Sandia National Laboratories

Exceptional service in the national interest

ASC DevOps Initiative

ASC Unified Environment and Trilinos

Presented by Scott Warnock

Content also provided by ASC DevOps Technical Leadership Team, ASC DevOps Management Team, & Andrew Younge

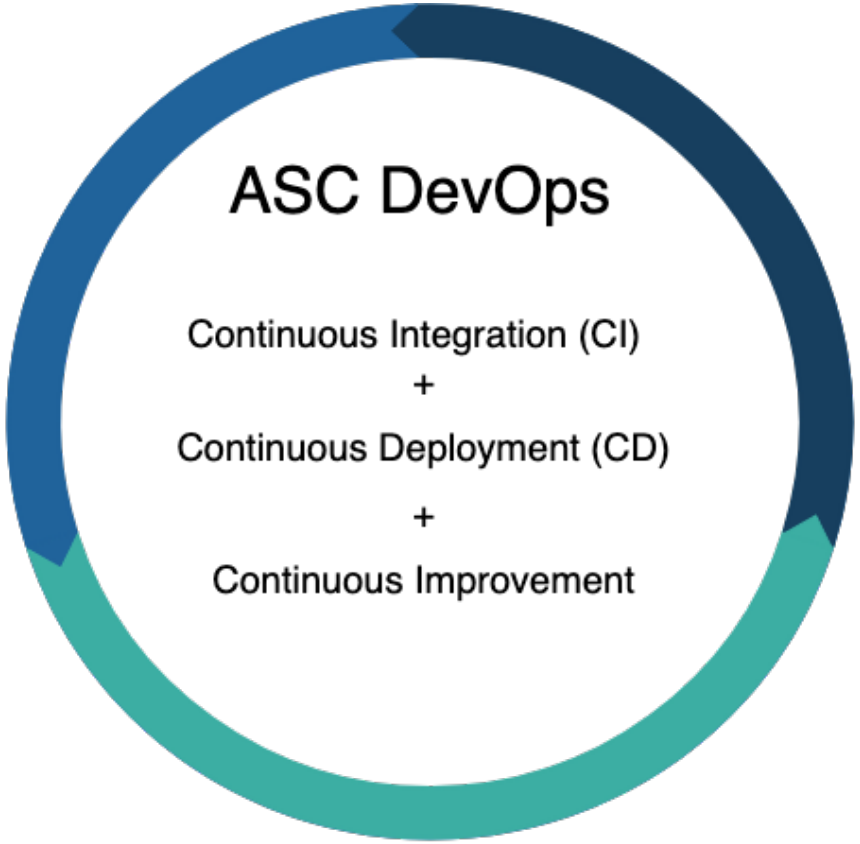
TUG, 26 October 2022, Albuquerque, NM

SAND2022-14822 C

AGENDA

- Overview of ASC DevOps Initiative
- Where Trilinos Fits
- ASC DevOps & Trilinos FY23

SNL Ecosystem-Wide Collaboration and Communication



Unified Environments

Coordinated Development Teams

Long-Term, Inter-ModSim-Team Collaboration/Communication

Hardware

Unified Environments + Containers + System Hardware

Long-Term, Collaboration/Communication w/System Hardware Teams (e.g. CEE, HPC Systems, HAPPs)

Containers

Unified Environments + Software Systems

Long-Term, Collaboration/Communication w/System Software Teams (e.g. TOSS, ATSE, ATS)

ASC DevOps' focus is to minimize { (time for CI/CD cycle) / (ASC-wide DevOps efficiency) } through ecosystem-wide, collaborative R&D

ASC Unified Environments

Trilinos Dependent ModSim Codes

RAMSES	CCR	CompSim
EMPIRE	Alegra	SPARC
GEMMA	Aleph	Sierra (DO, SM, TF, SD, InvOPT)
Charon	Dakota	Cubit/GMTK
Eiger	Catalyst	Plato
Xyce	Trilinos	

MVP Selected Codes

MVP/Tooling Infastructure

Spack Package Management

Spack Repo

Spack Tooling
- package.py
- environment.yaml

Spack Repo Branches

Pre-release → Development → Qualification → Production

Build Cache
- Tier 1 Compilers
- Tier 1 MPI

Deployed Modules
- Tier 1 Tooling
- Tier 1 Compilers
- Tier 1 MPI
- Tier 2 TPLs

ASC DevOps Supported Systems & Networks

Systems

CEE | CTS | TLCC | VAN | ATS

Networks

Multiple supported networks

Build & Test Simulation Code

Env. Build environment & Tooling
- Underlying OS provided
- VAN - ATSE
- HPC - TOSS
- CEE - RHEL
- Vortex -Coral

Tier 1 Tooling
- Anaconda/Python
- CMake
- git
- git-lfs
- LaTex
- Ninja
- Spack

1. **Build & Test Compiler**
2. **Build & Test MPI**
3. **Build & Test Trilinos Flavor TPLs**
4. **Build & Test Trilinos**
5. **Build & Test Code Dependent TPLs**
6. **Build Simulation Code**

ModSim Codes

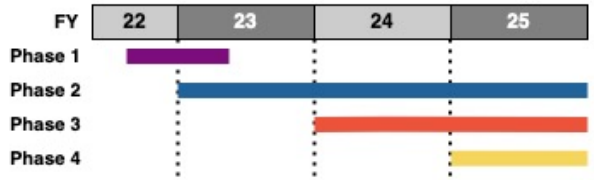
RAMSES	CCR	CompSim	IWF
ITS	LGR	CTH	IWF
Cheetah	SPARTA		SAW
Q	DARMA		
SCEPTRE	Kokkos		
	Paraview		
	Ensight		
	Slycat		
	SAW-Slycat Integration		

Tier 1 Tooling	Tier 1 Compilers	Tier 1 MPI
Anaconda/Python	GCC	OpenMPI
CMake	LLVM (Clang)	IntelMPI
git	Intel	Vendor
git-lfs	Arm	
LaTex	CUDA	
Ninja		
Spack		

- MVP/Tooling Infastructure
- Unified Environment Tier 1
- Trilinos Dependent TPLs
- Unified Environment Tier 2

Tier 2 TPLs
- Boost
- HDF5
- NetCDF
- Parallel NetCDF
- SuperLU
- CGNS
- SuiteSparse
- Hypre
- yamlcpp
- Metis/ParMetis
- Netlib BLAS & Lapack
- Accelerated Libraries
- libz

Unified Environments Roadmap



Timeline	Deliverable
Phase 1	**MVP/Tooling Infrastructure** (repos, caches, governance, environments, initial Spack configuration methodologies)
Phase 2	**External Packages** (e.g., compilers, MPI, Anaconda)
Phase 3	**Tooling** (Spack configuration)
Phase 4	**Unification** (modules on systems)

Containers Collaboration Framework

ASC DevOps Container Project

Scope

Runtime Support
Have ability to build containers! Currently, on track for Podman within RHEL. Docker is still the technology for macOS and Windows.

Base Images
Decide on minimum set of base container images and intended use cases (e.g., RHEL UBI will support external, deployments whereas Alpine will only support external).

Env Images
Create "env" images that add only the libraries and build utilities needed to run app built atop (3.). These are, ideally, as small as possible without too much effort w.r.t. breaking things on the base image.

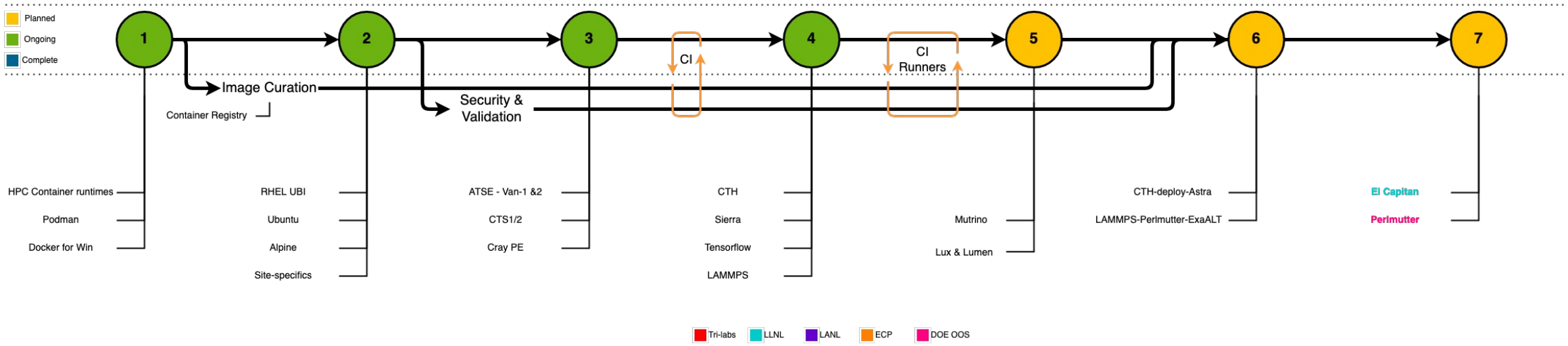
Dev Images
Create "developer" images that add PE atop relevant base images (e.g., some of these will mimic CTS-1/TOSS, some will mimic CEE/RHEL, some ATS-1/Trinity, some ASCIC). These will contain relevant compilers, MPI, and other TPLs/utilities for building and debugging the application.

Testing
Learn how to use images from (3.) and (4.) across the wide spectrum of needs and use cases. For example, what is the best way to use TotalView to debug? Another example, how would an analyst use a Sierra container and a CTH container to run Zapotec and post process results with a Python container?

Deployment Solutions
Investigate ways of heavily compressing (4.) to have something that is truly minimal.

Supercomputers
Now that sizes of images are understood, compute estimates for infrastructure needed to handle production rollout of this capability assuming *_all_* apps are interested (because they are).

Execution Sequence



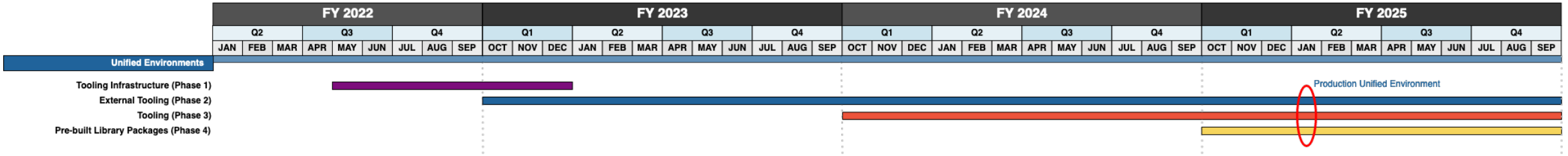
Planned or Ongoing Container Sub-projects

Collaboration

ASC DevOps and Trilinos

1. Adopt Supported Compilers + MPI
2. Align on selected TPLs

ASC DevOps 3-Year Roadmap



ASC Unified Environments (AUE) Tooling, Compilers + MPI

Tier 1 Tooling
- Anaconda/Python
- CMake
- git
- git-lfs
- LaTex
- Ninja
- Spack

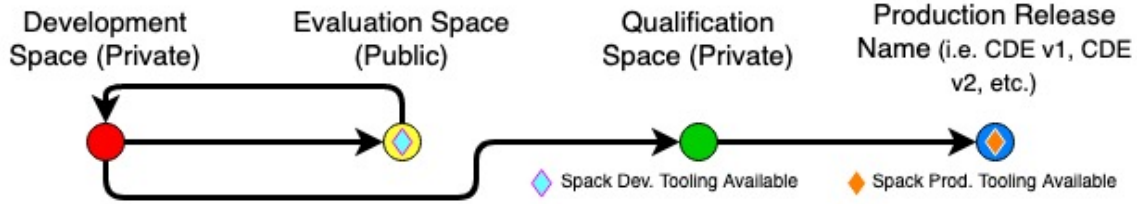
Tier 1 MPI
- OpenMPI
- IntelMPI
- Vendor

Tier 1 Compilers
- GCC
- LLVM (Clang)
- Intel
- Arm
- CUDA

Tier 1 Profilers/Debuggers
- HPCToolKit
- Trenza Survey
- TotalView

ASC DO Evaluation Space

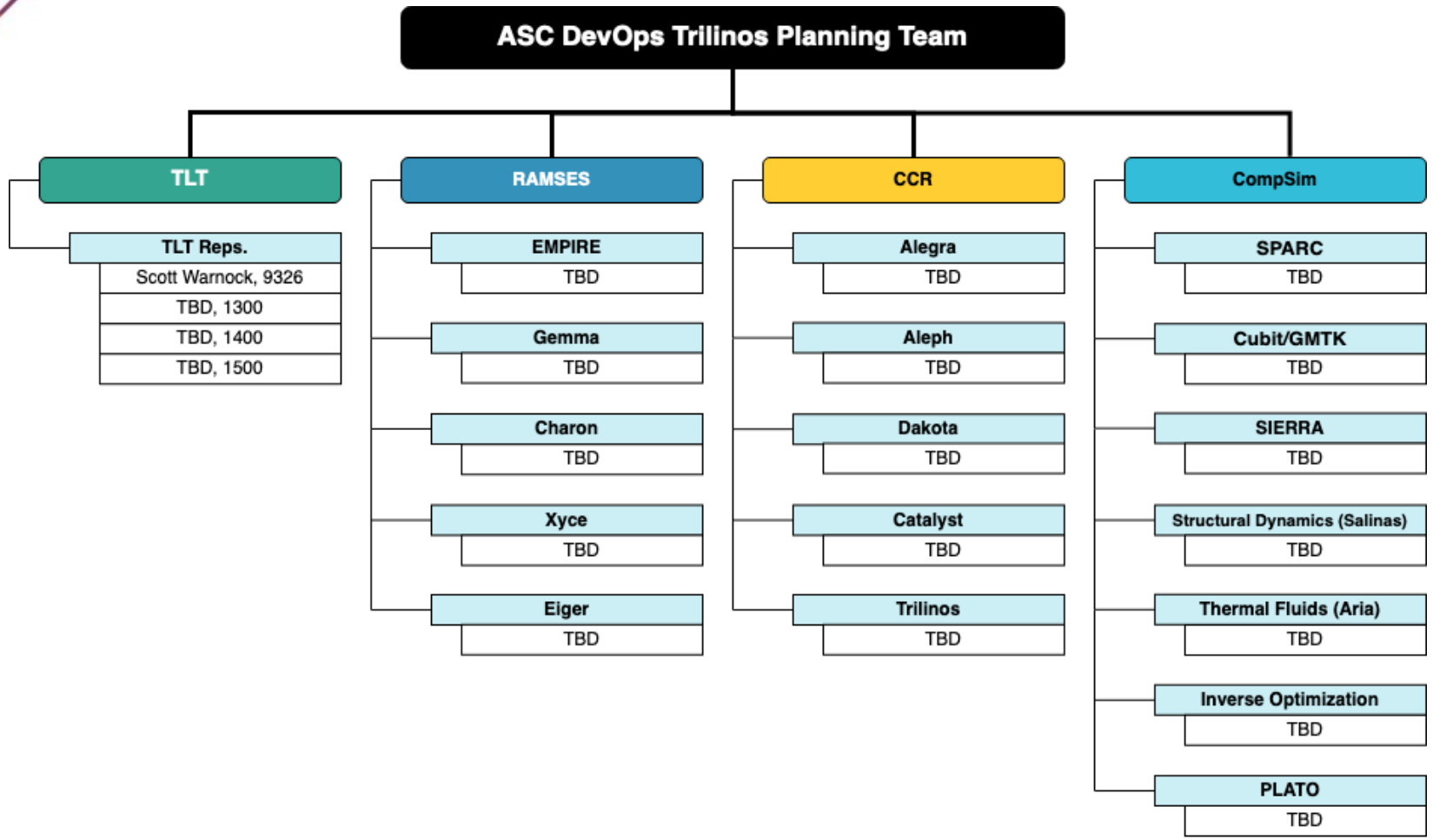
Development-to-Deployment Promotion Workflow



ASC Unified Environments (AUE) TPLs

Tier 2 TPLs
- Boost
- HDF5
- NetCDF
- Parallel NetCDF
- SuperLU
- CGNS
- SuiteSparse
- Hypre
- yamlcpp
- Metis/ParMetis
- Netlib BLAS & Lapack
- Accelerated Libraries
- libz

ASC DevOps and Trilinos FY23



- Setting up Planning Team
- Understand current state
- Map Tilinos builds by team
- Look for commonality
- Develop technical path forward
- FY23 Planning is seen as risk mitigation
- One hour a week
- All stakeholders in the room