Software Landscape

Learning Goals

- Understand the software environment on CoSTAR
- Learn how to install and manage software in user space
- Compile your own software in your user space
- Understand containers and Apptainer workflows
- Apply best practices for efficient HPC software usage
- Know where to find documentation and support

The HPC Software Environment

- **The Linux Shell is Your Command Centre:** interact with the cluster through command-line operations.
- **“No Root Access” Philosophy:** users cannot install software system-wide, keeping the cluster stable and secure.
- **How to get *software*:**
  - **User-installed software:** install tools in your own directories, including virtual environments and compiled software.
  - **Containers** such as Apptainer package complex software and dependencies for portability.

👉 Graphical Interface Option: Open OnDemand (OOD) provides a web portal for file management, job submission, and selected graphical applications (JupyterLab, VS_Code, MatLab etc.)

Custom Software in Your Own Space

Installing in your user directory:

- Compile your own code.
- Install language-specific packages such as Python and R packages.
- Make no system-wide changes; there is no root access.
- 👉 Users are responsible for managing their own software stack.

Virtual environments: Conda, venv, and poetry.

- Create isolated environments for specific projects.
- Manage dependencies independently per project.
- Prevent version conflicts and improve reproducibility.
- Use virtual environments for complex Python and R workflows.

Containers for Full Software Isolation

Container-first policy:

- No system-wide software or Lmod modules.
- Users bring and manage their own software stack.
- Containers are the default and recommended model.

What is a container?

- A portable package that bundles an application with its dependencies.
- Includes code, libraries, system dependencies, and runtime.
- Ensures consistent execution across environments.
- Provides isolation, portability, and strong reproducibility.

Containerised Workflows on CoSTAR

Containers on COSTAR use Apptainer, formerly Singularity.

- Docker is not available directly, but Docker images can run through Apptainer.
- Key idea: Docker builds; Apptainer runs.
- Native to HPC and integrates cleanly with Slurm.
- Required on CoSTAR because there is no centrally installed software.
- No root access required; containers run as the invoking user.
- Single-file .sif images are easy to manage and share.

Core principle

👉 **bring your own software stack.**

Working with Apptainer

Three Steps to Use Containers

1️⃣ Build Custom Images

- Build a container from a definition file (.def)
- The .def file describes how the container should be built
- The build process creates a .sif image (the container file you run)
- Images can be built: Locally or on the cluster (interactive session)

2️⃣ Pull Existing Images

- From public registries (e.g. DockerHub)
- Docker images are automatically converts to .sif

3️⃣ Run the Container

- Interactive exploration and debugging → shell
- Run a specific command (recommended for batch jobs) → exec
- Execute the container's default program → run

👉 Containers wrap and isolate your software.

.def → build → .sif
(recipe) (container image)

Best Practices for Containerised Research

- Pre-pull images to reduce job startup time.
- Version-control your .def files, for example in GitLab repositories.
- Test an interactive shell session before large-scale runs.
- Keep images minimal and purpose-driven using lightweight base images.
- Frequently empty your container cache.

Efficient Software Workflow

- Develop and test locally first: start small on your own machine or in a small interactive cluster session.
- Incremental testing: test with small datasets and short run times before large runs.
- Deploy when ready: submit long-running, large-scale jobs only when the code and configuration are ready.
- Why this matters: save compute time, power, and storage.
- Faster iteration: catch errors quickly with small tests.
- Green computing: use resources efficiently and sustainably.
- Think of it as: Quality assurance for your research – test thoroughly before a big launch!.

Support & Resources

1. 📘 Containers on CoSTAR documentation
2. 📗 Main CoSTAR docs (Software)
3. 🧰 CoSTAR: Slurm Examples
4. 🎫 CoSTAR Support Portal
5. 💬 CoSTAR MS Teams channel