July 28, 2025

2025 IRUG Technical Seminar

Upgrades to RELAP5-3D Version Control, Installation, and Testing Systems

Battelle Energy Alliance manages INL for the
U.S. Department of Energy's Office of Nuclear Energy

Idaho National Laboratory

RELAP5-3D Modernization

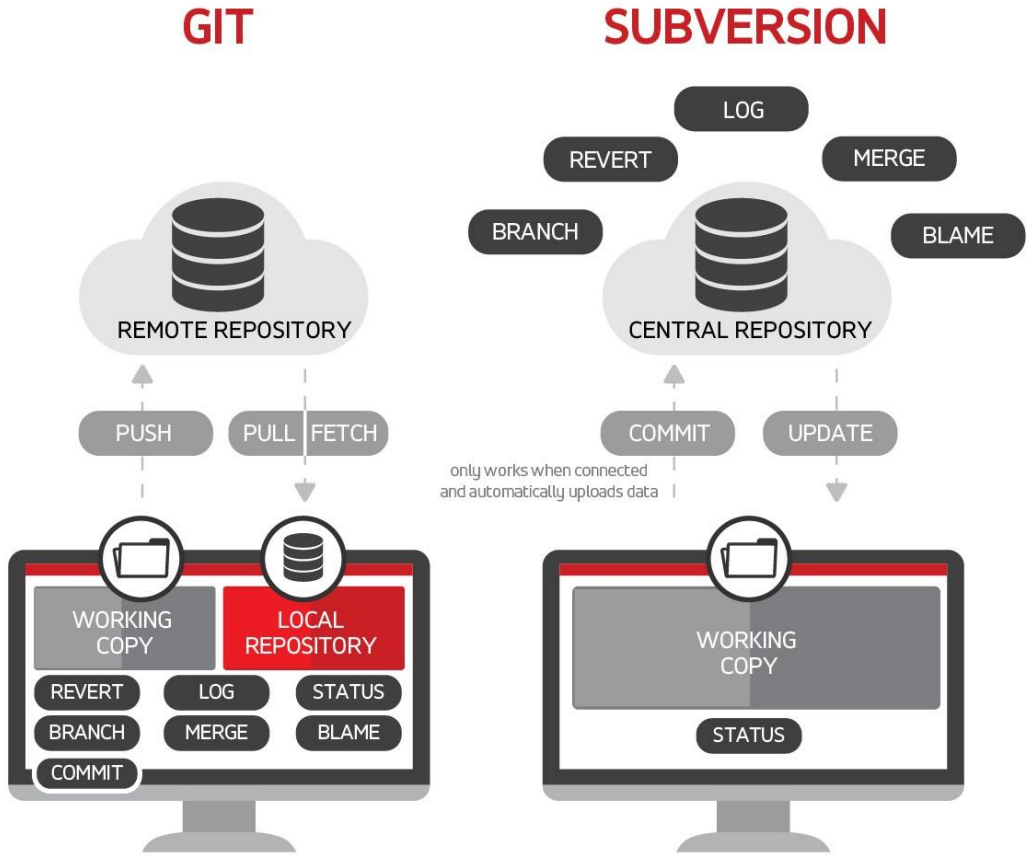
- RELAP5-3D has long been known for its robustness, reliability, and accuracy
- This is due to great coding standards and practices, and thorough verification and validation
- As the computing industry evolves, so too must programs that want to endure
- Having endured decades, RELAP5-3D required some modernization
- Keeping up with industry best practices:
  - Increases maintainability
  - Improves code quality
  - Enhances collaboration
  - Ensures the future of the software

Systemic Updates to RELAP5-3D

- Recently major updates have been made to the following aspects of RELAP5-3D:
  - Version control
    - Switched from Subversion to Git
  - Build system
    - Implemented CMake system
  - Test system
    - Developed in-house Python object-oriented system
    - Implemented continuous integration into development process

Benefits of Git over Subversion (SVN)

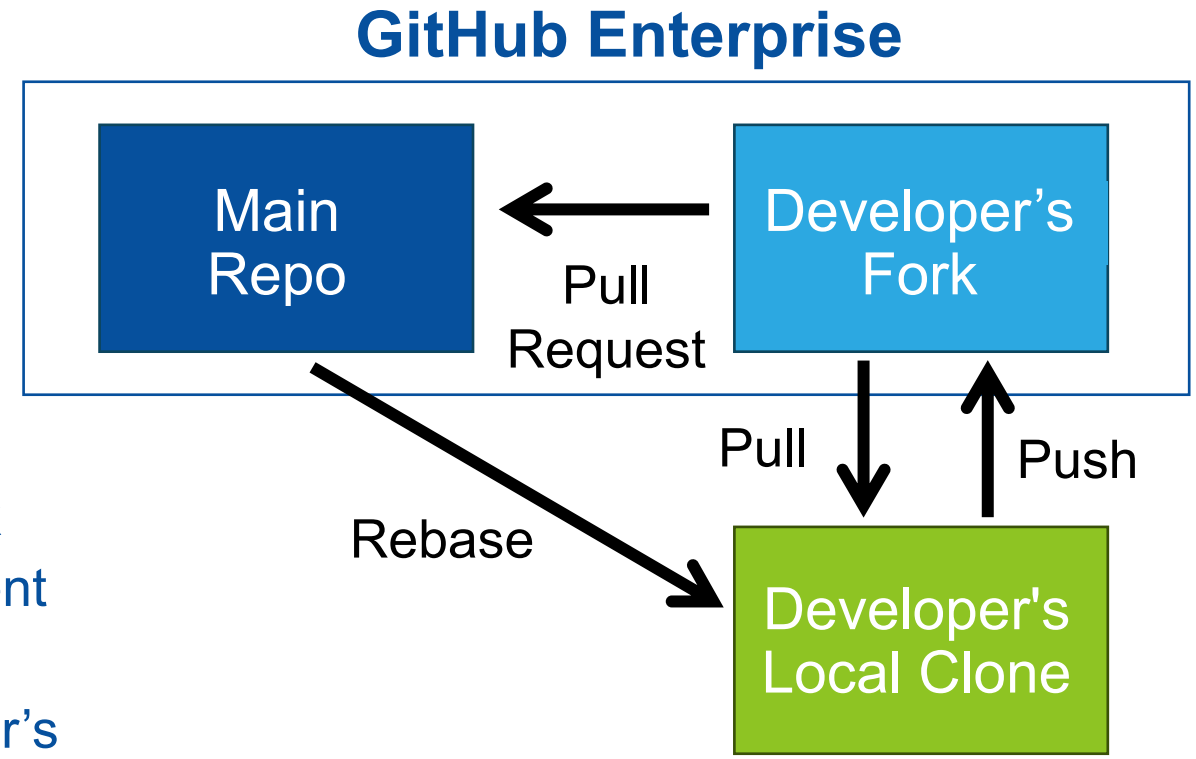
- Each developer has a full copy of the repository (repo)
  - Local operations are faster
  - Offline work
- Greater collaboration
- More flexible workflow
- Powerful conflict resolution tools
- Active community



http://anoneh.com/417.php

Git Collaboration

- Each developer creates a fork (private copy) of the entire main repo that resides remotely on GitHub
- The developer clones (downloads an identical copy of) their remote fork to a local repo
- Changes are made and tested in the local fork repo
- They can have one of 3 states:
  - Modified files are in the working area but not yet staged
  - Staged via “git add”. Not yet committed to local fork
  - Committed via “git commit” are files made permanent in the local fork
- Push via “git push” uploads committed files to developer’s remote fork
- Create Pull Request (PR) to transfer from developer’s remote fork into main repo



New Code Modification Process

Previous SVN Process

- Code custodian provides latest source code to programmer
- Programmer develops and tests updates
- Programmer provides all modified files to code custodian
- Code custodian handles merge conflicts with other's changes
- Code custodian pushes changes from all developers to SVN repository

New Git Process

- Code custodian manages access to the main remote Git repo
- Programmer makes fork of repo
- Programmer utilizes branches to implement and test changes
- Programmer creates pull request to upload updates to remote Git repo
- Code custodian and other approved programmers review changes
- Once changes are approved, updates are merged into the main branch.

Build System Update – Migration to CMake

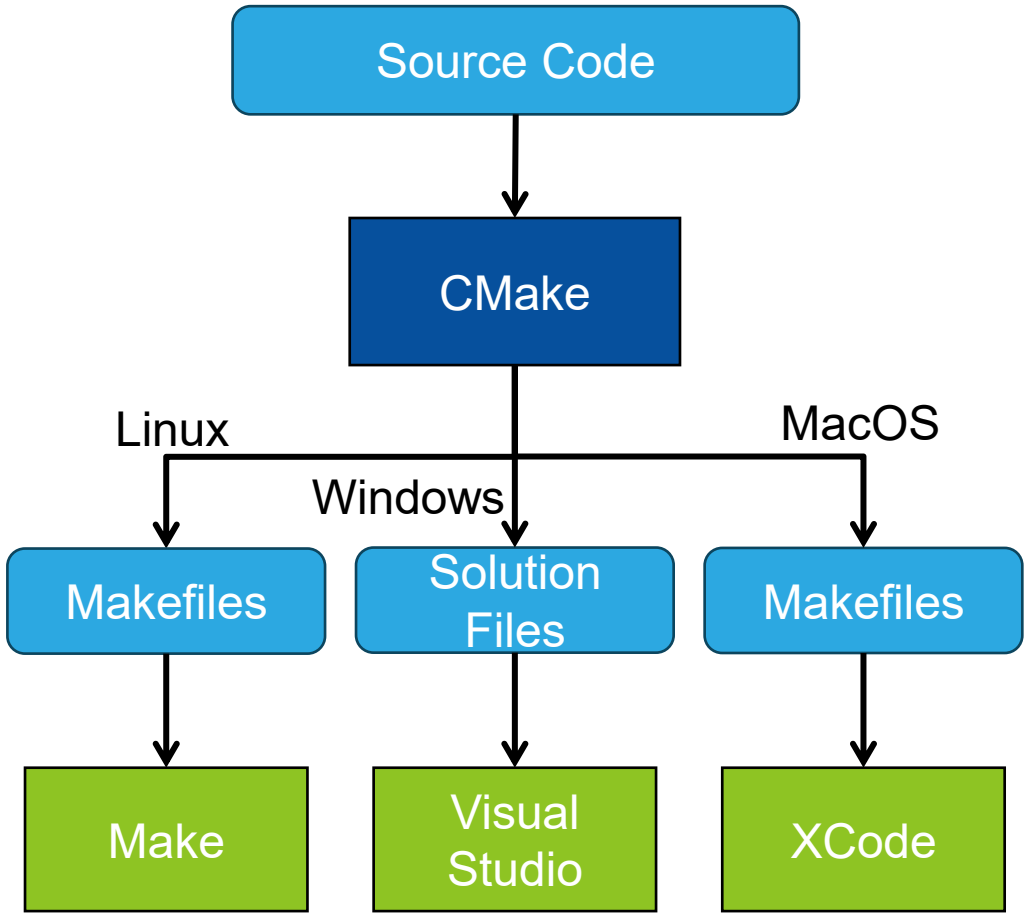
- Converted from a system of Makefiles and shell scripts to CMake
- Information System Laboratories (ISL) developed this CMake build system
  - Huge thanks to Doug Barber, Collin Leavitt, and the ISL team
- Major Benefits:
  - Streamlines configuration of build process
  - Simplifies cross platform compilation
    - Linux
    - Windows
    - MacOS

Historic Build System

- System of Make files and shell scripts
  - Manually define source code structure and dependencies in the make files
  - Use shell scripts to handle environment setup, pre-compilation tasks, and run the make system
  - Custom rules and scripts for every source code directory
- System became streamlined through years of use and adjustments
  - Hard for new developers to learn
- Windows compilation:
  - Extensive work to configure Microsoft visual studio solution files to compile on windows

CMake Build System

- CMake is a meta-build system
  - It generates system configuration files which are then compiled by the native build system
- Advantages:
  - One system works for all environments
  - Simpler to configure specific settings for folders and files
    - User only specifies project structure
    - Automatically manages dependencies
  - Out-of-source builds
    - Cleaner more organized
    - Allows multiple build configurations



Time savings with CMAKE

- Time savings from Cmake are due to:
  - Parallelization
  - Ensures optimal compiler settings are used
  - Utilizes precompiled modules
- The table below shows the time for compilation of the Total and IRUG version using the historic build system (dinstls) and CMake with 1, 3, and 20 processors

Version	Dinstls	Cmake – 1 processor	Cmake 3 – processors	Cmake – 20 processors
Total	8 min 21 s	6 min 54 s	2 min 26 s	46 s
IRUG	7 min 46 s	6 min 45 s	2 min 13 s	48 s

RELAP5-3D Extensive Testing

- More than 2,600 test cases, divided into 18+ different test suites
- Base, Athena, and Other test suites are the standard installation tests
- Verification test suite verifies 200 code capabilities are correct
- The Development Assessment performs Validation and Verification tests

Base cases	KinDt	TestBp
Athena	Merror	TestDt
Other	Mstable	TestMatrixDt
Extra	Nkerr	Verificaton_suite
Development Assessment	Nodal_Kinetics	Dissolved gas model
FlexWall	Pvm	Other proprietary tests

Historic Test Procedures

- Each test suite contains shell script to run the tests
- Most contain additional scripts that analyze the results
- Programmers would run standard installation tests to check changes
- Code Custodian runs all test suites to verify and validate new release
  - Test total version
  - Create client versions from total
  - Test all client versions
  - If errors are found – fix and re-test all versions
- This took weeks to complete for an externally released code

New Test System

- Runs tests in parallel
  - Drastically decreases test time
- Standardized format for all test setup in YAML format
- Each test requires:
  - Input deck to run
  - Checks to evaluate the run
- Checks are separate from the test script
  - Once a check is developed, can be applied to any test in any test suite
- Evaluation if test passed or failed is performed automatically

```
[1]  test/tests/base_cases/hex2dk_dt0  ..........
.....................................................................
.....................................................  Passed 7s
[1]  test/tests/base_cases/Athena/pb_ss_air  ....
.....................................................................
.....................................................  Passed 4s
[1]  test/tests/base_cases/scw_ni  ..............
.....................................................................
.....................................................  Passed 8s
[1]  test/tests/base_cases/typ12002  ............
.....................................................................
.....................................................  Passed 8s
[1]  test/tests/base_cases/typ1200295n  .........
.....................................................................
.....................................................  Passed 8s
[1]  test/tests/base_cases/typ12002_ni  .........
.....................................................................
.....................................................  Passed 8s
[1]  test/tests/base_cases/Other/tank  ..........
.....................................................................
.....................................................  Passed 5s
[1]  test/tests/base_cases/typ1200n2  ...........
.....................................................................
.....................................................  Passed 9s
Done executing tests with class in: ['short']

Elapsed time            : 13.65 seconds
Number of tests run     : 218
Number of tests skipped : 12
Number of failed tests  : 0
```

Continuous Integration

- CIVET (Continuous Integration, Verification, Enhancement, and Testing)
- Connects to repository on GitHub Enterprise
- Test's automatically start when pull request is initialized or modified
- PR cannot be merged without the tests passing
- Ensures modifications don't break existing functionality

Conclusion

- The modernization of RELAP5-3D is essential to it's continued excellence
- The updates presented here:
  - Increase maintainability
  - Streamline code modification process
  - Utilize powerful industry standard tools

Idaho National Laboratory

Battelle Energy Alliance manages INL for the U.S. Department of Energy's Office of Nuclear Energy. INL is the nation's center for nuclear energy research and development, and also performs research in each of DOE's strategic goal areas: energy, national security, science and the environment.

Git Flexible Workflow

- The following Git features provide a more flexible workflow than SVN
- A branch is a copy of the code that allows developers to:
  - Isolate work – make developments without affecting main codebase
  - Make modifications in parallel with multiple developers or in parallel with developer's own separate projects
  - Aids the merging of changes made by multiple developers
- Staging area allows greater control of what changes get committed
  - git add marks specified files to be committed
- While both Git and SVN have "commits":
  - SVN commits push code directly to remote repository
  - Git commits are applied locally
  - Developers can create multiple commits before merging with codebase