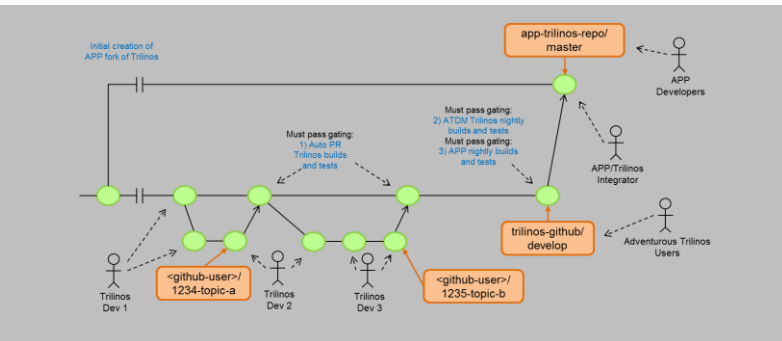


Exceptional service in the national interest



Group	Site	Build Name	Missing Status
ATDM	chance	Trilinos-atomic-chance-initial-debug-openssl	Build exists but no test results
ATDM	chance	Trilinos-atomic-chance-initial-debug-openssl	Build exists but no test results
ATDM	chance	Trilinos-atomic-chance-initial-debug-openssl	Build exists but no test results
ATDM	chance	Trilinos-atomic-chance-initial-debug-openssl	Build exists but no test results
ATDM	chance	Trilinos-atomic-chance-initial-debug-openssl	Build exists but no test results
ATDM	chance	Trilinos-atomic-chance-initial-debug-openssl	Build exists but no test results
ATDM	chance	Trilinos-atomic-chance-initial-debug-openssl	Build exists but no test results
ATDM	chance	Trilinos-atomic-chance-initial-debug-openssl	Build exists but no test results
ATDM	chance	Trilinos-atomic-chance-initial-debug-openssl	Build exists but no test results
ATDM	chance	Trilinos-atomic-chance-initial-debug-openssl	Build exists but no test results

ATDM Trilinos Testing and Integration

Trilinos Developers Meeting
October 25, 2018

Roscoe A. Bartlett, Joseph R. Frye
Sandia National Laboratories

SAND2018-12416 PE



Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Overview of ATDM Trilinos Testing and APP Integration

- **Trilinos / APP Git Workflows:**

- How git repositories and branches are set up, how merges occur, what git commands are run, etc.
- Different git workflows are used for Trilinos developers, APP developers, and APP/Trilinos co-developers
- Gating test suites can/should be run before each “merge” in the workflow

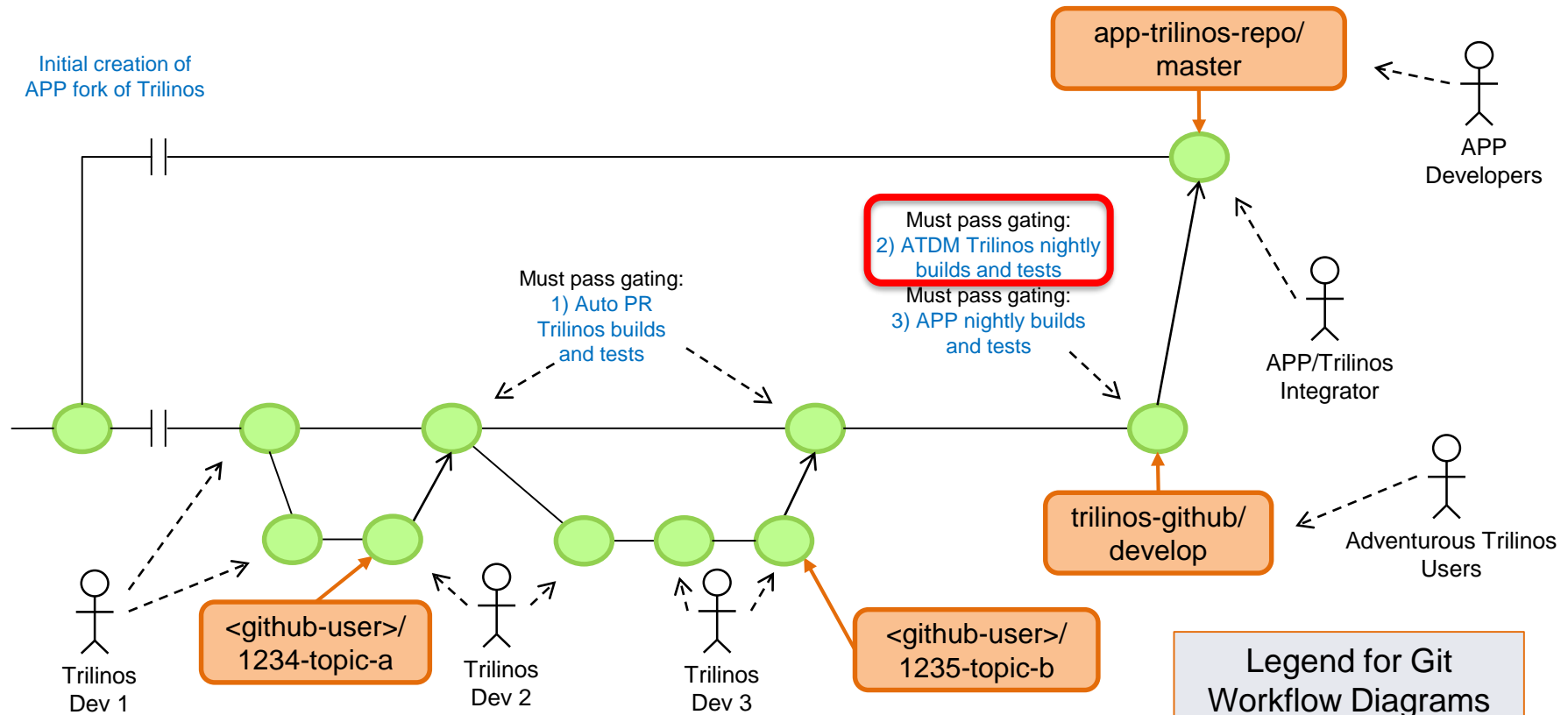
- **Testing gates for workflows:**

- Gating tests can be run manually or automated, daily or “every-so-often”, etc.
- Important test suites:
 - [1\) Auto PR Trilinos builds and tests](#) : Owned by the Trilinos Framework team
 - [2\) ATDM Trilinos nightly builds and tests](#): Jointly owned by ATDM DevOps and APP teams
 - [3\) APP nightly builds and tests](#): Owned by APP teams

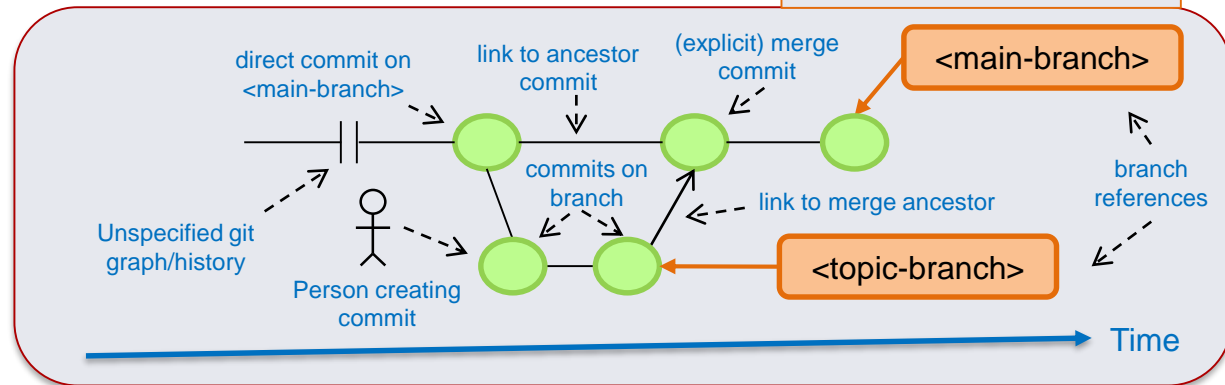
- **Triaging and fixing failed builds and tests:**

- Notification of new failures
- Triage failures
- Address failures
- Manage & Follow-up

Trilinos Development and APP-Integration Workflows



This is the goal, but the APPs are updating without checking that the native Trilinos test suite actually passes on the ATDM platforms!



Trilinos Development and APP-Integration Parts

- **Trilinos / APP Git Workflows:**

- Trilinos Pull Request (PR) testing & merging to 'develop' [**Done**]
- SPARC / Trilinos subteam workflow (manual testing by Micah H.) [**Done**]
- EMPIRE / Trilinos git.git workflow (EMPIRE-owned Jenkins pipeline for testing) [**Done**]

- **Testing gates for workflows:**

- 1) Auto PR Trilinos builds and tests :
 - **Current:** MPI GCC 4.8.4, GCC 4.9.3, Intel 17
 - **Future:** CUDA (see [Trilinos GitHub #2646](#))
- 2) ATDM Trilinos nightly builds and tests:
 - ATDM Trilinos builds for EMPIRE ... EMPIRE switchover in-progress/complete?
 - ATDM Trilinos builds for SPARC ... 'cee-rhel6' gnu, intel, and clang complete
- 3) APP nightly builds and tests
 - EMPIRE-PIC and EMPIRE-Fluid build and test suite: A few builds posting to Jenkins only
 - SPARC build and test suite: Submits to Sierra CDash site
- **Triaging and fixing failed builds and tests:**
 - **Notification of new failures:** Python email tool pulling data for ATDM Trilinos builds off [CDash site](#).
 - **Triage failures:** Filter out non-code failures then create [Trilinos GitHub Issues](#)
 - Joe Frye creates initial GitHub Issues, Product Areas Leads follow up from there
 - **Address failures :**
 - New builds: a) fix, b) allow to fail, c) temporally disabling non-critical tests
 - Existing builds: a) fix, b) allow to fail, c) temporally disable, or d) reverting PR from 'develop'
 - **Manage & Follow-up:** Someone must observe and ensure failures are addressed (???Who???)

2) ATDM Trilinos Nightly Builds and Tests (CDash)

ATDM																30 builds	
Site	Build Name	Update		Configure		Build		Test				Start Time	Labels				
		Revision	Time	Error	Warn	Time	Error	Warn	Time	Not Run	Fail			Pass			
sems-rhel8	Trilinos-atdm-sems-rhel8-gnu-opt-serial	254bd0	12s	0	19	1m 21s	0	29	30m 2s	0	0	1781	5m 13s	41m 31s	Oct 22, 2018 - 06:18 UTC	(25 labels)	
sems-rhel8	Trilinos-atdm-sems-rhel8-gnu-opt-openmp	254bd0	12s	0	19	1m 25s	0	29	35m 53s	0	0	1891	5m 42s	46m 56s	Oct 22, 2018 - 05:29 UTC	(25 labels)	
hansen	Trilinos-atdm-hansen-shiller-intel-opt-serial	254bd0	18s	0	19	3m 44s	0	80	1h 26m 20s	0	0	1780	5m 48s	1h 22m 10s	Oct 22, 2018 - 10:03 UTC	(25 labels)	
hansen	Trilinos-atdm-hansen-shiller-intel-debug-serial	254bd0	12s	0	19	3m 30s	0	50	1h 26m 17s	0	0	1777	5m 59s	1h 23m 51s	Oct 22, 2018 - 08:25 UTC	(25 labels)	
sems-rhel8	Trilinos-atdm-sems-rhel8-intel-opt-openmp	254bd0	12s	0	19	2m 10s	0	50	1h 39m 6s	0	0	1891	6m 1s	50m 13s	Oct 22, 2018 - 04:21 UTC	(25 labels)	
hansen	Trilinos-atdm-hansen-shiller-intel-opt-openmp	254bd0	18s	0	19	3m 43s	0	50	1h 29m 6s	0	0	1890	6m 24s	1h 31m 37s	Oct 22, 2018 - 11:04 UTC	(25 labels)	
white	Trilinos-atdm-white-ride-gnu-opt-openmp	254bd0	12s	0	19	3m 34s	0	15	24m 50s	0	0	1890	6m 32s	1h 31m 27s	Oct 22, 2018 - 08:10 UTC	(25 labels)	
waterman	Trilinos-atdm-waterman-gnu-opt-openmp	254bd0	24s	0	19	3m 20s	0	15	16m 58s	0	0	1890	8m 8s	2h 17m 32s	Oct 22, 2018 - 04:24 UTC	(25 labels)	
hansen	Trilinos-atdm-hansen-shiller-gnu-opt-openmp	254bd0	12s	0	19	2m 53s	0	4	32m 56s	0	0	1890	8m 38s	1h 41m 4s	Oct 22, 2018 - 12:45 UTC	(25 labels)	
hansen	Trilinos-atdm-hansen-shiller-intel-debug-openmp	254bd0	12s	0	19	3m 44s	0	50	1h 33m 30s	0	0	1889	12m 18s	3h 8m 15s	Oct 22, 2018 - 10:48 UTC	(25 labels)	
serrano	Trilinos-atdm-serrano-intel-opt-openmp	254bd0	1m 18s	0	19	2m 27s	0	50	3h 24m 43s	0	0	1889	14m 20s	1h 47m 27s	Oct 22, 2018 - 04:25 UTC	(25 labels)	
chama	Trilinos-atdm-chama-intel-debug-openmp	254bd0	2m 30s	0	19	5m 24s	0	50	7h 20m 11s	0	0	1890	14m 35s	1h 49m 42s	Oct 22, 2018 - 04:33 UTC	(25 labels)	
chama	Trilinos-atdm-chama-intel-opt-openmp	254bd0	1m 48s	0	19	5m 19s	0	50	5h 38m 50s	0	0	1891	14m 53s	1h 51m 1s	Oct 22, 2018 - 04:38 UTC	(25 labels)	
serrano	Trilinos-atdm-serrano-intel-debug-openmp	254bd0	1m 12s	0	19	2m 31s	0	50	4h 12m 54s	0	0	1888	15m 3s	1h 52m 43s	Oct 22, 2018 - 04:30 UTC	(25 labels)	
mutrino	Trilinos-atdm-mutrino-intel-opt-openmp-KNL-panzer	254bd0	1m	0	1	4m 14s	0	50	1h 31m 21s	0	0	159	16m 18s	2h 8m 23s	Oct 22, 2018 - 14:29 UTC	Panzer	
white	Trilinos-atdm-white-ride-gnu-debug-openmp	254bd0	18s	0	19	2m 34s	0	14	19m 16s	0	0	1888	16m 56s	4h 11m 29s	Oct 22, 2018 - 08:30 UTC	(25 labels)	
sems-rhel8	Trilinos-atdm-sems-rhel8-gnu-debug-openmp	254bd0	18s	0	19	1m 26s	0	29	30m 33s	0	0	1890	17m 3s	2h 9m 50s	Oct 22, 2018 - 06:14 UTC	(25 labels)	
sems-rhel8	Trilinos-atdm-sems-rhel8-gnu-debug-serial	254bd0	6s	0	19	1m 19s	0	29	29m 3s	0	0	1780	19m 15s	2h 14m 10s	Oct 22, 2018 - 04:30 UTC	(25 labels)	
hansen	Trilinos-atdm-hansen-shiller-gnu-debug-openmp	254bd0	12s	0	19	3m 10s	0	4	32m 39s	0	0	1890	19m 32s	1h 17m 36s	Oct 22, 2018 - 09:50 UTC	(25 labels)	
hansen	Trilinos-atdm-hansen-shiller-gnu-opt-serial	254bd0	12s	0	19	2m 34s	0	4	34m 14s	0	0	1778	21m 19s	5h 15m 19s	Oct 22, 2018 - 07:24 UTC	(25 labels)	
hansen	Trilinos-atdm-hansen-shiller-gnu-debug-serial	254bd0	12s	0	19	2m 57s	0	4	35m 26s	0	0	1778	30m 25s	6h 49m 21s	Oct 22, 2018 - 06:13 UTC	(25 labels)	
white	Trilinos-atdm-white-ride-cuda-9.2-opt	254bd0	12s	0	20	4m 3s	0	50	1h 3m 10s	0	1	1896	33m 4s	4h 18m 17s	Oct 22, 2018 - 06:47 UTC	(25 labels)	
white	Trilinos-atdm-white-ride-cuda-9.2-debug	254bd0	12s	0	20	3m 51s	0	50	1h 30m 31s	0	1	1895	46m 37s	6h 7m 22s	Oct 22, 2018 - 06:24 UTC	(25 labels)	
waterman	Trilinos-atdm-waterman-cuda-9.2-opt	254bd0	12s	0	20	4m 4s	0	50	55m 3s	0	0	1900	49m 28s	6h 31m 26s	Oct 22, 2018 - 04:18 UTC	(25 labels)	
waterman	Trilinos-atdm-waterman-cuda-9.2-debug	254bd0	24s	0	20	4m 25s	0	50	1h 25m 15s	0	0	1897	1h 18s	7h 55m 54s	Oct 22, 2018 - 04:58 UTC	(25 labels)	
hansen	Trilinos-atdm-hansen-shiller-cuda-9.0-debug	254bd0	12s	0	20	3m 48s	0	50	2h 44m 48s	0	1	1898	1h 13m 16s	9h 33m 11s	Oct 22, 2018 - 11:42 UTC	(25 labels)	
hansen	Trilinos-atdm-hansen-shiller-cuda-9.0-opt	254bd0	12s	0	20	3m 39s	0	50	1h 59m 21s	0	1	1897	1h 25m 11s	11h 8m 47s	Oct 22, 2018 - 12:40 UTC	(25 labels)	
hansen	Trilinos-atdm-hansen-shiller-cuda-8.0-opt	254bd0	18s	0	20	3m 34s	0	50	2h 1m 4s	0	1	1897	1h 27m 40s	11h 27m 54s	Oct 22, 2018 - 06:15 UTC	(25 labels)	
hansen	Trilinos-atdm-hansen-shiller-cuda-8.0-debug	254bd0	18s	0	20	3m 52s	0	50	3h 10m 45s	0	1	1896	1h 32m 41s	12h 7m 18s	Oct 22, 2018 - 06:14 UTC	(25 labels)	
mutrino	Trilinos-atdm-mutrino-intel-opt-openmp-HSW	254bd0	1m 6s	0	19	6m 33s	0	50	5h 52m 25s	0	3	1888	2h 39m 16s	21h 8m 4s	Oct 22, 2018 - 05:43 UTC	(25 labels)	
Items per page [All]																	
Specialized																	12 builds
Site	Build Name	Update		Configure		Build		Test				Start Time	Labels				
		Revision	Time	Error	Warn	Time	Error	Warn	Time	Not Run	Fail			Pass			
mutrino	Trilinos-atdm-mutrino-intel-opt-openmp-KNL	254bd0	1m	0	19	6m 14s	0	50	5h 54m 31s	0	10	1891	3h 53m 45s	1 day 6h 59m 52s	Oct 22, 2018 - 05:41 UTC	(25 labels)	
ride	Trilinos-atdm-white-ride-cuda-9.2-debug-pt	254bd0	18s	0	10	5m 53s	2	50	2h 20m 23s	2	88	2717	1h 10m 17s	9h 19m 15s	Oct 22, 2018 - 04:23 UTC	(54 labels)	
white	Trilinos-atdm-white-ride-cuda-9.2-debug-pt	254bd0	18s	0	10	5m 44s	2	50	2h 15m 54s	2	88	2717	1h 10m	9h 18m 19s	Oct 22, 2018 - 06:24 UTC	(54 labels)	
waterman	Trilinos-atdm-waterman-cuda-9.2-release-debug	254bd0	24s	0	20	3m 50s	0	50	1h 22m 23s	0	1	1897	54m 30s	7h 11m 16s	Oct 22, 2018 - 06:09 UTC	(25 labels)	
ride	Trilinos-atdm-white-ride-cuda-9.2-debug	254bd0	12s	0	20	3m 54s	0	50	1h 31m 20s	0	1	1895	46m 44s	6h 8m 19s	Oct 22, 2018 - 04:54 UTC	(25 labels)	
ride	Trilinos-atdm-white-ride-cuda-9.2-opt	254bd0	12s	0	20	3m 36s	0	50	1h 3m 5s	0	1	1896	32m 43s	4h 16m 45s	Oct 22, 2018 - 04:22 UTC	(25 labels)	
ride	Trilinos-atdm-white-ride-gnu-debug-openmp	254bd0	12s	0	19	3m 1s	0	14	19m 36s	0	0	1888	16m 57s	4h 12m 42s	Oct 22, 2018 - 04:24 UTC	(25 labels)	
waterman	Trilinos-atdm-waterman-gnu-release-debug-openmp	254bd0	12s	0	19	3m 11s	0	21	21m 49s	0	1	1887	7m 35s	2h 8m 13s	Oct 22, 2018 - 07:29 UTC	(25 labels)	
cee-rhel8	Trilinos-atdm-cee-rhel8-intel-opt-serial	254bd0	6s	0	0	2m 9s	0	50	1h 4m 48s	0	2	2004	6m 56s	54m 50s	Oct 22, 2018 - 07:38 UTC	(27 labels)	
ride	Trilinos-atdm-white-ride-gnu-opt-openmp	254bd0	18s	0	19	2m 44s	0	15	24m 45s	0	0	1890	6m 27s	1h 31m 2s	Oct 22, 2018 - 04:17 UTC	(25 labels)	
cee-rhel8	Trilinos-atdm-cee-rhel8-clang-opt-serial	254bd0	12s	0	0	1m 26s	0	50	36m 23s	0	4	2002	5m 50s	44m 14s	Oct 22, 2018 - 06:10 UTC	(27 labels)	
cee-rhel8	Trilinos-atdm-cee-rhel8-gnu-opt-serial	254bd0	12s	0	0	1m 22s	0	4	35m 15s	0	2	2008	5m 48s	43m 52s	Oct 22, 2018 - 06:55 UTC	(27 labels)	
Items per page [All]																	

- All cleaned-up builds promoted to "ATDM" CDash group and maintained (30 as of 10/22/2018).
- Every Trilinos developer can see details on build and test failures.
- Easy to query about behavior of tests over multiple days, multiple builds, etc.
- Easy for Trilinos developers to reproduce failing builds and tests on any ATDM platform.
- Pull down results using CDash API for automated workflows.
- Python tool pulling data off CDash and daily summary email.

Where to Catch Trilinos Defects on ATDM Systems?

- **Trilinos package native test suite running in ATDM platform**
 - **Best place to catch a Trilinos defect!**
 - Trilinos developers can triage and fix a defect before APP/Trilinos Integrators need to dig in to triage APP failures caused by these defects.
- **APP (EMPIRE, SPARC) native test site running on ATDM platform**
 - Less than best place to catch a Trilinos defect.
 - Requires APP/Trilinos Integrator and APP Developers to triage problems and communicate back to Trilinos developers.
- **APP developer or user when running APP code**
 - **The worst place to catch a Trilinos defect!**
 - APP Customer has to report problems back to APP Developers who have to triage the failure and then report back to Trilinos developers.

Example:

- SEACAS update <https://github.com/trilinos/Trilinos/issues/2650>
 - Broke Trilinos/SEACAS CUDA test suite.
 - Did **NOT** break the EMPIRE test suite.
 - Broke usage of EMPIRE!

If update of Trilinos was gated by 100% passing SEACAS tests, then EMPIRE developers may have never seen these defects!

Injecting Failures vs. Fixing Failures

Injecting New Failures and Fixing Failures: A Race!

- **Mean-time to fail:** Average time (in days) for when a new failure shows up in ‘develop’ branch in one or more promoted ATDM Trilinos builds.
- **Mean-time to fix:** Average time (in days) to discover, triage and fix a failure on the Trilinos ‘develop’ branch in the promoted ATDM Trilinos builds.
- **The core problem:** If “mean-time to fail” is less than “mean-time to fix”, then the ATDM Trilinos builds on ‘develop’ on average will **ALWAYS be broken** (and therefore block updates of Trilinos to the APP customers)!

Mean-time to fix

<

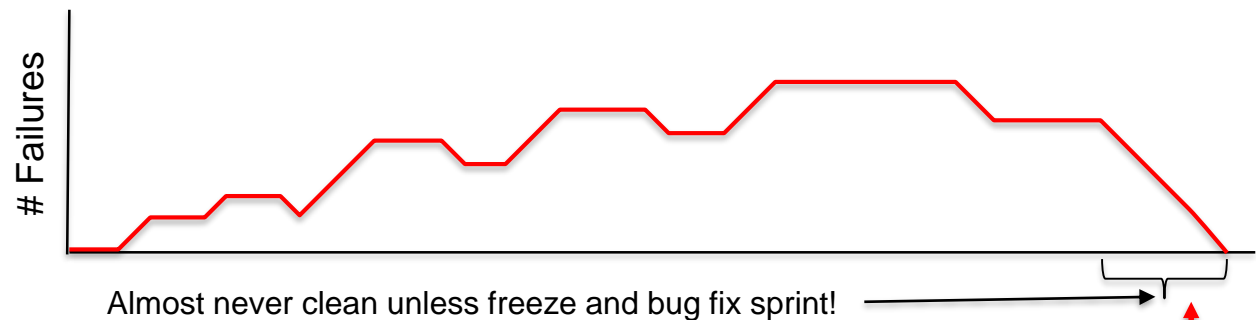
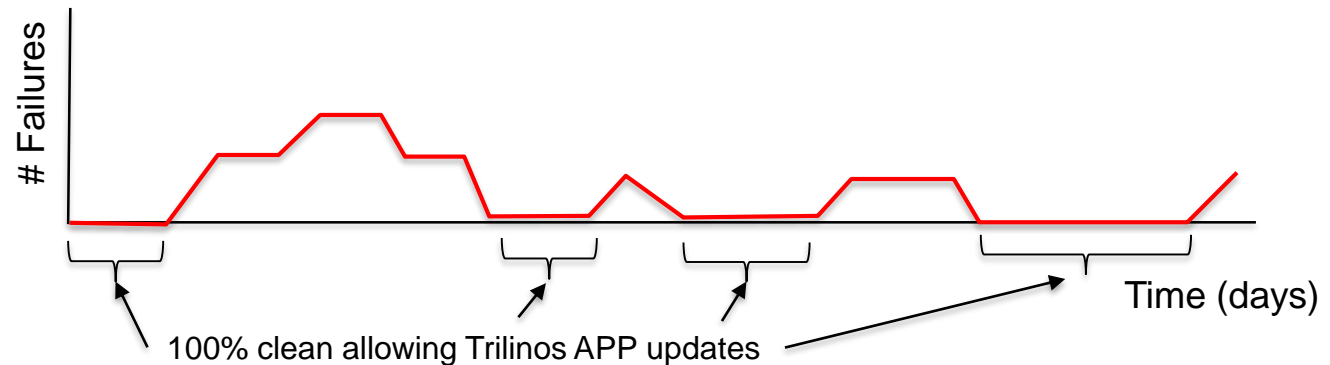
Mean-time to fail

Mean-time to fix

>

Mean-time to fail

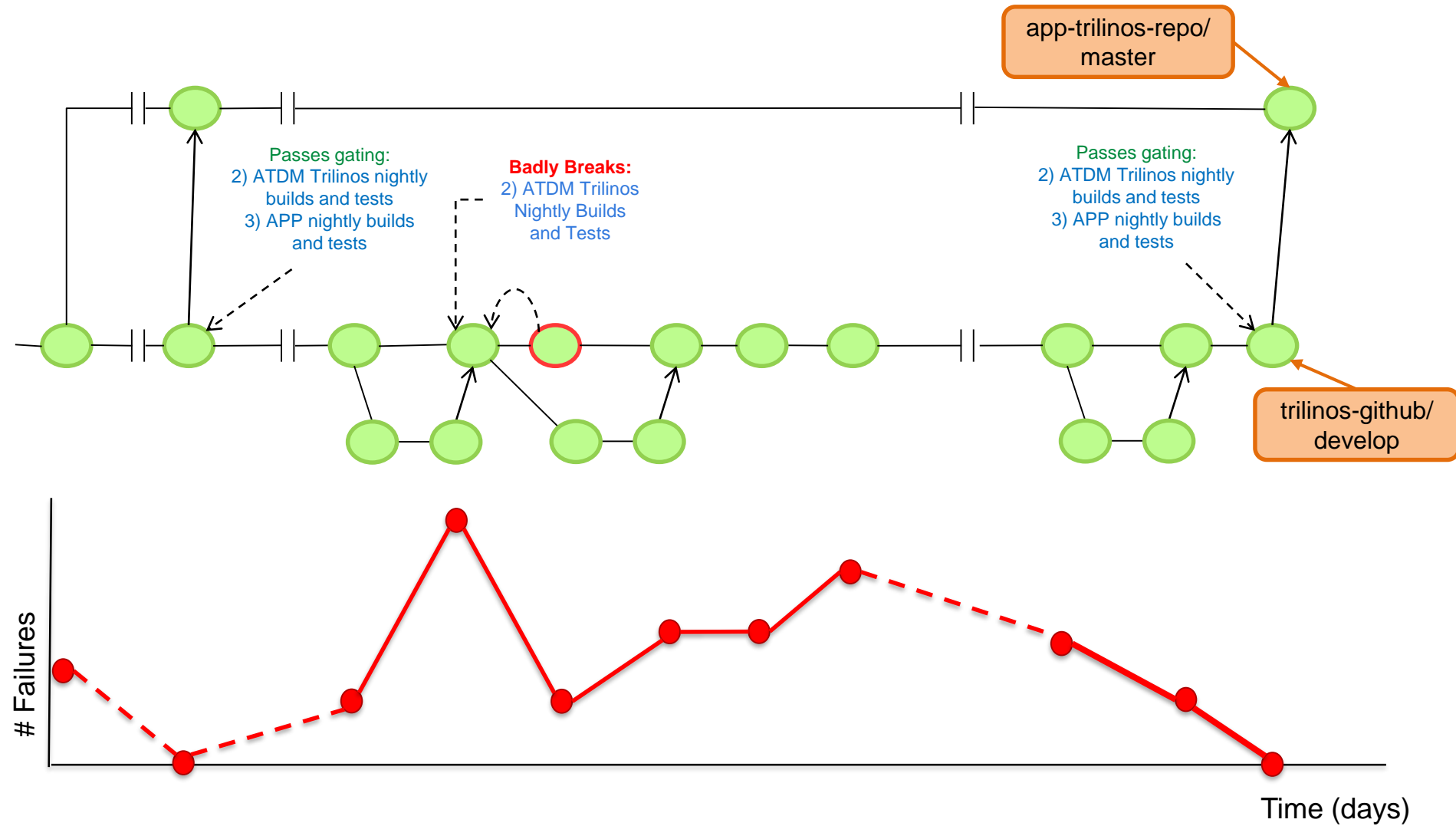
Promoted “ATDM”
Trilinos builds have been
continuously broken for
3+ months since
7/15/2018!



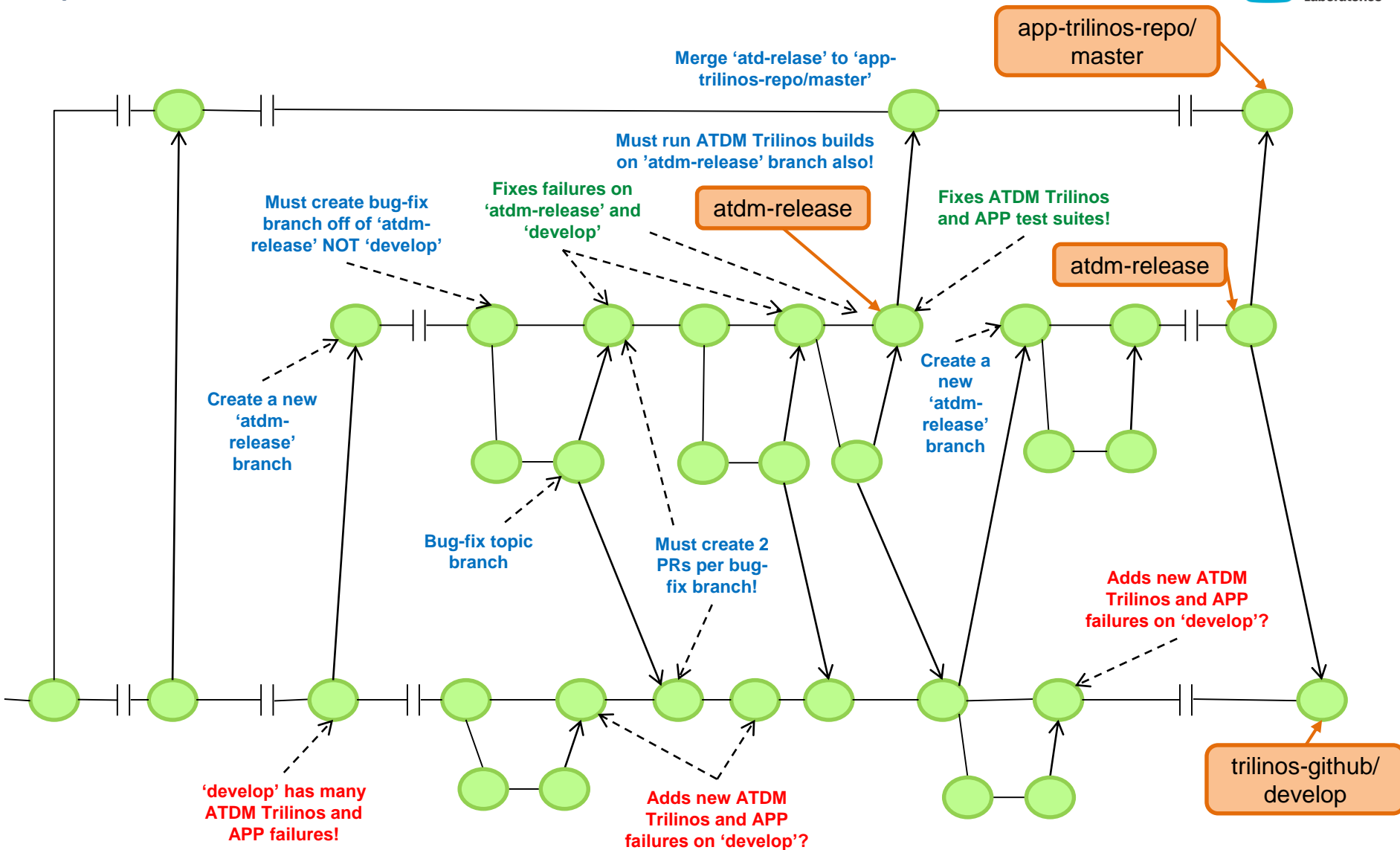
Freezing the ‘develop’ branch to fix failures is never going to happen!

- **Option-1: Make ATDM Trilinos builds clean on ‘develop’ periodically**
- **Option-2: Create ‘atdm-release’ branches and clean up there**

Option-1: Get clean ATDM Trilinos Builds on 'develop'

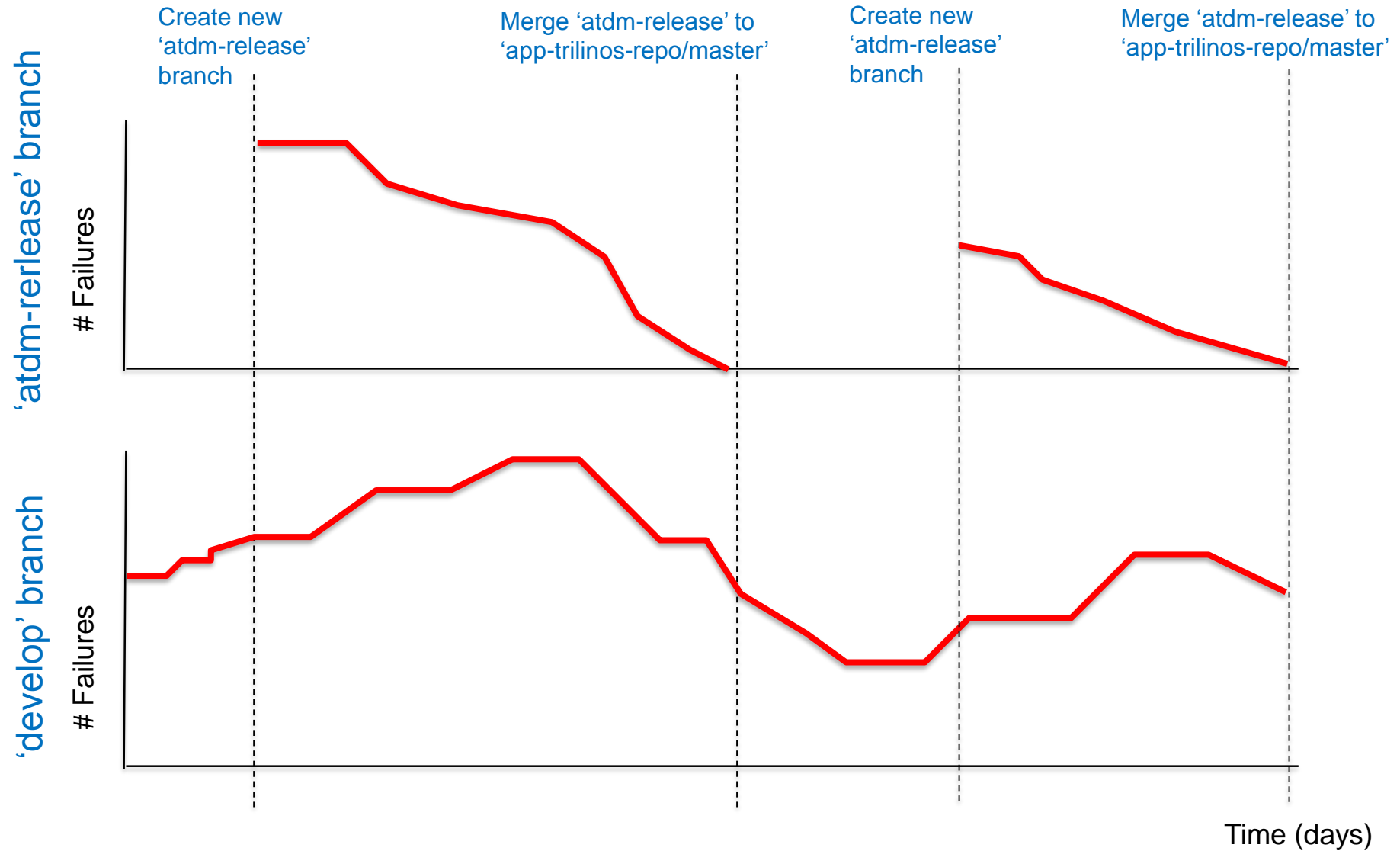


Option-2: Trilinos 'atdm-release' branches: Workflow



NOTE: Note this is really just an adaptation of the [gitworkflows\(7\)](#) release 'maint' branch!

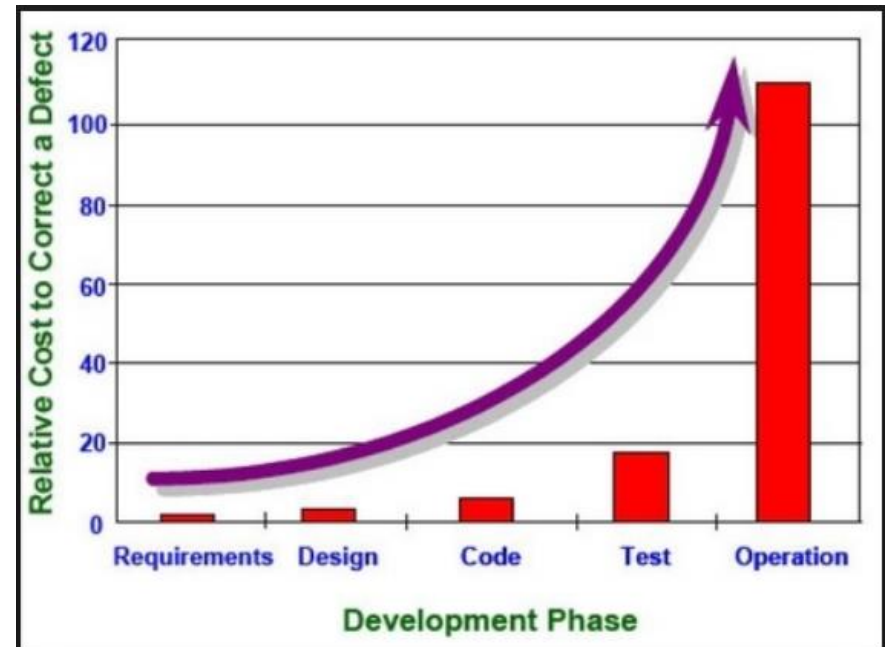
Option-2: Trilinos 'atdm-release' branches: Failures



- **Option-1: Make ATDM Trilinos builds clean on ‘develop’ periodically**
 - **Assumes:** “Mean-time to fix” is less than the “mean-time to fail” on ‘develop’ branch.
 - **Pro:** Requires **just one set of builds** on the ATDM platforms.
 - **Pro:** **Simpler workflow for Trilinos developers** merge bug fixes to ‘develop’ branch.
 - **Pro:** Provides **quicker APP updates of Trilinos**.
 - **Pro:** Allows APPs like EMPIRE to **co-develop Trilinos** and update Trilinos ‘develop’ and get updates to the APP fairly regularly.
 - **Con:** **Requires fast reaction time** to detect and triage new failures and then either a) fix, b) disable, or c) revert breaking PRs so that the “mean-time to fix” is less than “mean-time to fail”.
- **Option-2: Create ‘atdm-release’ branches and clean up there**
 - **Assumes:** “Mean-time to fix” is less than the “mean-time to failure” (not true right)
 - **Pro:** **More leisurely reaction time to fix defects** since no race with “mean-time to fail”.
 - **Pro:** Guaranteed periodic Trilinos updates with 100% clean ATDM Trilinos builds.
 - **Con:** Requires **double the number of builds**; one on ‘develop’, one on ‘atdm-release’
 - **Con:** **More complex workflow for Trilinos developers** to commit fixes to ‘atdm-release’ and then merge back to “develop” in two Trilinos PRs per bug-fix branch!
 - **Con:** **More complex workflow for APP Trilinos co-developers** involving branches, cherry-picks (e.g. EMPIRE git-git-like workflow and SPARC cherry-picking workflow).

General Software Engineering Principles for Defects

- **Cost of a defect goes up (significantly) the longer it takes to detect and correct a defect.**



- **Lean/Agile SE Practices for dealing with defects:**
 - Strong automated testing (have tests help new detect defects)
 - Continuous testing (reduce the time to detect new defects caught by tests)
 - Continuous integration (reduce time to detect conflict defects)
 - **STOP THE LINE** when a new defect gets into the main development branch
 - Fixing defects in previously working software is higher priority than developing new features!

Reducing Time to Detect, Triage, and Address Trilinos Failures

Reduce Time to Detect, Triage, and Fix Defects

- **Reduce time to detect and triage new Trilinos defects**
 - Run nightly ATDM Trilinos builds against 'develop' and run APP native tests against Trilinos 'develop' **[Much progress but more to do]**
 - Filter out non-code failures and create new ATDM Trilinos GitHub Issues with dedicated person(s) to do top-level triage. **[In Progress, Joe Frye]**
 - Python tool to keep track of new failures not already covered by Trilinos GitHub issues **[In Progress]**
 - Add categories according to severity (e.g. "critical", "blocker", "nonblocker") **[ToDo]**
- **Reduce time to fix defects**
 - Make it easy for Trilinos developers to reproduce failures for ATDM Trilinos builds **[Done]**
 - Send regular reminders to Trilinos Product Area Leads and assigned Trilinos developers about un-resolved ATDM Trilinos GitHub issues. **[ToDo]**

Detecting New Failures/Missing Results: CDash Email

jfrye@sandia.gov

FAILED (bme=1, twoi=1, twi=9): Promoted ATDM Trilinos Builds on 2018-10-24

7:18 AM

Build and Test results for Promoted ATDM Trilinos Builds on 2018-10-24

[Builds on CDash](#) (num=30)

[Nonpassing Tests on CDash](#) (num=10)

Missing expected builds: bme=1

Failing tests without issue tracker: twoi=1

Failing tests with issue tracker: twi=9

Failures in **red** require action!

- Missing test results!
- Failing test without issue tracker!

Missing expected builds: bme=1

Group	Site	Build Name	Missing Status
ATDM	chama	Trilinos-atdm-chama-intel-debug-openmp	Build exists but no test results

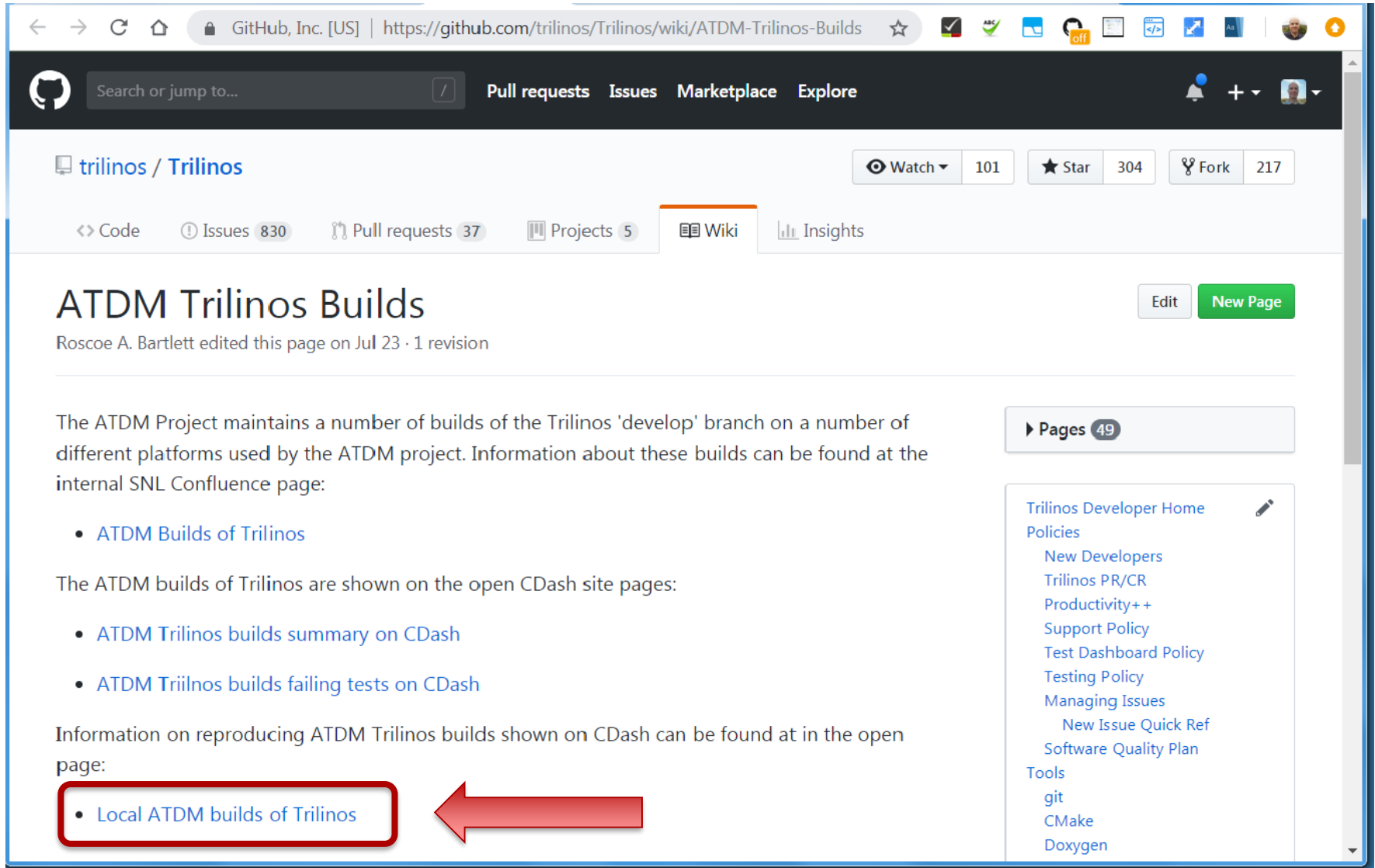
Failing tests without issue tracker (limited to 10): twoi=1

Site	Build Name	Test Name	Status	Details	# Fails last 30 Days	Previous Failure Date	Tracker
mutrino	Trilinos-atdm-mutrino-intel-opt-openmp-HSW	PanzerAdaptersSTK_tPointBasisValuesEvaluator_MPI_1	Failed	Completed (Failed)	1	None	

Failing tests with issue tracker: twi=9

Site	Build Name	Test Name	Status	Details	# Fails last 30 Days	Previous Failure Date	Tracker
mutrino	Trilinos-atdm-mutrino-intel-opt-openmp-HSW	Anasazi_Epetra_BKS_norestart_test_MPI_4	Failed	Completed (Failed)	30	2018-10-23	#3499
mutrino	Trilinos-atdm-mutrino-intel-opt-openmp-HSW	Anasazi_MultiVecTraitsTest2_MPI_4	Failed	Completed (Failed)	30	2018-10-23	#3499
mutrino	Trilinos-atdm-mutrino-intel-opt-openmp-HSW	Belos_gcrodr_hb_MPI_4	Failed	Completed (Failed)	30	2018-10-23	#3497
hansen	Trilinos-atdm-hansen-shiller-cuda-8.0-debug	PanzerAdaptersSTK_CurlLaplacianExample-ConvTest-Quad-Order-4	Failed	Completed (Timeout)	14	2018-10-23	#3579
hansen	Trilinos-atdm-hansen-shiller-cuda-8.0-opt	PanzerAdaptersSTK_CurlLaplacianExample-ConvTest-Quad-Order-4	Failed	Completed (Timeout)	14	2018-10-23	#3579
hansen	Trilinos-atdm-hansen-shiller-cuda-9.0-debug	PanzerAdaptersSTK_CurlLaplacianExample-ConvTest-Quad-Order-4	Failed	Completed (Timeout)	13	2018-10-23	#3579
hansen	Trilinos-atdm-hansen-shiller-cuda-9.0-opt	PanzerAdaptersSTK_CurlLaplacianExample-ConvTest-Quad-Order-4	Failed	Completed (Timeout)	14	2018-10-23	#3579

Reproducing ATDM Trilinos Builds: Trilinos Wiki



The screenshot shows the GitHub Trilinos Wiki page for "ATDM Trilinos Builds". The page header includes the GitHub logo, a search bar, and navigation links for Pull requests, Issues, Marketplace, and Explore. The repository name "trilinos / Trilinos" is displayed, along with statistics for Watch (101), Star (304), and Fork (217). The page tabs show Code, Issues (830), Pull requests (37), Projects (5), Wiki (selected), and Insights.

ATDM Trilinos Builds

Roscoe A. Bartlett edited this page on Jul 23 · 1 revision

The ATDM Project maintains a number of builds of the Trilinos 'develop' branch on a number of different platforms used by the ATDM project. Information about these builds can be found at the internal SNL Confluence page:

- [ATDM Builds of Trilinos](#)

The ATDM builds of Trilinos are shown on the open CDash site pages:

- [ATDM Trilinos builds summary on CDash](#)
- [ATDM Trilinos builds failing tests on CDash](#)

Information on reproducing ATDM Trilinos builds shown on CDash can be found at in the open page:

- [Local ATDM builds of Trilinos](#)

A red box highlights the link "Local ATDM builds of Trilinos", and a large red arrow points to it from the right.

On the right side of the page, there is a sidebar with a "Pages" section showing 49 pages. The "Trilinos Developer Home" section includes links to Policies, New Developers, Trilinos PR/CR, Productivity++, Support Policy, Test Dashboard Policy, Testing Policy, Managing Issues, New Issue Quick Ref, and Software Quality Plan. The "Tools" section includes links to git, CMake, and Doxygen.

Reproducing ATDM Trilinos Builds: README.md

GitHub, Inc. [US] | <https://github.com/trilinos/Trilinos/blob/develop/cmake/std/atdm/README.md#quick-start>



Quick-start

After [cloning the Trilinos git repo](#) on one of the supported ATDM machines, a local configure of Trilinos enabling a few packages is performed as:

```
$ cd <some_build_dir>/

$ source $TRILINOS_DIR/cmake/std/atdm/load-env.sh <job-name>

$ cmake \
  -GNinja \
  -DTrilinos_CONFIGURE_OPTIONS_FILE:STRING=cmake/std/atdm/ATDMDevEnv.cmake \
  -DTrilinos_ENABLE_TESTS=ON -DTrilinos_ENABLE_<Package1>=ON \
  $TRILINOS_DIR

$ make NP=16 # Uses ninja -j16

$ ctest -j16 # Might need to be run with srun or some other command, see below
```

The command:

```
$ source $TRILINOS_DIR/cmake/std/atdm/load-env.sh <job-name>
```

determines what machine you are on (using `hostname`) and then loads the correct environment automatically for that machine and for the build options passed through in `<job-name>` (or errors out if the current machine is not one of the supported machines).

The `<job-name>` argument is a single string of the form `XXX-<keyword0>-<keyword1>-...`. The standard order and format of this string is:

Specific instructions for each system

- [ride/white](#)
- [shiller/hansen](#)
- [chama/serrano](#)
- [mutrino](#)
- [SEMS rhel6 environment](#)
- [CEE rhel6 environment](#)
- [waterman](#)

ride/white

Once logged on to `white` (on the SON) or `ride` (on the SRN), one can directly configure and build on the login node (being careful not to overload the node). But to run the tests, one must run on the compute nodes using the `bsub` command to run if using a CUDA build. For example, to configure, build and run the tests for the `cuda-debug` build for say `MueLu` on `white`, (after cloning Trilinos on the `develop` branch) one would do:

```
$ cd <some_build_dir>/

$ source $TRILINOS_DIR/cmake/std/atdm/load-env.sh cuda-debug

$ cmake \
  -GNinja \
  -DTrilinos_CONFIGURE_OPTIONS_FILE:STRING=cmake/std/atdm/ATDMDevEnv.cmake \
  -DTrilinos_ENABLE_TESTS=ON -DTrilinos_ENABLE_MueLu=ON \
  $TRILINOS_DIR

$ make NP=16

$ bsub -x -Is -q rhel7F -n 16 ctest -j16
```

Reproducing ATDM Trilinos Builds: GitHub Issue

GitHub, Inc. [US] | <https://github.com/trilinos/Trilinos/issues/3681>

Steps to Reproduce

One should be able to reproduce this failure on waterman as described in:

- <https://github.com/trilinos/Trilinos/blob/develop/cmake/std/atdm/README.md>
More specifically, the commands given for waterman are provided at:
- <https://github.com/trilinos/Trilinos/blob/develop/cmake/std/atdm/README.md#waterman>
The exact commands to reproduce this issue should be:

```
$ cd <some_build_dir>/


$ source $TRILINOS_DIR/cmake/std/atdm/load-env.sh cuda-9.2-release-debug

$ cmake \
  -GNinja \
  -DTrilinos_CONFIGURE_OPTIONS_FILE:STRING=cmake/std/atdm/ATDMDevEnv.cmake \
  -DTrilinos_ENABLE_TESTS=ON -DTrilinos_ENABLE_Intrepid2=ON \
  $TRILINOS_DIR

$ make NP=20

$ bsub -x -Is -n 20 ctest -j20
```



 fryeguy52 added **bug** **Intrepid2** **ATDM** labels 4 days ago

Addressing Trilinos Failures

How to Address Trilinos Failures?

- **Keeping already cleaned-up promoted builds clean**
 - a) Fix the failures => **Best option!**
 - b) Mark failing tests as “expected may fail” and not trigger global failure in Python tool:
 - Only for non-blocking issues
 - Allows us to watch test run but not block updates of Trilinos to APPs
 - **Best for cases where someone is working (or soon is going to work) to fix non-blocking failures.**
 - c) (Temporarily) disable failing tests:
 - Only for non-blocking issues
 - **Best for cases where no-one is going to work on fixing the failures anytime soon.**
 - d) Revert the commit(s) (or PR merge) causing the failure:
 - **=> Perhaps best option for critical or blocking failures that can't be fixed soon!**
- **Initial failures from setting up new platforms**
 - a) Fix the failures
 - b) (Temporarily) disable failing tests (non-blocking issues only)
 - c) Mark failing tests as “expected may fail” and not trigger global failure in Python tool (non-blocking issues only)
 - **NOTE: Reverting commits is NOT an option for cleaning up failures that occur when setting up new builds on new platforms or envs on existing platforms.**

Following up on ATDM Trilinos GitHub Issues

- **Problems not working issues:**

- Ignoring new ATDM Trilinos GitHub Issues (e.g. no replies for weeks)
- Not following up on Issues after some initial investigation

- **Problems closing issues:**

- Closing Issues before getting confirmation on CDash (sometimes issue is ***not*** addressed)
- Not closing issues after issue has been addressed (clutter up list of active issues)

Proposed Solution => Send out weekly email listing

- Send out emails to Trilinos Product Area Leads for open ATDM Issues in their area
- Send out emails to assignees of open ATDM Trilinos issues?

(7) ATDM Trilinos Linear Solvers Product Area GitHub Issues on 2018-10-24

#Issue	ATDM Priority	Created	Last Updated	# Comments
Summary				
Next Action Status				
#3499	Blocker	2018-09-25	2018-10-03	5
Anasazi tests failing in ATDM build on mutrino Likely broken by PR #3481 merged to 'develop' on 9/21/2018 ...				
#3686	Blocker	2018-10-16	2018-10-16	0
Teko_ModALPreconditioner_MPI_1 Failing in ATDM cee-rhel6-clang-opt-serial build ???				
...				

- Python tool that pulls data off GitHub (query labels 'ATDM', 'TPA: Linear Solvers', ...)
- **To Who?** Send only to Trilinos Product Area Leads? Send email also to Issue assignees?
- **Frequency?** Once a week? Twice a week?

Wrapping Up

Observations and Open Questions

- **SNL ATDM is test-bed for components approach for exascale HPC software!**
- **Two approaches being compared:**
 - *Full buy-in to using and co-developing components:* **EMPIRE**
 - *Avoid deep dependencies or co-development of components:* **SPARC**
- **The role of Trilinos in ATDM will either be viewed as:**
 - **A) Successful:**
 - => Encouraging the usage of components approaches
 - => Leading to more future funding for Trilinos?
 - **B) Unsuccessful:**
 - => Discouraging the usage of components approaches
 - => Leading to less future funding for Trilinos?
- **Questions:**
 - At the end of ATDM, which approach will be viewed more successful?
 - What will that say about Trilinos?
 - What will this say about the future of components in CSE/HPC exascale?
 - **If we can't succeed with components with Trilinos in ATDM at SNL, how we expect this to work in the larger ECP project across labs, universities, etc.?**

Let's stabilize Trilinos for ATDM APPs and remove that as an excuse!

- **Trilinos / APP Git Workflows:**

- Trilinos Pull Request (PR) testing & merging to 'develop' [**Done**]
- SPARC / Trilinos subteam workflow (manual testing by Micah H.) [**Done**]
- EMPIRE / Trilinos git.git workflow (EMPIRE-owned Jenkins pipeline for testing) [**Done**]

- **Testing gates for workflows:**

- 1) Auto PR Trilinos builds and tests :
 - **Current:** MPI GCC 4.8.4, GCC 4.9.3, Intel 17
 - **Future:** CUDA (see [Trilinos GitHub #2646](#))
- 2) ATDM Trilinos nightly builds and tests:
 - ATDM Trilinos builds for EMPIRE ... EMPIRE switchover in-progress/complete?
 - ATDM Trilinos builds for SPARC ... 'cee-rhel6' gnu, intel, and clang complete
- 3) APP nightly builds and tests
 - EMPIRE-PIC and EMPIRE-Fluid build and test suite: A few builds posting to Jenkins only
 - SPARC build and test suite: Submits to Sierra CDash site
- **Triaging and fixing failed builds and tests:**
 - **Notification of new failures:** Python email tool pulling data for ATDM Trilinos builds off [CDash site](#).
 - **Triage failures:** Filter out non-code failures then create [Trilinos GitHub Issues](#)
 - Joe Frye creates initial GitHub Issues, Product Areas Leads follow up from there
 - **Address failures :**
 - New builds: a) fix, b) allow to fail, c) temporally disabling non-critical tests
 - Existing builds: a) fix, b) allow to fail, c) temporally disable, or d) reverting PR from 'develop'
 - **Manage & Follow-up:** Someone must observe and ensure failures are addressed (???Who???)

THE END