Software Deployment at Facilities

EXASCALE COMPUTING PROJECT

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

Ryan Adamson

National Nuclear Security Administration

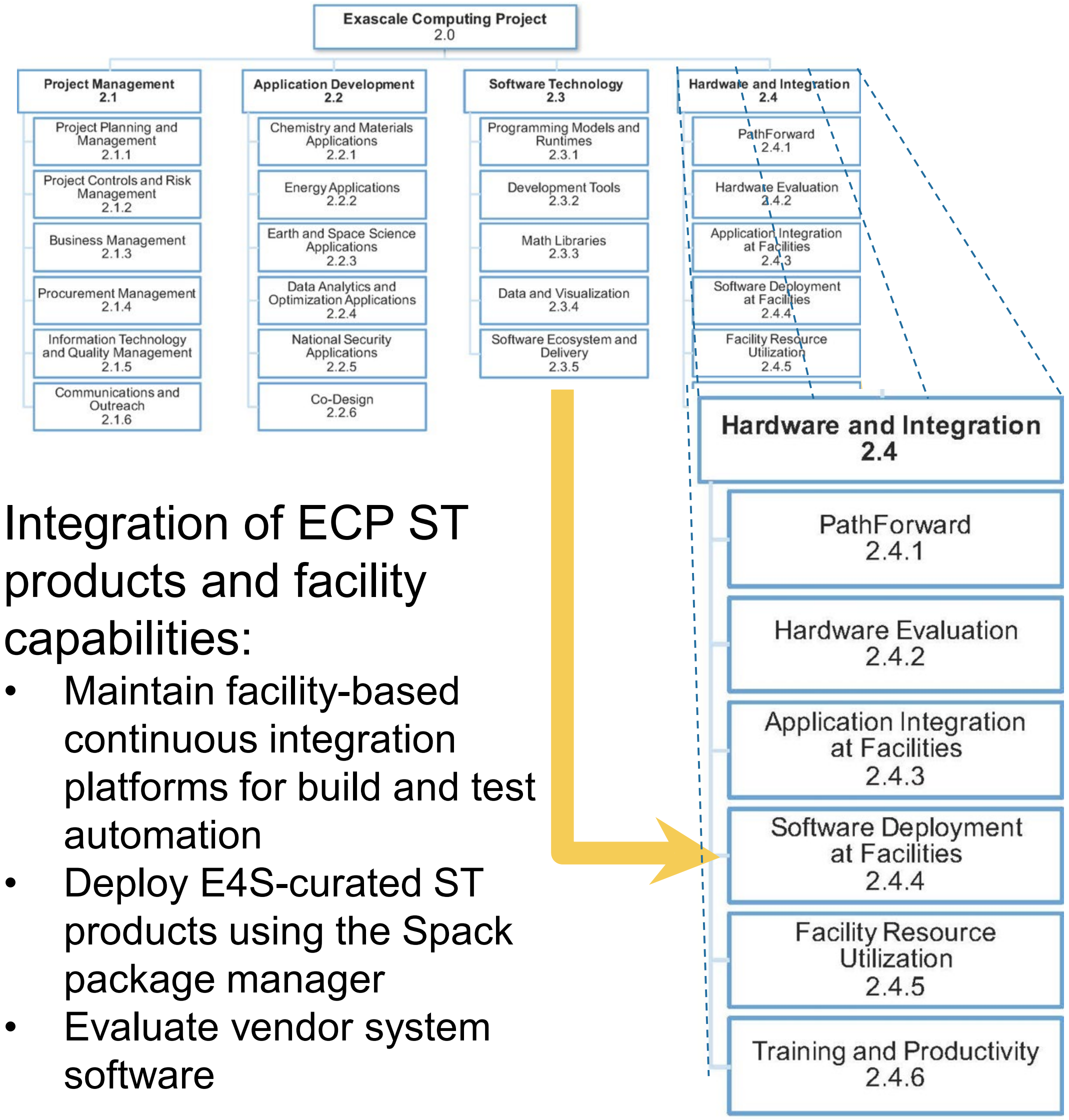
U.S. DEPARTMENT OF ENERGY | Office of Science

Overview:

ECP Software Deployment at Facilities (SD) Activity

2.4.4 Software Deployment at Facilities - Mission

Each software team and facility has individual (and often unique!) preferences, constraints, and strategies regarding scientific software development and deployment



Exascale Computing Project 2.0

Project Management 2.1
- Project Planning and Management 2.1.1
- Project Controls and Risk Management 2.1.2
- Business Management 2.1.3
- Procurement Management 2.1.4
- Information Technology and Quality Management 2.1.5
- Communications and Outreach 2.1.6

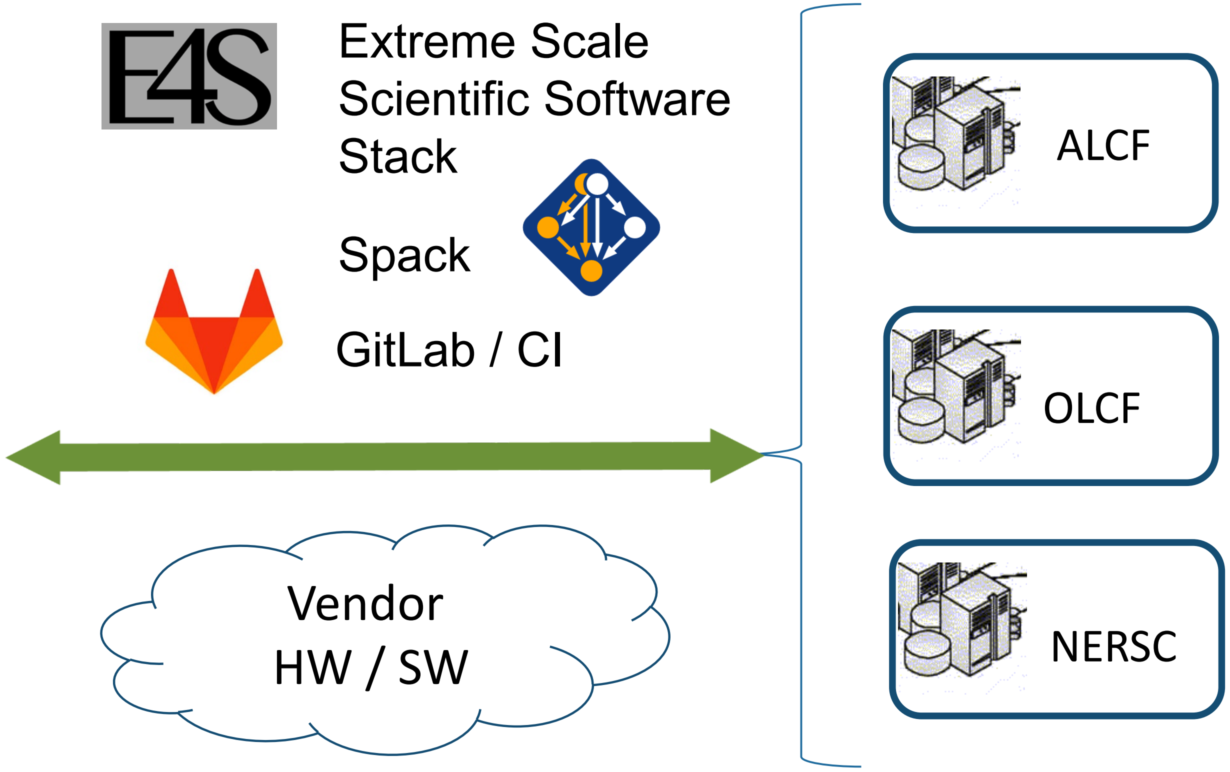
Application Development 2.2
- Chemistry and Materials Applications 2.2.1
- Energy Applications 2.2.2
- Earth and Space Science Applications 2.2.3
- Data Analytics and Optimization Applications 2.2.4
- National Security Applications 2.2.5
- Co-Design 2.2.6

Software Technology 2.3
- Programming Models and Runtimes 2.3.1
- Development Tools 2.3.2
- Math Libraries 2.3.3
- Data and Visualization 2.3.4
- Software Ecosystem and Delivery 2.3.5

Hardware and Integration 2.4
- PathForward 2.4.1
- Hardware Evaluation 2.4.2
- Application Integration at Facilities 2.4.3
- Software Deployment at Facilities 2.4.4
- Facility Resource Utilization 2.4.5
- Training and Productivity 2.4.6

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



E4S Extreme Scale Scientific Software Stack

Spack

GitLab / CI

Vendor HW / SW

ALCF

OLCF

NERSC

WBS 2.4.4: Software Deployment at Facilities

Project Short Name	PI Name, Inst	Short Description/Objective	Program Office(s)
2.4.4.01 Software Integration	Shahzeb Siddiqui (LBL)	Build/Test/Deploy ST products at facilities	ASCR
2.4.4.04 Continuous Integration	Paul Bryant (ORNL)	Develop and Deploy ECP CI infrastructure	ASCR
2.4.4.03 Shasta Testing	Jay Srinivasan (LBL)	Evaluate and expedite Shasta releases	ASCR
2.4.4.05 HPCM / Slingshot Testing	Scott Atchley (ORNL)	Explore HPCM and Slingshot as alternatives to Shasta	ASCR

Shahzeb Siddiqui

Paul Bryant

Jay Srinivasan

Scott Atchley

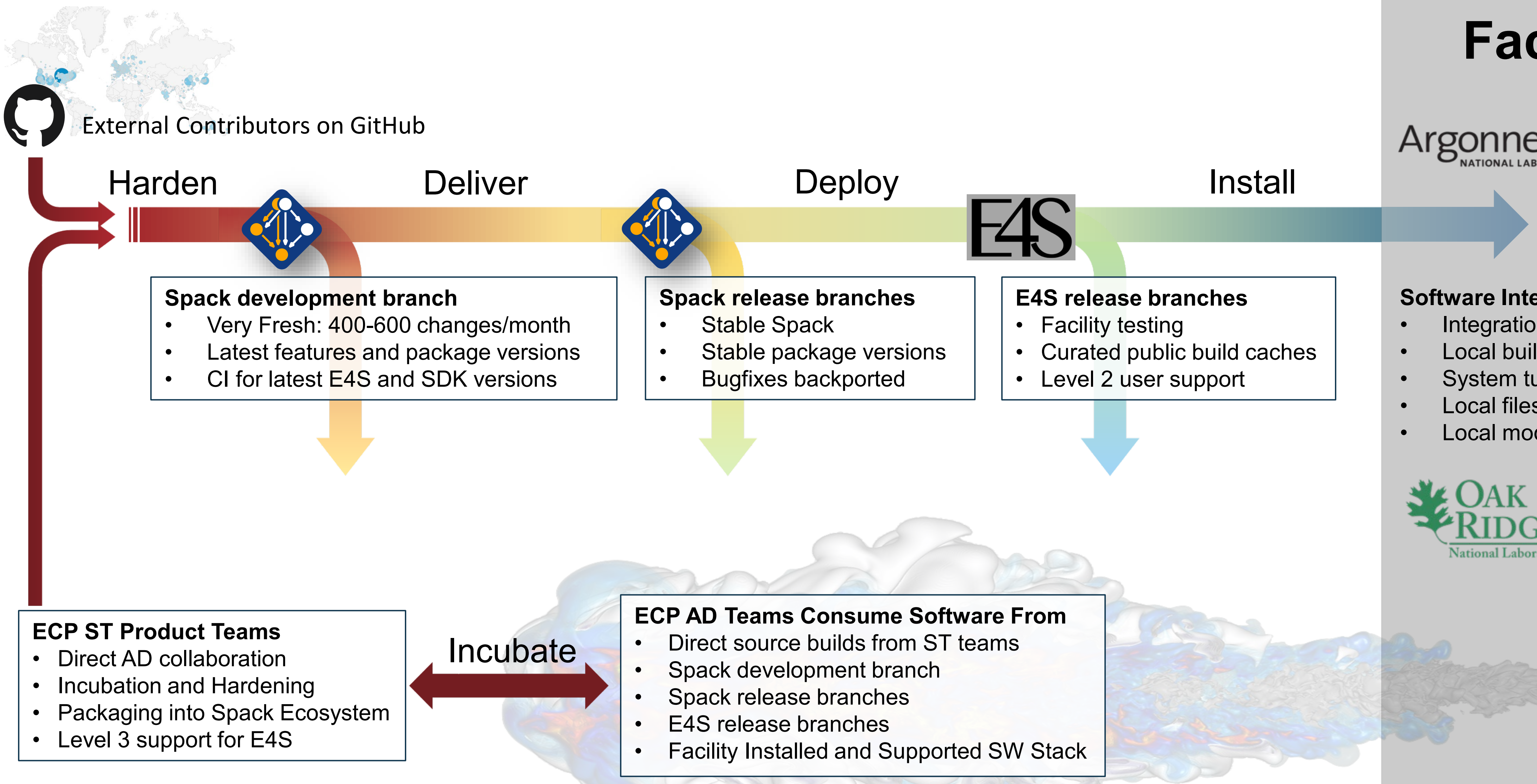
The Software Deployment team’s role is to package and integrate Software Technology products on exascale systems

PMR Core (17)	Compilers and Support (7)	Tools and Technology (11)	xSDK (16)	Visualization Analysis and Reduction (9)	Data mgmt, I/O Services, Checkpoint restart (12)	Ecosystem/E4S at-large (12)
QUO	openarc	TAU	hypre	ParaView	SCR	mpiFileUtils
Papyrus	Kitsune	HPCToolkit	FleSCI	Catalyst	FAODEL	TriBITS
SICM	LLVM	Dyninst Binary Tools	MFEM	VTK-m	ROMIO	MarFS
Legion	CHiLL autotuning comp	Gotcha	Kokkoskernels	SZ	Mercury (Mochi suite)	GUFI
Kokkos (support)	LLVM openMP comp	Caliper	Trilinos	zfp	HDF5	Intel GEOPM
RAJA	OpenMP V & V	PAPI	SUNDIALS	VisIt	Parallel netCDF	BEE
CHAI	Flang/LLVM Fortran comp	Program Database Toolkit	PETSc/TAO	ASCENT	ADIOS	FSEFI
PaRSEC*		Search (random forests)	libEnsemble	Cinema	Darshan	Kitten Lightweight Kernel
DARMA		Siboka	STRUMPACK	ROVER	UnifyCR	COOLR
GASNet-EX		C2C	SuperLU		VeloC	NRM
Qthreads		Sonar	ForTrilinos		IOSS	ArgoContainers
BOLT			SLATE		HXHIM	Spack
UPC++			MAGMA			
MPICH			DTK			
Open MPI			Tasmanian			
Umpire			TuckerMPI			
AML						

Legend

- PMR
- Tools
- Math Libraries
- Data and Vis
- Ecosystems and delivery

ECP Software Stream: Incubation to Installation



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 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

Facilities

Argonne National Laboratory

Berkeley Lab

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

Oak Ridge National Laboratory

Los Alamos National Laboratory EST. 1943

Sandia National Laboratories

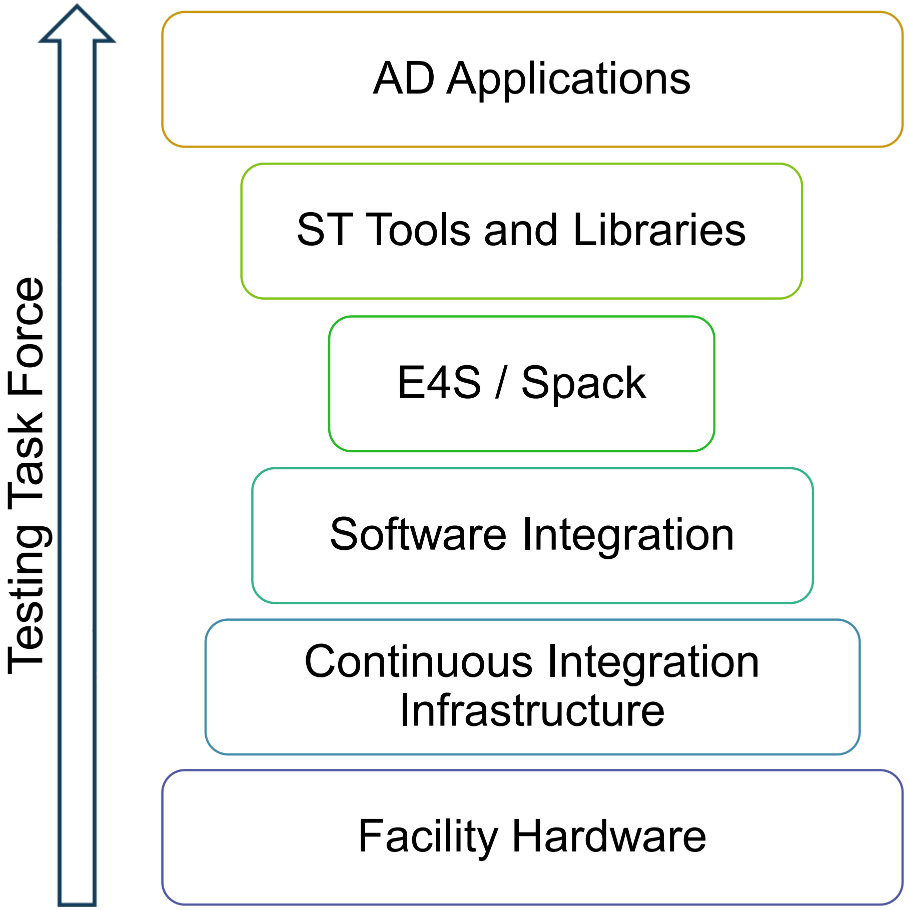
Lawrence Livermore National Laboratory

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

https://spack.io/ → https://e4s-project.github.io/ → https://gitlab.com



E4S and Spack are the 'spanning layers' of the ECP Software Ecosystem Hourglass

Scientific Software Ecosystem roles and responsibilities

ST Developers

- AD and ST teams implement ECP CI regression testing as appropriate into existing CI frameworks and merge new build/test recipes into spack/develop
  - https://e4s-project.github.io/policies.html
- 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
  - https://ideas-productivity.org/ideas-ecp/

E4S Software Curators

- Spack Team implements CI build and smoke testing for spack/develop
  - https://github.com/spack/spack
  - https://cdash.spack.io
- 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

SI / Facilities

- SI Team Deploys packages selected from a tested and versioned E4S release onto Facility Resources.
  - https://docs.nersc.gov/applications/e4s/cori/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/

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

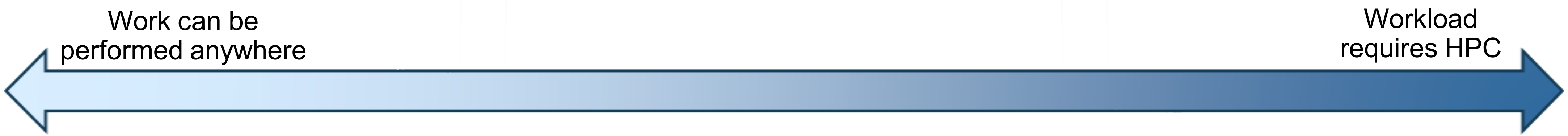
- Regression Testing
  - Software failures are detected when new code is introduced. This prevents latent bugs from existing well after a feature has been implemented.
  - Correctness of results is assessed by examining changes in the output of a well-understood and tested problem.
- Performance Testing
  - 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.



Work can be performed anywhere

Workload requires HPC

There is a Value Per Cycle tradeoff

Types of testing found within the ECP Software Ecosystem

Test type	Complexity	Notes
Build	Low	Build tests check that all software components have been built successfully by using the appropriate compilers and tunables.
Validation	Low	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.
Integration	Medium	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.
Regression	High	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.
Performance	High	Performance tests are implemented to detect regressions in runtime or degradation in time to solution or network bandwidth.

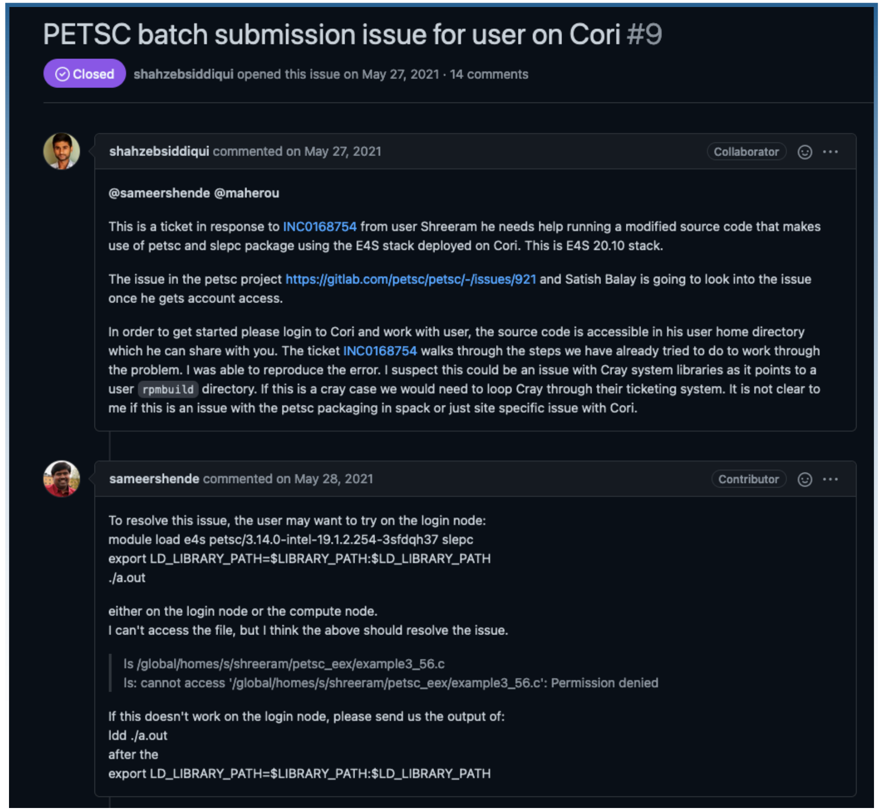
Teams and Testing Assurance Layers

- 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)
- 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



PETSC batch submission issue for user on Cori #9

Closed shahzebsiddiqui opened this issue on May 27, 2021 · 14 comments

shahzebsiddiqui commented on May 27, 2021 Collaborator

@sameershende @maherou

This is a ticket in response to INC0168754 from user Shreeram he needs help running a modified source code that makes use of petsc and slepc package using the E4S stack deployed on Cori. This is E4S 20.10 stack.

The issue in the petsc project https://gitlab.com/petsc/petsc/-/issues/921 and Satish Balay is going to look into the issue once he gets account access.

In order to get started please login to Cori and work with user, the source code is accessible in his user home directory which he can share with you. The ticket INC0168754 walks through the steps we have already tried to do to work through the problem. I was able to reproduce the error. I suspect this could be an issue with Cray system libraries as it points to a user `rpmbuild` directory. If this is a cray case we would need to loop Cray through their ticketing system. It is not clear to me if this is an issue with the petsc packaging in spack or just site specific issue with Cori.

sameershende commented on May 28, 2021 Contributor

To resolve this issue, the user may want to try on the login node:
module load e4s petsc/3.14.0-intel-19.1.2.254-3sfdqh37 slepc
export LD_LIBRARY_PATH=$LIBRARY_PATH:$LD_LIBRARY_PATH
./a.out

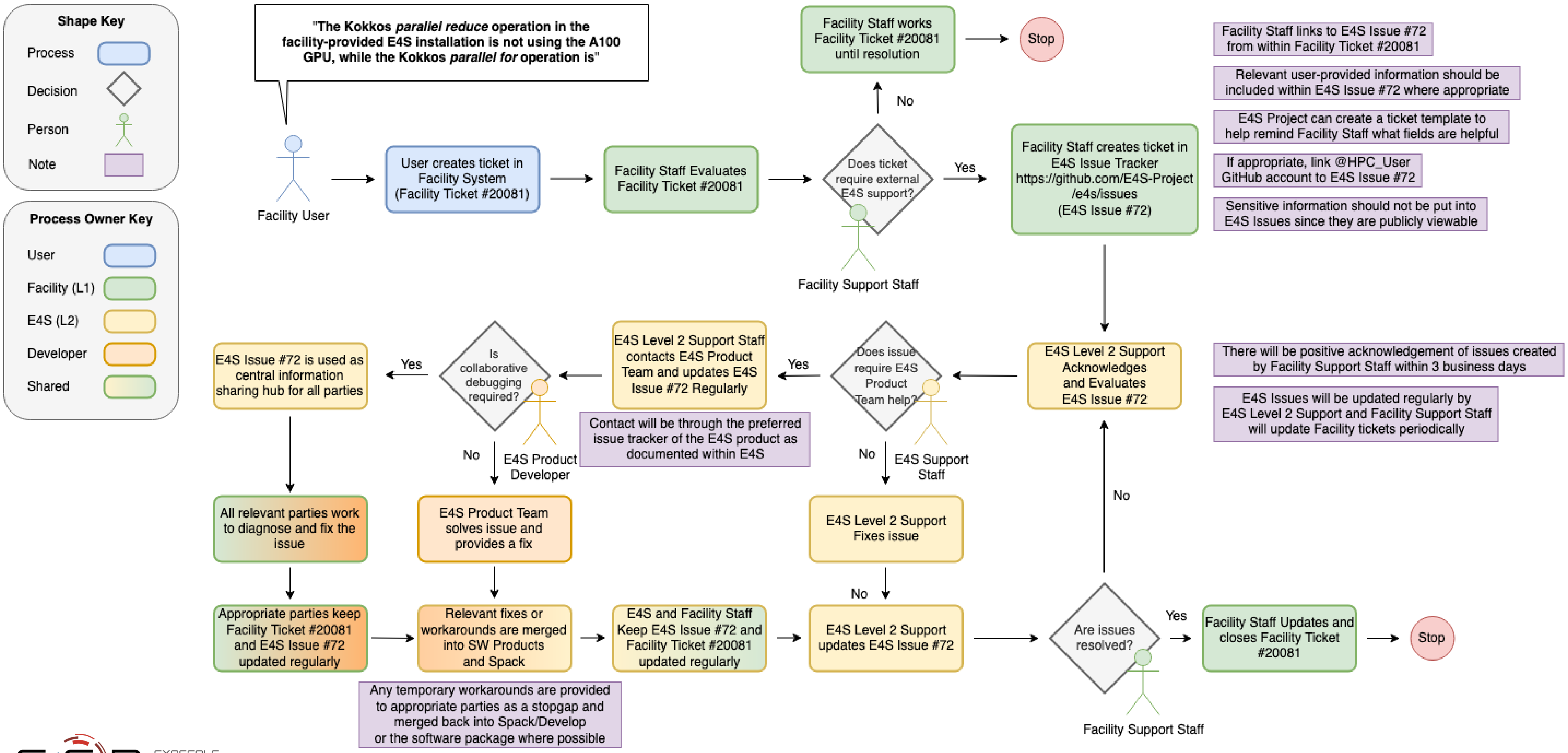
either on the login node or the compute node.
I can't access the file, but I think the above should resolve the issue.

> ls /global/homes/s/shreeram/petsc_eex/example3_56.c
> ls: cannot access '/global/homes/s/shreeram/petsc_eex/example3_56.c': Permission denied

If this doesn't work on the login node, please send us the output of:
ldd ./a.out
after the
export LD_LIBRARY_PATH=$LIBRARY_PATH:$LD_LIBRARY_PATH

Example issue of this process in action: https://github.com/E4S-Project/e4s/issues/9

E4S / Facility Software Support Model



Shape Key

Process

Decision

Person

Note

Process Owner Key

User

Facility (L1)

E4S (L2)

Developer

Shared

"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 User

User creates ticket in Facility System (Facility Ticket #20081)

Facility Staff Evaluates Facility Ticket #20081

Does ticket require external E4S support?

Facility Support Staff

No

Facility Staff works Facility Ticket #20081 until resolution

Stop

Yes

Facility Staff creates ticket in E4S Issue Tracker https://github.com/E4S-Project/e4s/issues (E4S Issue #72)

Facility Staff links to E4S Issue #72 from within Facility Ticket #20081

Relevant user-provided information should be included within E4S Issue #72 where appropriate

E4S Project can create a ticket template to help remind Facility Staff what fields are helpful

If appropriate, link @HPC_User GitHub account to E4S Issue #72

Sensitive information should not be put into E4S Issues since they are publicly viewable

E4S Level 2 Support Acknowledges and Evaluates E4S Issue #72

There will be positive acknowledgement of issues created by Facility Support Staff within 3 business days

E4S Issues will be updated regularly by E4S Level 2 Support and Facility Support Staff will update Facility tickets periodically

Does issue require E4S Product Team help?

E4S Support Staff

Yes

E4S Level 2 Support Staff contacts E4S Product Team and updates E4S Issue #72 Regularly

Contact will be through the preferred issue tracker of the E4S product as documented within E4S

Is collaborative debugging required?

E4S Product Developer

Yes

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

No

E4S Product Team solves issue and provides a fix

Relevant fixes or workarounds are merged into SW Products and Spack

Any temporary workarounds are provided to appropriate parties as a stopgap and merged back into Spack/Develop or the software package where possible

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

No

E4S Level 2 Support Fixes issue

No

E4S Level 2 Support updates E4S Issue #72

Are issues resolved?

Facility Support Staff

No

Yes

Facility Staff Updates and closes Facility Ticket #20081

Stop

EXASCALE COMPUTING PROJECT

Thank you

https://www.exascaleproject.org

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.

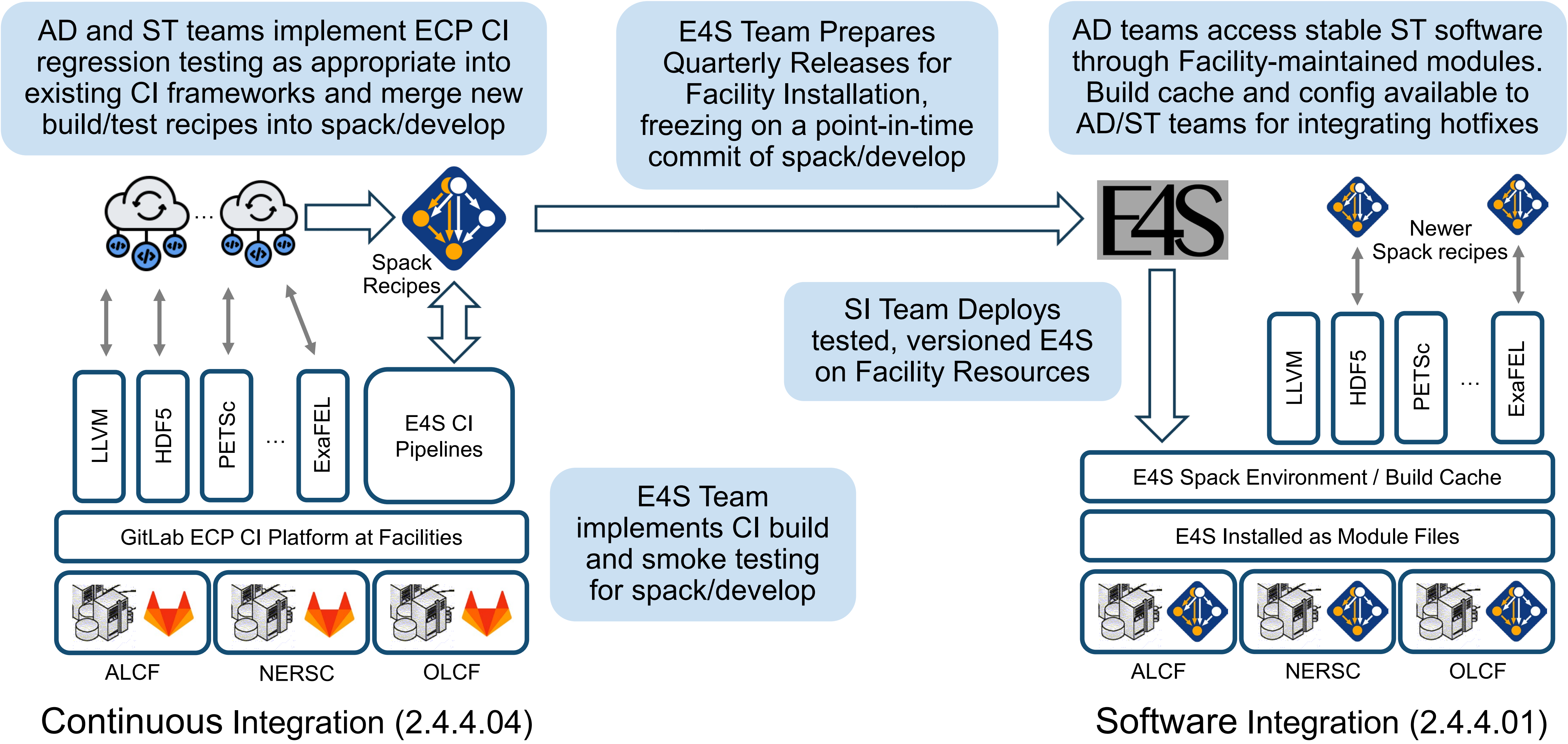
ECP

EXASCALE COMPUTING PROJECT

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

…

Spack Recipes

E4S

Newer Spack recipes

SI Team Deploys tested, versioned E4S on Facility Resources

LLVM

HDF5

PETSc

…

ExaFEL

E4S CI Pipelines

GitLab ECP CI Platform at Facilities

ALCF

NERSC

OLCF

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

LLVM

HDF5

PETSc

…

ExaFEL

E4S Spack Environment / Build Cache

E4S Installed as Module Files

ALCF

NERSC

OLCF

Continuous Integration (2.4.4.04)

Software Integration (2.4.4.01)

Software feature development lifecycle model for an ST product

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

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

	Phase			
	Incubate	Harden	Deliver	Deploy
Nalu-Wind & hypre	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	hypre team ingests prototype GS smoother, provides coverage tests	Available to all hypre users	Coming soon
SUNDIALS & Ginkgo	SUNDIALS team collaborates with Ginkgo team on interface for batched sparse solvers needed for Pele-C, others	Gingko refactors existing APIs, tests	Available to all Gingko users	Coming soon
ALPINE & VTK-m	ALPINE team prototypes a distributed contour tree algorithm within their fork of VTK-m	VTK-m team merges into their develop branch for hardening	VTK-m promotes to release branch	In E4S V21.11

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