

The application of TriBITS to the Software Development and Integration Processes of Larger Componentized Multi-Organization Scientific and Engineering Software Projects

Roscoe A. Bartlett

Multiphysics Applications (org. 1446), Sandia National Labs

CASL Physics Integration Infrastructure Lead

Trilinos Software Engineering Technologies and Integration Lead

ATDM Software Tools and Environment Lead

Sandia Computational Science Seminar Series

June 9, 2016



Overview of CASL VERA Development

Overview of CASL



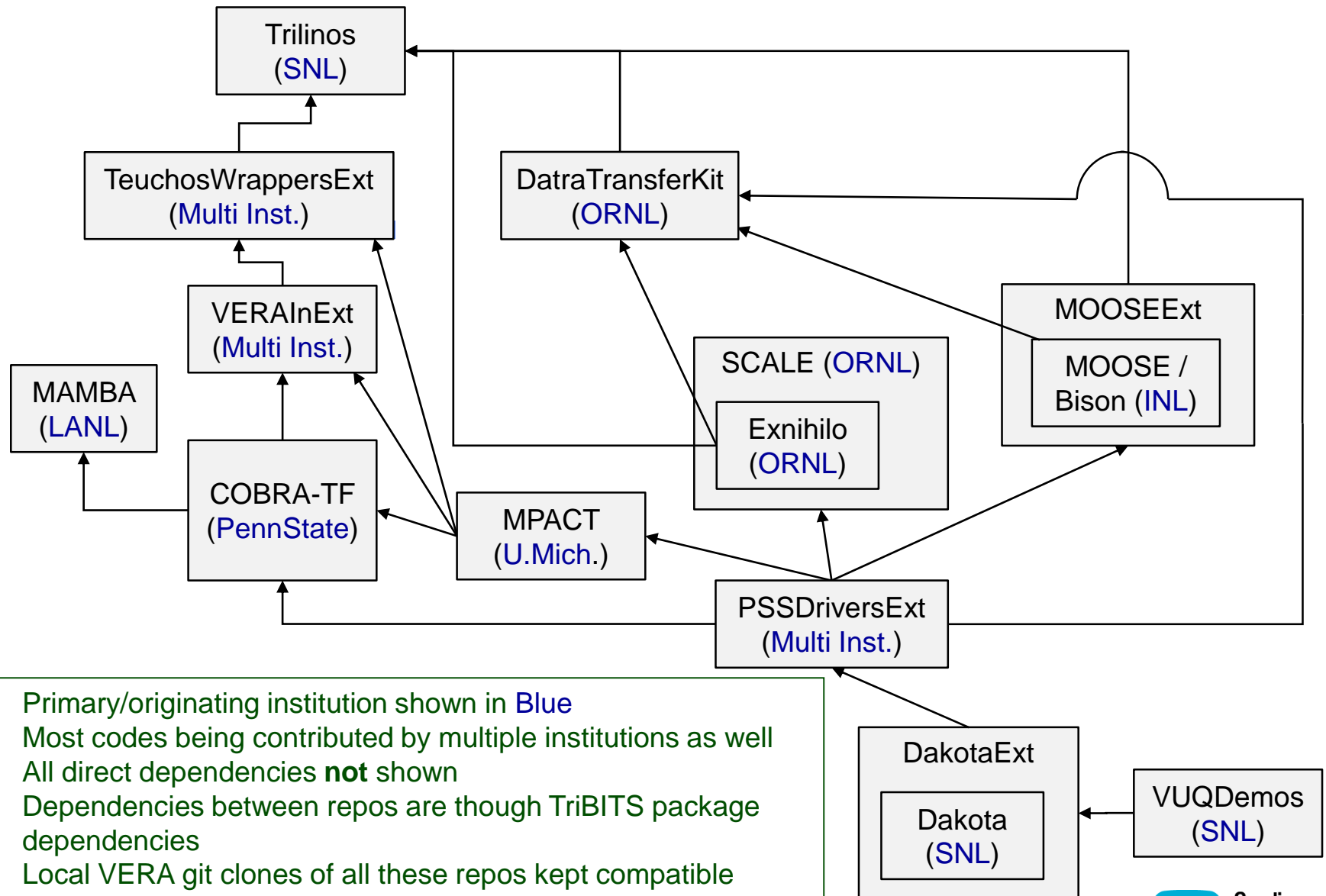
- **CASL:** Consortium for the **A**dvanced **S**imulation of **L**ightwater reactors
- DOE Innovation Hub including DOE labs, universities, and industry partners
- Goals:
 - Advance modeling and simulation of lightwater nuclear reactors
 - Produce a set of simulation tools to model lightwater nuclear reactor cores to provide to the nuclear industry: **VERA: Virtual Environment for Reactor Applications.**
- Phase 1: July 2010 – June 2015
- Phase 2: July 2015 – June 2020
- Organization and management:
 - ORNL is the hub of the Hub
 - Milestone driven (6 month plan-of-records (PoRs))
 - Focus areas: **Physics Integration (PHI)**, Thermal Hydraulic Methods (THM), Radiation Transport Methods (RTM), Advanced Modeling Applications (AMA), Materials Performance and Optimization (MPO), Validation and Uncertainty Quantification (VUQ)



CASL VERA Development Overview

- VERA development is complicated in almost every way ☹
- VERA Currently Composed of:
 - 21 different git repositories on casl-dev.ornl.gov (clones of other repos) most with a different access list (NDAs, Export Control, IP, etc.)
 - Integrating development efforts from many teams from 9+ institutions
- Large single CMake build system using TriBITS CMake Framework:
 - Very large full source code base:
 - 55K source and script files
 - 12M lines of code (not comments)
 - 2,700 CMakeLists.txt files
 - 229 packages + subpackages enabled (out of 496 total) ≈ **46% of full code base**
 - Most CMake developer reconfigures take place in less than 30 seconds!
- VERA Software Development Process:
 - VERA integration maintained by continuous and nightly testing:
 - Pre-push CI testing: checkin-test-vera.sh, cloned VERA git repos
 - Post-push CI testing: CTest/CDash, all VERA git repos
 - Nightly testing: MPI and Serial builds, Debug and Release builds, ...
 - Main 100% passing builds and tests most days!
 - Many internal and external repository integrations on daily basis
 - VERA releases are taken off of stable 'master' branches on casl-dev git repos.
 - Very low maintenance cost of the infrastructure

Dependencies Between Selected VERA Repositories





Why CMake?

Why TriBITS?



Why CMake?

Open-source tools maintained and used by a large community and supported by a professional software development company (Kitware).

CMake:

- Simplified build system, easier maintenance
- Improved mechanism for extending capabilities (CMake language)
- Support for all major C, C++, and Fortran compilers.
- Automatic full dependency tracking (headers, src, mod, obj, libs, exec)
- Good Fortran support (parallel builds with modules with src => mod => object tracking, C/Fortran interoperability, etc.)
- Shared libraries on all platforms and compilers (support for RPATH)
- Faster configure times (e.g. > 10x faster than autotools)
- Generates different backend builds: Makefiles, **Ninja**, Visual Studio, Eclipse, XCode, ...
- Portable support for cross-compiling

CTest:

- Parallel running and scheduling of tests and test time-outs
- Memory testing (Valgrind)
- Line coverage testing (GCC GCOV)
- Better integration between the test system and the build system

CDash:

- Web server for display and archive of build, test, memory, and coverage results
- Flexible query and filtering of build and test results
- Automated failure email notifications



Componentized CMake-based Projects Approaches

CMake, CTest, and CDash are great, **but raw usage does not scale very well to large projects and multiple repositories and teams!**

- **Multiple CMake projects:**

- Manual builds and linking through <Package>Config.cmake files (e.g. Albany & Trilinos)
- CMake ExternalProject:
 - + Provided as standard CMake module ExternalProject.cmake
 - + Allows non-CMake subprojects
 - Does not do full dependency tracking between component CMake projects
- Google Catkin (used for the Google Robotics Operating System (ROS) project)
 - + Automatic dependency handling logic (implemented in Python)
 - + Parallel builds of CMake (sub)projects and posting to CDash
 - Requires Python for basic usage
- CMakeBasis
 - + Standardize creation and usage of <Package>Config.cmake files
 - Core functionality requires Python

- **Single CMake project:**

- Kitware VTK Modules:
 - Does not support optional dependencies
 - Slow due to need to sort submodules based on dependencies
- TriBITS:
 - + Support multiple repos
 - + Core functionality depends only on CMake 2.8.11+



Why TriBITS?

TriBITS: **Tri**bal (or **Tri**linos) **B**uild, **I**ntegration, and **T**est **S**ystem

- Framework for large, distributed multi-repository CMake projects
- Reduce boiler-plate CMake code and enforce consistency across large distributed projects
- Subproject dependencies and namespacing architecture (packages)
- Automatic package dependency handling (directed acyclic graph)
- Additional functionality missing in raw CMake
- Change default CMake behavior when necessary
- Additional tools for agile software development processes (e.g. Continuous Integration (CI))

History of TriBITS:

- 2007: Initially developed as a CMake package architecture for Trilinos
- 2011: Generalized and extended for CASL VERA
- 2014: Source code hosted on GitHub



Raw CMake vs. TriBITS

Example Raw CMakeLists.txt File

Build and install library

```
set(HEADERS hello_world_lib.hpp)
set(SOURCES hello_world_lib.cpp)
add_library(hello_world_lib ${SOURCES})
install(TARGETS hello_world_lib DESTINATION lib)
install(FILES ${HEADERS} DESTINATION include)
```

Build and install user executable

```
add_executable(hello_world hello_world_main.cpp)
target_link_libraries(hello_world hello_world_lib)
install(TARGETS hello_world DESTINATION bin)
```

Test the executable

```
add_test(hello_world ${CMAKE_CURRENT_BINARY_DIR}/hello_world)
set_tests_properties(hello_world PROPERTIES PASS_REGULAR_EXPRESSION "Hello World")
```

Build and run some unit tests

```
add_executable(unit_tests hello_world_unit_tests.cpp)
target_link_libraries(unit_tests hello_world_lib)
add_test(unit_test ${CMAKE_CURRENT_BINARY_DIR}/unit_tests)
set_tests_properties(unit_test PROPERTIES PASS_REGULAR_EXPRESSION "All unit tests passed")
```

**Executable and
test names must
be globally
unique!**

Example TriBITS Package CMakeList.txt File

```
tribits_package(HelloWorld)
tribits_add_library(hello_world_lib HEADERS hello_world_lib.hpp SOURCES hello_world_lib.cpp)
tribits_add_executable(hello_world NOEXEPREFIX SOURCES hello_world_main.cpp INSTALLABLE)
tribits_add_test(hello_world NOEXEPREFIX PASS_REGULAR_EXPRESSION "Hello World")
tribits_add_executablea_and_test(unit_tests SOURCES hello_world_unit_tests.cpp
    PASS_REGULAR_EXPRESSION "All unit tests passed")
tribits_package_postprocess()
```

- Less duplication and boiler-plate code
- Fewer commands
- **Build command wrappers:**
 - Install by default (most common)
 - Optionally Install libraries and headers or just executables?
 - Optional global prefixing of libraries
 - And more ...
- **CTest command wrappers:**
 - Automatic namespacing of tests and test executables
 - Classification of tests (BASIC, CONTINUOUS, NIGHTLY, ...)
 - Uniform handling of timeouts (and scaling of timeouts)
 - And more ...

Maintain consistency and add/change behavior across different independent repositories and packages and 1000s of CMakeLists.txt files!



TriBITS Structural Units

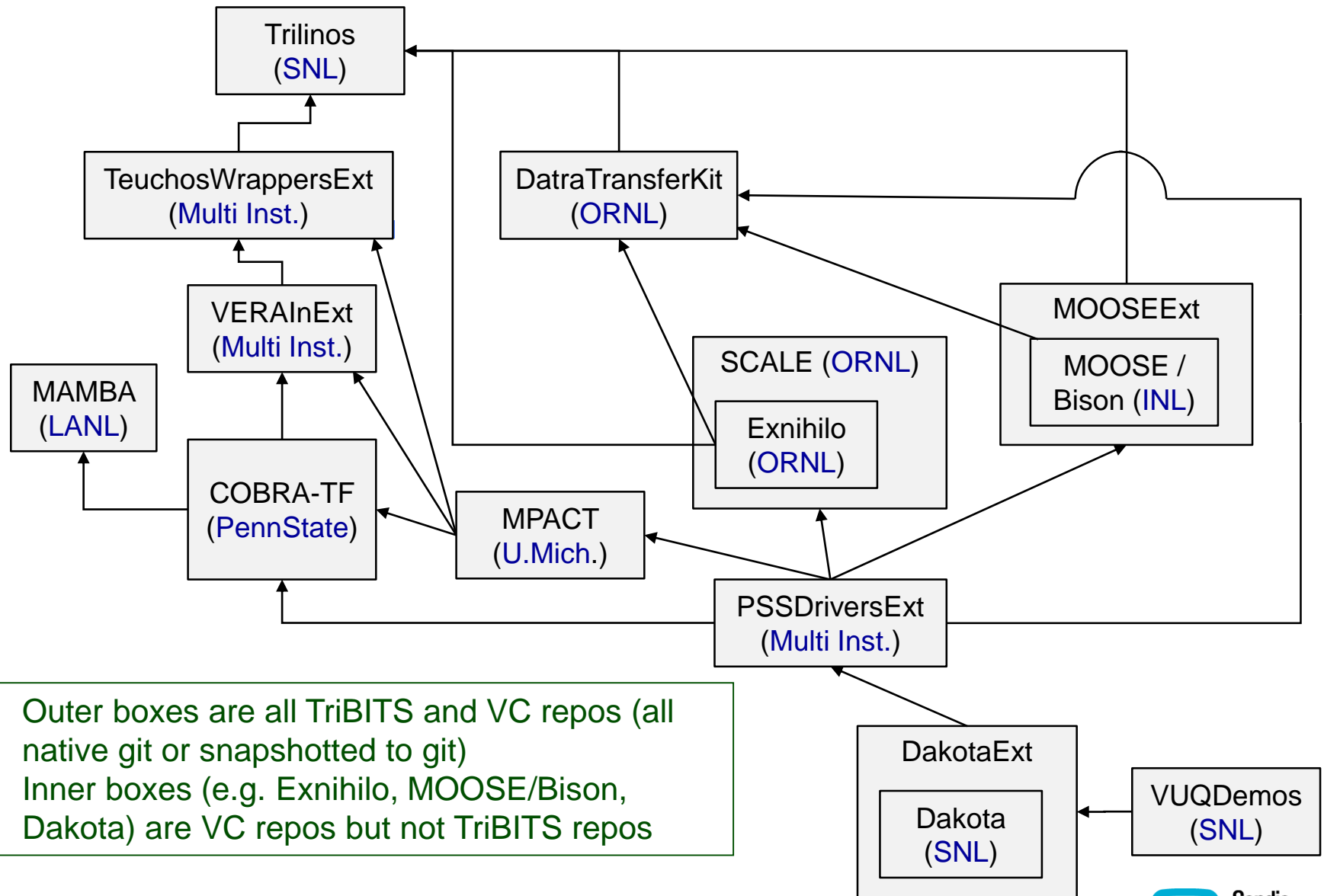
TriBITS Structural Units

- **TriBITS Project:**
 - Complete CMake “Project”
 - Overall projects settings
- **TriBITS Repository:**
 - Collection of **Packages** and **TPLs**
 - Unit of distribution and integration
 - Typically a version control (git) repository
- **TriBITS Package:**
 - Encapsulated collection of related software & tests
 - Unit of testing, namespacing, documentation, and reuse
 - Lists dependencies on **SE Packages** & **TPLs**
- **TriBITS Subpackage:**
 - Optional partitioning of package software & tests
 - Primarily for dependency management (SE principles)
- **TriBITS TPLs (Third Party Libraries):**
 - Specification of external dependencies (libs)
 - Required or optional dependency
 - Single definition across all packages
 - Can use native CMake Find<Package>.cmake modules

**Packages
+ Subpackages
=
Software
Engineering (SE)
Packages**

See: <https://tribits.org/doc/TribitsDevelopersGuide.html>

TriBITS and VC Repos for CASL VERA



- Outer boxes are all TriBITS and VC repos (all native git or snapshotted to git)
- Inner boxes (e.g. Exnihilo, MOOSE/Bison, Dakota) are VC repos but not TriBITS repos



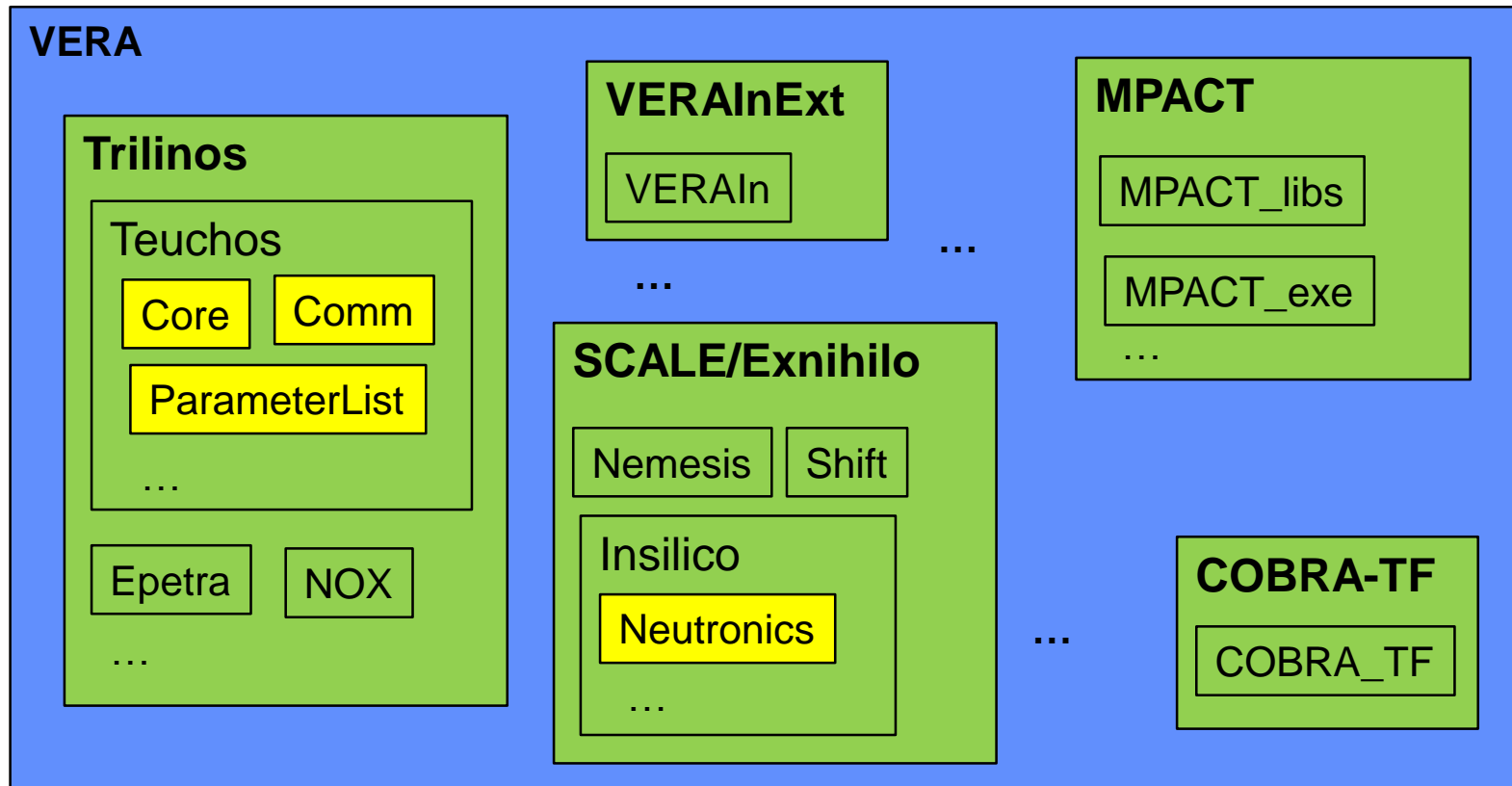
VERA/cmake/ExtraRepositoriesList.cmake

```
tribits_project_define_extra_repositories(  
  TriBITS          ""      GIT  git@casl-dev:TriBITS          ""      ${TriBITS_REPO_TYPE}  
  Trilinos          ""      GIT  git@casl-dev:Trilinos          ""      Continuous  
  TeuchosWrappersExt ""      GIT  git@casl-dev:TeuchosWrappersExt ""      Continuous  
  MAMBA             ""      GIT  git@casl-dev:MAMBA             ""      Continuous  
  Cicada            ""      GIT  git@casl-dev:Cicada            ""      Continuous  
  COBRA-TF          ""      GIT  git@casl-dev:COBRA-TF          ""      Continuous  
  VERAIInExt         ""      GIT  git@casl-dev:VERAIInExt         ""      Continuous  
  VERADData          ""      GIT  git@casl-dev:VERADData          NOPACKAGES  ${VERADATA_CAT}  
  DataTransferKit    ""      GIT  git@casl-dev:DataTransferKit ""      Continuous  
  MOOSEExt           ""      GIT  git@casl-dev:MOOSEExt           ""      Continuous  
  MOOSE              MOOSEExt/MOOSE  GIT  
    git@casl-dev:MOOSE  NOPACKAGES  Continuous  
  SCALE             ""      GIT  git@casl-dev:SCALE             ""      Continuous  
  Exnihilo           SCALE/Exnihilo  GIT  
    git@casl-dev:Exnihilo          NOPACKAGES  Continuous  
  XSTools            ""      GIT  git@casl-dev:XSTools            ""      Nightly  
  MPACT              ""      GIT  git@casl-dev:MPACT              ""      Continuous  
  PSSDriversExt      ""      GIT  git@casl-dev:PSSDriversExt      ""      Continuous  
  DakotaExt          ""      GIT  git@casl-dev:DakotaExt          ""      Continuous  
  Dakota  DakotaExt/Dakota  GIT  git@casl-dev:Dakota  NOPACKAGES  Continuous  
  VUQDemos           ""      GIT  git@casl-dev:VUQDemos           ""      Nightly  
  VERAView           ""      GIT  git@casl-dev:VERAView           ""      Continuous  
)
```

Official version of VERA in on master branch used for CI & Nightly testing

- Partial set of repos can be cloned (protected by different groups)
- Non-git repos are converted into git repos: **Dakota (svn)**, **SCALE (hg)**, **MOOSE (git submodules)**

VERA Meta-Project, Repositories, Packages & Subpackages



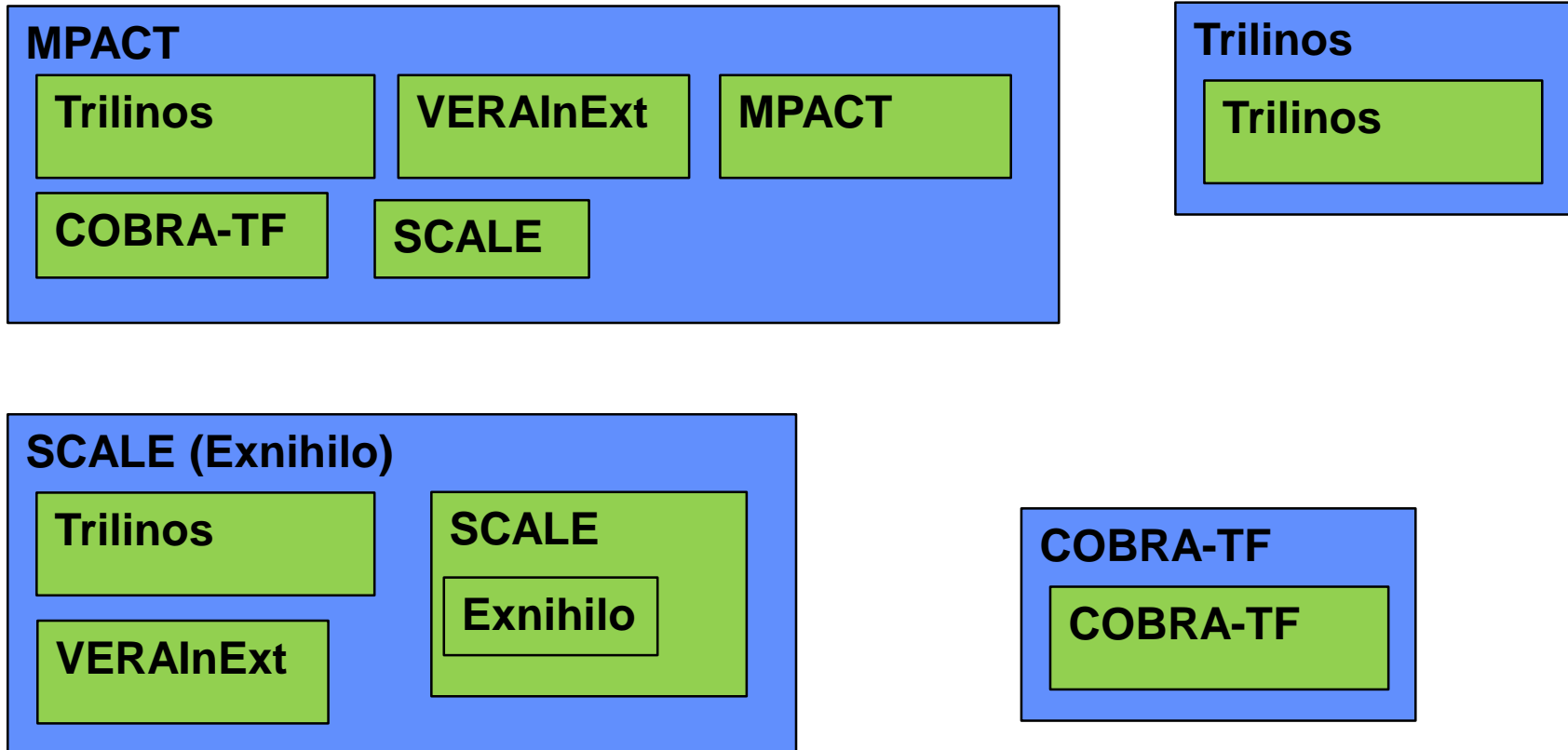
- **VERA meta-project (1):** Git repository and TriBITS meta-project (contains no packages)
- **TriBITS repos (17):** Trilinos, VERAInExt, COBRA-TF, MPACT, SCALE ...
- **TriBITS packages (93 enabled out of 213 total):** Teuchos, Epetra, VERAIn, Insilico, COBRA_TF, MPACT_exe, ...
- **TriBITS subpackages:** TeuchosCore, InsilicoNeutronics ...
- **TriBITS SE packages (229 enabled out of 496 total):** TeuchosCore, Teuchos, VERAIn, Insilico, InsilicNeutronics, ...



Current Adoption and Usage of TriBITS in CASL

- Repositories of independent projects using **TriBITS** primary build/test system:
 - **Trilinos**: SNL
 - **SCALE**: ORNL
 - Requires GCC 4.6.1+ and Intel 13.1+
 - Mixed Fortran, C, C++
 - Linux builds
 - Windows builds
 - **Exnihilo**: ORNL
 - Mostly C++ with some Fortran 90/77, Python, etc.
 - Contains Denovo, Shift, Insilico
 - **MPACT**: Univ. of Mich.
 - Requires GCC 4.6.1+ and Intel 13.1+
 - Mostly Fortran
 - Windows builds
 - **COBRA-TF**: Penn. State Univ.
 - Mostly Fortran 77 and 90
- Native TriBITS repos providing packages: **TeuchosWrappersExt**, **VERAInExt**, **DataTransferKit**
- VERA Repositories/packages not using TriBITS as native build system but have **secondary native TriBITS support**: **MAMBA**
- VERA Repositories not providing a TriBITS build: **MOOSE/Bison**, **DAKOTA**

Flexibility in TriBITS Projects and Repositories

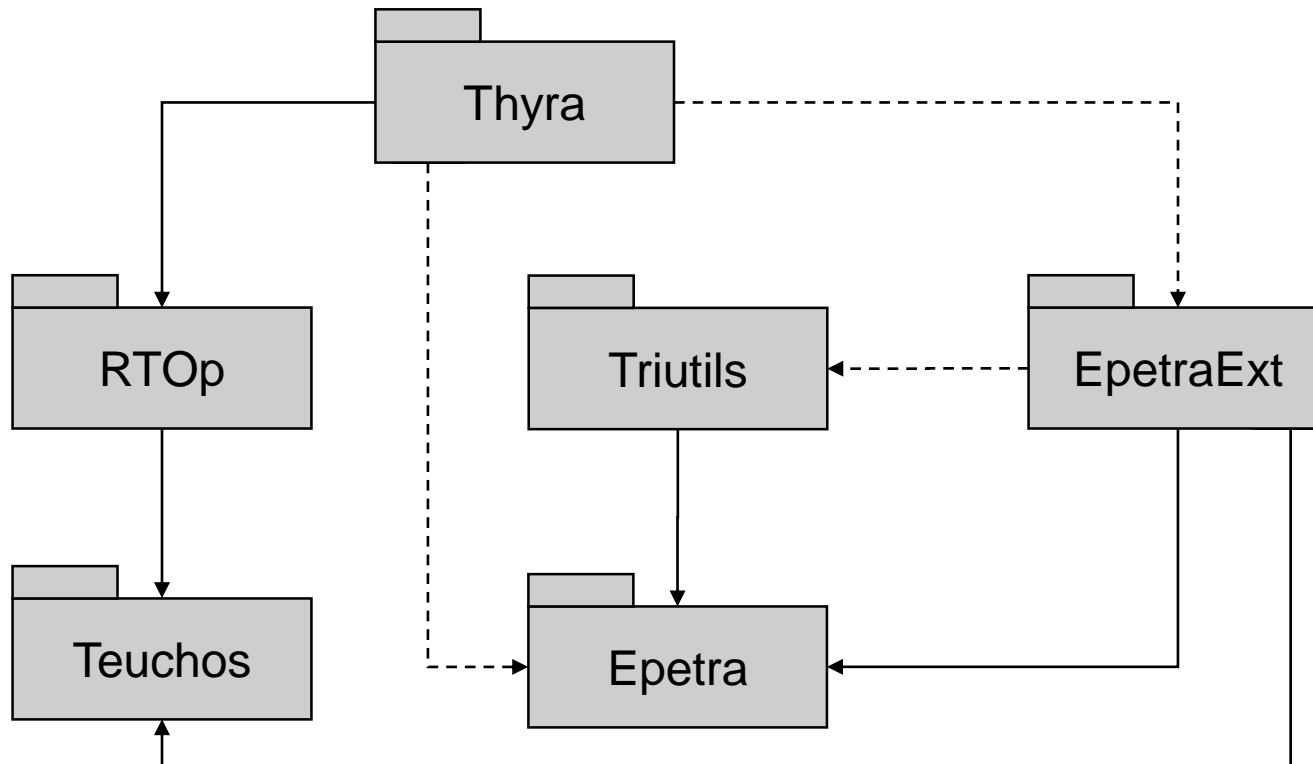


The same TriBITS repositories can be rearranged into multiple TriBITS CMake projects



TriBITS Automated Package and TPL Dependency Handling

Package Dependency Structure (Example: Trilinos)



Required Dependence →

Optional Dependence - - - - ->



Package Dependencies.cmake Files

Teuchos

```
tribits_package_define_dependencies(  
  LIB_REQUIRED_TPLS BLAS LAPACK  
  LIB_OPTIONAL_TPLS Boost )
```

Epetra

```
tribits_package_define_dependencies(  
  LIB_REQUIRED_TPLS BLAS LAPACK )
```

RTOp

```
tribits_package_define_dependencies(  
  LIB_REQUIRED_PACKAGES Teuchos )
```

Triutils

```
tribits_package_define_dependencies(  
  LIB_REQUIRED_PACKAGES Epetra )
```

EpetraExt

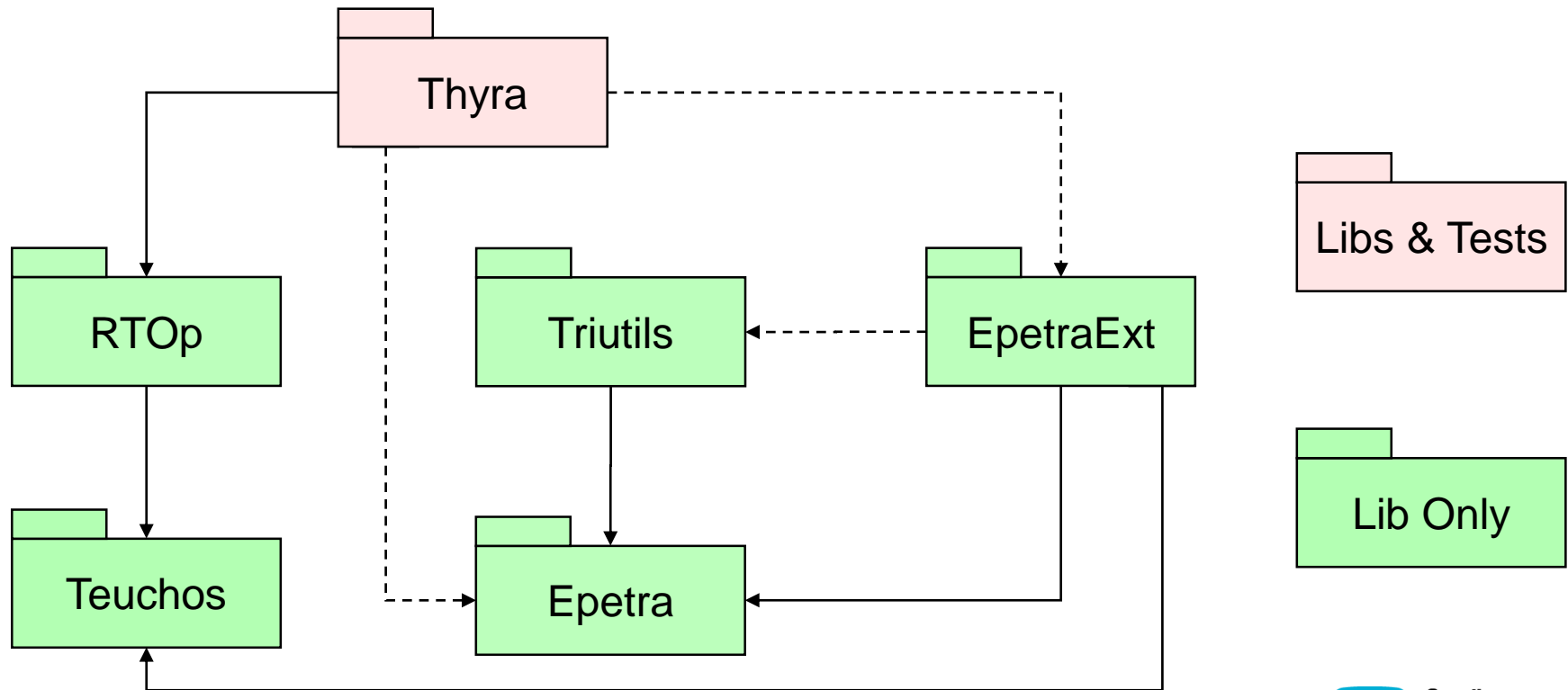
```
tribits_package_define_dependencies(  
  LIB_REQUIRED_PACKAGES Epetra Teuchos  
  LIB_OPTIONAL_PACKAGES Triutils )
```

Thyra

```
tribits_package_define_dependencies(  
  LIB_REQUIRED_PACKAGES RTOp Teuchos  
  LIB_OPTIONAL_PACKAGES EpetraExt Epera )
```

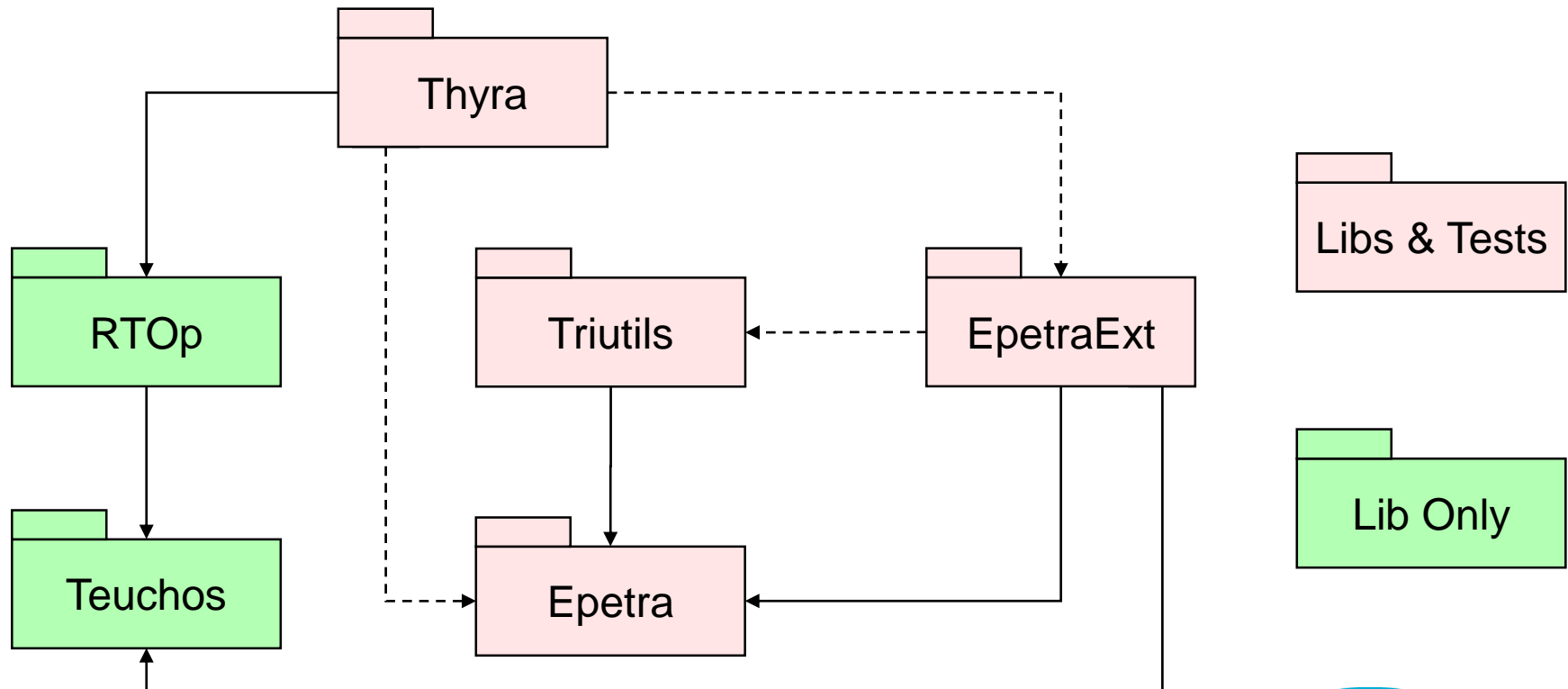
Enabling a Package its Tests

```
$ cmake \  
-D Trilinos_ENABLE_Thyra=ON \  
-D Trilinos_ENABLE_TESTS=ON \  
...
```



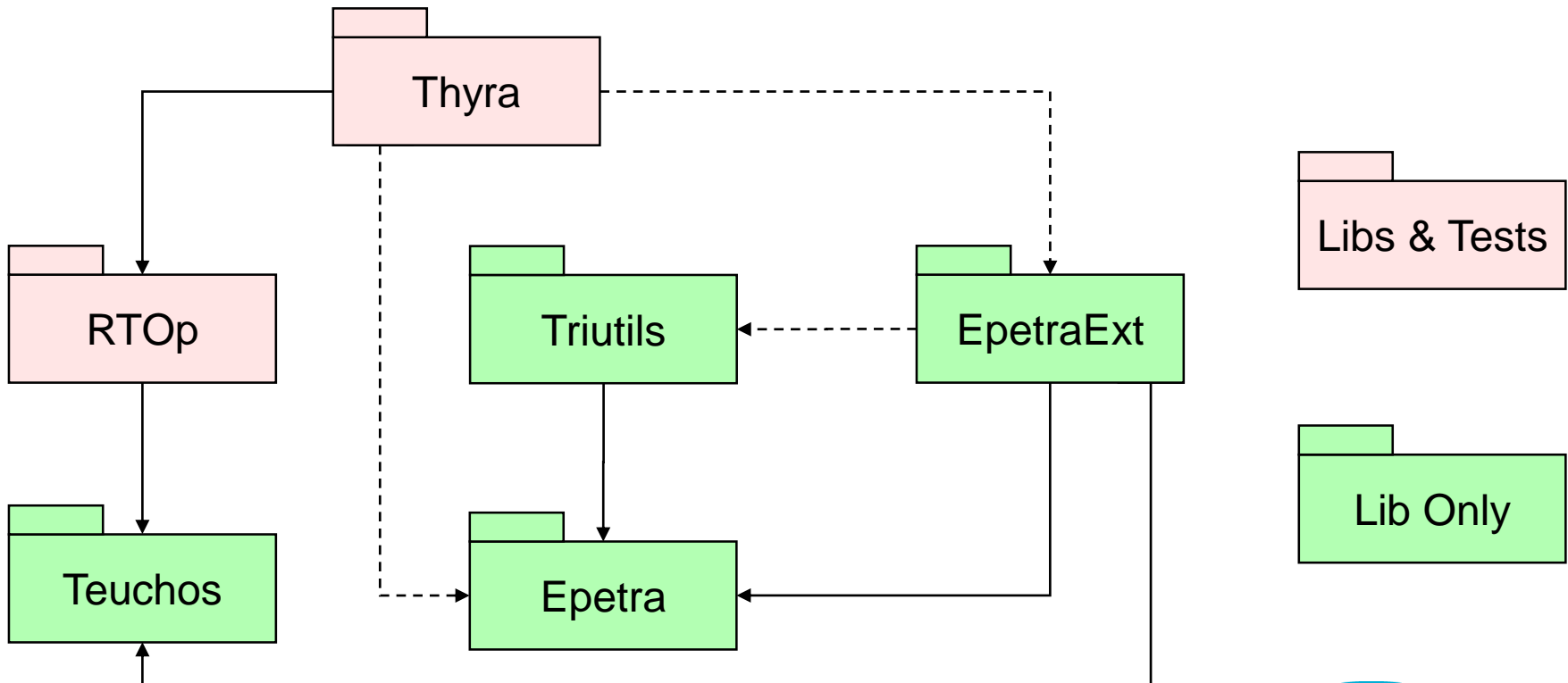
CI Testing: Change Epetra

```
$ cmake \  
-D Trilinos_ENABLE_Epetra=ON \  
-D Trilinos_ENABLE_ALL_FORWARD_DEP_PACKAGES=ON \  
-D Trilinos_ENABLE_TESTS=ON \  
...
```



CI Testing: Change RTOp

```
$ cmake \  
-D Trilinos_ENABLE_RTOp=ON \  
-D Trilinos_ENABLE_ALL_FORWARD_DEP_PACKAGES=ON \  
-D Trilinos_ENABLE_TESTS=ON \  
...
```





TriBITS Packages and Subpackages



Software Engineering Theory about Packaging

Package Cohesion OO Principles:

- REP (Release-Reuse Equivalency Principle): The granule of reuse is the granule of release.
- CCP (Common Closure Principle): The classes in a package should be closed together against the same kinds of changes. A change that affects a closed package affects all the classes in that package and no other packages.
- CRP (Common Reuse Principle): The classes in a package are used together. If you reuse one of the classes in a package, you reuse them all.

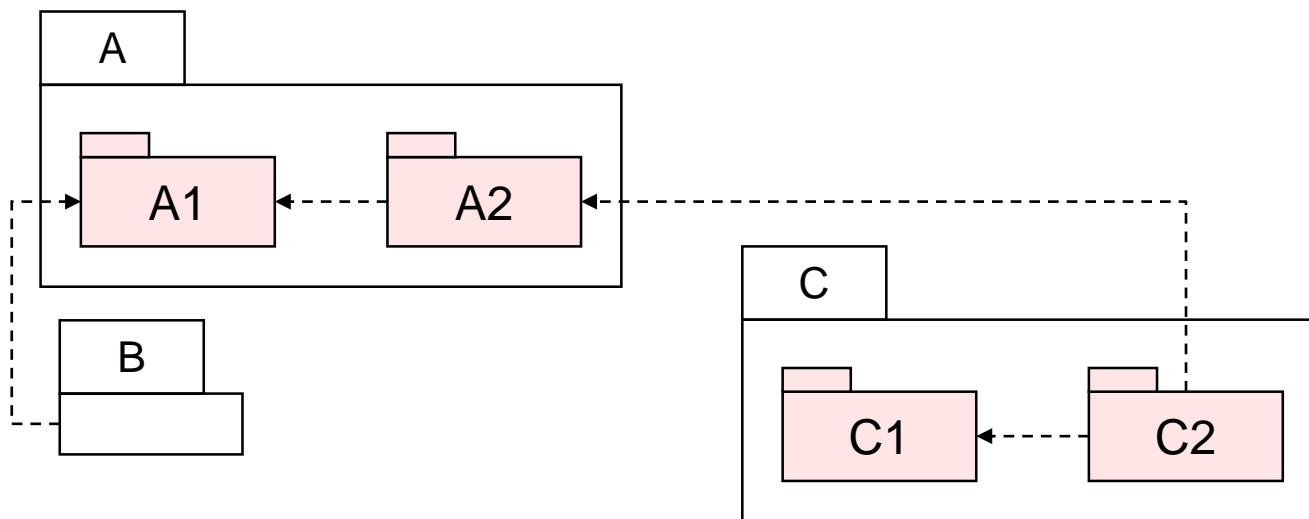
Package Coupling OO Principles:

- ADP (Acyclic Dependencies Principle): Allow no cycles in the package dependency graph.
- SDP (Stable Dependencies Principle): Depend in the direction of stability.
- SAP (Stable Abstractions Principle): A package should be as abstract as it is stable.

The TriBITS (e.g Trilinos) definition of a “Package” is not consistent with SE packaging principles most importantly the CRP

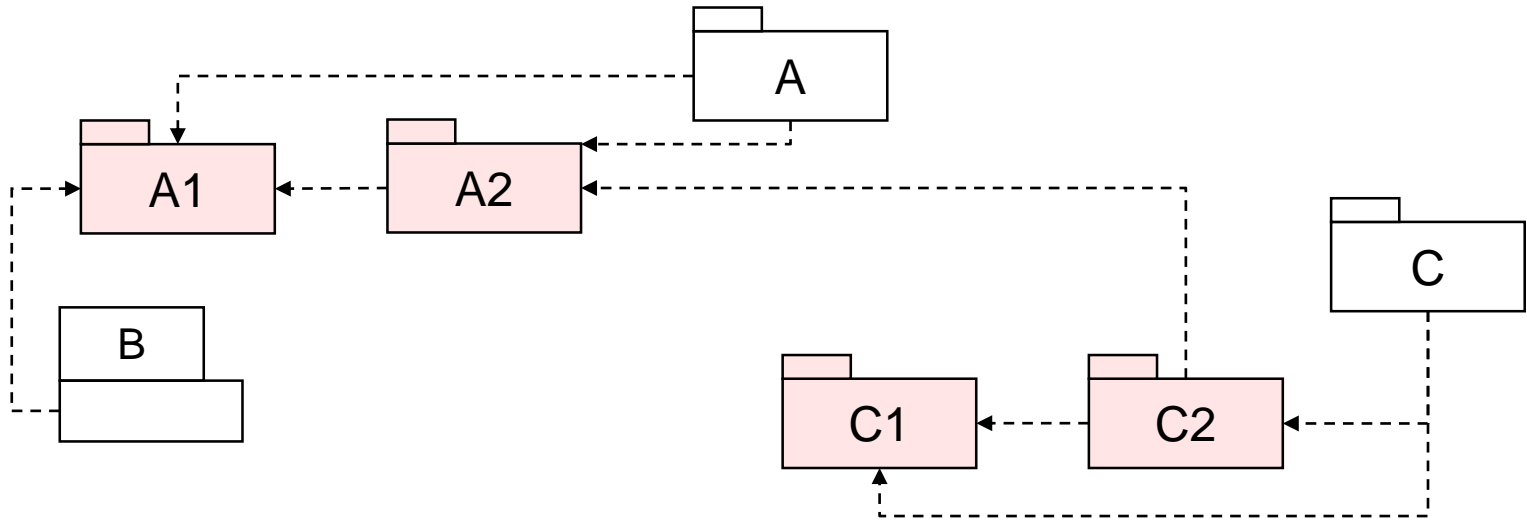
Source: Martin, Robert C. *Agile Software Development (Principles, Patterns, and Practices)*. Prentice Hall, 2003

TriBITS Packages and Subpackages: Overview



- **TriBITS Parent Package:**
 - Collection of related subpackages
 - Community of tightly integrated developers
 - Unit of documentation, package-by-package CTest driver (single email address)
 - Downstream (SE) packages should **not** list parent package as a dependency!
- **TriBITS Subpackage:**
 - Lightweight encapsulated collection of tightly related libs and tests/examples
 - Lightweight use the options of the parent package
 - Lists dependencies on upstream **SE Packages & TPLs**
 - Primary unit for dependency management!

TriBITS Packages and Subpackages: Dependencies

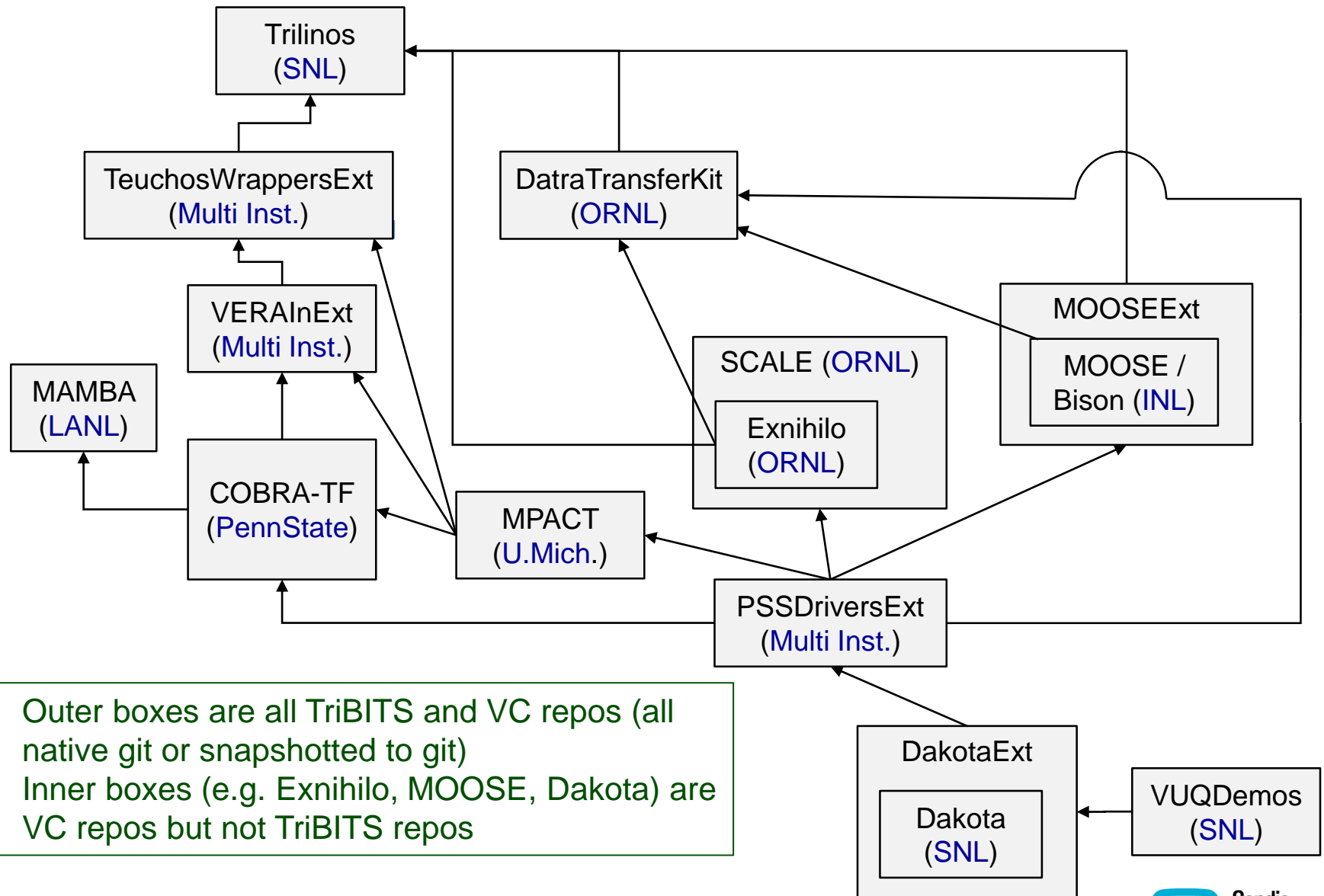


- This is how the TriBITS dependency management looks at packages and subpackages => **Packages and Subpackages are just are all just SE packages!**
- The parent package is just an SE Package that depends (optional or required) on all of its subpackages
 - New TriBITS packages should only have optional subpackages, **have no required subpackages!**
 - Support for **required subpackages is to maintain backward compatibility** when packages are broken into subpackages (when optional packages are disabled).



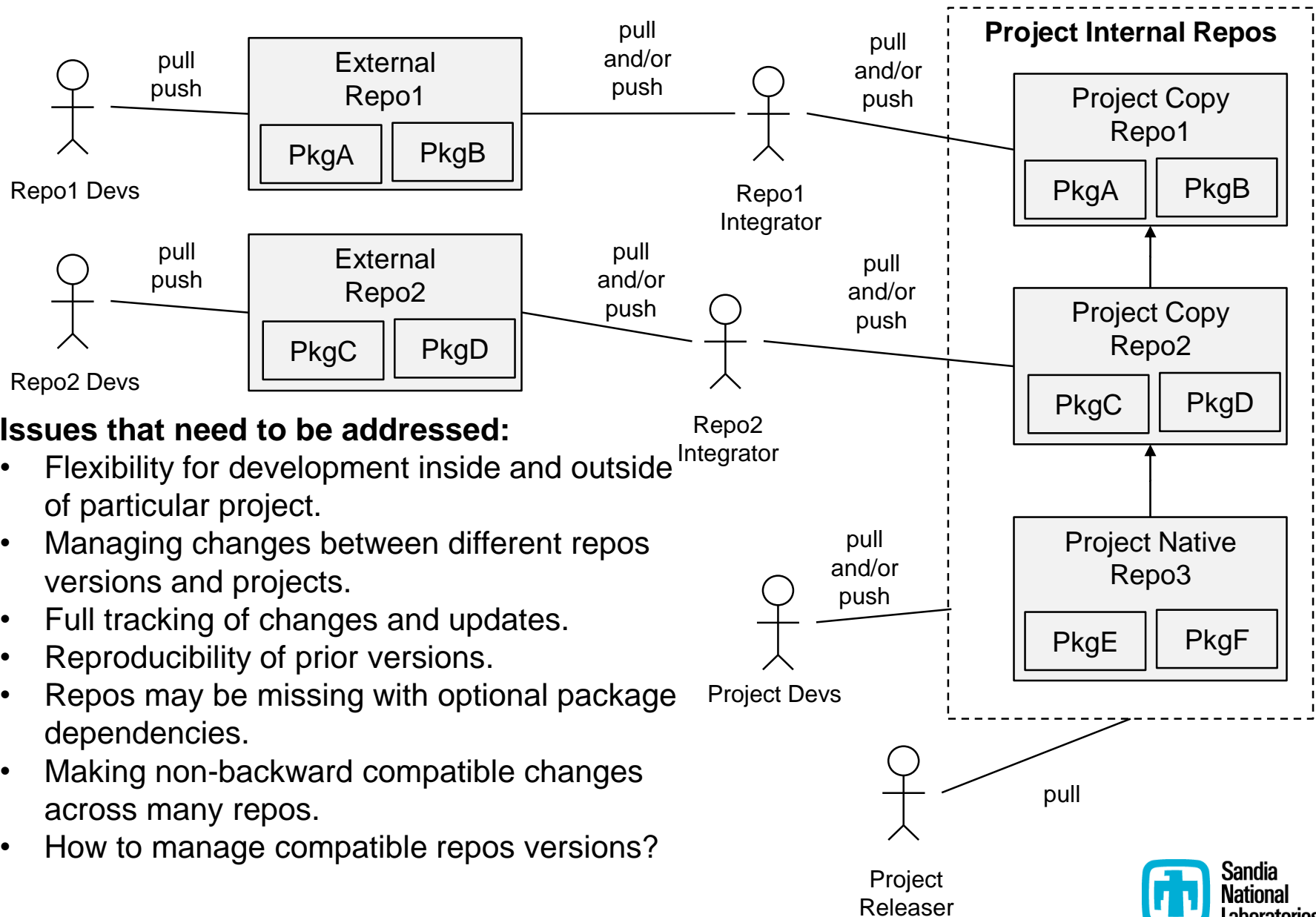
TriBITS Multi-Repository Support and Handling

TriBITS and VC Repos for CASL VERA



- Outer boxes are all TriBITS and VC repos (all native git or snapshotted to git)
- Inner boxes (e.g. Exnihilo, MOOSE, Dakota) are VC repos but not TriBITS repos

Managing Compatible Repos and Repo Versions





Managing Multiple Compatible Git Repositories

- **Combine all repos into one big git repo** (e.g. SIERRA/Trilinos style)
 - **git subtree & snapshotting** (e.g. TriBITS **snapshot_dir.py**):
 - **Pro**: One set of SHA1s, easy to do git bisect
 - **Pro**: One repo to pull and build final version
 - **Pro**: Can pull in non-git repos (e.g. Hg, svn)
 - **Con**: Harder to coordinate changes back and forth to native repos
 - **Con**: Does not allow for partitioning based on access control
 - **Directly deal with set of separate git repos:**
 - **Manual**: (e.g. SIERRA code, test, & doc repos; Albany & Trilinos)
 - **Pro**: No extra tool dependence
 - **Cons**: Does not scale, easy to make mistakes
 - **git submodule**:
 - **Pro**: Built-in git support and documentation
 - **Pro**: Individual repos stay independent (let git do its job).
 - **Pro**: Tracks compatible versions of repos, allows basic git bisect
 - **Con**: Extra commands to pull component repos
 - **Con**: Updating repos versions is complex for non-git savvy developers
 - **Con**: Does not support co-development well at all
 - **Treat set of repos as one big repo** (Google Repo, TriBITS **gitdist**)
 - **Pro**: Simple single-repo workflow
 - **Con**: Must track compatible versions as an add-on (e.g. to support git bisect)

CASL VERA => Uses **gitdist for most git repos, uses snapshotting for non-native or more complex git repos.**

gitdist: Treat collection of git repos as one

- **gitdist**: Simple stand-alone Python tool for distributing git commands across multiple git repos.

Usage: `gitdist [gitdist options] <raw-git-command> [git options]`

- `.gitdist.default` file committed to base git repo:

```
Trilinos
SCALE
SCALE/Exnihilo
...
```

Example:

```
$ gitdist status
*** Base Git Repo: VERA
On branch master
...
*** Git Repo: Trilinos
On branch master
...
*** Git Repo: SCALE
# On branch master
...
*** Git Repo: SCALE/Exnihilo
# On branch master
...
```

- Same workflow as single git repo:
 - `$ gitdist checkout master`
 - `$ gitdist pull`
 - `<create commits to repos>`
 - `$ gitdist status`
 - `$ gitdist push`
- No complexities of git submodules
- Keeps full history of separate repos for easy merging, etc. (unlike snapshots)
- Special commands for many repos:
 - Show compact status table:
 - `$ gitdist-status`
 - Show only changed repos:
 - `$ gitdist-mod-status`
 - `$ gitdist-mod local-stat`

gitdist: Show summary status table

\$ gidist-status # alias 'gitdist dist-repo-status'

ID	Repo Dir	Branch	Tracking Branch	C	M	?
0	VERA (Base)	master	origin/master	2	1	
1	TriBITS	master	origin/master			
2	Trilinos	master	origin/master			
3	TeuchosWrappersExt	master	origin/master			2
4	MAMBA	master	origin/master			
5	COBRA-TF	master	origin/master			
6	VERAInExt	master	origin/master			3
7	DataTransferKit	master	origin/master			
8	MOOSEExt	master	origin/master			
9	MOOSEExt/MOOSE	master	origin/master			
10	SCALE	master	origin/master			
11	SCALE/Exnihilo	master	origin/master			
12	MPACT	master	origin/master	2		
13	LIMEExt	master	origin/master			
14	hydrath	master	origin/master			
15	PSSDriversExt	master	origin/master		4	
16	DakotaExt	master	origin/master			
17	DakotaExt/Dakota	master	origin/master			
18	VUQDemos	master	origin/master			

gitdist: Show summary status table, modified only

```
$ gidist-mod-status # alias `gitdist dist-repo-status --dist-mod-only`
```

ID	Repo Dir	Branch	Tracking Branch	C	M	?
0	VERA (Base)	master	origin/master	2	1	
3	TeuchosWrappersExt	master	origin/master			2
6	VERAInExt	master	origin/master			3
12	MPACT	master	origin/master	2		
15	PSSDriversExt	master	origin/master		4	

- Compress out repos with no changes
- Manage changes on many repos at once even when there are 100s of repos!

gitdist: Run commands only on modified repos

```
$ gitdist-mod log --oneline --name-status HEAD ^@{u}
```

```
*** Base Git Repo: VERA
```

```
0ed6639 Minor comment update
```

```
M      cmake/ctest/drivers/fissile4/load_dev_env_gcc-4.8.3.sh
```

```
*** Git Repo: TriBITS
```

```
db7152a Add better gitdist-mod-status command
```

```
M      test/python_utils/gitdist_UnitTests.py
```

```
M      tribits/devtools_install/load_dev_env.csh.in
```

```
M      tribits/devtools_install/load_dev_env.sh.in
```

```
M      tribits/python_utils/gitdist
```

- See detailed changes only in changed repos
- Manage changes on many repos at once even when there are 100s of repos!

TriBITS: clone_extra_repos.py

```
$/clone_extra_repos.py
```

...

ID	Repo Name	Repo Dir	VC	Repo URL	Category
1	TriBITS	TriBITS	GIT	git@casl-dev:TriBITS	Continuous
2	Trilinos	Trilinos	GIT	git@casl-dev:Trilinos	Continuous
3	TeuchosWrappersExt	TeuchosWrappersExt	GIT	git@casl-dev:TeuchosWrappersExt	Continuous
4	MAMBA	MAMBA	GIT	git@casl-dev:MAMBA	Continuous
5	COBRA-TF	COBRA-TF	GIT	git@casl-dev:COBRA-TF	Continuous
6	VERAInExt	VERAInExt	GIT	git@casl-dev:VERAInExt	Continuous
7	VERAData	VERAData	GIT	git@casl-dev:VERAData	Nightly
8	DataTransferKit	DataTransferKit	GIT	git@casl-dev:DataTransferKit	Continuous
9	MOOSEExt	MOOSEExt	GIT	git@casl-dev:MOOSEExt	Continuous
10	MOOSE	MOOSEExt/MOOSE	GIT	git@casl-dev:MOOSE	Continuous
11	SCALE	SCALE	GIT	git@casl-dev:SCALE	Continuous
12	Exnihilo	SCALE/Exnihilo	GIT	git@casl-dev:Exnihilo	Continuous
13	XSTools	XSTools	GIT	git@casl-dev:XSTools	Nightly
14	MPACT	MPACT	GIT	git@casl-dev:MPACT	Continuous
15	PSSDriversExt	PSSDriversExt	GIT	git@casl-dev:PSSDriversExt	Continuous
16	DakotaExt	DakotaExt	GIT	git@casl-dev:DakotaExt	Continuous
17	Dakota	DakotaExt/Dakota	GIT	git@casl-dev:Dakota	Continuous
18	VUQDemos	VUQDemos	GIT	git@casl-dev:VUQDemos	Nightly
19	VERAView	VERAView	GIT	git@casl-dev:VERAView	Continuous

...

```
Running: git clone git@casl-dev:TriBITS TriBITS
```

```
Running: git clone git@casl-dev:Trilinos Trilinos
```

...

- Reads from ExtraRepositoriesList.cmake
- Only clones the repos that the user/developer has access to clone

TriBITS: Keeping track of compatible sets of repos

How to keep track of compatible sets of repos?

=> TriBITS support for <Project>RepoVersion.txt file:

```
*** Base Git Repo: SomeBaseRepo
e102e27 [Mon Sep 23 11:34:59 2013 -0400] <author1@someurl.com>
First summary message
*** Git Repo: ExtraRepo1
b894b9c [Fri Aug 30 09:55:07 2013 -0400] <author2@someurl.com>
Second summary message
*** Git Repo: ExtraRepo1/ExtraRepo2
97cflac [Thu Dec 1 23:34:06 2011 -0500] <author3@someurl.com>
Third summary message
```

...

Setting **-D<PROJECT>_GENERATE_REPO_VERSION_FILE=ON** results in <Project>RepoVersion.txt getting:

- generated in the build base directory,
- echoed in the configure output (therefore archived to CDash),
- installed in the base install directory,
- included in the source tarball ('**make package_source**'),
- installed in the base install directory from the untarred source.

Use with gitdist to access compatible versions:

```
$ gitdist --dist-version-file=<Project>RepoVersion.<somedate>.txt \  
[git command and arguments]
```



Using <Project>RepoVersion.txt for Dev Distributions

- Known “good” versions of the Project code are recorded as <Project>RepoVersion.txt files (e.g. archived on CDash, committed directly to base repo, etc.).
- **Example:** If a given Nightly build of Project passed on all platforms then we can give the associated <Project>RepoVersion.txt file as the “version” for a client to use.
- Send client file <Project>RepoVersion.<newdate>.txt for “good” version to install.
- Client gets updated version:

```
$ cd VERA/  
$ gitdist fetch  
$ gitdist --dist-version-file=~/<Project>RepoVersion.<newdate>.txt \  
  checkout _VERSION_
```
- Client can see changes since a previous installs of <Project>:

```
$ gitdist fetch  
$ gitdist \  
  --dist-version-file=~/<Project>RepoVersion.<newdate>.txt \  
  --dist-version-file2=${INSTALL_BASE}/<olddate>/<Project>RepoVersion.txt \  
  log-short --name-status _VERSION_ ^_VERSION2_
```
- NOTE: A variation of the <Project>RepoVersion.txt file could be committed to the base repo (at well-defined points) to allow using **git bisect** to search for introduction of defects and other changes.

Dealing with Missing Repos giving Missing Packages

What if **Repo2** is missing? Can we still configure and build the remaining packages in Repo3?

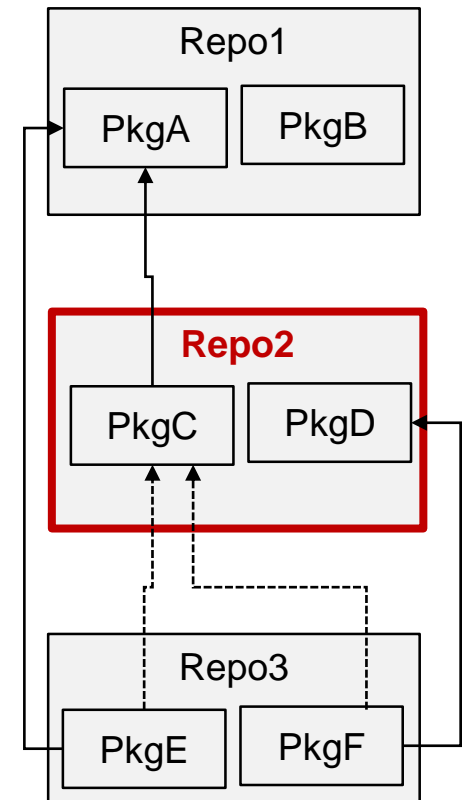
```
# Repo3/PkgE/cmake/Dependencies.cmake
LIB_REQUIRED_PACKAGES PkgA
LIB_OPTIONAL_PACKAGES PkgC
...
TRIBITS_ALLOW_MISSING_EXTERNAL_PACKAGES(PkgC)

# Repo3/PkgF/cmake/Dependencies.cmake
LIB_REQUIRED_PACKAGES PkgD
LIB_OPTIONAL_PACKAGES PkgC
...
TRIBITS_ALLOW_MISSING_EXTERNAL_PACKAGES(PkgC PkgD)
```

Now when configuring the Project with Repo2 missing
TriBITS automatically adjusts:

NOTE: PkgC is being ignored since its directory is missing and
PkgC_ALLOW_MISSING_EXTERNAL_PACKAGE = TRUE!

NOTE: Setting <PROJECT>_ENABLE_PkgF=OFF because PkgD is a
required missing package!





Primary Meta-Project Packages (PMPP)

- Some packages are “primary” to the project and are under development by the project. Other packages are just there to satisfy downstream dependencies.

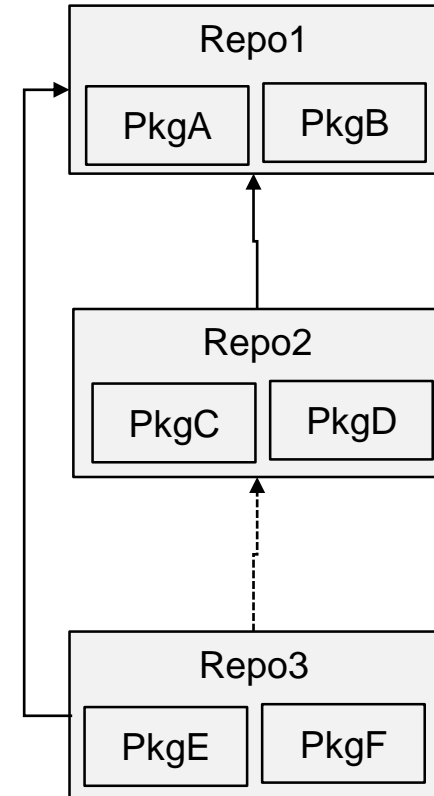
Example: VERA

```
set(Trilinos_NO_PRIMARY_META_PROJECT_PACKAGES TRUE)
set(SCALE_NO_PRIMARY_META_PROJECT_PACKAGES TRUE)
set(SCALE_NO_PRIMARY_META_PROJECT_PACKAGES_EXCEPT Insilico)
```

- **-DVERA_ENABLE_ALL_PACKAGES=ON**: Only explicitly enable the PMPPs for VERA and not packages in Trilinos, SCALE (except Insilico), etc.
- **-DVERA_ENABLE_ALL_TESTS=ON**: Only enable tests for PMPPs that are explicitly enabled and not others that might be enabled in Trilinos, SCALE, etc.
- Used in:
 - Automated CTest/CDash testing (only process PMPPs)
 - Pre-push CI testing with checkin-test.py (tests for only PMPPs)
 - Creating source tarball distributions (excludes packages not enabled)

Incorporating Externally Configured/Built Software

- **Motivation:** For some software, it may not be practical or maintainable to create a (secondary) native TriBITS build for a piece of software.
- **Goal:**
 - Use another configure/build tool for external software.
 - Put in CMake/TriBITS hooks to incorporate into TriBITS build with other packages.
- **Easy case:** No upstream TriBITS packages => **Repo1**
- **Medium case:** No downstream TriBITS packages => **Repo3** (e.g. **Dakota**)
- **Hard case:** Both upstream and downstream TriBITS packages => **Repo2**
- **Example:** INL MOOSE/Bison developers not willing to support a native TriBITS build of **libmesh** and **MOOSE/Bison** for CASL VERA. Libmesh depends on DataTransferKit and therefore Trilinos ☹
Tpetra <= DataTransferKit <= **libmesh** <= **MOOSE/Bison** <= Tiamat
- **Technical challenges addressed in TriBITS:**
 - Generate export makefile for upstream Trilinos packages
 - Create CMake rules to produce libs/executables
 - Add dependencies for changes to upstream code
 - Add dependencies for modified external project files





Development Workflow for Projects with TriBITS



COBRA-TF: Clone, Configure, Build, Test

```
$ git clone git@github.com:CASL/COBRA-TF
$ cd COBRA-TF/
$ mkdir BUILD/
$ cd BUILD/
$ cmake \
  -DCOBRA-TF_ENABLE_TESTS=ON \
  [other standard cmake options] .. # Compilers and BLAS found in PATH
$ make -j16
$ ctest -j16
```

Key Points:

- TriBITS is snapshotted in, no extra dependencies.
- Just looks like a regular simple CMake project to most users and developers.



COBRA-TF: Pull, Change, Test, Commit, Push

Source Directory Terminal

```
$ cd COBRA-TF/  
$ git pull  
  
$ emacs <files-to-change>  
  
$ git status  
$ git commit -a
```

Build Directory Terminal

```
$ cd BUILD/  
$ make -j16  
$ ctest -j16 # Passed!  
  
$ ../checkin-test.py --do-all --push
```

Key Points:

- Just looks like a regular simple CMake project to most developers.

VERA: Clone, Configure, Build, Test

```
# Set up to find compilers, TPLs, etc.
$ source /projects/vera/gcc-4.8.3/load_dev_env.sh

# Get the source repos
$ git clone git@casl-dev:VERA
$ cd VERA/
$ ./clone_vera_repos.py      # Calls ./clone_extra_repos.py

# Configure, build and test
$ export VERA_STD=$PWD/cmake/std/gcc-4.8.3 # Standard configurations
$ mkdir BUILD/
$ cd BUILD/
$ cmake \
  -DVERA_CONFIGURE_OPTIONS_FILE=$VERA_STD/mpi-release-shared-options.cmake \
  -DVERA_ENABLE_TESTS=ON \
  -DVERA_ENABLE_Tiamat=ON \
  [other standard cmake options] \
  ..
$ make -j16
$ ctest -j16
```

Key Points:

- Looks like a regular CMake project for most users and developers (except for cloning extra repos).

VERA: Pull, Change, Test, Commit, Push

Source Directory Terminal

```
# Pull updates from remote repos
$ cd VERA/
$ gitdist pull # All 21 repos!

# Modify files in three git repos!
$ cd COBRA-TF/cobra-tf/
$ emacs \
  preprocessor_src/ASCII_read.f90
$ cd ..
$ cd VERAInExt/verain/scripts/
$ emacs Templates/COBRA-TF.yml [more]
$ cd ..
$ cd ../../PSSDriversExt/Tiamat/
$ emacs src/Tiamat.cpp [more]
$ cd ..

# Check status and commit
$ gitdist-mod status
$ cd COBRA-TF/; git commit -a; ..
$ cd VERAInExt/; git commit -a; ..
$ cd PSSDriversExt/; git commit -a; ..
```

Key Points:

- Just a single CMake project!
- Same workflow as for a single git repo except for:
 - `git pull => gitdist pull`
 - `git status => gitdist-mod status`
- Developers with very little git knowledge can do this!

Build Directory Terminal

```
# Local build and test
$ cd BUILD/PSSDrivers/tiamat/
$ make -j16 # Magic!
$ ctest -j16 # Passed!

$ ../checkin-test.py --do-all --push
```




Example Development Workflow For Project without TriBITS

Example Trilinos Configure for Downstream APP

```
cmake \  
-D Teuchos_ENABLE_LONG_LONG_INT:BOOL=ON \  
\  
-D Trilinos_ENABLE_Teuchos:BOOL=ON \  
-D Trilinos_ENABLE_Shards:BOOL=ON \  
-D Trilinos_ENABLE_Sacado:BOOL=ON \  
-D Trilinos_ENABLE_Epetra:BOOL=ON \  
-D Trilinos_ENABLE_EpetraExt:BOOL=ON \  
-D Trilinos_ENABLE_Ipack:BOOL=ON \  
-D Trilinos_ENABLE_AztecOO:BOOL=ON \  
-D Trilinos_ENABLE_Amesos:BOOL=ON \  
-D Trilinos_ENABLE_Anasazi:BOOL=ON \  
-D Trilinos_ENABLE_Belos:BOOL=ON \  
-D Trilinos_ENABLE_ML:BOOL=ON \  
-D Trilinos_ENABLE_Phalanx:BOOL=ON \  
-D Trilinos_ENABLE_Intrepid:BOOL=ON \  
-D Trilinos_ENABLE_Intrepid2:BOOL=ON \  
-D Trilinos_ENABLE_NOX:BOOL=ON \  
-D Trilinos_ENABLE_Stratimikos:BOOL=ON \  
-D Trilinos_ENABLE_Thyra:BOOL=ON \  
-D Trilinos_ENABLE_Rythmos:BOOL=ON \  
-D Trilinos_ENABLE_MOOCHO:BOOL=ON \  
-D Trilinos_ENABLE_Stokhos:BOOL=ON \  
-D Trilinos_ENABLE_Piro:BOOL=ON \  
-D Trilinos_ENABLE_Teko:BOOL=ON \  
\  
-D Trilinos_ENABLE_STKIO:BOOL=ON \  
-D Trilinos_ENABLE_STKMesh:BOOL=ON \  
-D TPL_ENABLE_Boost:BOOL=ON \  
-D Boost_INCLUDE_DIRS:FILEPATH="$BOOSTDIR/include" \  
-D Boost_LIBRARY_DIRS:FILEPATH="$BOOSTDIR/lib" \  
-D TPL_ENABLE_BoostLib:BOOL=ON \  
-D BoostLib_INCLUDE_DIRS:FILEPATH="$BOOSTDIR/include" \  
-D BoostLib_LIBRARY_DIRS:FILEPATH="$BOOSTDIR/lib" \  
\  
-D Trilinos_ENABLE_SEACASIOss:BOOL=ON \  
-D Trilinos_ENABLE_SEACASExodus:BOOL=ON \  
-D TPL_ENABLE_Netcdf:BOOL=ON \  
-D Netcdf_INCLUDE_DIRS:PATH="$NETCDFDIR/include" \  
-D Netcdf_LIBRARY_DIRS:PATH="$NETCDFDIR/lib" \  
-D TPL_ENABLE_HDF5:BOOL=ON \  
-D HDF5_INCLUDE_DIRS:PATH="$HDF5DIR/include" \  
-D HDF5_LIBRARY_DIRS:PATH="$HDF5DIR/lib" \  
\  
-D Trilinos_ENABLE_Tpetra:BOOL=ON \  
-D Trilinos_ENABLE_Kokkos:BOOL=ON \  
-D Trilinos_ENABLE_Ipack2:BOOL=ON \  
-D Trilinos_ENABLE_Amesos2:BOOL=ON \  
-D Trilinos_ENABLE_Zoltan2:BOOL=ON \  
-D Trilinos_ENABLE_MueLu:BOOL=ON \  
-D Amesos2_ENABLE_KLU2:BOOL=ON \  
\  
-D Trilinos_ENABLE_EXPLICIT_INSTANTIATION:BOOL=ON \  
-D Tpetra_INST_INT_LONG_LONG:BOOL=ON \  
-D Tpetra_INST_INT_INT:BOOL=ON \  
-D Tpetra_INST_DOUBLE:BOOL=ON \  
-D Tpetra_INST_FLOAT:BOOL=OFF \  
-D Tpetra_INST_COMPLEX_FLOAT:BOOL=OFF \  
-D Tpetra_INST_COMPLEX_DOUBLE:BOOL=OFF \  
-D Tpetra_INST_INT_LONG:BOOL=OFF \  
-D Tpetra_INST_INT_UNSIGNED:BOOL=OFF \  
\  
-D Trilinos_ENABLE_Kokkos:BOOL=ON \  
-D Trilinos_ENABLE_KokkosCore:BOOL=ON \  
-D Phalanx_KOKKOS_DEVICE_TYPE:STRING="SERIAL" \  
-D Phalanx_INDEX_SIZE_TYPE:STRING="INT" \  
-D Phalanx_SHOW_DEPRECATED_WARNINGS:BOOL=OFF \  
-D Kokkos_ENABLE_Serial:BOOL=ON \  
-D Kokkos_ENABLE_OpenMP:BOOL=OFF \  
-D Kokkos_ENABLE_Pthread:BOOL=OFF \  
[more options] ..
```

... [Continued] ...

55 separate enables and disables!

APP + Trilinos: Clone, Configure, Build, Test

```
# Set up to find compilers, TPLs, etc.
$ source <some-env-script>

# Clone the repos
$ git clone git@github.com:trilinos/Trilinos.git
$ git clone <some-url>:<app>

# Set up build dirs
$ mkdir BUILD/ ; cd BUILD/ ; mkdir Trilinos ; cd .. ; mkdir <app> ; ..

# Configure, build and install Trilinos
$ export TRIINOS_INSTALL_DIR=$PWD/trilinos-install
$ cd Trilinos/
$ cd ../BUILD/Trilinos/
$ cmake [55 enables/disables] [other options] \
  -DCMAKE_INSTALL_PREFIX=$TRIINOS_INSTALL_DIR \
  ../../Trilinos
$ make -j16 install

# Configure, build, and test APP
$ cd ../../BUILD/<app>/
$ cmake \      # Pulls in TrilinosConfig.cmake for most info!
  -D<APP>_TRILINOS_DIR=$TRIINOS_INSTALL_DIR \
  ../../<app>
$ make -j16
$ ctest -j16
```

APP+Trilinos: Pull, Change, Commit, Test, Push

Source Directory Terminal

```
# Pull updates from remote repos
$ cd Trilinos/ ; git pull ; cd ..
$ cd <app>/ ; git pull ; cd ..

# Modify files in Trilinos and APP
$ cd Trilinos/packages/sacado/
$ emacs src/Sacado_Fad_DFad.hp
$ cd ../../..
$ cd <app>/
$ emacs <some-app-files>
$ cd ..

# Check status and commit
$ Trilinos/ ; git local-stat
$ git commit -a
$ cd ../<app>/ ; git local-stat
$ git commit -a

$ git pull --rebase; git push
```

Key Points:

- This is just 2 git repos and CMake projects **but other examples at SNL involve 3, 4, 5+ repos!**
- **How well does this scale?**
=> CMake ExternalProject?

Build Directory Terminal

```
$ cd BUILD/Trilinos/
$ make -j16 install
$ cd ../<app>/
$ make -j16
$ ctest -j16 # Passed!

$ cd ../Trilinos_CHECKIN/
$ ./checkin-test-<machine>.sh \
    --do-all --push
$ cd ../<app>/
$ make -j16 ; ctest -j16 # Pass
```



TriBITS Testing Support & CTest, CDash

TriBITS Standardized CTest Handling

```
tribits_add_test( <exeRootName>  [NOEXEPREFIX]  [NOEXESUFFIX]  
  [NAME <testName> | NAME_POSTFIX <testNamePostfix>]  
  [COMM [serial] [mpi]]  
  [NUM_MPI_PROCS <numMpiProcs>]  
  [NUM_TOTAL_CORES_USED <numTotalCoresUsed>]  
  [CATEGORIES <category0> <category1> ...]  
  [HOST <host0> <host1> ...]  [XHOST <host0> <host1> ...]  
  [HOSTTYPE <hosttype0> <hosttype1> ...]  [XHOSTTYPE <hosttype0> <hosttype1> ...]  
  [TIMEOUT <maxSeconds>]  
  ...  
)
```

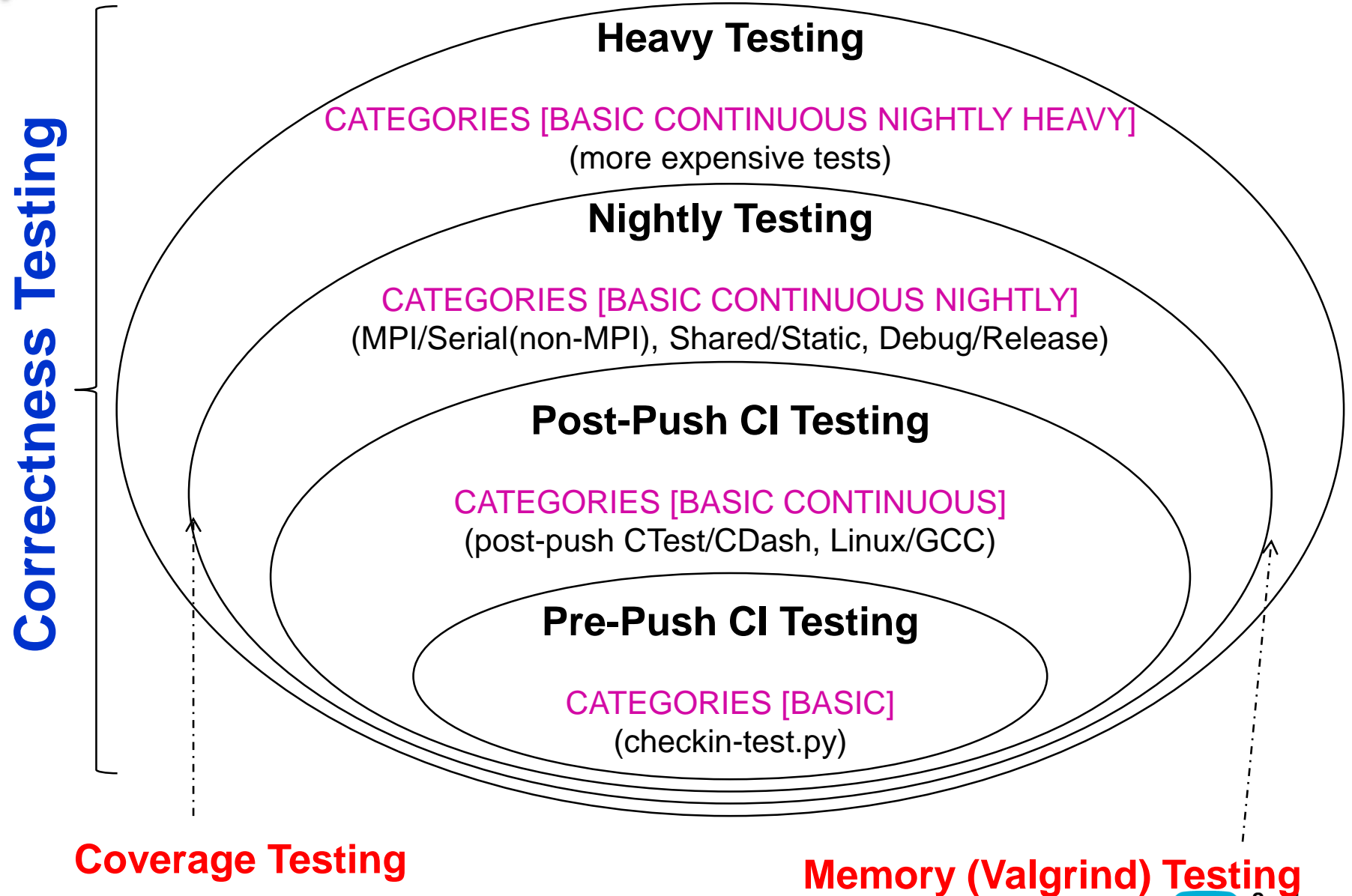
- Automatic namespacing of tests
- Automatic running of parallel tests with MPI
- Include/exclude tests based on number of MPI processes, number of cores, MPI/serial, host type (Linux, Windows, OSX, ...),, specific hosts machines, ...
- Standard classification of tests (BASIC, CONTINUOUS, NIGHTLY, HEAVY, ...) for inclusion/exclusion of tests
- Uniform handling of timeouts (and scaling of timeouts)
- And more ...

TriBITS CTest Extensions

```
tribits_add_advanced_test( <testNameBase>
    TEST_0 (EXEC <execTarget0> | CMND <cmndExec0>)
        [ARGS <arg1> <arg2> ... <argn>]      [MESSAGE "<message>"]      [WORKING_DIRECTORY <workingDir>]
        [NUM_MPI_PROCS <numProcs>]      [NUM_TOTAL_CORES_USED <numTotalCoresUsed>]
        [OUTPUT_FILE <outputFile>]      [NO_ECHO_OUTPUT]
        [PASS_ANY | PASS_REGULAR_EXPRESSION "<regex>"
            | PASS_REGULAR_EXPRESSION_ALL "<regex1>" "<regex2>" ... "<regexn>"
            | FAIL_REGULAR_EXPRESSION "<regex>" | STANDARD_PASS_OUTPUT ] ...
    [TEST_1 (EXEC <execTarget1> | CMND <cmndExec1>) ...]
    ...
    [TEST_N (EXEC <execTargetN> | CMND <cmndExecN>) ...]
    [OVERALL_NUM_MPI_PROCS <overallNumProcs>]
    [OVERALL_NUM_TOTAL_CORES_USED <overallNumTotalCoresUsed>]
    [CATEGORIES <category0> <category1> ...]
    [HOST <host0> <host1> ...]      [XHOST <host0> <host1> ...] ...
    [TIMEOUT <maxSeconds>]
    ...
)
```

- Run tests defined by multiple steps, .e.g. preprocessing, post-processing, etc.
- All the same behaviors/features as tribits_add_test() plus ...
- Timing of individual test steps
- Creates `cmake -P` script that is added with add_test()
- And more ...

Standard TriBITS Test Categories and Layers





Pre-Push CI Testing: checkin-test.py

```
$ checkin-test.py --do-all --push
```

- Checks that git repos are “clean”
- Checks that all git repos on are tracking branches
- Integrates (i.e. pulls) latest version in remote git repositories
- Figures out modified packages

```
Modified file: 'packages/teuchos/CMakeLists.txt'
```

```
=> Enabling 'Teuchos'!
```

- Enables all forward/downstream packages & tests
- Configures, builds, and runs tests
- Does the push (if all builds/tests pass)
- Sends notification emails
- Fully customizable (enabled packages, build cases, etc.)
- Documentation: [checkin-test.py --help](#)

CDash: VERA Packages

Login All Dashboards

Tuesday, June 07 2016 23:42:21



VERA

Dashboard

Calendar

Previous

Current

Project

Project

Project	Configure			Build			Test			Last submission
	Error	Warning	Pass	Error	Warning	Pass	Not Run	Fail	Pass	
VERA	0	159	159	0	132	27	0	2	9968	2016-06-07 21:44:04

SubProjects

SubProject	Configure			Build			Test			Last submission
	Error	Warning	Pass	Error	Warning	Pass	Not Run	Fail	Pass	
CASL_MOOSE	0	8	8	0	6	2	0	0	73	2016-06-07 18:20:49
Cicada	0	8	8	0	6	2	0	0	30	2016-06-07 17:23:49
COBRA_TF	0	8	8	0	8	0	0	0	2666	2016-06-07 17:24:08
CTeuchos	0	8	8	0	8	0	0	0	28	2016-06-07 17:21:17
DakotaExt	0	8	8	0	8	0	0	0	41	2016-06-07 19:26:33
DataTransferKit	0	8	8	0	8	0	0	0	434	2016-06-07 17:58:42
ForTeuchos	0	8	8	0	8	0	0	0	28	2016-06-07 17:23:17
Insilico	0	8	8	0	8	0	0	0	798	2016-06-07 18:21:51
MAMBA	0	8	8	0	8	0	0	0	32	2016-06-07 17:23:35
MPACT_Drivers	0	8	8	0	8	0	0	0	40	2016-06-07 18:57:38
MPACT_exe	0	8	8	0	3	5	0	0	1639	2016-06-07 19:06:51
MPACT_libs	0	8	8	0	8	0	0	0	1064	2016-06-07 18:47:34
Tiamat	0	8	8	0	6	2	0	0	47	2016-06-07 19:26:05
TriBITS	0	8	8	0	0	8	0	0	1632	2016-06-07 17:19:05
VeraAPI	0	8	8	0	8	0	0	2	270	2016-06-07 19:24:31
VERAIn	0	8	8	0	8	0	0	0	1064	2016-06-07 17:55:35
VERAout	0	8	8	0	8	0	0	0	8	2016-06-07 17:58:18
VUQCore	0	8	8	0	8	0	0	0	32	2016-06-07 19:35:49
VUQDemos	0	7	7	0	7	0	0	0	2	2016-06-07 19:36:29

CDash: VERA Package Builds (VeraAPI)

Login

All Dashboards

Tuesday, June 07 2016 23:51:19

CDash

Dashboard

Calendar

Previous

Current

Project

VERA


No file changed as of Tuesday, June 07 2016 - 00:59 EDT

Nightly

Site	Build Name	Update	Configure		Build		Test			Start Time	Labels
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass		
	Linux-GCC-4.8.3-SERIAL_RELEASE_SHARED	0	0	4	0	2	0	0	3 ¹	4 hours ago	VeraAPI
	Linux-GCC-4.8.3-SERIAL_RELEASE_DEBUG_SHARED	0	0	4	0	2	0	0	3 ¹	7 hours ago	VeraAPI
	Linux-GCC-4.8.3-MPI_RELEASE_STATIC	0	0	3	0	6 ₋₂	0	1 ¹	49	8 hours ago	VeraAPI
	Linux-GCC-4.8.3-MPI_RELEASE_DEBUG_STATIC	0	0	3	0	6 ₋₂	0	1 ¹	49	9 hours ago	VeraAPI
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	0	0	3	0	6 ₋₂	0	0	54	11 hours ago	VeraAPI
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED	0	0	3	0	6 ₋₂	0	0	49	13 hours ago	VeraAPI
	Linux-GCC-4.8.3-MPI_RELEASE_DEBUG_SHARED	0	0	3	0	6 ₋₂	0	0	49	17 hours ago	VeraAPI
	Linux-GCC-4.8.3-MPI_DEBUG_DEBUG_SHARED	0	0	3	0	8 ₋₄	0	0	14	18 hours ago	VeraAPI

CDash: VERA (Collapsed) Builds

[Login](#)
[All Dashboards](#)
Tuesday, June 07 2016 23:54:10



VERA
[Dashboard](#)
[Calendar](#)
[Previous](#)
[Current](#)
[Project](#)

No file changed as of **Tuesday, June 07 2016 - 00:59 EDT**

Filters
[Help](#)

Match the following rule:

Site
 contains

[Apply](#)
[Clear](#)
[Create Hyperlink](#)

Nightly

Site	Build Name	Update	Configure		Build		Test			Start Time ▼	Labels
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass		
	Linux-GCC-4.8.3-SERIAL_RELEASE_SHARED	0	0	72	0	224	0	0	1080	6 hours ago	(20 labels)
	Linux-GCC-4.8.3-SERIAL_RELEASE_DEBUG_SHARED	0	0	72	0	226	0	0	1080	10 hours ago	(20 labels)
	Linux-GCC-4.8.3-MPI_RELEASE_STATIC	0	0	65	0	251	0	1	1273	13 hours ago	(20 labels)
	Linux-GCC-4.8.3-MPI_RELEASE_DEBUG_STATIC	0	0	65	0	253	0	1	1273	15 hours ago	(20 labels)
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED	0	0	65	0	252	0	0	1273	18 hours ago	(20 labels)
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	0	0	56	0	251	0	0	1796	22 hours ago	(19 labels)
	Linux-GCC-4.8.3-MPI_RELEASE_DEBUG_SHARED	0	0	65	0	254	0	0	1273	22 hours ago	(20 labels)
	Linux-GCC-4.8.3-MPI_DEBUG_DEBUG_SHARED	0	0	66	0	385	0	0	920	22 hours ago	(20 labels)

Package-by-Package Build/Test Results

Site

Build Name: Linux-GCC-4.8.3-MPI_RELEASE_STATIC

Number of SubProjects Built: 20

Nightly

SubProject	Update		Configure			Build			Test				Start Time
	Files	Time	Error	Warn	Time	Error	Warn	Time	Not Run	Fail	Pass	Time	
VUQDemos	0	6s	0	9	14s	0	1 ⁺¹ ₋₁	42s	0	0	0	1s	5 hours ago
VUQCore	0	6s	0	9	16s	0	1 ⁺¹ ₋₁	48s	0	0	4	7s	5 hours ago
DakotaExt	0	6s	0	8	36s	0	1 ⁺¹ ₋₁	11m 6s	0	0	5	1m	5 hours ago
Tiamat	0	6s	0	3	4m 18s	0	13 ⁺⁷ ₋₇	2h 18m 12s	0	0	7 ⁺¹	27m 43s	7 hours ago
VeraAPI	0	6s	0	3	41s	0	6 ⁺² ₋₂	54m 54s	0	1 ⁺¹	49	39m 42s	8 hours ago
MPACT_exe	0	6s	0	2	27s	0	0	27m 54s	0	0	180	23m 40s	9 hours ago
MPACT_Drivers	0	6s	0	2	13s	0	4 ⁺¹ ₋₁	11m 12s	0	0	5	9s	9 hours ago
MPACT_libs	0	6s	0	2	24s	0	15 ⁺¹ ₋₁	11m 24s	0	0	135	1m 55s	9 hours ago
XSTools	0	6s	0	2	6s	0	0	1m 18s	0	0	5	47s	9 hours ago
Insilico	0	6s	0	3	1m 2s	0	50	27m 54s	0	0	109	1m 42s	10 hours ago
CASL_MOOSE	0	6s	0	3	2m 55s	0	5 ⁺² ₋₂	1h 59m 12s	0	0	13 ⁺²	7m 46s	12 hours ago
DataTransferKit	0	6s	0	3	22s	0	27 ⁺¹ ₋₁	23m 54s	0	0	62	1m 59s	12 hours ago
VERAout	0	6s	0	2	5s	0	1 ⁺¹ ₋₁	54s	0	0	1	1s	12 hours ago
VERAIn	0	6s	0	2	9s	0	5 ⁺⁴ ₋₄	2m 36s	0	0	133	1m 26s	12 hours ago
COBRA_TF	0	6s	0	2	15s	0	91 ⁺⁴¹ ₋₄₁	27m 42s	0	0	344	26m 22s	13 hours ago
MAMBA	0	6s	0	2	5s	0	17 ⁺⁴ ₋₄	24s	0	0	4	0s	13 hours ago
Cicada	0	6s	0	2	8s	0	6 ⁺¹ ₋₁	2m 42s	0	0	5	21s	13 hours ago
ForTeuchos	0	6s	0	2	6s	0	7 ⁺⁴ ₋₄	1m 12s	0	0	4	1s	13 hours ago
CTeuchos	0	6s	0	2	9s	0	1	2m 48s	0	0	4	0s	13 hours ago
TriBITS	0	6s	0	2	28s	0	0	2m 36s	0	0	204	1m 58s	13 hours ago

Post-Push Testing: TRIBITS_CTEST_DRIVER()

My CDash All Dashboards Log Out Monday, May 05 2014 20:48:48 EDT

VERA

Dashboard Calendar Previous Current Next Project Settings

No update data as of Sunday, April 06 2014 - 23:00 EDT Show Filters Advanced View Auto-refresh Help

Nightly

Site	Build Name	Update	Configure		Build		Test			Build Time	Labels
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass		
	Linux-GCC-4.6.1-MPI_RELEASE_GCC461	0	0	74	0	60	0	1	625	Apr 07, 2014 - 01:11 EDT	(13 labels)
	Linux-GCC-4.6.1-MPI_RELEASE_GCC461_WEEKLY	0	0	74	0	60	0	1	634	Apr 07, 2014 - 01:15 EDT	(13 labels)
	Linux-GCC-4.6.1-MPI_DEBUG_GCC461	0	0	74	0	61	0	0	626	Apr 07, 2014 - 01:11 EDT	(13 labels)

Continuous

Site	Build Name	Update	Configure		Build		Test			Build Time	Labels
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass		
	Linux-GCC-4.6.1-MPI_DEBUG_GCC461_CI	0	0	15	0	1	0	0	320	Apr 07, 2014 - 21:38 EDT	(2 labels)
	Linux-GCC-4.6.1-MPI_DEBUG_GCC461_CI	0	0	15	6	1	0	0	53	Apr 07, 2014 - 19:14 EDT	(2 labels)
	Linux-GCC-4.6.1-MPI_DEBUG_GCC461_CI	0	0	44	0	120	0	0	594	Apr 07, 2014 - 04:06 EDT	(11 labels)

VERA CDash Dashboard for 4/6/2014

- Collapsed summaries for each build case
- Nightly, CI, Experimental build cases

My CDash All Dashboards Log Out Monday, May 05 2014 20:48:52 EDT

VERA

Dashboard Calendar Previous Current Next Project Settings

No update data as of Sunday, April 06 2014 - 23:00 EDT Show Filters Advanced View Auto-refresh Help

Continuous

Site	Build Name	Update	Configure		Build		Test			Build Time	Labels
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass		
	Linux-GCC-4.6.1-MPI_DEBUG_GCC461_CI	0	0	13	0	1 ⁺⁵⁴ ₋₁₄	0	0	53 ₋₄₇	Apr 07, 2014 - 19:18 EDT	VRIPSS
	Linux-GCC-4.6.1-MPI_DEBUG_GCC461_CI	0	0	2	6 ⁺⁶	0				Apr 07, 2014 - 19:14 EDT	COBRA_TF

VERA CDash CI Iterations

- Individual packages built in sequence
- Targeted emails for failed package build & tests
- Failed packages disabled in downstream packages

=> Don't propagate failures!

My CDash All Dashboards Log Out Monday, May 05 2014 20:51:59 EDT

VERA

Dashboard Calendar Previous Current Next Project Settings

No update data as of Sunday, April 06 2014 - 23:00 EDT Show Filters Advanced View Auto-refresh Help

Continuous

Site	Build Name	Update	Configure		Build		Test			Build Time	Labels
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass		
	Linux-GCC-4.6.1-MPI_DEBUG_GCC461_CI	0	0	13	0	1 ⁺⁵ ₋₃₄	0	0	100 ⁺⁴⁷	Apr 07, 2014 - 21:45 EDT	VRIPSS
	Linux-GCC-4.6.1-MPI_DEBUG_GCC461_CI	0	0	2	0 ₋₆	0 ⁺²	0	0	220 ⁺²²⁰	Apr 07, 2014 - 21:38 EDT	COBRA_TF

CDash Queries: Killer Feature

Show the HEAVY builds for the last two weeks:

FiltersHelp

Match

all

 of the following rules:

Build Name

contains

HEAVY

-

+

Build Time

is after

2 weeks weeks ago

-

+

Apply

Clear

Create Hyperlink

Nightly

Site	Build Name	Update	Configure		Build		Test			Start Time ▾	Labels
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass		
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	0	0	56	0	251	0	1	1796	21 hours ago	(19 labels)
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	0	0	56	0	251	0	0	1796	Jun 07, 2016 - 01:10 EDT	(19 labels)
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	0	0	56	0	251	0	1	1795	Jun 06, 2016 - 01:10 EDT	(19 labels)
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	0	0	56	0	251	0	0	1796	Jun 05, 2016 - 01:10 EDT	(19 labels)
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	0	0	56	0	251	0	0	1796	Jun 04, 2016 - 01:10 EDT	(19 labels)
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	0	0	56	0	251	0	1	1794	Jun 03, 2016 - 01:10 EDT	(19 labels)
	Linux-GCC-4.8.3-	1	0	56	0	251	0	0	1795	Jun 02, 2016 - 01:10 EDT	(19 labels)

CDash Queries: Killer Feature

Show failures for test CASL_MOOSE_bison_from_vera_ifba in last 4 weeks:

Query Tests: 5 matches

[Hide Filters](#)

Filters

[Help](#)

Match **all** of the following rules:

Test Name	is	CASL_MOOSE_bison_from_vera_ifba	-	+
Status	is not	Passed	-	+
Build Time	is after	4 weeks ago	-	+

Site	Build Name	Test Name	Status	Time	Details ▼	Build Time ▼
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED	CASL_MOOSE_bison_from_vera_ifba	Failed	227.75	Completed (Failed)	2016-06-08T06:05:15 EDT
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	CASL_MOOSE_bison_from_vera_ifba	Failed	183.38	Completed (Failed)	2016-06-08T01:53:38 EDT
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED	CASL_MOOSE_bison_from_vera_ifba	Failed	266.87	Completed (Failed)	2016-06-02T06:10:12 EDT
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	CASL_MOOSE_bison_from_vera_ifba	Failed	168.28	Completed (Failed)	2016-05-30T01:52:45 EDT
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	CASL_MOOSE_bison_from_vera_ifba	Failed	3.57	Completed (Failed)	2016-05-24T01:52:50 EDT

This test is failing randomly ☹

CDash Queries: Killer Feature

Show most expensive tests yesterday:

Query Tests: 12291 matches

[Hide Filters](#)

Filters

[Help](#)

Match **all** of the following rules:

Build Time is after 2 days ago

Build Time is before 1 day ago

Apply

Clear

Create Hyperlink

Site	Build Name	Test Name	Status	Time ▼	Details	Build Time
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	MPACT_exe_testProgression_Problems_9-mini	Passed	13111.8	Completed	2016-06-07T03:10:34 EDT
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	MPACT_exe_testProgression_Problems_8-mini	Passed	12943.4	Completed	2016-06-07T03:10:34 EDT
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	VeraAPImpact_p6a_mpack_dep	Passed	5739.74	Completed	2016-06-07T12:48:23 EDT
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	MPACT_exe_testMVS_ap1000_IFBAOnly	Passed	4886.6	Completed	2016-06-07T03:10:34 EDT
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	MPACT_exe_testMVS_ap1000_Region4	Passed	4106.07	Completed	2016-06-07T03:10:34 EDT
	Linux-GCC-4.8.3-MPI_RELEASE_SHARED_HEAVY	MPACT_exe_testMVS_ap1000_Region5	Passed	4012.66	Completed	2016-06-07T03:10:34 EDT



TriBITS Specialized CDash Support

TriBITS Core:

- Provides ready-made partitioning into CTest/CDash Subprojects (i.e. TriBITS Packages)
- Gives CDash regression emails for each Package (i.e. CDash Subproject)
- Exports XML file with all TriBITS Package dependencies

TriBITS CTest Driver Support:

- Generates CDashSubproject.xml file and automatically submits to CDash
- Determines set of changed packages for fast CI rebuilds and tests
- Posts updated commits from git pull as CDash build notes files



Multi-Repo Git Updates Shown on CDash

Displays pulled commits as “Notes” file for Subproject (Package) build

...

*** VERAINExt updates:

c23eff9: Add training slides

Author: Scott Palmtag <pmy@ornl.gov>

Date: Tue Jun 7 08:59:00 2016 -0400

A verain/docs/VERA-Input-Training-2012.pdf

A verain/docs/VERA-Input-Training-May2016.pdf

*** VERADData updates:

...



TriBITS Summary



Recent CMake/TriBITS Progress

- **Speed of configuring with CMake 3.3+**
 - CMake 3.3+ eliminates the cost of TriBITS dependency handling:
 - Example: The cost for building CASL VERA dependency graph of 467 SE packages and then processing enables/disables **dropped from 2 minutes with CMake 2.8.11 to less than 2.5 seconds for CMake 3.3**
 - CMake 3.6 (soon to be released) to be even faster!
- **TriBITS with CMake 3.3+:**
 - Reconfigure time for large CASL VERA coupled code Tiamat < 30 sec
- **CDash:**
 - Addressed numerous issues/defects with ORNL CASL Kitware contract



In Progress and Upcoming work

In-Progress Work:

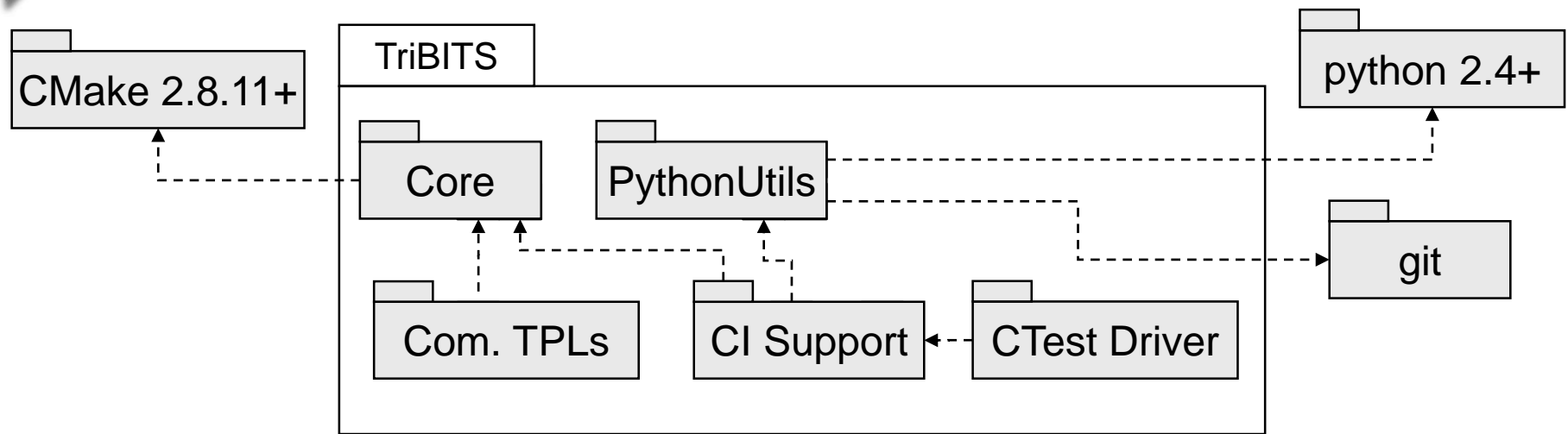
- Transition to use Ninja instead of Makefiles
 - Ninja Advantages:
 - Faster dependency analysis: E.g. Drekar dependency check [dropped from 2+ minutes to 2 seconds](#))
 - More parallelization of builds vs. recursive Makefiles
 - Ninja support for Fortran (fork exists, waiting for Google to accept)
 - Dummy Makefiles to make Ninja easy to use (same usage)
 - Address issues with Ninja and NVCC/CUDA
- Configure, build, and test all packages at once and submit to CDash

Upcoming Work:

- Combining concepts of packages and TPLs for large meta-projects ([TriBITS #63](#))
- High-level and tutorial documentation
- Enable usage of `git bisect` with gitdist (automate with checkin-test.py)
- A few of the backlog issues (see [TriBITS Issues](#) and [TriBITS Backlog](#))

Once these are done => TriBITS will be a good candidate for a universal meta-build and installation system for a **very** large amount of CSE software

TriBITS Partitioning and Dependencies



- **TriBITS Core** ([tribits/core/](https://tribits.org/tribits/core/)): Core TriBITS package-based architecture for CMake projects includes configure, build, test, install, deploy (tarballs) for multi-repo projects. (1M size)
- **TriBITS Python Utils** ([tribits/python_utils/](https://tribits.org/tribits/python_utils/)): Some basic Python utilities that are not specific to TriBITS (e.g. [gitdist](#), [snapshot_dir.py](#)).
- **TriBITS CI Support** ([tribits/ci_support/](https://tribits.org/tribits/ci_support/)): Support code for pre-push continuous integration testing (e.g. [checkin-test.py](#)).
- **TriBITS CTest Driver** ([tribits/ctest_driver/](https://tribits.org/tribits/ctest_driver/)): Support for package-by-package testing driven by CTest submitting to CDash (e.g. [TribitsCTestDriverCore.cmake](#)).
- **TriBITS Common TPLs** ([tribits/common_tpls/](https://tribits.org/tribits/common_tpls/)): Used by many independent TriBITS projects (e.g. [FindTPLBLAS.cmake](#), [FindTPLLAPACK.cmake](#), [FindTPLHDF5.cmake](#), ...)
- ...



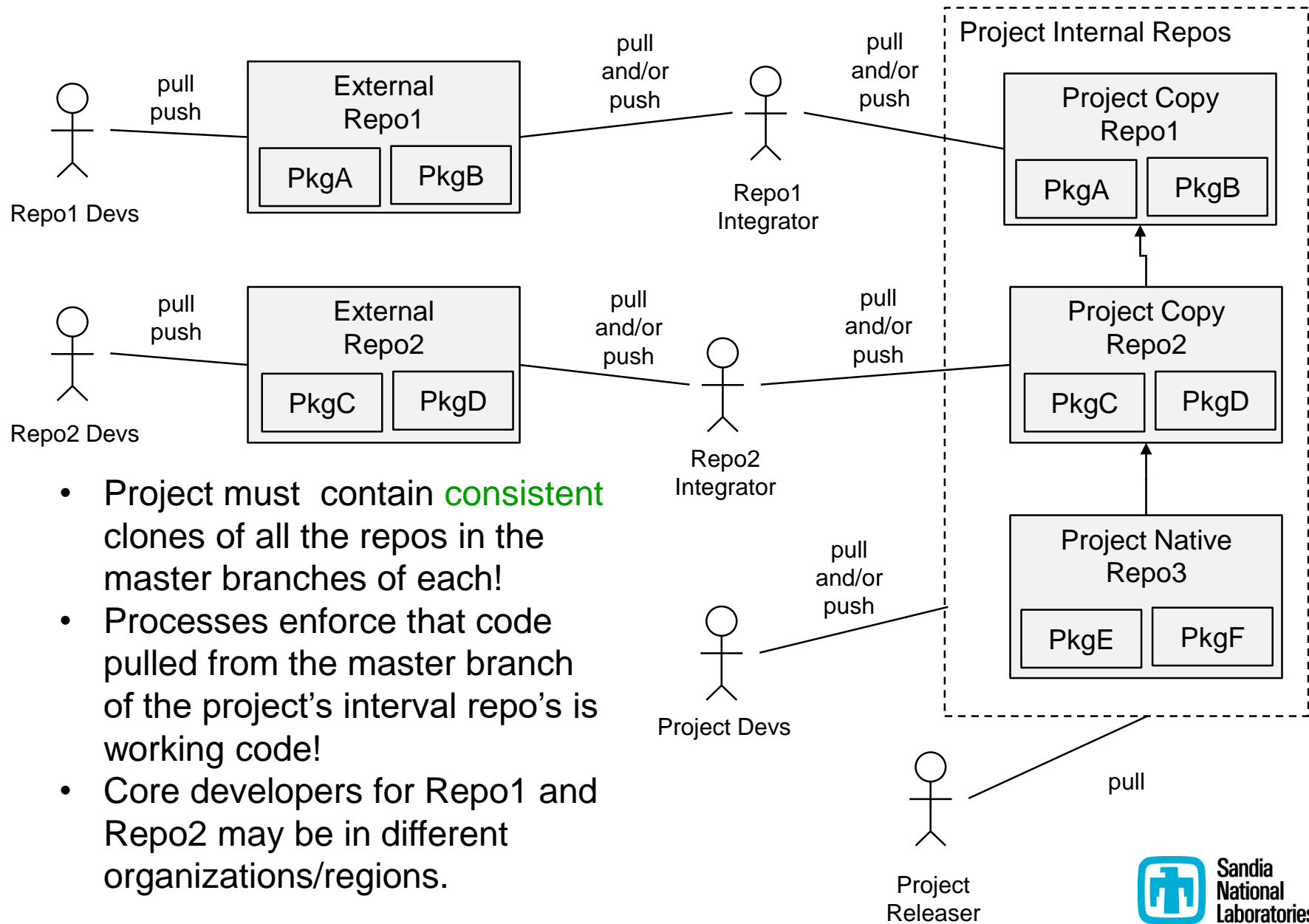
TriBITS Summary

- **TriBITS System Partitioning and Dependencies:**
 - TriBITS Core: Basic configure, build, test, install, and creating distributions
 - => Only requires raw CMake 2.8.11+
 - => 10K lines of CMake code (1M of disk space)
 - TriBITS CI Support (checkin-test.py, clone_extra_repos.py,...)
 - => Requires Git (1.7.0.4+) and Python 2.4
 - See TriBITS Developers Guide for more details (<http://tribits.org>)
- **Usage of TriBITS:**
 - Trilinos (SNL, originating project)
 - ORNL: SCALE, Exnihilo, DataTransferKit
 - Non-ORNL: MPACT (Univ. of Misc.), COBRA-TF (Penn. State)
 - CASL-Related: VERA
- **TriBITS Development & Distribution:**
 - 3-clause BSD-like license, Copyright SNL
 - Main source hosted on GitHub (<https://github.com/TriBITSPub/TriBITS>)
 - Documentation hosted on <http://tribits.org>
- **Near-term Future Work:**
 - Support for Ninja with Fortran, NVCC, etc.
 - More flexibility on pre-building packages and linking in as TPLs
 - Define a standard installation of TriBITS
 - Put out a TriBITS release (release with CMake in the future?)
 - Finish overview document and tutorials
 - More error checking to catch user mistakes



Multi-Repository Integration Models and Processes

Integrating Repos into Project: External and Internal

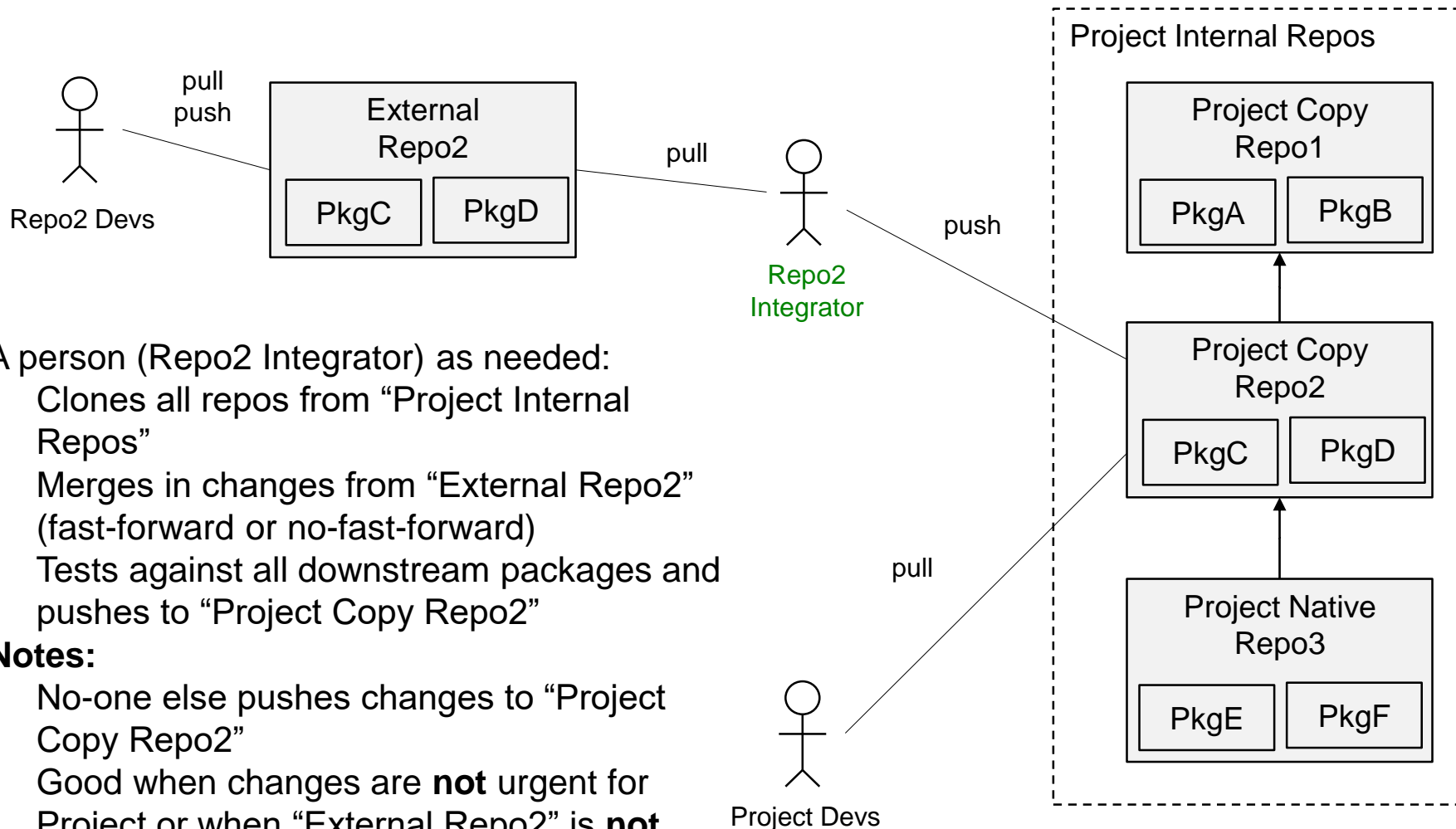




Multi-Repository Integration Models

- Range of development and sync models (external dev to internal dev):
 - External repo is manually synced into project/master as needed.
 - External repo is synced automatically using sync server into project/master.
 - Both external and internal repos pushed to by different development groups with sync servers running one way or both ways.
 - Internally managed repo is synced to an external repo on some schedule to make available to other developers and users and changes from external repo may or may not be synced back into internal repo.
 - Internally managed repo not shared externally
- A given repo may shift between different integration models at different periods of time (e.g. Trilinos, COBRA-TF, VERAIn)
- Integration of different repos should be done independently if possible (e.g. errors trying to integrate new MPACT version should not stop integration of new versions of SCALE/Exnihilo and visa versa).
- Non-backward compatible changes for multiple repos require coordinated development and combined pushing to project/master

External Repo is manually synced



A person (Repo2 Integrator) as needed:

- Clones all repos from “Project Internal Repos”
- Merges in changes from “External Repo2” (fast-forward or no-fast-forward)
- Tests against all downstream packages and pushes to “Project Copy Repo2”

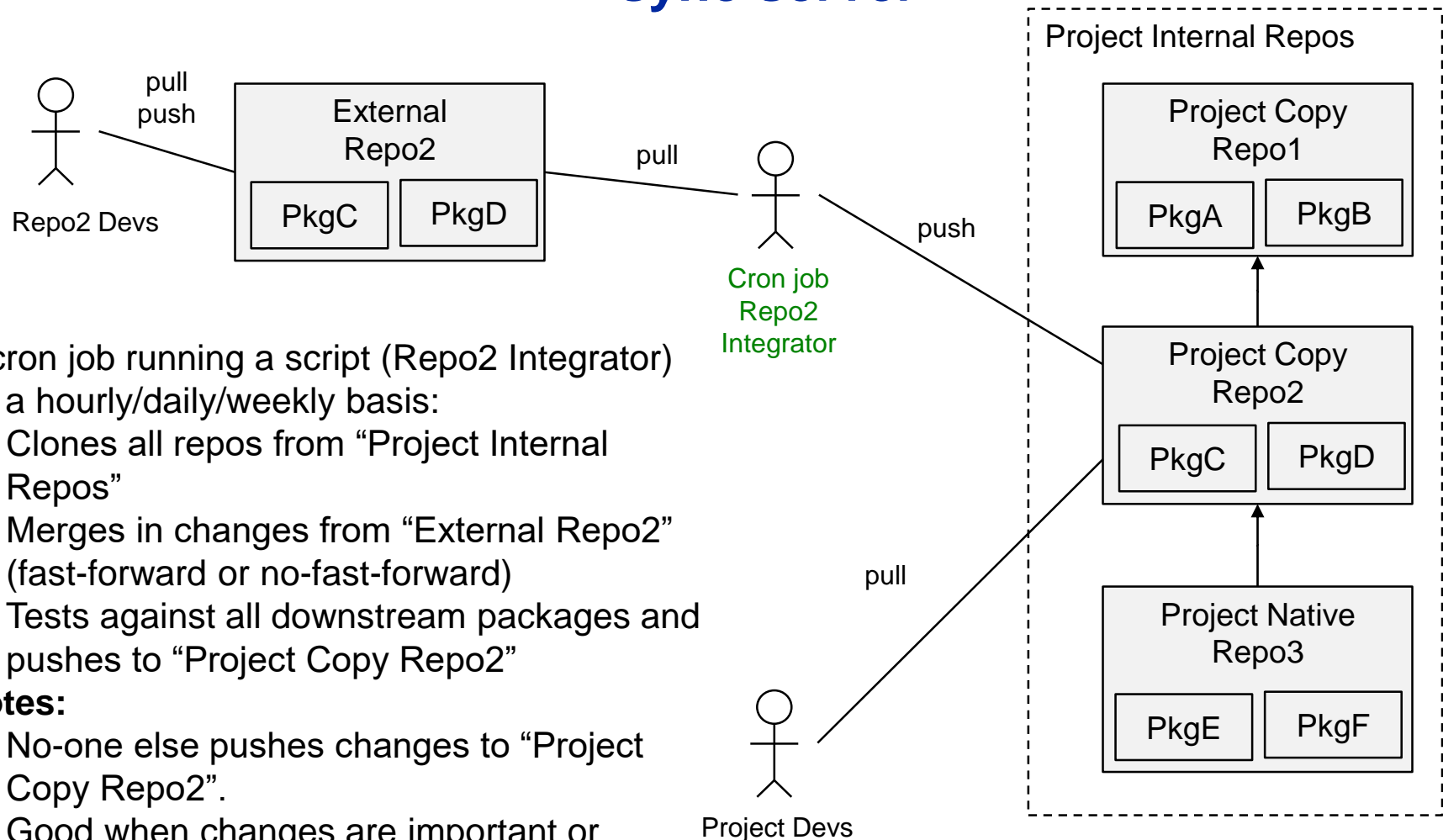
Notes:

- No-one else pushes changes to “Project Copy Repo2”
- Good when changes are **not** urgent for Project or when “External Repo2” is **not** stable

VERA Repos: Trilinos, DataTransferKit,

MAMBA

External repo is synced automatically using sync server



A cron job running a script (Repo2 Integrator) on a hourly/daily/weekly basis:

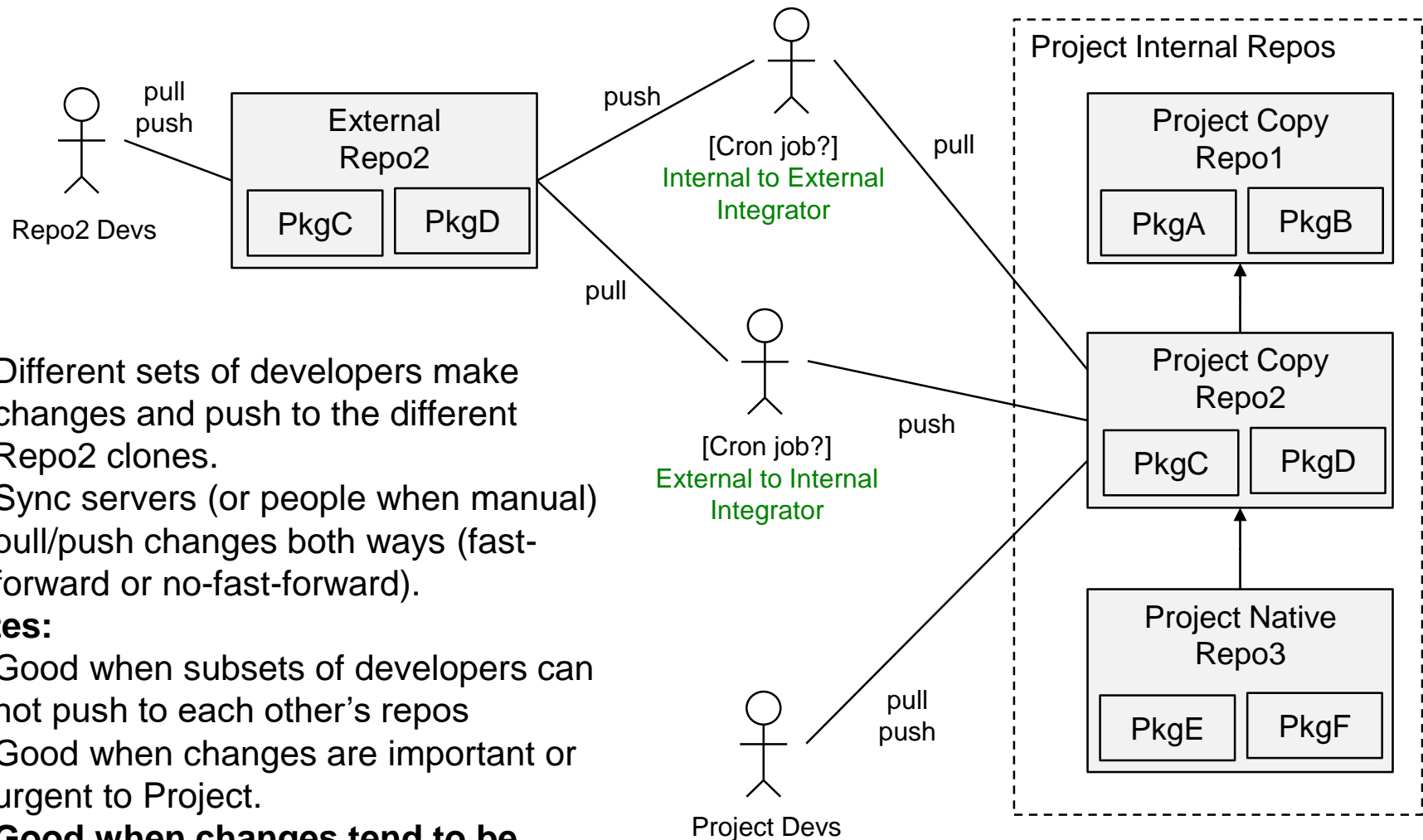
- Clones all repos from “Project Internal Repos”
- Merges in changes from “External Repo2” (fast-forward or no-fast-forward)
- Tests against all downstream packages and pushes to “Project Copy Repo2”

Notes:

- No-one else pushes changes to “Project Copy Repo2”.
- Good when changes are important or urgent to Project and “External Repo2” is fairly stable.

VERA Repos: SCALE/Exnihilo, MPACT, MOOSE/Bison

Both external and internal repos pushed to



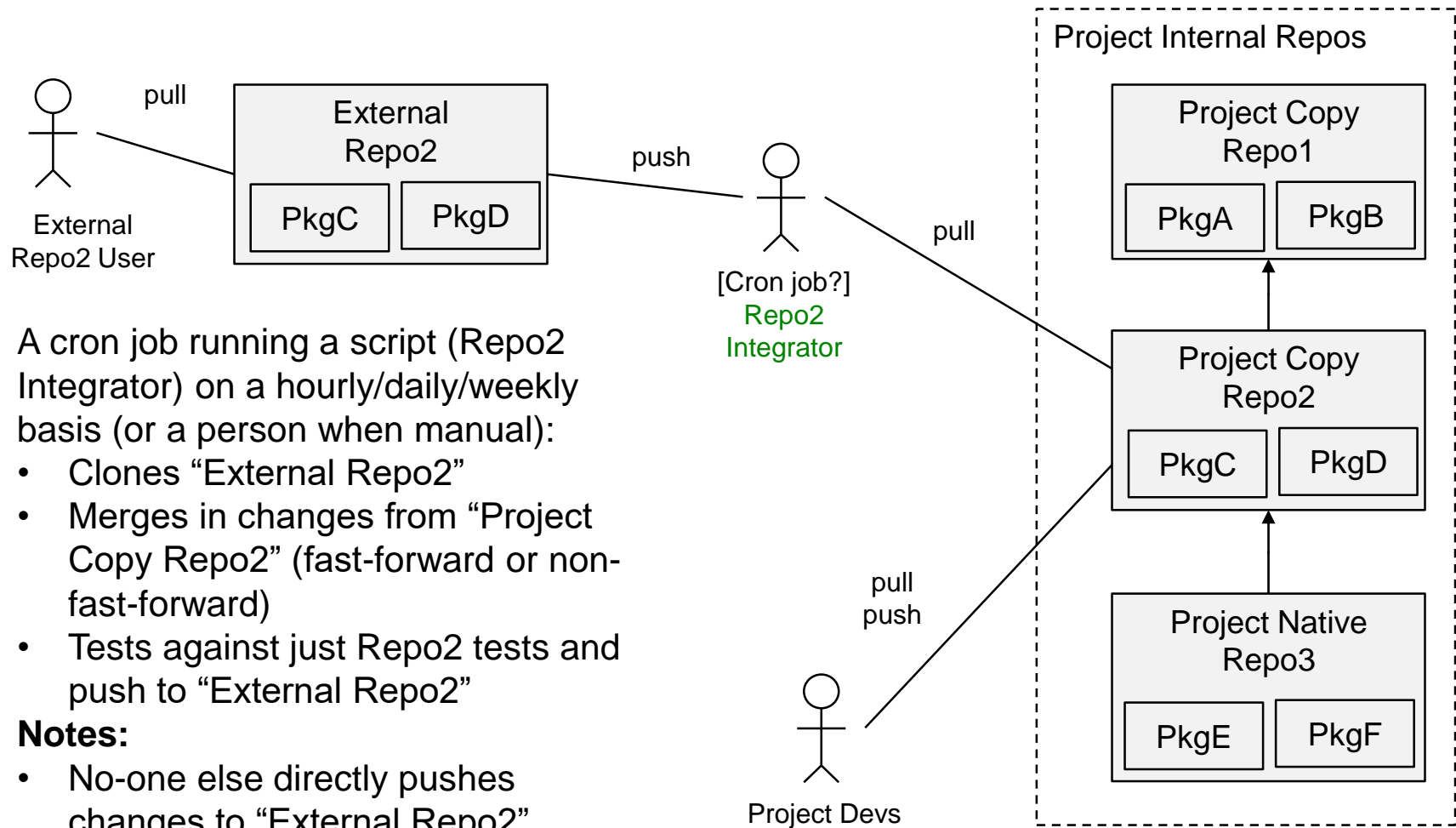
- Different sets of developers make changes and push to the different Repo2 clones.
- Sync servers (or people when manual) pull/push changes both ways (fast-forward or no-fast-forward).

Notes:

- Good when subsets of developers can not push to each other's repos
- Good when changes are important or urgent to Project.
- **Good when changes tend to be independent** made to Internal and External repos.
- **Most complex and danger of merge conflicts that someone has to resolve!**

VERA Repos: TriBITS
(sometimes **COBRA-TF, MPACT, Trilinos, ...**)

Internally managed repo is synced to an external repo



A cron job running a script (Repo2 Integrator) on a hourly/daily/weekly basis (or a person when manual):

- Clones “External Repo2”
- Merges in changes from “Project Copy Repo2” (fast-forward or non-fast-forward)
- Tests against just Repo2 tests and push to “External Repo2”

Notes:

- No-one else directly pushes changes to “External Repo2”
- Good when you just want to make changes available to external users on a continuous/frequent basis.

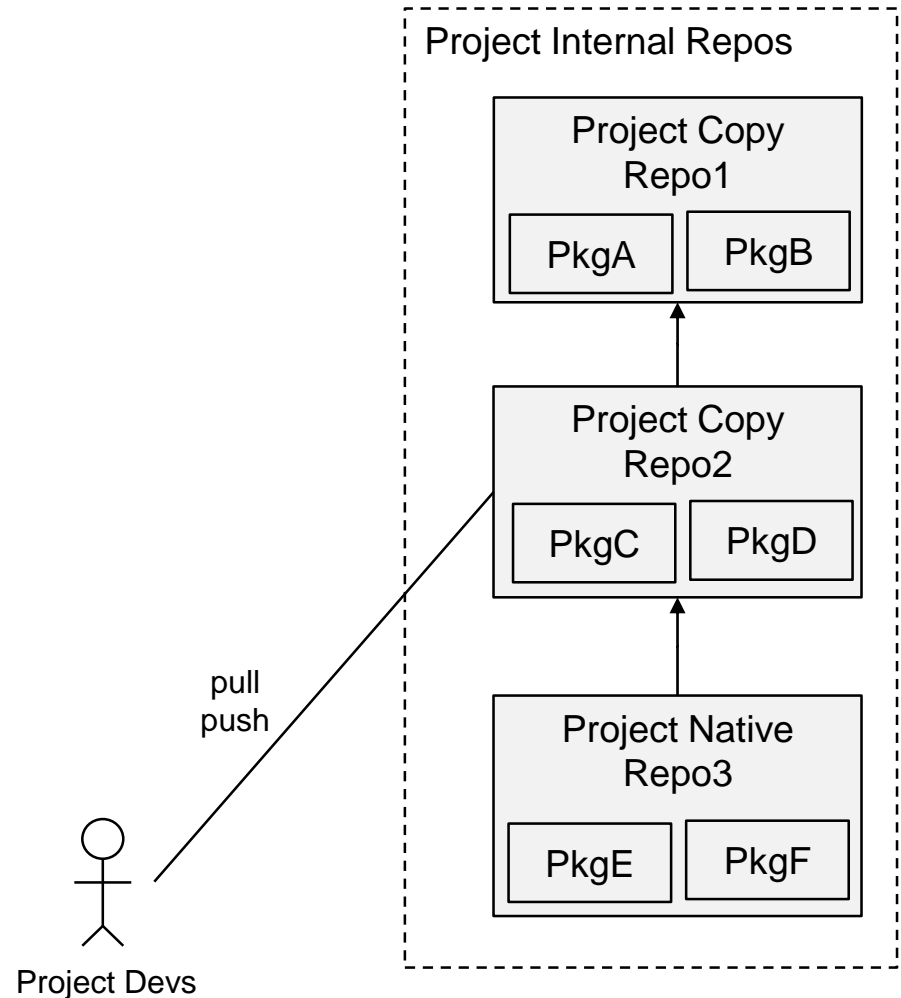
VERA Repos: COBRA-TF, TeuchosWrappersExt, VERAIn

Internally managed repo

- Simple single-repo development model

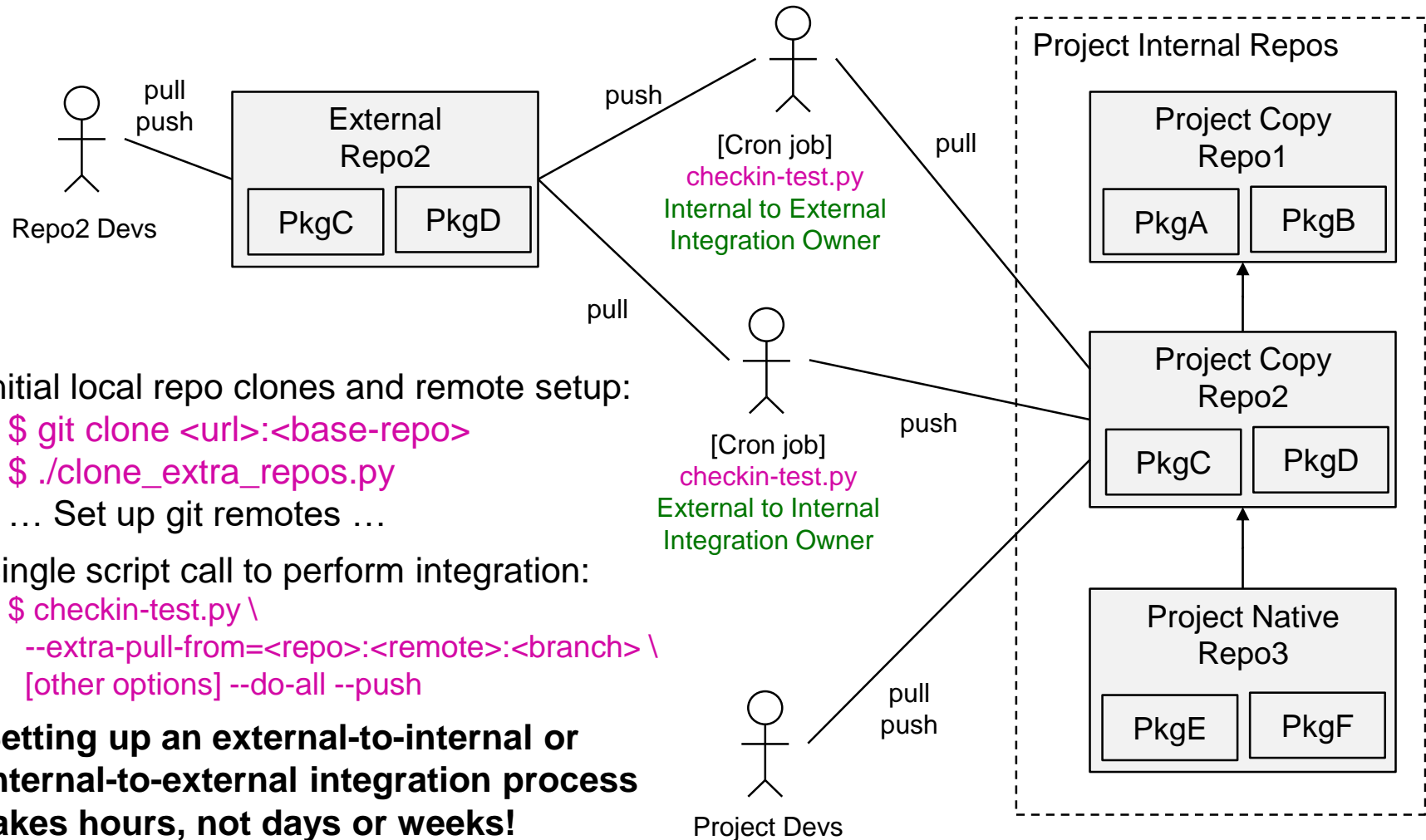
Notes:

- Good when you don't need to coordinate with developers outside of the Project



VERA Repos: MOOSEExt, PSSDriversExt ,
DakotaExt, VUQDemos

TriBITS Multi-Repository Integration Processes Support



- Initial local repo clones and remote setup:
`$ git clone <url>:<base-repo>`
`$./clone_extra_repos.py`
... Set up git remotes ...
- Single script call to perform integration:
`$ checkin-test.py \`
`--extra-pull-from=<repo>:<remote>:<branch> \`
`[other options] --do-all --push`
- Setting up an external-to-internal or internal-to-external integration process takes hours, not days or weeks!**
- Integration Owners** for the integration processes watch over the process, triage issues, and make sure they are addressed.



THE END