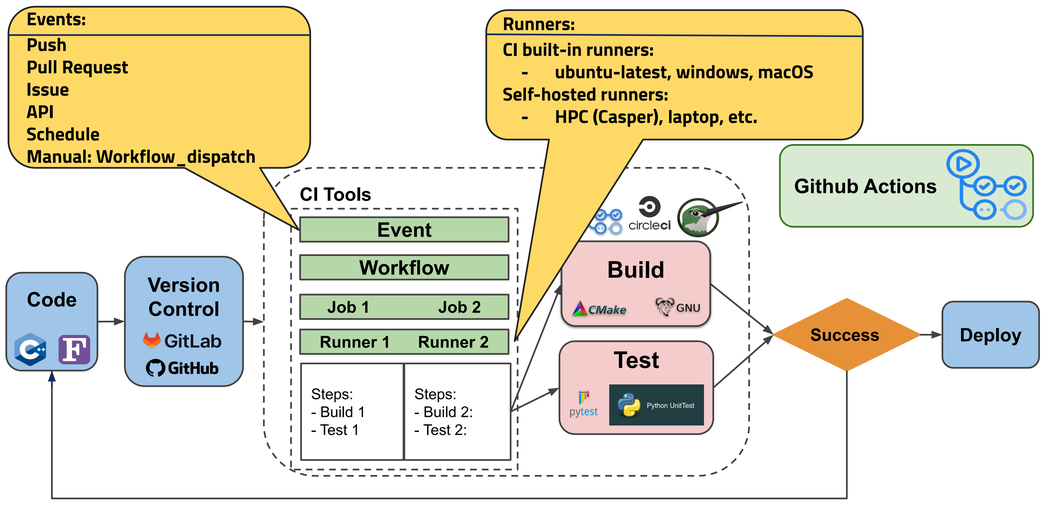
Why CI/CD Rocks:

- Dependency management
- Library versions: Dealing with CUDA and OpenMPI versions
- GPU compatibility
- Frequent automated testing
- Early detection of bottlenecks
- Faster Integration
- Speedy Development

Investigated Tools: CircleCI, and Github Actions

Runners: built-in runners, self-hosted runners (HPC, laptop, Rescale nodes, RAPIDS)

CI PIPELINE



WHY GITHUB ACTIONS?

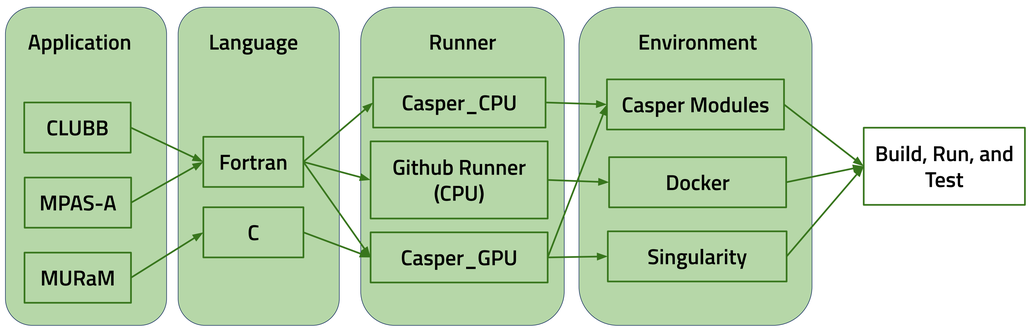
- **Flexibility:** GitHub Actions can handle a wide variety of tasks.
- **Built for GitHub:** No need for external tools. GitHub Actions are part of GitHub itself.
- **Powerful Workflows:** Design workflows to run on triggers with high automation capabilities
- **Diverse Environment Support:** GitHub Actions work with Windows, Linux, or MacOS.
- **Community Support:** Large community creating and sharing actions.
- **Self-Hosted Runners:** Can set up our own runners with specific hardware or software.
- **Free for Public Repositories:** Unlimited free minutes for open source projects.

CLUBB, MPAS-A, and MURaM

MURaM: simulates aspects of the Sun's activity. (Code in C/C++, MPI and GPU-enabled)

CLUBB: models various types of clouds and atmospheric layers, which aids our understanding of weather patterns and climate changes. (Code in Fortran, MPI and GPU-enabled)

MPAS-A: It is like a flexible weather magnifying glass. It can focus on specific regions without losing sight of the bigger picture. (Code in Fortran, GPU-enabled)



CHALLENGES

- Learning about and creating specific documentation for Make, CMake and Autoconf
- Developing or finding a suitable Docker image including both NVHPC and OpenMPI libraries.
- Choosing the cheapest way to set up self-hosted runners:
  - CircleCI: Organization level self-hosted runner,
  - Jacamar: Only for Gitlab repositories,
  - Rescale GPU resources: Expensive,
  - RAPID GPU self-hosted runners: Required subscription.

CONCLUSION AND FUTURE WORK

We achieved the two main goals of the project this summer:

- Documentation and Examples
- Focus on implementing CI for multiple ASAP applications.

Future work:

- Implementing CD as the next step for automating the software deployment and release.
- Building CI workflows for all the ASAP applications.
- Change the repository level self-hosted runner to Organization level self-hosted runner and continuously running the runner to make it easier to run the workflows.

ACKNOWLEDGEMENTS

Thank you to Carl Ponder (NVIDIA) for providing the Docker container used in this project. Additional thanks to CISL, the SIParCS team, and the SIParCS 2023 cohort for an amazing and unforgettable summer.

Examples and Documentations:

hkashgar@uwyo.edu