This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

Exceptional service in the national interest

TRILINOS DEVOPS PIPELINE PLANNING

Sam Browne, Jonathan Hu, Curt Ober, Roger Pawlowski, Jim Willenbring

Trilinos Users Group 2023

9:15 am October 31st, 2023
ACKNOWLEDGEMENTS

We would like to thank everyone who has helped us ...  
• Michael Wolf  
• Mike Heroux  
• Raisa Koshkin  
• Phyllis Rutka  
• Trilinos Survey Respondents  
• Ross Bartlett  
• Phil Sakievich  
• Todd Gamblin  
• Others ...
There have been many complaints and perceived issues:
- "Trilinos is too big. Trilinos is too complex."
- "Why does Trilinos use TriBITS when there is Modern CMake and Spack?"
- "Trilinos takes too long to build."
- "Trilinos needs more testing."
- "Need to deliver code changes more quickly to applications."
- ...

These touch every component of the Trilinos DevOps Pipeline (TDOP)

So the basic question is "How do we improve the Trilinos DevOps Pipeline?"
LEADERSHIP GUIDANCE

Key Goals

• **Goal 1**: Have a broadly maintained and sustainable Trilinos DevOps Pipeline
  • Avoid the single person (point of failure) problem
  • Major risk for Trilinos and ASC

• **Goal 2**: Have one-day turn-around time from Trilinos commits to ASC applications
  • The pipeline should not stand in way of meeting this goal in the future
  • I.e., support stakeholder integration testing of Trilinos

Secondary Considerations

• Use standard tools/software when possible
  • We should question ourselves when we feel that we need one-off, in-house solutions

• Should we partition Trilinos?
  • Perception that Trilinos is “too large”

• What can we do to facilitate external adoption of Trilinos?
  • Issues encountered by ECP applications in building/testing Trilinos
  • Is there anything that we can do to lower the barrier to adoption?
EXECUTIVE SUMMARY (BLUF)

Trilinos plans to ...

- Remain single repository to maintain developer productivity
- Retain key capabilities of TriBITS and form a support team

- Utilize ASC DevOps common Trilinos configurations (e.g., RAMSES and CompSim)
- Provide/maintain a Spack recipe that others can use (e.g., ASC Stakeholders and Spack)

- Maintain/support CMake+TriBITS and Spack builds
- Incorporate Containers and GitHub Actions to catch build errors and keep builds clean

- Add Integration testing for Trilinos packages (e.g., Kokkos and Kokkos Kernels)
- Support application’s integration testing of Trilinos to mitigate integration issues

- Support both delivery (Trilinos GitHub) and deployment (Spack)
- Steward Trilinos’s Spack recipe with support from Framework and Trilinos Developers
TRILINOS BACKGROUND AND CONTEXT
"The Trilinos Project is an effort to facilitate the design, development, integration and ongoing support of mathematical software libraries within an object-oriented framework for the solution of large-scale, complex multi-physics engineering and scientific problems. Trilinos addresses two fundamental issues of developing software for these problems: (i) Providing a streamlined process and set of tools for development of new algorithmic implementations and (ii) promoting interoperability of independently developed software."


"The Trilinos project was established to address two important needs: (1) bringing teams of library developers together in order to leverage commonalities and produce compatible software components, formally called packages and (2) to amortize the cost and efforts associated with more formal software engineering requirements. With a modest level of coordination and without unduly compromising package team autonomy, Trilinos project members could leverage each other’s efforts, consolidate commonly needed tools, make packages compatible, and define a common set of software engineering tools and processes."

WHY IS TRILINOS A SINGLE REPOSITORY?

- Utilize a common set of tools and processes.
  - Trilinos DevOps Pipeline, SQE practices, ...
- Promote interoperability of independently developed software
  - A lot of code reuse!
  - A lot of capabilities built from multiple packages!
- Rapid/Coupled software development requires “SHA sync’ing”
- Partitioning the DAG introduces integration overhead
- Complexity only reduced through fewer packages/dependencies

Excluding archived/Framework packages
135 Packages and Subpackages
566 Required (solid) and Optional (dashed) Dependencies
OVERVIEW OF CURRENT TRILINOS DEVOPS PIPELINE

Drive PR process
Check out Trilinos (merge with develop)
Determine packages to build
Report pass/fail
Pull code
Check for PRs/apply rules

Open
SRN

Develop

Deploy

Tools used
• Jenkins
• AutoTester*
• GenConfig*
• SEMS modules*
• SimpleTestingRefactor*
• TriBITS*
• CMake/CTest/CDash

Configure  Build  Test
UPSTREAM PACKAGE INTEGRATION PROCESS

- Stakeholders request code changes
- Upstream packages may need to
  - Develop code
  - Fix bugs
  - Iterate with Stakeholders
  - Collaborate with other upstream packages
- Upstream packages then perform internal and/or integration testing
- If a simple internal fix, most packages can turn this around in a few hours
- Things that cause delays
  - Integration Testing
  - Package coordination
  - Cascading changes / issues
  - Will get worse with more packages
- Ways to improve
  - Incorporate downstream build configurations and representative use-cases
TRILINOS
DEVOPS
PIPELINE PLAN
TRILINOS REMAIN SINGLE REPO; RETAIN KEY CAPABILITIES OF TRIBITS

- What are the complaints and perceived issues?
  - "Trilinos is too big. Trilinos is too complex." → Too many packages/dependencies
  - Should Trilinos be “partitioned”?  
  - Why do we have TriBITS?

- Points to Consider
  - Survey says ... **76% - favored single Trilinos repository**
    - Large loss of productivity with multiple repos for developers and applications
    - TriBITS provides CMake layer of common/consistent functionality across Trilinos packages
      - Most large software projects have some CMake layer (e.g., LNLL’s BLT and VTK modules)
      - Recent and ongoing TriBITS improvements (e.g., modern CMake and external builds)

- Planned solution/response
  - Remain single repository for Trilinos to maintain developer productivity
  - Retain key capabilities of TriBITS and form a support team
  - Review packages/dependencies to ensure we have the minimum set
  - Consider not using (and removing) subpackages capability from TriBITS/Trilinos
USE COMMON TRILINOS CONFIGURATIONS; PROVIDE SPACK RECIPE

• What are the complaints and perceived issues?
  • Too many configuration options to get the build you would like
  • Not easy to configure/build for a single/few package(s)

• Points to Consider
  • Survey says ...
    • Configuration preference - CMake/TriBITS (35%), CMake/Scripting (32%), CMake/Package Manager/Scripting (23%)
    • Documentation on configuring/building needs improvement. New users have difficulty.

• Planned solution/response
  • Improve documentation for configuration options and provide examples
  • Provide "recommended" configure defaults based on PR testing
  • Utilize ASC DevOps common Trilinos configurations (e.g., RAMSES and CompSim)
  • Review dependencies to ensure required and optional dependencies are correctly defined
  • Provide/maintain a Spack recipe that others can use (e.g., ASC Stakeholders and Spack)
  • Generate intra-package/TPL dependencies with TriBITS for CMake+TriBITS and Spack recipe
MAINTAIN CMAKE+TRIBITS AND SPACK BUILDS; INCORPORATE CONTAINERS AND GITHUB ACTIONS

• What are the complaints and perceived issues?
  • Build times are long.

• Points to Consider
  • Survey says …
    • Trilinos build times are not long compared to application build times. (21 to 9 respondents)
    • >90% build times less than an hour
  • Container usage is dependent on CEE LAN RHEL 8

• Planned solution/response
  • Continue CMake+TriBITS builds (more for developers/power users)
  • Expand support for Spack+CMake builds (more for Stakeholders/future usage/sysadmins)
  • Add a Spack PR build (meaning a Spack build must pass to commit)
  • Incorporate Containers and GitHub Actions to catch build errors and keep builds clean
  • Utilize build caches (ccache) to increase speed
ADD INTEGRATION TESTING FOR TRILINOS PACKAGES; AND SUPPORT INTEGRATION TESTING OF TRILINOS

• What are the complaints and perceived issues?
  • Pull Request (PR) testing results not visible to foreign nationals and external developers
  • Trilinos needs greater performance testing
  • Lengthy process to snapshot packages like Kokkos → integration takes too long

• Points to Consider
  • Trilinos testing covers CTS/testbed (SRN) platforms
  • There are gaps in PR testing due to hardware availability (e.g., AMD GPUs)
  • Integration testing between
    • Trilinos packages would introduce additional overhead
    • Upstream packages/Trilinos/applications can reduce turn-around time
  • Trilinos has internal “performance” dashboard

• Planned solution/response
  • Add Integration testing for Trilinos packages (e.g., Kokkos and Kokkos Kernels)
  • Support application's integration testing of Trilinos to mitigate integration issues
  • Add application use cases for performance monitoring
  • Expand performance testing and memory monitoring (e.g., Watchr)
  • Move PR and Nightlies to SON/OHPC for foreign nationals and external collaborators
  • Leverage cloud and other institution resources to provide external testing
SUPPORT BOTH TRILINOS GITHUB AND SPACK; STEWARD TRILINOS’S SPACK RECIPE

• What are the complaints and perceived issues?
  • "Need to deliver code changes more quickly to applications."
  • Difficult to use Spack’s recipe for Trilinos

• Points to Consider
  • Survey Says ...
    • Preference is to use CMake installation over package manager (20 to 6 respondents)
    • Package Manager (e.g., Spack) is path forward for the future
    • In the past, primarily only delivered Trilinos and applications deployed.

• Planned solution/response
  • Support both delivery (Trilinos GitHub) and deployment (Spack)
  • Steward Trilinos’s Spack recipe with support from Framework and Trilinos Developers
PLANNED TRILINOS DEVOPS PIPELINE – LEVERAGE EXISTING TOOLS

- Drive PR process
- Set up build environment
- Determine packages to build (generate spack variants)
- Build/test/report
- Report pass/fail

Tools used:
- GitHub Actions
- Podman
- Spack
- TriBITS*
- CMake/CTest/CDash

Additional points:
- Move from 5 internally-developed products to 1!
- Manage configurations as opposed to tools
- Move results to SON/OHPC to allow for better communication and development
BENEFITS OF PLANNED TRILINOS DEVOPS PIPELINE

• Lower maintenance costs
  • Use industry-standard externally-developed software solutions (e.g., GitHub Actions)
  • Maintain configurations of external toolkits (e.g., Dockerfiles) than owning toolkits (e.g., LoadEnv and SEMS modules)

• Allows easier reproduction of automated builds
  • Container images will be able to be shared vs. SEMS modules only existing on internal machines
  • Spack is a more-common language than GenConfig across ECP-related customers

• GitHub Actions maintains current security requirements of AutoTester

• Enhances current security
  • Force all jobs to run in container, which is more-isolated from the system
  • Also run as new entity account (unprivileged) which only has very select machine accounts
  • Also run on SON/OHPC vs. SRN, greatly reducing any concerns about ECI/etc.

• Standardized tooling may not support current capabilities
  • E.g., per-package breakdown on CDash may require specialized code in/around Spack

Overall, drive towards more community-supported tooling, more nimble automation, and more visible results to enable a wider community to develop more quickly on Trilinos
**PIPELINE – ADDRESSING KEY GOALS**

**Goal 1:** Have a broadly maintained and sustainable Trilinos DevOps Pipeline  
- Framework team is mentoring and spreading knowledge across new team members  
- Retain key capabilities of TriBITS and form a support team

**Goal 2:** Work towards one-day turn-around time for Trilinos commits to ASC applications  
- Basic Trilinos changes are merged (Delivered) within a few hours  
- Utilize ASC DevOps common Trilinos configurations (e.g., RAMSES and CompSim)  
- Add Integration Testing for packages (e.g., Kokkos and Kokkos Kernels)  
- Add ASC use cases for performance monitoring and improved integration testing

**Secondary Considerations**

- Use standard tools/software when possible  
  - Replace 4 internally developed tools with standard tools  
  - Utilize Containers and GitHub Actions to catch build errors and keep builds clean

- Should we partition Trilinos?  
  - Remain a single repository to maintain developer productivity  
  - Review packages/dependencies to ensure we have the minimum needed set  
  - Consider not using (and removing) subpackages capability from TriBITS/Trilinos  
  - Review dependencies to ensure required and optional dependencies are correctly defined

- What can we do to facilitate external adoption of Trilinos?  
  - Improve documentation for configuration options and provide examples  
  - Maintain/support CMake+TriBITS and Spack builds  
  - Provide/maintain a Spack recipe that others can use (e.g., ASC Stakeholders and Spack)
NEXT STEPS ...

- Timeline
  - FY24
    - Epetra-Tpetra Transition for Trilinos packages
    - Current planned tasks
      - Test/support CompSim and RAMSES Trilinos configurations (coordinating with ASC DevOps)
      - Ownership/support of Spack-based Trilinos build added to nightly/PR testing
      - Transition from SEMS Autotester to GitHub Actions + Containers
      - Transition from GenConfig to Spack for configuration management*
    - Continued planning of TDOP
  - FY25
    - Epetra-Tpetra transition for Stakeholders
    - Start other planned TDOP responses
EXECUTIVE SUMMARY (BLUF)

**Trilinos plans to ...**

- Remain single repository to maintain developer productivity
- Retain key capabilities of TriBITS and form a support team
- Utilize ASC DevOps common Trilinos configurations (e.g., RAMSES and CompSim)
- Provide/maintain a Spack recipe that others can use (e.g., ASC Stakeholders and Spack)
- Maintain/support CMake+TriBITS and Spack builds
- Incorporate Containers and GitHub Actions to catch build errors and keep builds clean
- Add Integration testing for Trilinos packages (e.g., Kokkos and Kokkos Kernels)
- Support application’s integration testing of Trilinos to mitigate integration issues
- Support both delivery (Trilinos GitHub) and deployment (Spack)
- Steward Trilinos’s Spack recipe with support from Framework and Trilinos Developers
SUMMARY OF PLANNED TDOP RESPONSES

DEVELOP

• Develop
  • Remain single repository for Trilinos to maintain developer productivity
  • Retain key capabilities of TriBITS and form a support team
  • Review packages/dependencies to ensure we have the minimum set
  • Consider not using (and removing) subpackages capability from TriBITS/Trilinos

CONFIGURE

• Configure
  • Improve documentation for configuration options and provide examples
  • Provide "recommended" configure defaults based on PR testing
  • Utilize ASC DevOps common Trilinos configurations (e.g., RAMSES and CompSim)
  • Review dependencies to ensure required and optional dependencies are correctly defined
  • Provide/maintain a Spack recipe that others can use (e.g., ASC Stakeholders and Spack)
  • Generate intra-package/TPL dependencies with TriBITS for Cmake+TriBITS and Spack recipe

BUILD

• Build
  • Continue CMake+TriBITS builds (more for developers/power users)
  • Expand support for Spack+CMake builds (more for Stakeholders/future usage/sysadmins)
  • Add a Spack PR build (meaning a Spack build must pass to commit)
  • Incorporate Containers and GitHub Actions to catch build errors and keep builds clean
  • Utilize build caches (ccache) to increase speed

TEST

• Test
  • Add Integration testing for Trilinos packages (e.g., Kokkos and Kokkos Kernels)
  • Support application's integration testing of Trilinos to mitigate integration issues
  • Add application use cases for performance monitoring
  • Expand performance testing and memory monitoring (e.g., Watchr)
  • Move PR and Nightlies to SON/OHPC for foreign nationals and external collaborators
  • Leverage cloud and other institution resources to provide external testing

DELIVER DEPLOY

• Deliver and Deploy
  • Support both delivery (Trilinos GitHub) and deployment (Spack)
  • Steward Trilinos's Spack recipe with support from Framework and Trilinos Developers