Unifying Software Distribution in ECP

Approved for public release

Todd Gamblin
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory

1st Workshop on NSF and DOE High Performance Computing Tools
Eugene, Oregon
July 11, 2019

ECP
EXASCALE COMPUTING PROJECT

NNSA National Nuclear Security Administration

Lawrence Livermore National Laboratory

CASC

U.S. DEPARTMENT OF ENERGY | Office of Science

What is the Exascale Computing Project (ECP)?

ECP is an accelerated research and development project funded by the US Department of Energy (DOE) to ensure all necessary pieces are in place to deliver the nation’s first, capable, exascale ecosystem, including mission critical applications, **an integrated software stack**, and advanced computer system engineering and hardware components.

ECP by the Numbers

7
YEARS

$1.7B

A seven-year, $1.7 billion R&D effort that launched in 2016

6
CORE DOE
LABS

6 core DOE National Laboratories: Argonne, Oak Ridge, Berkeley, Lawrence Livermore, Los Alamos, Sandia

- Staff from most of the 17 DOE national laboratories take part in the project

3
TECHNICAL
FOCUS
AREAS

3 technical focus areas:

Application Development, Software Technology, Hardware and Integration

100
R&D TEAMS

1000
RESEARCHERS

More than 100 top-notch R&D teams

Argonne NATIONAL LABORATORY

OAK RIDGE National Laboratory

BERKELEY LAB

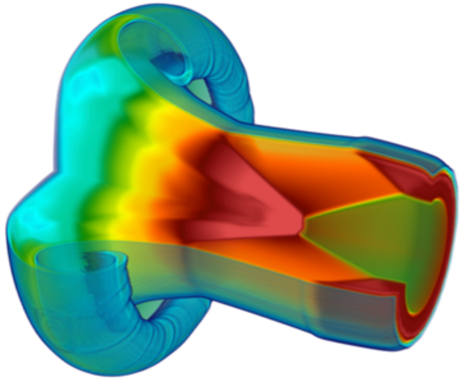
Los Alamos NATIONAL LABORATORY EST. 1943

Lawrence Livermore National Laboratory

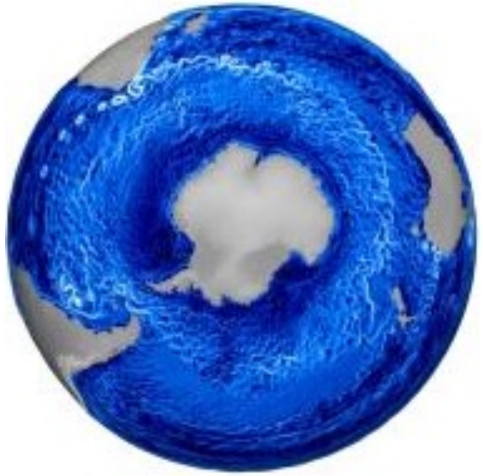
Sandia National Laboratories

ECP EXASCALE COMPUTING PROJECT

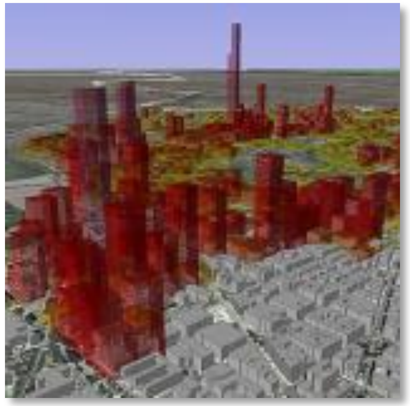
Exascale machines will support a wide range of science applications



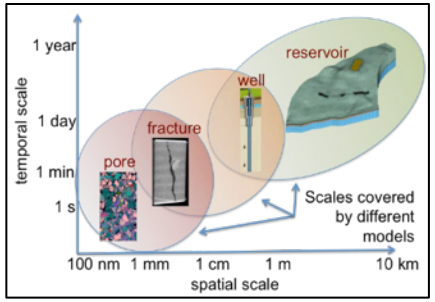
Compressible flow (MARBL)



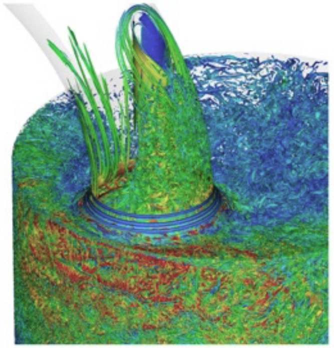
Climate (E3SM)



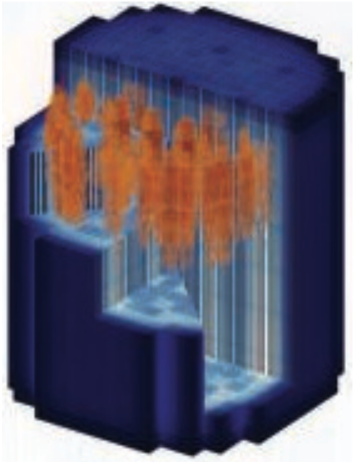
Urban systems (Urban)



Subsurface (GEOS)



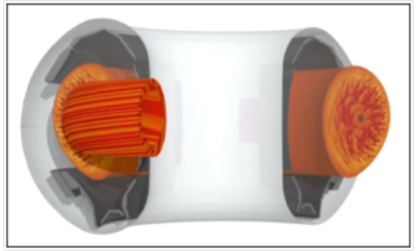
Combustion (Nek5000)



Modular Nuclear Reactors (ExaSMR)

Wind Energy (ExaWind)

Additive Manufacturing (ExaAM)



Magnetic Fusion (WDMApp)

Open source projects lay the foundation for DOE simulations

Parallel Programming Models

OpenMPI

Kokkos

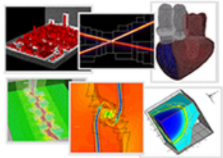
BERKELEY UPC UNIFIED PARALLEL C

MPICH

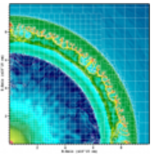
RAJA

GASNet-EX

Meshing / Finite Elements



SAMRAI



CHOMBO

MFEM

AMReX

Scientific Visualization

ParaView

VisIt

Ascent

VTK-m

Filesystems & I/O

ZFS ON LINUX

Lustre

ADIOS

Packaging/Build

Spack

bLT

SHIFTER

Charliecloud

Resource Managers

slurm workload manager

flux

Parallel Solvers

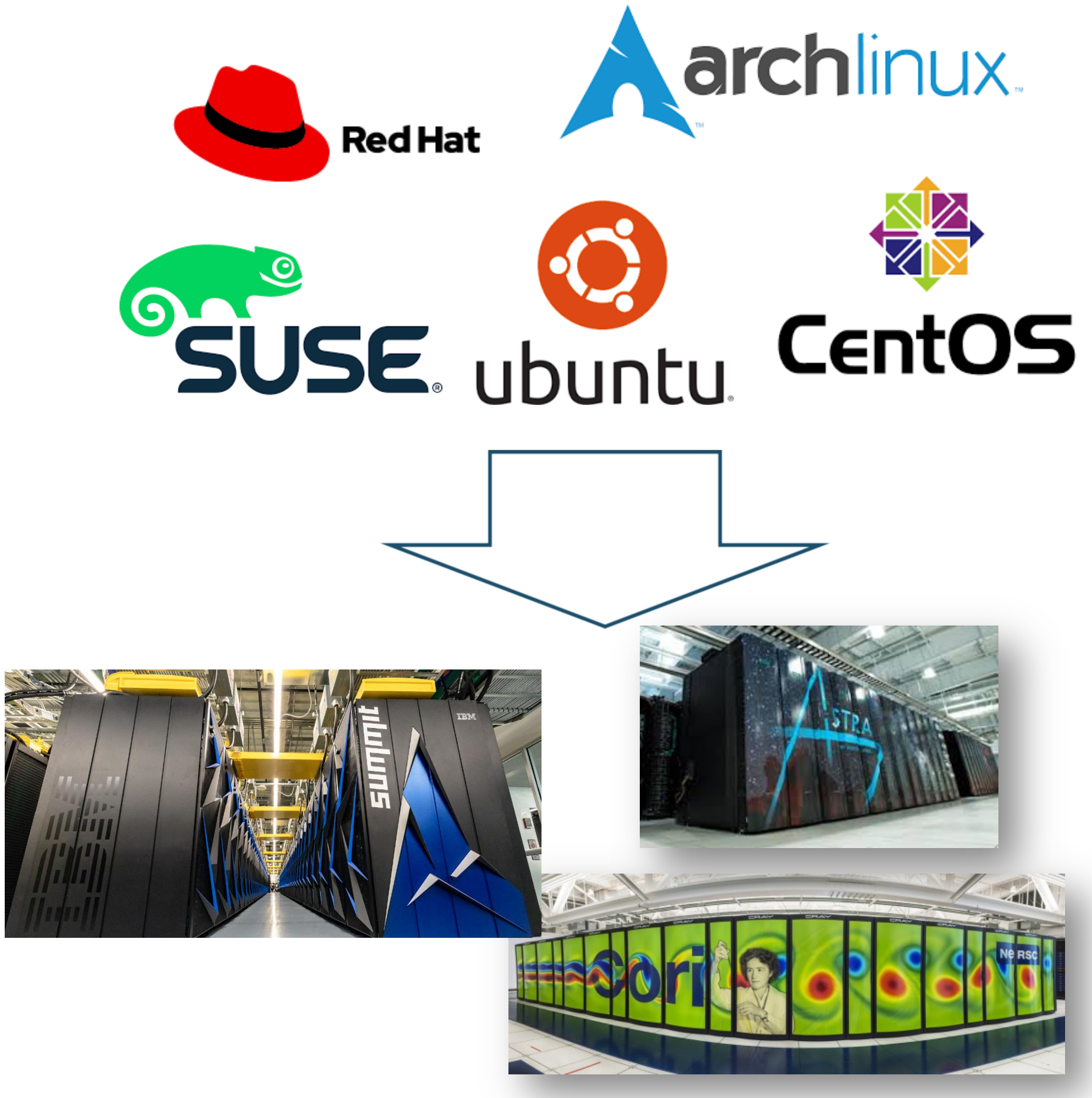
hypre

TRILINOS

PETSc

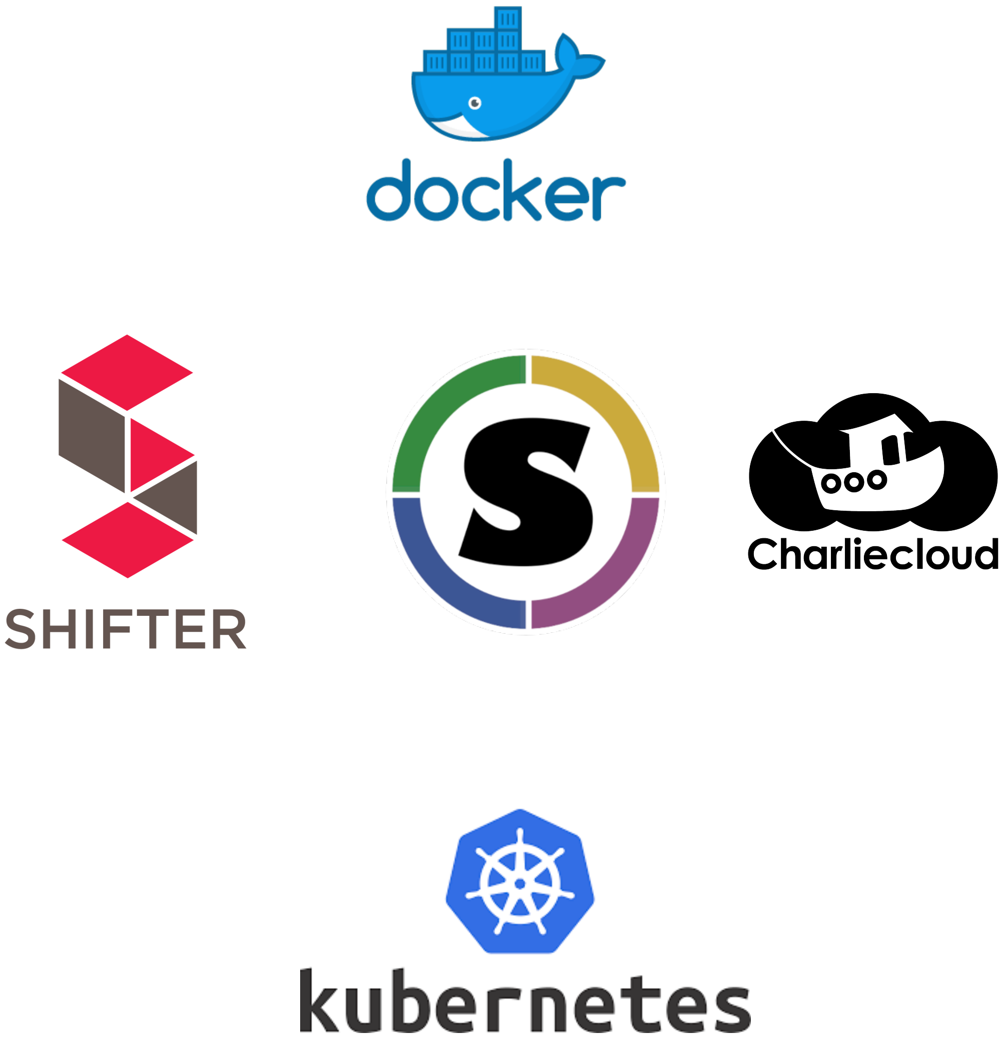
ECP is building a **software distribution** for Exascale

- “Nation’s first capable exascale ecosystem”
- Distribution effort required is similar to efforts like Red Hat, Debian, Ubuntu, etc.
  - Curation and vetting of software
  - Packaging, building
  - Wide distribution
- Not as mainstream, not quite as widespread
- Platform-wise, ECP is more complex and broader
  - Many (often unique) platforms
  - Many software ecosystems
  - From-source distribution
  - Must support Optimization, GPUs, fast networks



We’re also currently at the intersection of cloud and HPC

- Cloud has largely moved to containerized deployment
- Containers (moreso than VMs) look viable for HPC, but there are challenges
  - Optimized containers
  - Deeper integration with host architecture/network
  - Building containers in secure environments
  - Support at HPC centers for DevOps automation
- Still need to support traditional, modules-based HPC workflows
  - Bare-metal deployments will continue to be mainstream at HPC centers for some time



ECP is working to unify software distribution

- Three main requirements:
  1. **Research & Development**
     - New software to drive automation
     - New capabilities to enable distribution of optimized artifacts
     - Workflow automation for facilities and users
  2. **Infrastructure**
     - Automated build farms to produce artifacts
     - Hosting + bandwidth for distribution
     - HPC cycles to do the building
  3. **Process**
     - Humans have to be involved in the software curation process
     - SDKs, E4S, and facility collaboration are ECP's software curation vehicles

SHIFTER

Charliecloud

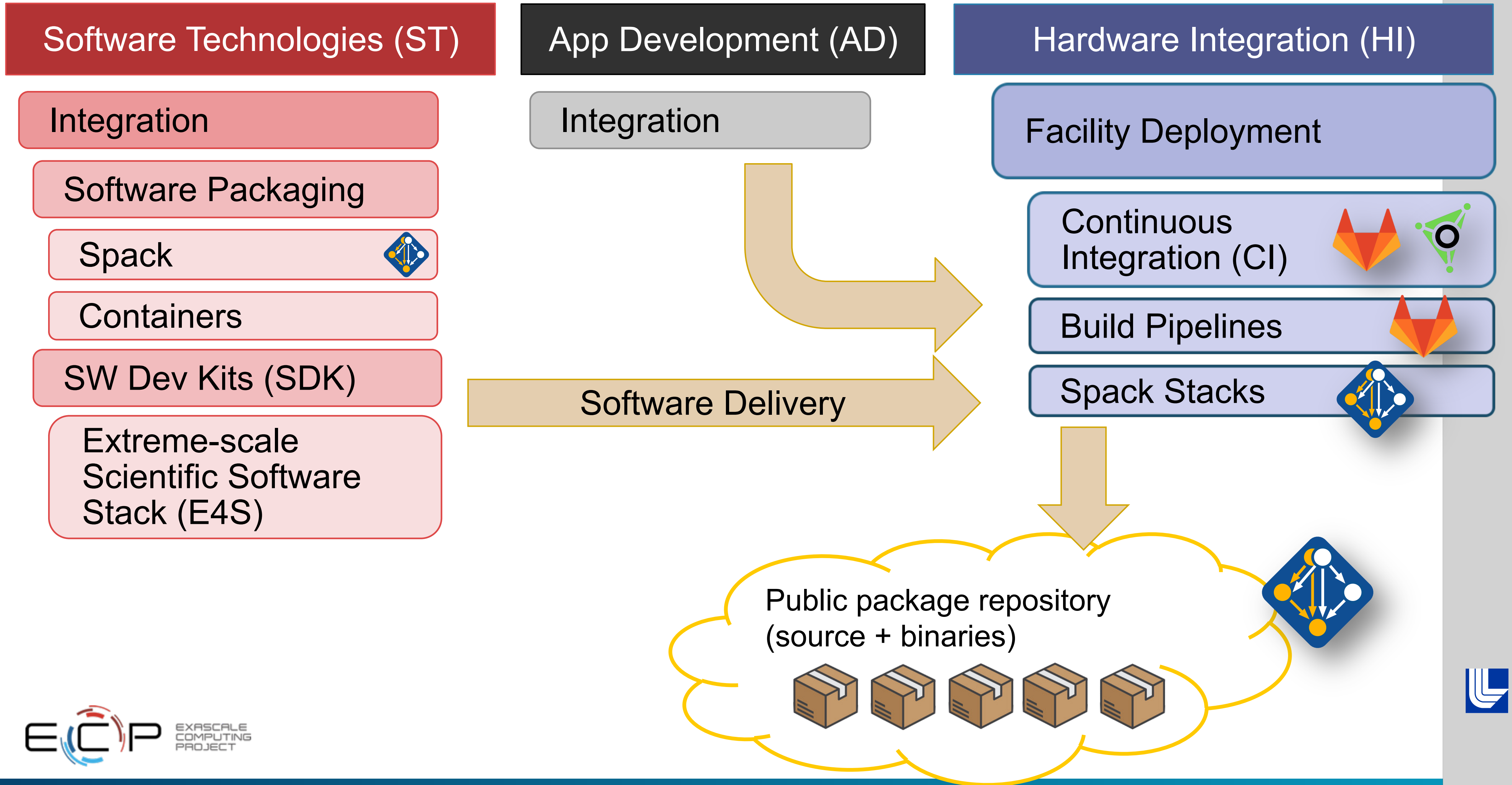
Containers

Packaging

Bare metal
HPC

Infrastructure

There are many activities around software distribution and deployment within ECP



Software Technologies (ST)

Integration

Software Packaging

Spack

Containers

SW Dev Kits (SDK)

Extreme-scale Scientific Software Stack (E4S)

App Development (AD)

Integration

Software Delivery

Hardware Integration (HI)

Facility Deployment

Continuous Integration (CI)

Build Pipelines

Spack Stacks

Public package repository (source + binaries)

Facilities

Argonne National Laboratory

Oak Ridge National Laboratory

Berkeley Lab

Los Alamos National Laboratory Est. 1943

Sandia National Laboratories

Lawrence Livermore National Laboratory

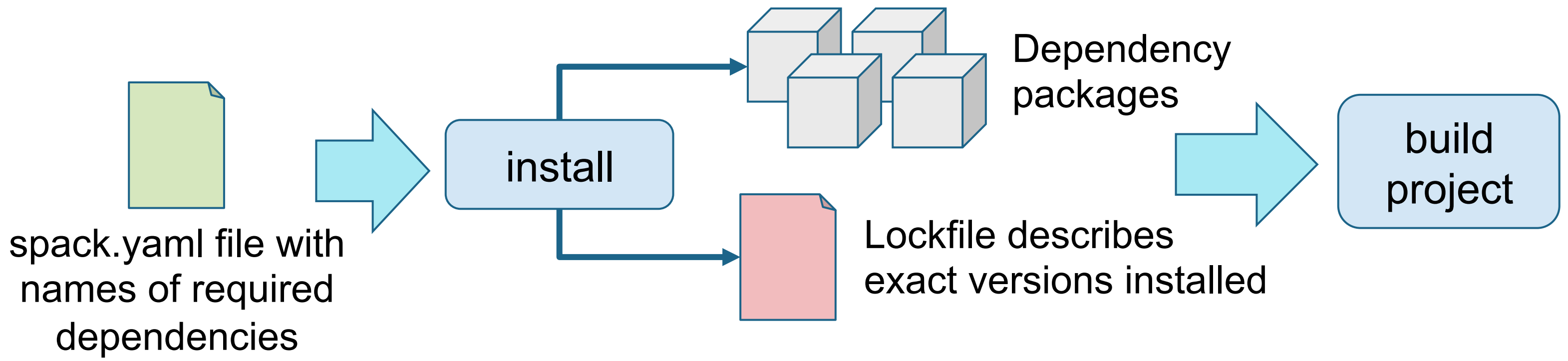
Following three talks give deep dives of each of these areas

- Andrew Younge - Containers
- Dave Montoya – Facility Deployment
- Sameer Shende – E4S

Specific architecture target information – in progress

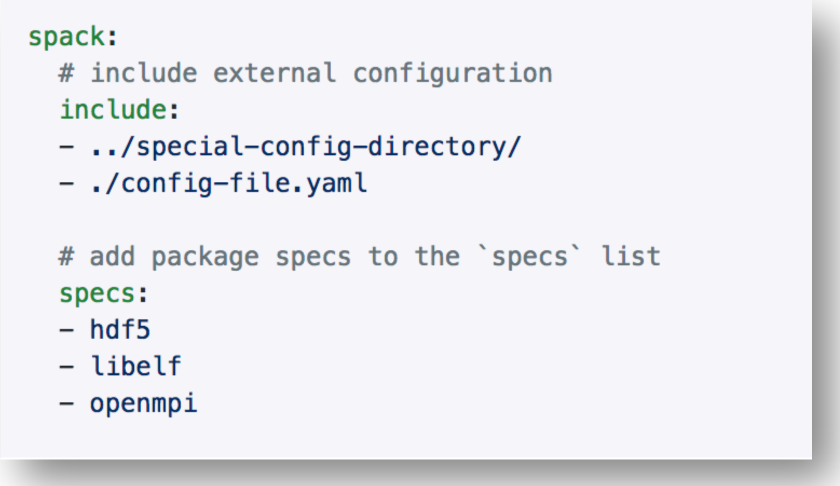
- We want to provide optimized builds for Spack packages and containers
  - Code level choices (O2, O3)
  - Architecture specific choices (-mcpu=cortex-a7, -march=haswell)
- Architectures vary as to how much they expose features to users
  - x86 exposes feature sets in /proc/cpuinfo
  - Arm hides many features behind revision number
- Methods for accessing architecture optimizations
  - Vary by both compiler and architecture
    - Gcc –mcpu vs. –march, for example
    - Relies on architectures providing a programmatic way to get information
- We want to expose the names users understand
  - Thunderx2, cortex-a7 for arm
  - Power8, power9 for IBM
  - Haswell, skylake for Intel

Spack has added environments and spack.yaml / spack.lock


spack.yaml file with names of required dependencies
install
Dependency packages
Lockfile describes exact versions installed
build project

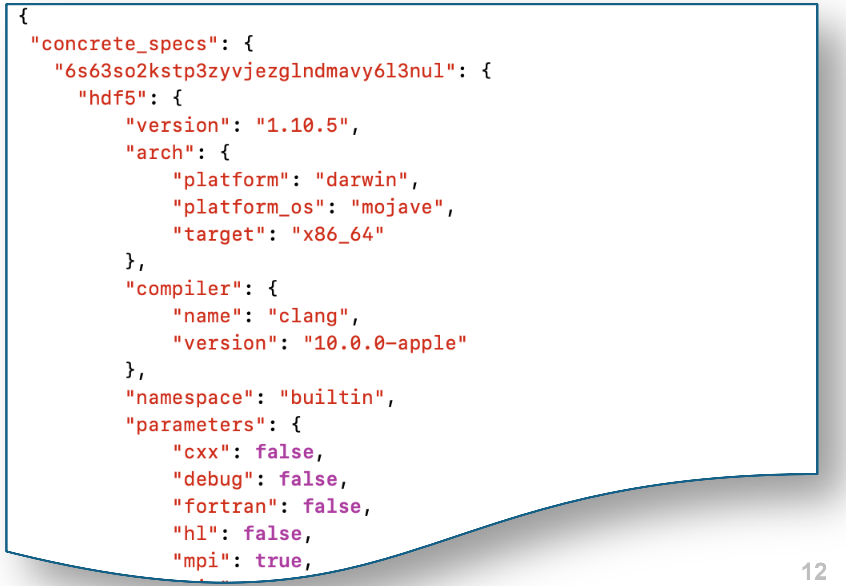
- Allows developers to bundle Spack configuration with their repository
- Can also be used to maintain configuration together with Spack packages.
  - E.g., versioning your own local software stack with consistent compilers/MPI implementations
- Manifest / Lockfile model pioneered by Bundler is becoming standard
  - spack.yaml describes project requirements
  - spack.lock describes exactly what versions/configurations were installed, allows them to be reproduced.

Simple spack.yaml file



```yaml
spack:
  # include external configuration
  include:
  - ../special-config-directory/
  - ./config-file.yaml

  # add package specs to the `specs` list
  specs:
  - hdf5
  - libelf
  - openmpi
```

Concrete spack.lock file (generated)



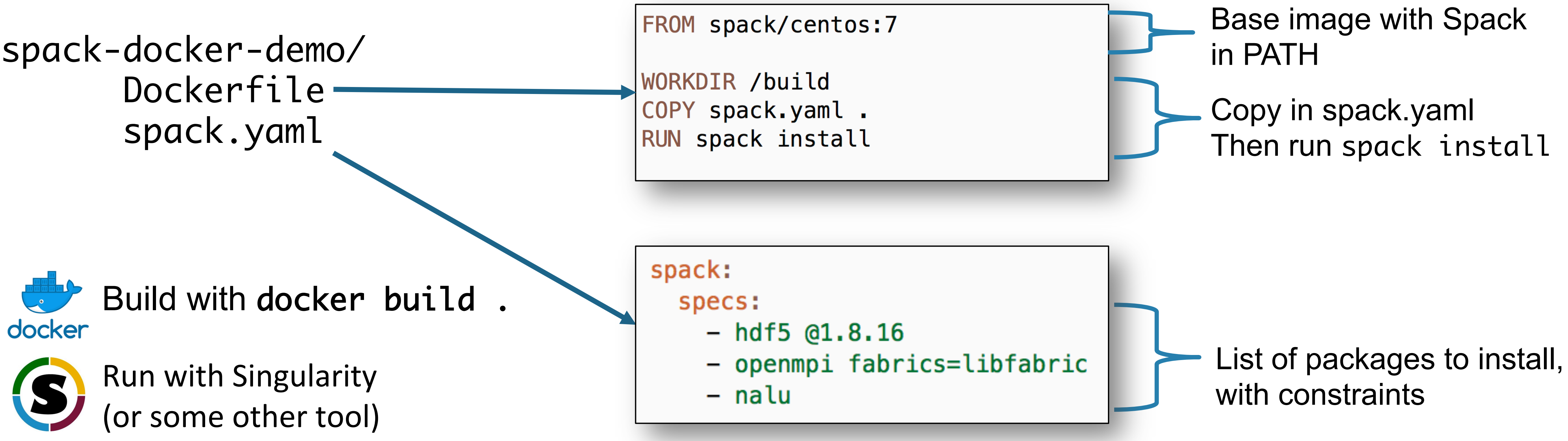
```json
{
 "concrete_specs": {
   "6s63so2kstp3zyvjezglndmavy6l3nul": {
     "hdf5": {
         "version": "1.10.5",
         "arch": {
             "platform": "darwin",
             "platform_os": "mojave",
             "target": "x86_64"
         },
         "compiler": {
             "name": "clang",
             "version": "10.0.0-apple"
         },
         "namespace": "builtin",
         "parameters": {
             "cxx": false,
             "debug": false,
             "fortran": false,
             "hl": false,
             "mpi": true,
```

EXASCALE COMPUTING PROJECT

Spack environments help with building containers

- We recently started providing base images on DockerHub with Spack preinstalled.

- **Very** easy to build a container with some Spack packages in it:



```
spack-docker-demo/
        Dockerfile
        spack.yaml
```

```
FROM spack/centos:7

WORKDIR /build
COPY spack.yaml .
RUN spack install
```

Base image with Spack in PATH

Copy in spack.yaml
Then run `spack install`

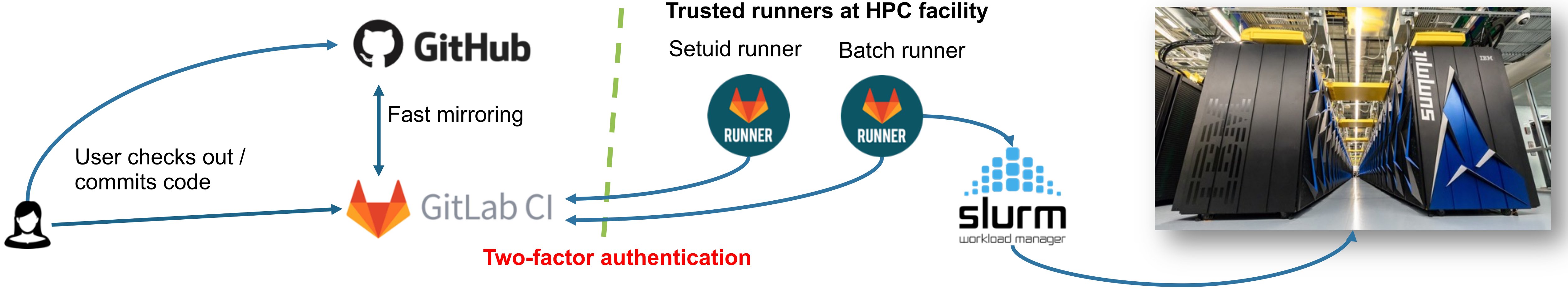
```
spack:
  specs:
    - hdf5 @1.8.16
    - openmpi fabrics=libfabric
    - nalu
```

List of packages to install, with constraints

Build with **`docker build .`**

Run with Singularity
(or some other tool)

Through ECP, we are working with Onyx Point to deliver continuous integration for HPC centers



Trusted runners at HPC facility

Setuid runner

Batch runner

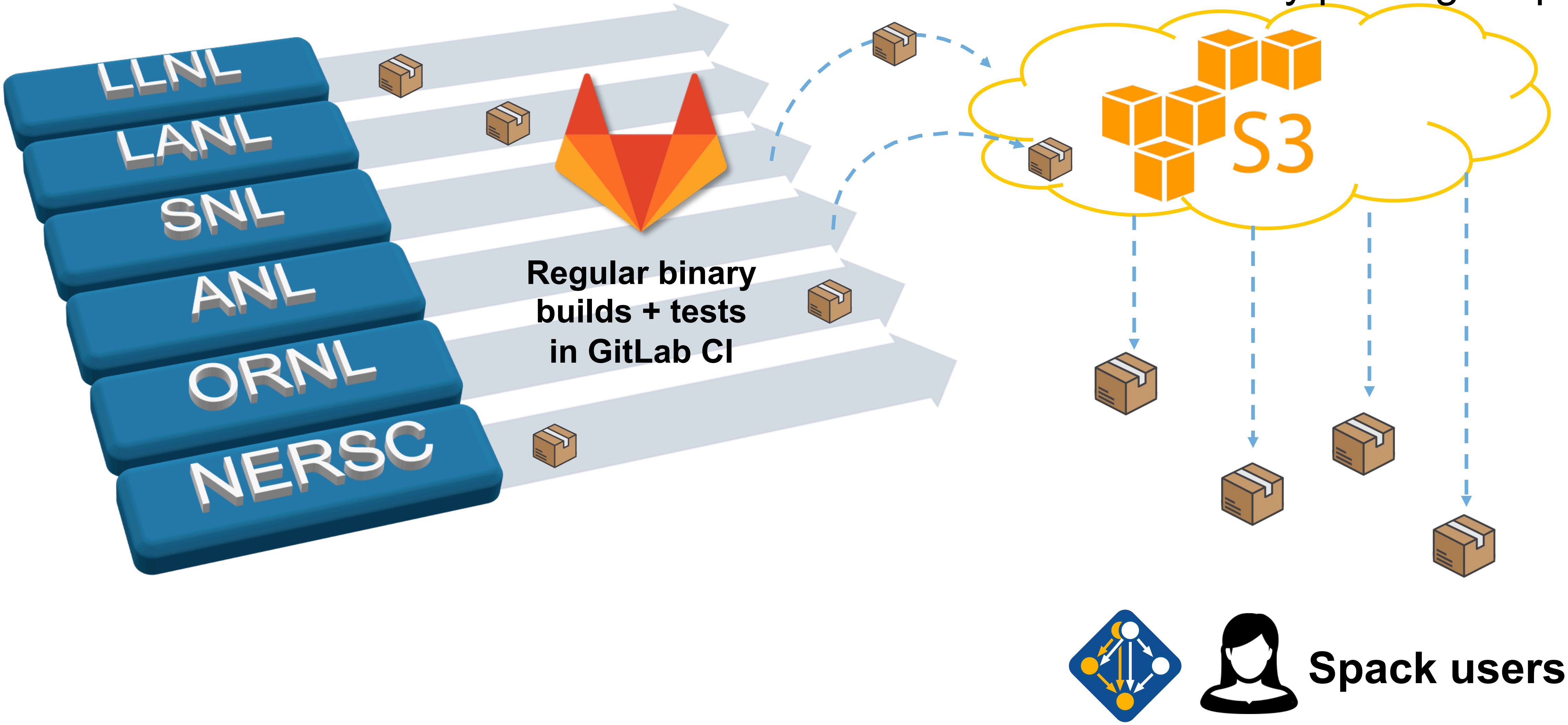
Fast mirroring

User checks out / commits code

Two-factor authentication

- CI at HPC centers is notoriously difficult
  - Security concerns prevent most CI tools from being run by staff or by users
  - HPC centers really need to deploy trusted CI services for this to work
- We are developing a secure CI system for HPC centers:
  - Setuid runners (run CI jobs as users); Batch integration (similar, but parallel jobs); multi-center runner support
- Onyx Point will upstream this support into GitLab CI
  - Initial rollout in FY19 at ECP labs: ANL, ORNL, NERSC, LLNL, LANL, SNL
  - Upstream GitLab features can be used by anyone!

Builds under ECP will be automated with continuous integration

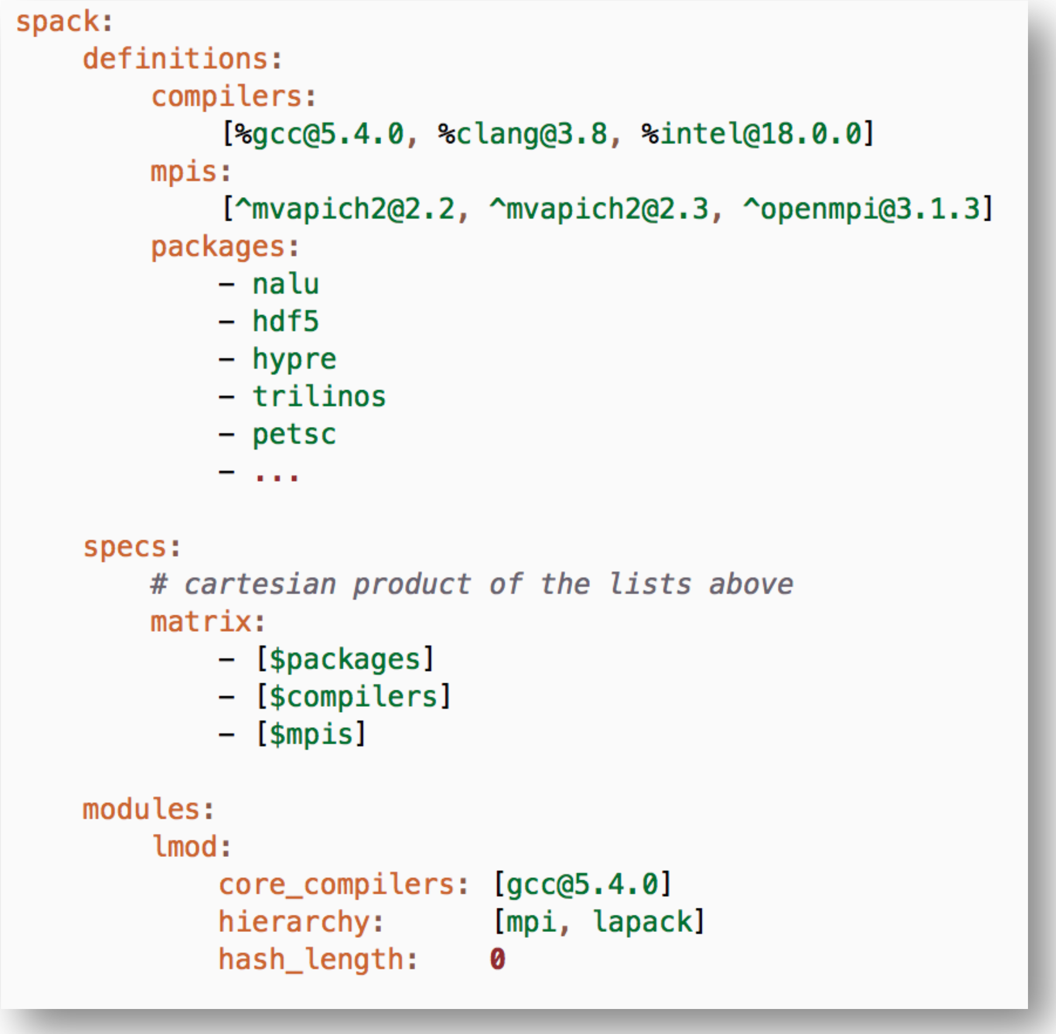


Public binary package repo

Regular binary builds + tests in GitLab CI

Spack users

Spack **stacks**: combinatorial environments for entire facility deployments

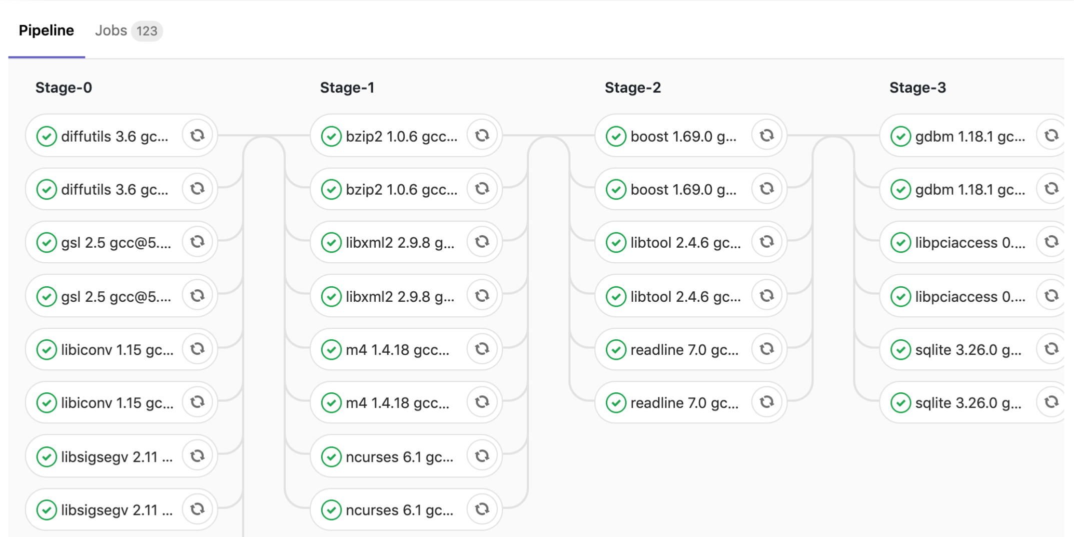


```yaml
spack:
    definitions:
        compilers:
            [%gcc@5.4.0, %clang@3.8, %intel@18.0.0]
        mpis:
            [^mvapich2@2.2, ^mvapich2@2.3, ^openmpi@3.1.3]
        packages:
            - nalu
            - hdf5
            - hypre
            - trilinos
            - petsc
            - ...

    specs:
        # cartesian product of the lists above
        matrix:
            - [$packages]
            - [$compilers]
            - [$mpis]

    modules:
        lmod:
            core_compilers: [gcc@5.4.0]
            hierarchy:      [mpi, lapack]
            hash_length:    0
```

- Allow users to easily express a huge cross-product of specs
  - All the packages needed for a facility
  - Generate modules tailored to the site
  - Generate a directory layout to browse the packages
- Build on the environments workflow
  - Manifest + lockfile
  - Lockfile enables reproducibility
- Relocatable binaries allow the same binary to be used in a stack, regular install, or container build.
  - Difference is how the user interacts with the stack
  - Single-PATH stack vs. modules.

GitLab CI Integration for Binary Build Pipelines

- Further builds on environments
  - Support auto-generating GitLab CI jobs
  - Can run in a Kube cluster or on bare metal runners at an HPC site
  - Sends progress to CDash
- **See PR #11612**

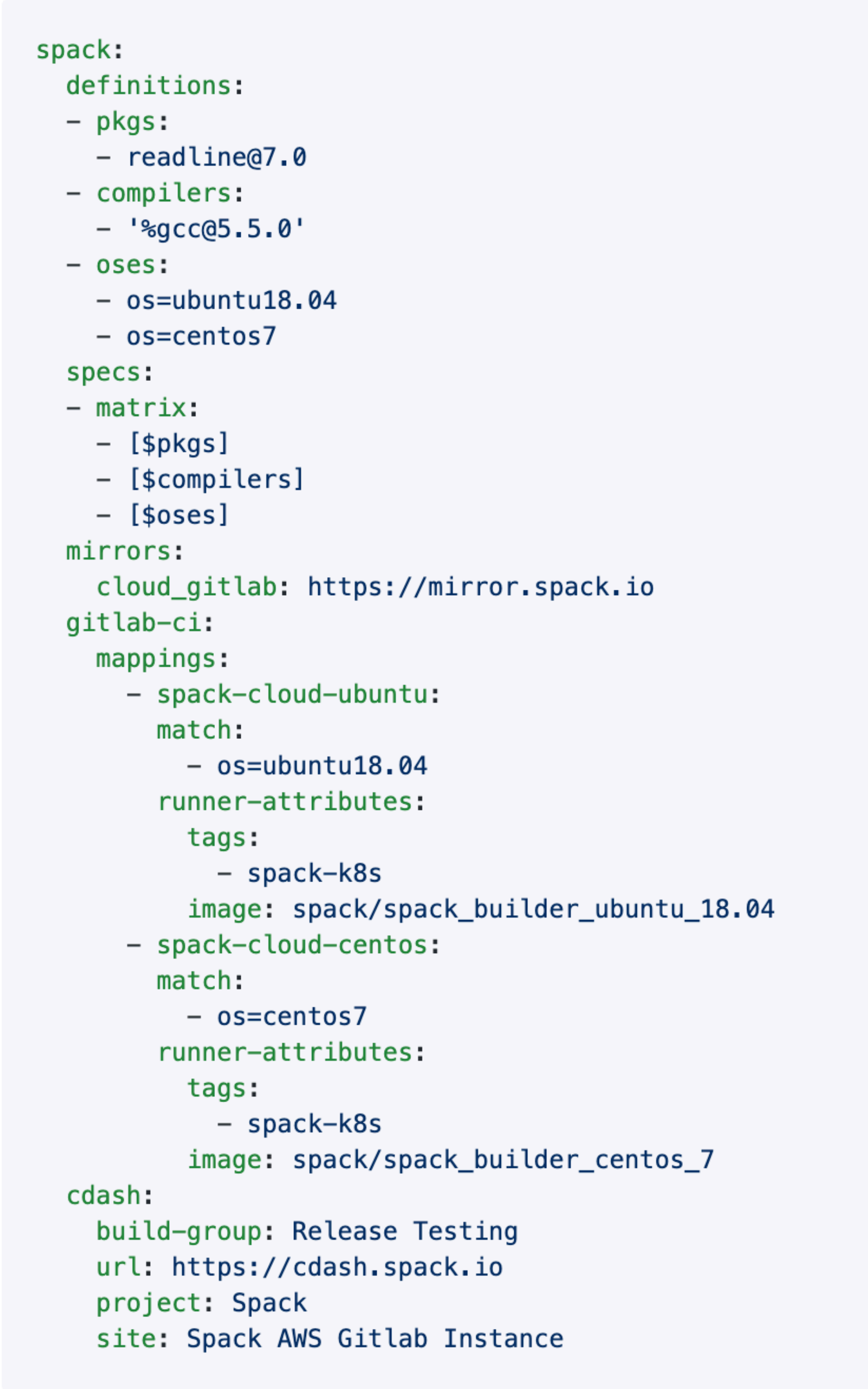


Stage-0
- diffutils 3.6 gc...
- diffutils 3.6 gc...
- gsl 2.5 gcc@5....
- gsl 2.5 gcc@5....
- libiconv 1.15 gc...
- libiconv 1.15 gc...
- libsigsegv 2.11 ...
- libsigsegv 2.11 ...

Stage-1
- bzip2 1.0.6 gcc...
- bzip2 1.0.6 gcc...
- libxml2 2.9.8 g...
- libxml2 2.9.8 g...
- m4 1.4.18 gcc...
- m4 1.4.18 gcc...
- ncurses 6.1 gc...
- ncurses 6.1 gc...

Stage-2
- boost 1.69.0 g...
- boost 1.69.0 g...
- libtool 2.4.6 gc...
- libtool 2.4.6 gc...
- readline 7.0 gc...
- readline 7.0 gc...

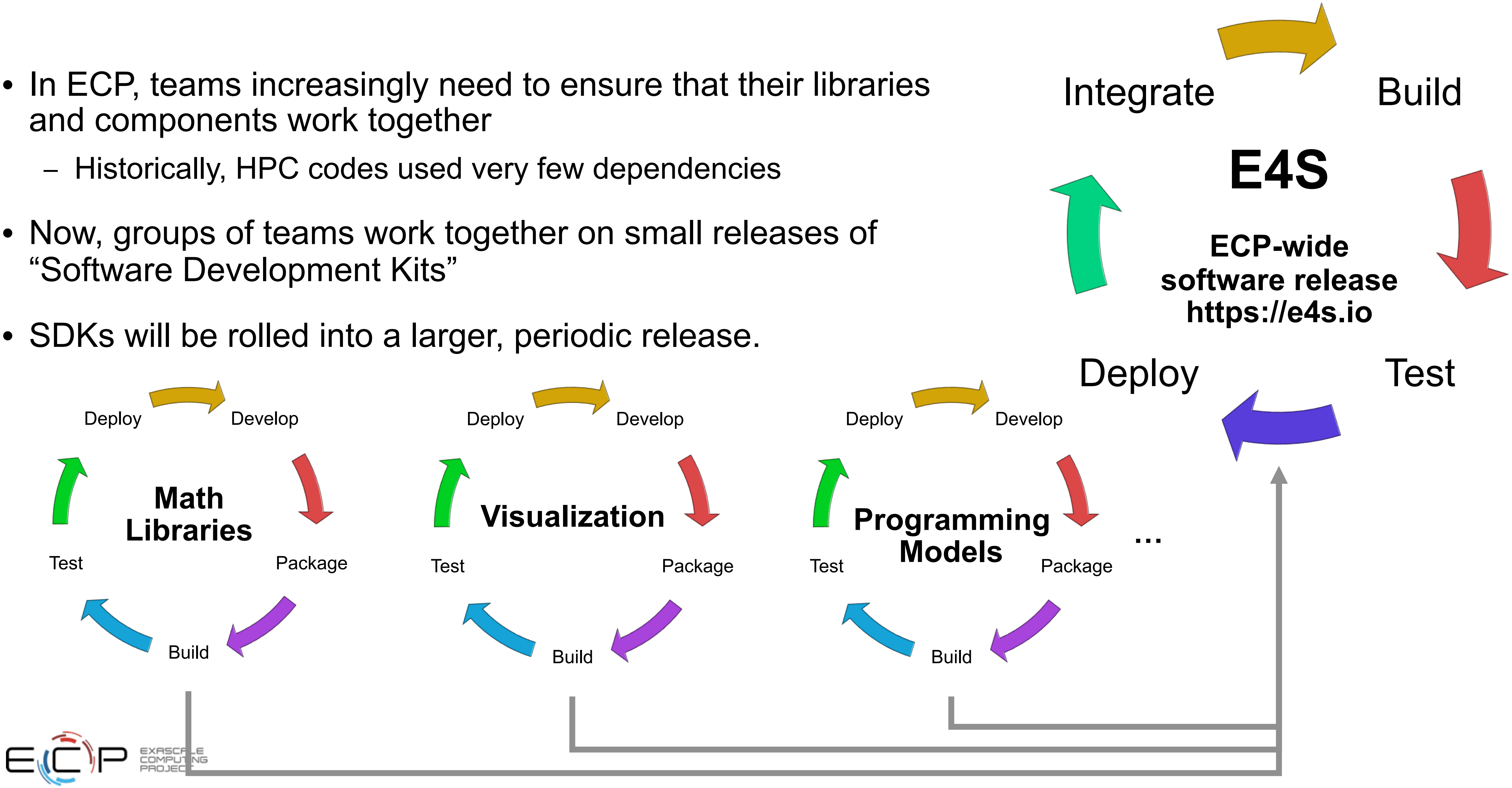
Stage-3
- gdbm 1.18.1 gc...
- gdbm 1.18.1 gc...
- libpciaccess 0....
- libpciaccess 0....
- sqlite 3.26.0 g...
- sqlite 3.26.0 g...



```yaml
spack:
  definitions:
  - pkgs:
    - readline@7.0
  - compilers:
    - '%gcc@5.5.0'
  - oses:
    - os=ubuntu18.04
    - os=centos7
  specs:
  - matrix:
    - [$pkgs]
    - [$compilers]
    - [$oses]
  mirrors:
    cloud_gitlab: https://mirror.spack.io
  gitlab-ci:
    mappings:
      - spack-cloud-ubuntu:
        match:
          - os=ubuntu18.04
        runner-attributes:
          tags:
            - spack-k8s
          image: spack/spack_builder_ubuntu_18.04
      - spack-cloud-centos:
        match:
          - os=centos7
        runner-attributes:
          tags:
            - spack-k8s
          image: spack/spack_builder_centos_7
  cdash:
    build-group: Release Testing
    url: https://cdash.spack.io
    project: Spack
    site: Spack AWS Gitlab Instance
```

ECP is working towards a periodic, hierarchical release process

- In ECP, teams increasingly need to ensure that their libraries and components work together
  - Historically, HPC codes used very few dependencies
- Now, groups of teams work together on small releases of "Software Development Kits"
- SDKs will be rolled into a larger, periodic release.



What will ECP’s software legacy be?

All of this is still in progress, but hopefully:

1. Continuous integration for HPC users actross DOE (and elsewhere)
2. A robust, widely available, and tested HPC software distribution
3. Support for optimized packaging and containers across diverse architectures