Mohamed Basuony

DevOps-Integrated HPC Workflow

with Apptainer, GitLab CI/CD, and SLURM

Agenda

Why DevOps in HPC?

- HPC workflows are traditionally manual, error-prone, and hard to reproduce.
- DevOps brings automation, testing, portability, and collaboration.
- Objective: Treat HPC pipelines like production code — testable, portable, versioned.

DevOps Pipeline Variants: Cloud vs HPC

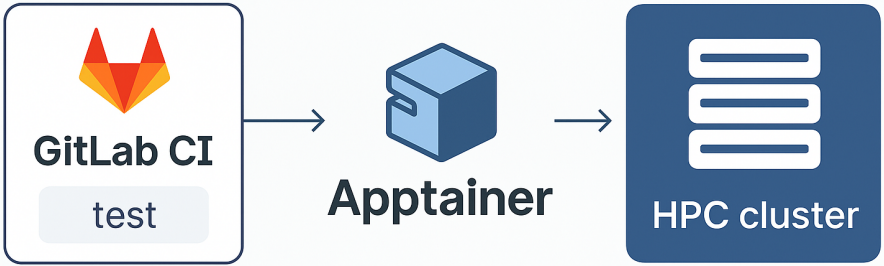
HPC Workflow (Reproducible, Secure, SLURM-Compatible)

Cloud Workflow (Docker-Native, Scalable, CI/CD-Driven)

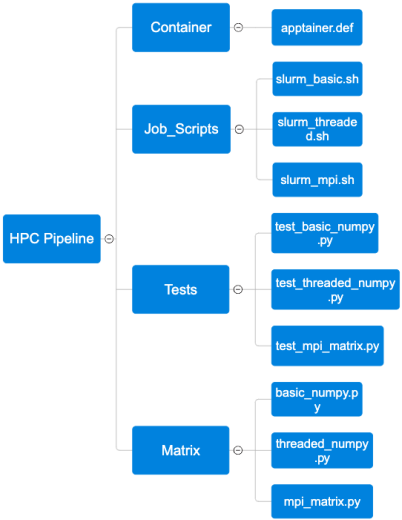
Project Goals

- Design an HPC matrix workload pipeline.
- Use Apptainer for reproducible containers.
- Automate build + test + deployment with GitLab CI/CD.
- Submit and run jobs via SLURM.
- Identify CI/CD limits in HPC environments.

Pipeline Overview



Directory Structure



Python Workloads

- `basic_numpy.py` Multiplies two **1000×1000** matrices on a single core. Used to validate the container and CI pipeline with minimal compute overhead.
- `threaded_numpy.py` Multiplies two **4000×4000** matrices using OpenBLAS threading. Benchmarks multi-core performance on a single node (`OMP_NUM_THREADS=8`).
- `mpi_matrix.py` Distributed multiplication of **4000×4000** matrices using `mpi4py`. Each MPI process handles a chunk of rows. Validates multi-process logic under SLURM.

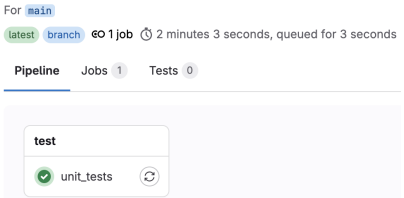
Python Workloads

- `basic_numpy.py` Multiplies two **1000×1000** matrices on a single core. Used to validate the container and CI pipeline with minimal compute overhead.
- `threaded_numpy.py` Multiplies two **4000×4000** matrices using OpenBLAS threading. Benchmarks multi-core performance on a single node (`OMP_NUM_THREADS=8`).
- `mpi_matrix.py` Distributed multiplication of **4000×4000** matrices using `mpi4py`. Each MPI process handles a chunk of rows. Validates multi-process logic under SLURM.

Purpose: Verify pipeline correctness and scalability across compute levels.

Automated Testing with Pytest

- Unit tests run locally and in GitLab CI (no SLURM required).
- Each workload is covered:
  - `basic_numpy.py` — validates shape and values
  - `threaded_numpy.py` — checks correctness under multithreading
  - `mpi_matrix.py` — verifies result consistency across MPI processes



Pytest output (CI snippet)

Containerization with Apptainer

- Environment defined in `container/apptainer.def`
  - Installs NumPy, `mpi4py`, OpenMPI, Pytest and Includes source code and tests
- Container built manually on login node:
  apptainer build matrix-demo.sif container/apptainer.def
- Used in all SLURM jobs with:
  apptainer exec matrix-demo.sif python3 ...

```
[2025-pchpc] u17408@glogin13 hpc-pipeline-demo $ apptainer exec matrix-demo.sif
python3 matrix/basic_numpy.py
Running single-threaded NumPy multiplication for 1000×1000 matrices...
Elapsed time: 0.053283 seconds
```

Terminal output: container build and execution

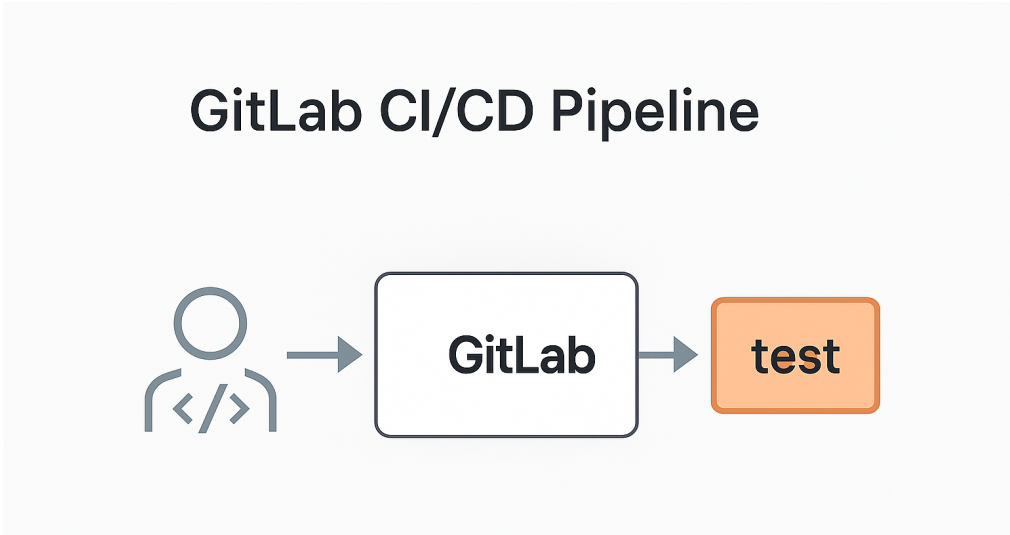
SLURM Job Submission

- Three job scripts for single-core, multi-core, and distributed MPI
- Each script loads Apptainer and runs:
  srun apptainer exec matrix-demo.sif python3 matrix/...
- Submission command:
  sbatch job_scripts/slurm_threaded.sh
- SLURM handles resource allocation, CPU binding, and execution

```
[2025-pchpc] u17408@glogin13 hpc-pipeline-demo $ sbatch job_scripts/slurm_thread
ed.sh
Submitted batch job 9904417
```

SLURM job submission and output snippet

GitLab CI/CD Pipeline



CI/CD Benefits

- All changes tested immediately.
- Tracks failures via logs.
- No manual setup steps on dev machine.
- Repeatable build + run on cluster.

Why GitLab Deploy Failed

Deployment hurdles on cluster:

- SSH private key format issues.
- GitLab masks multiline secrets by default.
- Cluster-side restrictions on write access.
- Cannot run `module load` from non-interactive shell.

Fix: Switched to manual scp + sbatch workflow after GitLab tests.

Workaround Deployment Strategy

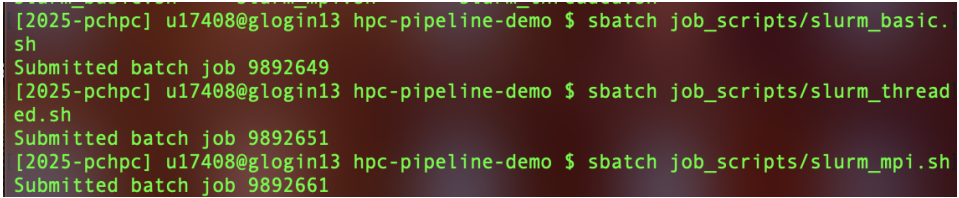
- CI/CD tests and builds locally.
- Manual steps for:
  - Copying code to HPC cluster
  - Building container
  - Submitting SLURM jobs
- Logs and outputs pulled for inspection.

Key Results

- ✓ **Automated testing works:** *Pytest runs reliably on every commit via GitLab CI*
- ✓ **Reproducible containers:** *Same Apptainer image builds locally and on HPC nodes*
- ✓ **Scalable SLURM jobs:** *Workloads run from 1 to 4 nodes without modification*
- **SSH deploy failed:** *GitLab CI can't push to cluster due to SSH/key issues*

SLURM Job Submission

- Submitted three jobs via `sbatch`:
  - `slurm_basic.sh` – 1 core
  - `slurm_threaded.sh` – 8 cores
  - `slurm_mpi.sh` – 2 nodes, 4 ranks
- Each job used `apptainer exec` to launch code inside container
- SLURM handled resource allocation and logging



Execution Timing Results

- `basic_numpy.py` — single-threaded, 1000×1000 matrix **0.0486s**
- `threaded_numpy.py` — OpenBLAS threads, 4000×4000 **1.26s**
- `mpi_matrix.py` — MPI, 4000×4000 **0.29s**

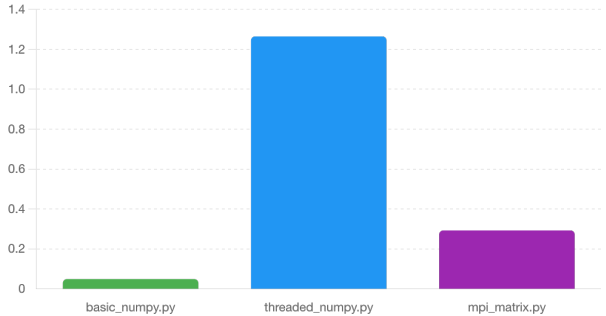
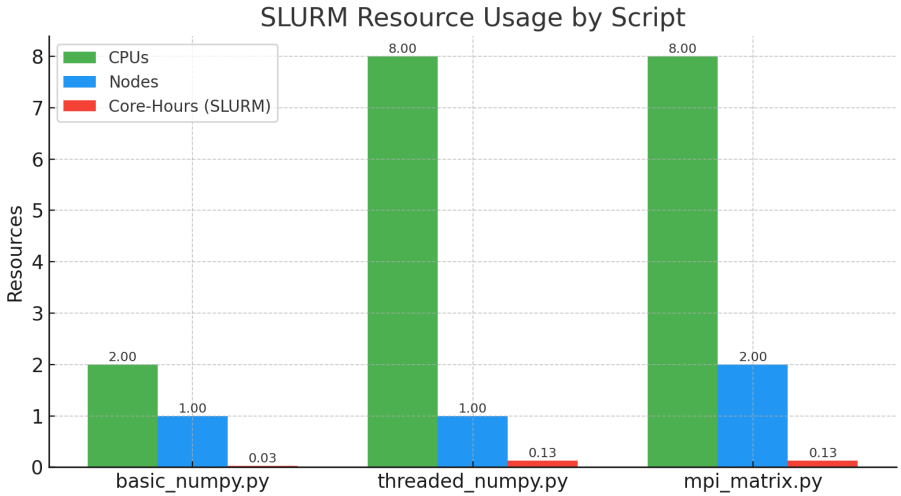


Chart: Execution time comparison from real container runs

SLURM Job Logs: Resources Billing



What's Next?

- Use GitLab runners directly on the HPC cluster.
- Automate secure deploy using GitLab deploy keys.
- Add ML/DL workloads to increase realism and diversity.
- Profile jobs using CPU/memory benchmarks for bottlenecks.
- Add automatic result validation post-SLURM runs.
- Integrate result logging and visualization for traceability.

Takeaways

- CI/CD workflows improve stability and reproducibility in HPC.
- Testing and containerization reduce human error and setup time.
- Manual fallback is essential in restricted environments.
- Full automation is possible — but only with system-level support.
- Infrastructure constraints matter as much as tooling.

DevOps isn’t just for the cloud.

References

- Apptainer Docs: https://apptainer.org/docs/
- GitLab CI/CD Pipelines: https://docs.gitlab.com/ee/ci/
- SLURM Workload Manager: https://slurm.schedmd.com/documentation.html

Thank You!

Questions?