Optimizing Shared Programming Environments on Tri-lab HPC Resources: Standardizing Uenv

Bodhi Rubinstein | Los Alamos National Laboratory | HPC-ENV
Mentors: Francine Lapid, Shivam Mehta, Paul Ferrell

Abstract

High-performance computing (HPC) systems are designed to provide a reliable and large-scale computing platform for a wide variety of scientific applications. The programming environment is a key component that provides a stable foundation on which software for a cluster is built. Being able to build, maintain, configure, and deploy accessible software to meet the wide variety of users' needs is very involved and can be an improvised, fragile, and messy process. At Los Alamos National Laboratory (LANL), an additional challenge arises with the need to share reproducible software stacks for use at Lawrence Livermore National Laboratory (LLNL) and Sandia National Lab (SNL), as part of the NNSA Tri-lab collaboration. Uenv is a tool built by the Swiss National Supercomputing Centre (CSCS) aimed to address this challenge. Uenv optimizes this process by packaging and preserving entire scientific software stacks as "user environments" inside of a single Squashfs file, which are stored in a shared container artifact registry. Although Uenv has proved to be a very useful tool on CSCS' Alps supercomputer, much of their implementation is system specific and not generically accessible to HPC teams at other institutions. HPC systems with higher security considerations, such as those at the Tri-labs, pose even more of a challenge for deploying Uenv. This presentation showcases the deployment of a standardized Uenv proof of concept on a LANL HPC system, and the contributions back to CSCS to improve the accessibility of their production Uenv tool.

Uenv - CSCS

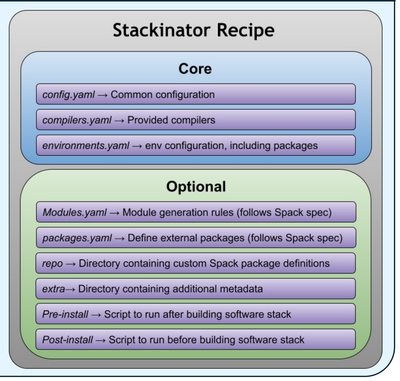
Uenv is a tool that provides "user environments" containing scientific software stacks for use on shared HPC resources:

- Developed and maintained by the Swiss National Supercomputing Centre (CSCS).
- Pseudo-containerized software stacks stored as a single Squashfs file (compressed directory tree).
- Builds using Stackinator, a CSCS tool for building Spack software stacks.
- Stored inside a shared OCI (Open Container Initivate) artifact registry.
- ORAS (OCI Registry As Storage) used to push/pull Squashfs artifacts from registry.
- Deployed by mounting a Squashfs file inside a non-initial mount namespace, using a small setuid binary developed by CSCS called Squashfs-mount.

Stackinator

Stackinator is a tool that builds Spack software stacks from a "recipe":

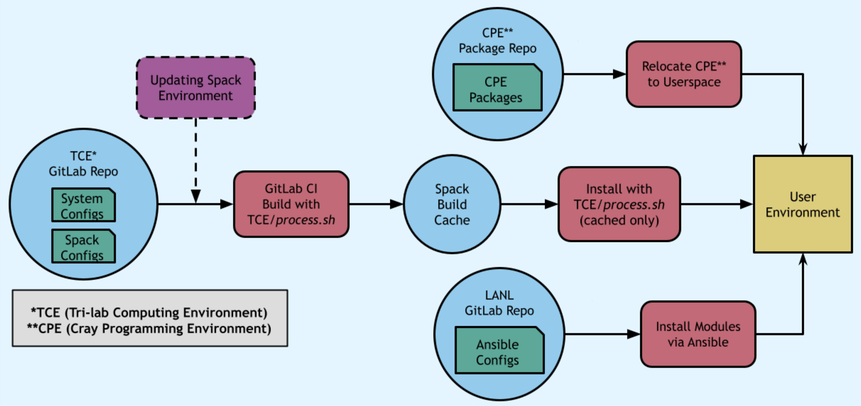
- Specifically designed for HPE Cray EX systems.
- Wraps Spack, the package manager for supercomputers developed at Lawrence Livermore National Lab (LLNL).
- Generates Spack configs and make files needed to build the Spack environments that are packaged into a software stack.
- Stackinator is very similar to the *process.sh* script in LANL's Tri-lab Computing Environment (TCE).
- Outputs a single Squashfs file containing the stack and its meta data.



Motivation

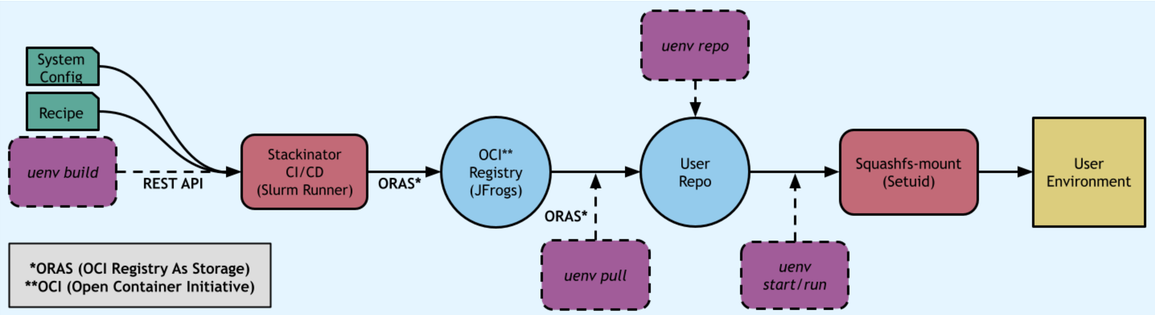
- LANL's current programming environment framework is an improvised solution to the difficulties with maintaining the Cray Programming Environment (CPE) that Cray EX HPC systems provide. Most institutions running Cray EX systems have had to build their own improvised solutions to this problem, so standardizing Uenv provides a common tool for deploying alternative user environments on those systems.
- Spack is designed in a way that makes it great at building throw-away software, but bad for building long term, supportable environments due to the many problems involved in getting it to be reasonably consistent and stable in what it builds. Uenv builds software using Spack the way it was intended: as throw-away, preserved environments, whereas TCE struggles with trying to support its Spack builds long-term.
- Unlike TCE, when a user loads a uenv, they can be assured a consistent environment. There's no chance of accidentally "breaking" a piece of software once it's been built.
- Provides users a modular workflow where they can mix and match toolchains and even mount more than one uenv at a time.
- Allows for integration with containerization services (Charliecloud). Uenv are packaged as a single Squashfs file, so they can be used as the Filesystem Layer when building a container.

TCE/CPE Workflow - LANL

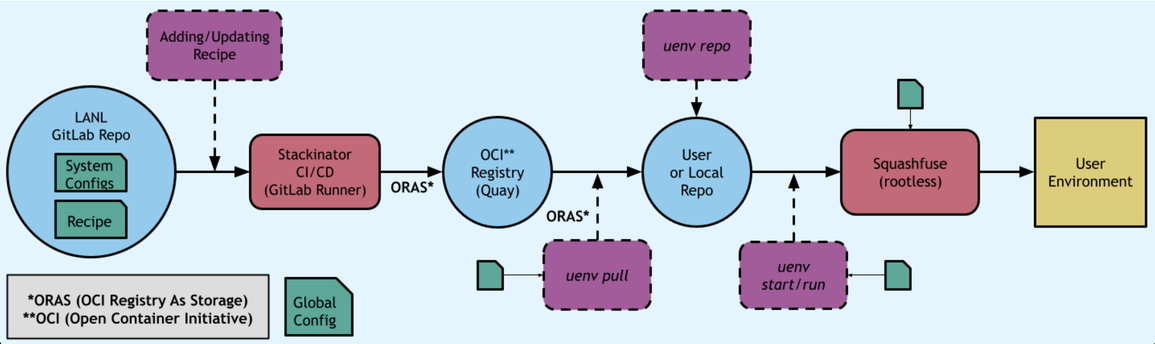


- LANL relocates the Cray Programming Environment (CPE) into user space as packaged executables inside the */CPE* directory (top path).
- Additional packages are built with Spack inside the Tri-lab Computing Environment (TCE; middle path).
- All Cray module files are managed by Ansible (bottom path).
- TCE uses five independent Spack configs in 1000+ line YAML files to manage eight LANL HPC systems. Especially problematic since Spack is bad at incremental updates to its environment.
- A significant amount of time and effort goes into supporting and troubleshooting software built with TCE.

Uenv Workflow - CSCS



Uenv Workflow - LANL



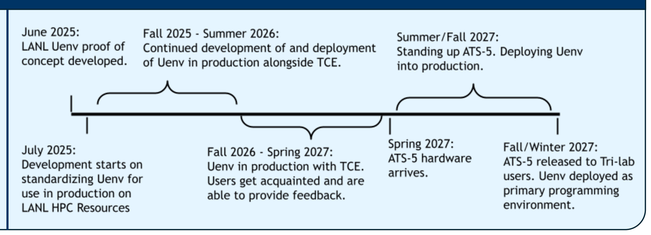
Standardizing Uenv

Feature	Original (CSCS)	Changes (LANL)
Install	1. Meson builds subproject dependencies using *wrapdb.mesonbuild.com*. 2. Install script uses CC→gcc-12 and CCX→gcc++-12.	1. Meson builds subproject dependencies from local files or GitHub (within LANL firewall). 2. Install script uses CC→cc and CCX→CC for Cray compiler wrappers.
Config	User level runtime configuration in: *$HOME/.config/uenv/config* • *Default Uenv repo* • *Colors*	Same user level runtime configuration. Additional global runtime configuration: • TOML config file; Parsed using **toml++**. • Configures: Mounting options, local repos, and OCI registry location.
Build	1. `uenv build` uploads the uenv label and recipe as a tarball to its (hardcoded) CSCS CI/CD build pipeline. 2. The CI/CD pipeline uses a Slurm runner to initialize the Stackinator build process inside a batch job. 3. Once the Slurm runner finishes building, it pushes the Squashfs file to the OCI artifact registry.	1. Any commit to the LANL Uenv GitLab adding or changing an existing Uenv recipe triggers the CI/CD build pipeline. 2. The CI/CD pipeline uses a GitLab runner that configures and builds the Uenv(s) using Stackinator (submodule). Runs inside a pre-built container. 3. Once the GitLab runner finishes building the Uenv, it pushes the Squashfs file to the LANL Quay OCI artifact registry.
Registry	• Uses JFrog OCI artifact registry. URL is hardcoded inside of Uenv. • Uses ORAS tool to push/pull uenv images. • Registry uses three namespaces: *build*, *deploy*, and *service*.	• Uses Quay OCI artifact registries. One registry is accessible to the open (non-classified) network, and the other is accessible to the restricted (classified) network. • OCI registry location can be defined inside the global configuration file instead of hardcoded.
Repos	• Default user repository found in *$HOME/.uenv/repo*, but can be changed inside the user configuration file. • A separate Uenv repository can be specified when running any Uenv command using `–repo <repo_path>`.	• Same user repository features. • Adds the ability to define a list of (pre-existing) local system repositories in global config file. • Local repos are included by default in addition to the user repo when listing available uenv (`uenv image ls`). • When loading a uenv (either `uenv run` or `uenv start`), if it cannot be found in the user repo, the local repos are sequentially searched by default.
Mount	• Passes Uenv environment (`execvpe`) to CSCS tool, Squashfs-mount, which uses recursive `mount` system calls to mount each given pair of `<squashfs>:<mount_point>` • Setuid to root (EUID->0) to enter a non-initial mount namespace (`unshare` system call). The uenvs are only mounted for the user that loaded it. • Returns to user and executes either a shell (`uenv run`) or a command (`uenv start`).	• If Squashfuse option (rootless) in the global config is set to true, Uenv passes the `--squashfuse` flag to the LANL patched Squashfs-mount tool. Otherwise, it uses the setuid method. • Enters non-initial user and mount namespace. EUID/EGID mapped to root (0) inside user namespace to run recursive FUSE (Filesystem in Userspace) mounts without needing any outside privileges. • Forks process, then child executes shell/command inside another (nested) non-initial user namespace with EUID/EGID mapped to outside user. User is illusioned that nothing has changed. • Parent waits for child to terminate, then recursively unmounts all uenv to clean up leftover Squashfuse processes.

Future Work

The end goal is to transition to using Uenv and Stackinator as the primary framework for the programming environment in production on the next NNSA Advanced Technology System (ATS-5). Some of the features that need to be implemented include:

- Parameterizing recipes inside of Ansible.
- Develop non-CI/CD, local build pipeline for users.
- Add OpenMPI as an MPI option for Stackinator recipes (currently Cray MPICH).
- Incorporate existing Spack build cache into Uenv build pipelines.



Collaboration

Swiss National Supercomputing Center (CSCS):

- Contributing LANL proof of concept standardization patches back to CSCS.
- Working with Uenv developers to implement features that transition Uenv into a generic and widely accessible production tool.

National Nuclear Security Administration (NNSA) Tri-labs:

- Los Alamos National Laboratory (LANL), Lawerence Livermore National Laboratory (LLNL), and Sandia National Lab (SNL).
- Collaborating with HPC teams at the Tri-labs to eventually replace the current Tri-lab Computing Environment (TCE) with shared OCI artifact registries containing Uenv.
- Uenv would be shared and consistent between Tri-lab HPC systems, benefiting scientists running applications across multiple Tri-lab resources.

Los Alamos NATIONAL LABORATORY | NNSA National Nuclear Security Administration
Managed by Triad National Security, LLC, for the U.S. Department of Energy's NNSA.
ROSY Submission ID: #88f75a72 LA-UR-25-28268
University of Colorado Boulder