Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Disclaimer
This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.
DevOps Survey Team

Gregory Becker   Cyrus Harrison   Rich Hornung   Chris White
What is DevOps?

“DevOps is a combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver software at high velocity: evolving and improving products at a faster pace than organizations using traditional software development processes. This speed enables organizations to better serve their customers and compete more effectively.”

https://aws.amazon.com/devops/what-is-devops/
LLNL’s WSC Computational Physics (CP) program develops production HPC multi-physics simulation applications

Snapshot of Scale and Complexity of LLNL’s WSC CP Program:

- Development Efforts
  - ~120 Developers
    (Physicists, Engineers, Computer Scientists, Software Quality, etc...)
  - ~15 project teams
  - ~15 – 30+ year application lifetimes
  - ~12 million lines of production code across projects
    (with more than 100 third-party dependencies)

- Diversity of Programming Languages and HPC Architectures
  - C++/C, Fortran, Python, Lua
  - Distributed Memory plus Multi-core or Many-core

- Diversity of Data
  - Scalars, Arrays, Tables, Contours, CAD Geometry, Meshes

CP’s efforts are a microcosm of the broader HPC simulation community
The software ecosystem supporting HPC multi-physics simulations is very complex

Multi-physics Simulation Applications

CS Infrastructure
- Input Parsing
- Steering
- Communication
- Parallelism Abstractions
- I/O
- In Situ Coupling

Physics Packages
- Hydrodynamics
- Chemistry
- Thermal radiation
- \{and many more \ldots\}

Physics Libraries
- Material Properties
- Material Models

Numerical Libraries
- Linear Algebra
- Finite Elements

Workflow Applications

Problem Setup and Meshing
- Computational Geometry
- Mesh Generation
- Mesh Decomposition
- Mesh Overlay

Visualization and Analysis
- Mesh Rendering
- Feature Extraction
- Simulated Diagnostics

Uncertainty Quantification
- Ensemble Generation
- Parametric Studies
- Statistical Models

Simulation & Data Management
- Workflow Capture
- Data Organization
- Provenance
WSC Integrated Codes rely on wide variety of internal and external software

- **Integrated Codes**
  - MARBL, Ares, ALE3D

- **Internal LLNL Packages**
  - Single Physics/Infrastructure
  - Axom, RAJA, MFEM, hypre, Spack, BLT, etc.

- **Open-source LLNL Packages**
  - System packages, compilers, MPI, etc.

- **External Packages**

WSC control, influence, and integration complexity
Third-party library (TPL) build complexities

- Integrated Codes can rely on upwards of 80 TPLs
- Each individual TPL has its own build system and configuration options
Integrated code teams and library development teams have different DevOps concerns

**Integrated Code Teams**
- Large teams and DevOps expertise
- Impacted by many decisions made downstream

**Library teams**
- Often small teams and may have DevOps expertise
- Supports multiple teams and use cases
Smaller teams can’t always stand-up DevOps best practices

- Lack people and/or expertise
- Manually testing and rely on users to report issue
- Fully rely on larger projects
We conducted a DevOps survey to get a handle on the state of our software teams

- Projects have grown their DevOps solutions organically over the life of their projects (some are decades old ...)
- Best practices have changed over the years
- Once a solution is working, there is a resistance to move to a newer or “better” solution
- There is not a large overlap of employees across projects

Without knowing the landscape, we can’t move forward in a positive and common direction.
Focus of the Survey

- **Third Party Libraries**
  - Spack or project-grown systems
  - Identifying internal libraries that could be improved

- **Continuous Integration (CI)**
  - Testing (correctness and/or project integration)
  - Throughput of testing
  - Open-source/LC platforms and resources

- **Other Topics**
  - Documentation
  - Build Systems (CMake, Autoconf, Make, etc.)
  - Programming Languages and Compilers
  - Testing and Workflow Tools (ATS, Maestro, etc.)
We surveyed 32 project teams across LLNL programs about their DevOps practices

- Teams vary in size and DevOps expertise
- They all have perspectives and opinions on best practices
Teams spend significant resources on DevOps tasks

Projects estimate they spend a total of ~13 Full-Time Equivalent Staff per year managing their third-party libraries alone

13 FTEs is more than any standalone code project!

Complexity is growing!

- Advanced architectures
- More platforms to support at a given time
- Increased software modularity
Projects utilize various tools to manage TPL build complexity

- **Spack**
  - 17 projects
  - ~10 projects actively working towards

- GNU Make
- Python
- Bash/Shel
We can reduce maintenance burden by recognizing TPL overlap

Total Unique TPLs: 114
ALE3D: 77
Ares: 69
Marbl: 76
Recommendations for the Future

1) Share knowledge
   • DevOps expertise varies from teams
   • Smaller teams do not have resources to stand up DevOps best practices

2) Minimize burden of common tasks
   • Reduce number of teams solving the same problems
   • We should solve these in a singular spot

3) Solve problems at the source before they spread
   • Increase testing and correctness
   • Increase confidence in upgrading to newer libraries
   • Reduce burden of building and testing newer libraries
Shared Knowledge

**Hold WSC DevOps meetings**
- Identifies and track common goals
- Large group of experts available for quickly removing roadblocks
- Regular presentations from group

**Multiple Chat Rooms**
- Increase audience for one-off questions and advice
- Don’t necessarily have to go to the same person every time

**Tutorials**
- Give targeted presentations to spread knowledge
- Examples: CMake importing/exporting targets, GPU programming, etc.

**Share vetted compiler configurations, build recipes, and bug fixes**
Internal Documentation Build and Hosting Server

Doc Build Server

Project Repositories

Monitors user-defined branches

Builds Docs

Host docs in a shared and common location

Basically, an internal ReadTheDocs.org
Cross-team Third Party Library Build Farm

- Query for and use existing libraries
- Submit new releases and desired configurations

DevOps experts + project developers work on issues

Reduce individual project burden for overlap in common libraries!
Cross-project CI Testing

Original project (RAJA) know best what changed and how to solve potential problems and can catch problems before user is impacted.