A.Y. 2024-2025 Software Engineering for HPC
DevOps Project:
goal, schedule, and rules
READ THIS VERY CAREFULLY
NO EXCUSE FOR IGNORING WHAT WE WRITE HERE

Goal and approach

The objective of this project is to apply in practice what you have learned during the second part of the course about automating testing and build processes, containerization, and usage of HPC clusters through SLURM.

The exercise consists of testing and automating a DevOps pipeline for a software project that includes converting color images into greyscale, following the steps described below.

For simplicity, we suggest you to keep the same teams as for the previous project, but this is not mandatory. In particular, we encourage the students who have developed the first project alone to join forces with the others in the same situation. At the end of your work, one and only one of the students belonging to the group must fill in the form available here https://forms.office.com/e/RnaZicBjWQ containing the links to your repositories.

The work is meant to be done mainly in class, but you may need more time to fix any bug in your pipeline.

The deadline for submission is Wednesday June 2 at 23.59 CEST.

Step 1 – testing

The goal is to test a specific buggy implementation of the image conversion software. Your objectives are:

- To identify test cases able to spot the errors.
- To automate the execution of tests using Google Test.

The implementation to be tested is a function that converts a color image in PPM format (represented in memory) into grayscale as a PGM file.

The function is made available to you as compiled object code with the following signature:

```
void convertToGrayscale(const std::vector<std::vector<std::array<int, 3>>>& rgbImage, int rows, int cols, GrayscaleMethod method, std::vector<std::vector<int>>& grayscaleImage);
```

Note that this signature follows the Vulkan[1] coding style.

Use the template available at
https://github.com/SimoneReale/SE4HPC_project_2025_first_part

to create your own GitHub repository. Name it from the surnames of all group participants, for instance: BrownRossiXiang_DevOps_first_part if the surnames of the three group members are Brown, Rossi and Xiang, respectively. Set it as private. Add all team members and the accounts SimoneReale and dinitto as contributors. The project includes:

- Under the lib folder, the compiled library libimage_processing.a containing the convertToGrayscale function.
- The header defining the function signature under the include folder.
- The file test_image_conversion.cpp under the test folder, containing a simple test case.
- A Makefile for building the software.
- A file working_image_conversion.cpp under the src folder containing a correct implementation of the image conversion for reference.
- A series of Python scripts that generate random PPM images or convert existing ones.

Your task is to:

- Extend the **test_image_conversion.cpp** file with your test cases, including clear comments explaining why each test case was chosen and what it is expected to validate. Note that you do not need to modify the other pieces of code.
- Push your results on your GitHub repository.
- Execute the code. Note that the library has been compiled for Linux. If your local environment does not support it, consider using GitHub Actions and their runners—this will also help with the next step.

The assessment of this step will be based on the quality of your testing code and of your test cases and associated comments.

For further details about the software to be tested see the readme.md file in the repository.

[1] Vulkan is a 3D graphics low level API https://www.vulkan.org/

Step 2 – From build to release and manual job execution

Now focus on the correct implementation of the `color-to-grayscale` image conversion, found in the following template:
https://github.com/SimoneReale/SE4HPC_project_2025_second_part

Your tasks:

Preparation:

- Use the template above to create your private GitHub repository. Name it from the surnames of all group participants, for instance: `BrownRossiXiang_DevOps_second_part` if the surnames of the three group members are Brown, Rossi and Xiang, respectively. Add all team members and the accounts `SimoneReale` and `dinitto` as contributors.
- Add your Step 1 tests to the repository.

Automating the build, test, and release processes:

- Create a CI/CD pipeline that builds and tests the project on every push.

Containerizing the application:

- Write a Singularity container definition for the image conversion tool and include it in your repo.
- Extend the CI/CD pipeline to build the container image from its definition.

Executing on the Galileo 100 cluster:

- Write a `job.sh` script to run your containerized application on SLURM.
- Ensure standard output and error are redirected to text files.
- Transfer the script and container to Galileo100.
- Submit the job and verify the results.
- Push `job.sh` and the output files to your GitHub repository.

Step 3 – Automating Job Submission with Containerization

Extend the CI/CD pipeline to fully automate the process from a GitHub push to execution of the containerized application on SLURM.

You must:

- Move the container from the GitHub runner to the cluster (e.g., using scp or a Singularity registry).
- Handle credentials securely using GitHub Secrets (**never hard-code tokens or passwords**).

Step 2 and 3 will be assessed by analysing the defined pipeline and executing it to verify that it works as expected.

Finalization

- Clean up your repositories as needed.
- Associate an open source license to each of your repositories and the files you have created in these repositories. In [1] you find general information about open source licenses, in [2] you find those that are approved by the Open Souce Initiative (OSI), in [3] you find guidelines on how to include a specific license into your repository.
- List in the Readme the names of all team members and the role of each one in the project, describe your test cases and your pipeline, present the difficulties you have faced. For those you have overcome, explain how you did it. For the others describe the attempts you made.
- Fill in the submission form https://forms.office.com/e/RnaZicBjWQ by indicating, once again, the information about all team members and the links to your two repositories. Only one single representative of each team will fill in this form on behalf of all his/her mates. Please avoid submitting duplicates. You can modify the form after submission if needed.

[1] Wikipedia. Open-source license. https://en.wikipedia.org/wiki/Open-source_license

[2] Open Source Initiative. OSI Approved Licenses. https://opensource.org/licenses

[3] GitHub. Adding a license to a repository. https://docs.github.com/en/communities/setting-up-your-project-for-healthy-contributions/adding-a-license-to-a-repository