Software Deployment at Facilities

E4S at NERSC 2022, Aug 25, 2022
Aug 25, 2022

Ryan Adamson
Overview:

ECP Software Deployment at Facilities (SD) Activity
Integration of ECP ST products and facility capabilities:

- Maintain facility-based continuous integration platforms for build and test automation
- Deploy E4S-curated ST products using the Spack package manager
- Evaluate vendor system software

Each software team and facility has individual (and often unique!) preferences, constraints, and strategies regarding scientific software development and deployment.
## WBS 2.4.4: Software Deployment at Facilities

<table>
<thead>
<tr>
<th>Project Short Name</th>
<th>PI Name, Inst</th>
<th>Short Description/Objective</th>
<th>Program Office(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4.4.01 Software Integration</td>
<td>Shahzeb Siddiqui (LBL)</td>
<td>Build/Test/Deploy ST products at facilities</td>
<td>ASCR</td>
</tr>
<tr>
<td>2.4.4.04 Continuous Integration</td>
<td>Paul Bryant (ORNL)</td>
<td>Develop and Deploy ECP CI infrastructure</td>
<td>ASCR</td>
</tr>
<tr>
<td>2.4.4.03 Shasta Testing</td>
<td>Jay Srinivasan (LBL)</td>
<td>Evaluate and expedite Shasta releases</td>
<td>ASCR</td>
</tr>
<tr>
<td>2.4.4.05 HPCM / Slingshot Testing</td>
<td>Scott Atchley (ORNL)</td>
<td>Explore HPCM and Slingshot as alternatives to Shasta</td>
<td>ASCR</td>
</tr>
</tbody>
</table>
The Software Deployment team’s role is to package and integrate Software Technology products on exascale systems.
ECP Software Stream: Incubation to Installation

**Facilities**
- Argonne National Laboratory
- Berkeley Lab
- Oak Ridge National Laboratory
- Los Alamos National Laboratory
- Lawrence Livermore National Laboratory

**Software Integration at Facilities**
- Integration of Vendor Stack
- Local builds and testing
- System tuned configurations
- Local filesystem installation
- Local module scheme

**External Contributors on GitHub**

**ECP ST Product Teams**
- Direct AD collaboration
- Incubation and Hardening
- Packaging into Spack Ecosystem
- Level 3 support for E4S

**ECP AD Teams Consume Software From**
- Direct source builds from ST teams
- Spack development branch
- Spack release branches
- E4S release branches
- Facility Installed and Supported SW Stack

**Spack development branch**
- Very Fresh: 400-600 changes/month
- Latest features and package versions
- CI for latest E4S and SDK versions

**Spack release branches**
- Stable Spack
- Stable package versions
- Bugfixes backported

**E4S release branches**
- Facility testing
- Curated public build caches
- Level 2 user support

**ECP Software Stream**
- **Incubate**
- **Harden**
- **Deliver**
- **Deploy**
- **Install**
The Scientific Software Ecosystem model

Software Deployment Efforts

- Software Integration interfaces with the Testing Task Force, E4S, and ST to adapt builds, tests, and deployments to facility systems
- Continuous Integration Infrastructure supports the automation of all activities on facility systems
- Shasta and HPCM testing provides insight into facility hardware from a system functionality perspective
- The Testing Task Force is the vertical integration of all these components

E4S and Spack are the ‘spanning layers’ of the ECP Software Ecosystem Hourglass
Scientific Software Ecosystem roles and responsibilities

<table>
<thead>
<tr>
<th>ST Developers</th>
<th>E4S Software Curators</th>
<th>SI / Facilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>• AD and ST teams implement ECP CI regression testing as appropriate into existing CI frameworks and merge new build/test recipes into spack/develop</td>
<td>• Spack Team implements CI build and smoke testing for spack/develop</td>
<td>• SI Team Deploys packages selected from a tested and versioned E4S release onto Facility Resources.</td>
</tr>
</tbody>
</table>
|   - [https://e4s-project.github.io/policies.html](https://e4s-project.github.io/policies.html) |   - https://github.com/spack/spack  
   - https://cdash.spack.io |   - [https://docs.nersc.gov/applications/e4s/cori/21.05/](https://docs.nersc.gov/applications/e4s/cori/21.05/) |
| • These software development teams typically have other automated CI pipelines, developer-driven unit testing, and other software assurance best practices as determined by their internal policies | • E4S Team Prepares Quarterly Releases for Facility Installation, freezing on a point-in-time commit of spack/develop  
   - [https://github.com/E4S-Project/e4s/tree/master/environments/21.05](https://github.com/E4S-Project/e4s/tree/master/environments/21.05) | • Facility software and operations teams have additional verification and validation tests that are performed according to facility software and operations policies  
   - [https://www.exascaleproject.org/event/buildtest-21-09/](https://www.exascaleproject.org/event/buildtest-21-09/) |

Each role has tailored processes for testing responsibilities provided by that role
Leadership software is *almost* turnkey: early Perlmutter lessons

Vendor Updates to PE can break software stacks

- **Examples of required vendor fixes:**
  - Some older versions of cray provided modules were removed: cray-mpich, cray-libsci, libfabrics, and nvhpc
  - There were some changes to the way certain modulefiles work, such as combining multiple nvhpc modules
  - Missing paths in vendor provided nvhpc modulefiles needed to be fixed

- **Examples of required Spack / E4S fixes:**
  - Spack was unable to find the correct nvhpc, which necessitated a Spack issue ticket and subsequent patch
  - Lmod module creation via Spack resulted in placement in the wrong hierarchy for hypre because it relied on external cray-mpich packages

Lessons Learned from Perlmutter

- It is useful for vendors to use a spack.yaml configuration that we provide to build software as a gating mechanism for a PE release
- This would catch breaking changes that sites would need to troubleshoot regardless
- If packages are not easy to install, we know that they won’t be installed!
- However, software stack installations on Leadership systems are significantly easier *now* than they ever have been
There are three main use cases for CI on facility systems

1) Software Development Testing
   - **Regression Testing**
     - Software failures are detected when new code is introduced. This prevents latent bugs from existing well after a feature has been implemented.
     - Correctness of results is assessed by examining changes in the output of a well-understood and tested problem.
   - **Performance Testing**
     - Performance regressions are caught in the exact same environment where performance is important.

2) Ecosystem Integration
   - **Extreme Scale Scientific Software Stack (E4S) integration tests**
     - Regular builds of spack versions and release candidates of E4S ensure that facility software integrators will not experience issues during installation.
     - Recurring tests of E4S team installed software stacks detect underlying issues that change over time.
   - **Individual software ‘build’ recipes**
     - Individual products can be built periodically on facility systems and failures addressed by developers or integrators as appropriate.

3) Facility Operations Assurance
   - **Regular tests of installed facility software stack**
     - Environments drift over time. CI can catch issues related to hardware, vendor PE, or other Facility service updates.
   - **Regular tests of user managed container images**
     - Security assessments can be automated with CI pipelines to inspect container images, run static analysis tools, and pass security unit tests.

There is a Value Per Cycle tradeoff

Work can be performed anywhere

Workload requires HPC
Types of testing found within the ECP Software Ecosystem

<table>
<thead>
<tr>
<th>Test type</th>
<th>Complexity</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build</td>
<td>Low</td>
<td>Build tests check that all software components have been built successfully by using the appropriate compilers and tunables.</td>
</tr>
<tr>
<td>Validation</td>
<td>Low</td>
<td>Validation tests are run to demonstrate that a piece of software is installed correctly and to clearly demonstrate usage of the appropriate unique hardware features of the system, such as accelerators or high-speed interconnects.</td>
</tr>
<tr>
<td>Integration</td>
<td>Medium</td>
<td>Integration tests are implemented to ensure that versions of libraries along with compile time options do not conflict adversely with other related libraries as part of a larger software ecosystem.</td>
</tr>
<tr>
<td>Regression</td>
<td>High</td>
<td>Regression tests are maintained and run as frequently as developers create new software changes to determine whether the changes impact any already working features of the software.</td>
</tr>
<tr>
<td>Performance</td>
<td>High</td>
<td>Performance tests are implemented to detect regressions in runtime or degradation in time to solution or network bandwidth.</td>
</tr>
</tbody>
</table>

### Teams and Testing Assurance Layers

- ST teams drive implementation of **all types of tests** on developer, cloud, or product specific build and test CI pipelines.
- ST teams may run selected, high-value tests of **all types of tests** on facility CI platforms.
- Spack CI pipelines test **builds** of packages in the cloud and at some facilities.
- E4S team performs **build, validation, and integration** tests on facility systems and elsewhere.
- SI teams run **build, validation, and integration** tests on facility systems.
- Facility users run **validation** tests such as ‘spack test’ and the E4S test suite.
E4S / Facility support model is prepared to last beyond ECP

The E4S Level 2 support plan has funding and facility buy-in

- The E4S issue tracker on GitHub is the single point of collaboration. ([https://github.com/E4S-Project/e4s/issues](https://github.com/E4S-Project/e4s/issues))
- A new-issue template identifies facility submitted issues
- E4S L2 support will address facility-submitted E4S issues by E4S L2 support staff within 3 business days
- Facilities can raise priority of major issues
- E4S issues will be updated by E4S product teams as progress is being made
- E4S L2 support will provide regular reporting of E4S support metrics to facility staff
- We expect the support model to evolve as needs change

Example issue of this process in action: [https://github.com/E4S-Project/e4s/issues/9](https://github.com/E4S-Project/e4s/issues/9)
E4S / Facility Software Support Model

The Kokkos parallel reduce operation in the facility-provided E4S installation is not using the A100 GPU, while the Kokkos parallel for operation is.

Facility Staff works Facility Ticket #20081 until resolution.

Facility Staff creates ticket in E4S Issue Tracker: https://github.com/E4S-Project/E4S-Issues (E4S Issue #72)

Facility Support Staff

E4S Level 2 Support Acknowledges and Evaluates E4S Issue #72

E4S Level 2 Support Fixes Issue

E4S Issue #72 is used as central information sharing hub for all parties

All relevant parties work to diagnose and fix the issue

Appropriate parties keep Facility Ticket #20081 and E4S Issue #72 updated regularly

E4S Product Team solves issue and provides a fix

Relevant fixes or workarounds are merged into SW Products and Spack

E4S and Facility Staff keep E4S Issue #72 and Facility Ticket #20081 updated regularly

E4S Level 2 Support updates E4S Issue #72

Are issues resolved?

Facility Staff Updates and closes Facility Ticket #20081

Stop
Thank you

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy’s Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation’s exascale computing imperative.

Thank you to all collaborators in the ECP and broader computational science communities. The work discussed in this presentation represents creative contributions of many people who are passionately working toward next-generation computational science.
Questions?
Scientific Software Ecosystem in detail

- **AD and ST teams implement ECP CI regression testing as appropriate into existing CI frameworks and merge new build/test recipes into spack/develop**

- **E4S Team Prepares Quarterly Releases for Facility Installation, freezing on a point-in-time commit of spack/develop**

- **AD teams access stable ST software through Facility-maintained modules. Build cache and config available to AD/ST teams for integrating hotfixes**

Continuous Integration (2.4.4.04)

- E4S Team Prepares Quarterly Releases for Facility Installation, freezing on a point-in-time commit of spack/develop

Software Integration (2.4.4.01)

- **SI Team Deploys tested, versioned E4S on Facility Resources**

- E4S Team implements CI build and smoke testing for spack/develop

- **E4S Installed as Module Files**

- **E4S Spack Environment / Build Cache**

- **Newer Spack recipes**

- **E4S Installed as Module Files**

- **E4S Spack Environment / Build Cache**

- **Newer Spack recipes**
## Software feature development lifecycle model for an ST product

<table>
<thead>
<tr>
<th>Phase</th>
<th>Incubate</th>
<th>Harden</th>
<th>Deliver</th>
<th>Deploy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developers</td>
<td>Explore new algorithms &amp; implementations</td>
<td>Refactor and merge exploratory code, complete tests, documentation</td>
<td>Promote to release branch, integrate into SDK and E4S</td>
<td>Finalize smoke tests, DocPortal, community policy, engage SD, facilities</td>
</tr>
<tr>
<td>Integrators</td>
<td>Collaborative work in a sandbox environment</td>
<td>Product team integrates feature</td>
<td>Product team promotes to release, collaborates with SDK/E4S team</td>
<td>Product team works with E4S team, SD and facilities staff</td>
</tr>
<tr>
<td>Users</td>
<td>Early collaborators, co-develop with app partners</td>
<td>Friendly app teams prepared to work with the ST team on debugging</td>
<td>App teams looking for latest stable features and staffed to incorporate new features</td>
<td>Apps looking for stable functionality in turnkey environment</td>
</tr>
<tr>
<td>Availability</td>
<td>Forked repo or local branch</td>
<td>Develop (pre-release) repo branch</td>
<td>Directly from main repo branch</td>
<td>Product release site, E4S, vendor, facility, community repo, or combo</td>
</tr>
</tbody>
</table>

Some Takeaways:

- AD users of ST products tend to engage in the incubate and harden phases of ST feature development
- First priority for ST teams must be engaging AD users
- First EAS/Exascale ports happen in incubation/harden phases
- For many ST features, sustainable deployment is not E4S (even though E4S inclusion is important)
# Software feature development lifecycle model: Examples

<table>
<thead>
<tr>
<th>Phase</th>
<th>Incubate</th>
<th>Harden</th>
<th>Deliver</th>
<th>Deploy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nalu-Wind &amp; hypre</td>
<td>Nalu-Wind team forks hypre repo, prototypes GPU-friendly 2-stage Gauss-Seidel smoother consulting with hypre team, also fixes GPU-only bugs in other parts of hypre</td>
<td>hypre team ingests prototype GS smoother, provides coverage tests</td>
<td>Available to all hypre users</td>
<td>Coming soon</td>
</tr>
<tr>
<td>SUNDIALS &amp; Ginkgo</td>
<td>SUNDIALS team collaborates with Ginkgo team on interface for batched sparse solvers needed for Pele-C, others</td>
<td>Gingko refactors existing APIs, tests</td>
<td>Available to all Ginkgo users</td>
<td>Coming soon</td>
</tr>
<tr>
<td>ALPINE &amp; VTK-m</td>
<td>ALPINE team prototypes a distributed contour tree algorithm within their fork of VTK-m</td>
<td>VTK-m team merges into their develop branch for hardening</td>
<td>VTK-m promotes to release branch</td>
<td>In E4S V21.11</td>
</tr>
</tbody>
</table>

**Key points:**

- Many of the most impactful ST capabilities are developed using this collaborative development pattern
- Incubation and hardening occur on the early-access systems since these systems are driving the need
- Ingesting for sustainable production use ends with promotion to the official release branch for delivery
- E4S integration occurs quarterly, several months after new features are available from product release branch