



# Git Workflows & Continuous Integration

Better Scientific Software Tutorial

Jared O'Neal

Argonne National Laboratory

ECP Annual Meeting

January 14, 2019



See slide 2 for  
license details

# License, citation, and acknowledgments



## License and Citation

- This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/) (CC BY 4.0).
- Requested citation: Jared O'Neal, Git Workflows & Continuous Integration, in Better Scientific Software Tutorial, Exascale Computing Project Annual Meeting, Houston, Texas, 2019. DOI: [10.6084/m9.figshare.7581746](https://doi.org/10.6084/m9.figshare.7581746)

## Acknowledgements

- This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.
- This work was performed in part at the Argonne National Laboratory, which is managed by UChicago Argonne, LLC for the U.S. Department of Energy under Contract No. DE-AC02-06CH11357
- Anshu Dubey, Klaus Weide, Saurabh Chawdhary, and Carlo Graziani
- Iulian Grindeanu
- Alicia Klinvex



# Git Workflows

# Goals

Development teams would like to use version control to collaborate productively and ensure correct code

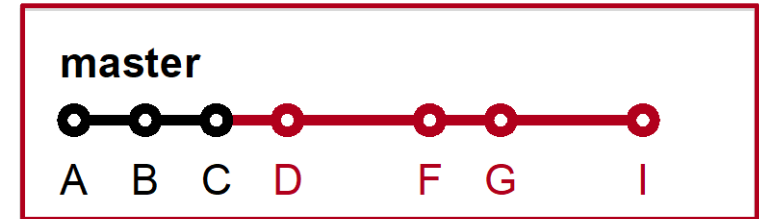
- Understand challenges related to parallel code development *via* distributed version control
- Understand extra dimensions of distributed version control & how to use them
  - Local vs. remote repositories
  - Branches
  - Issues, Pull Requests, & Code Reviews (earlier talk)
- Exposure to workflows of different complexity
- What to think about when evaluating different workflows
- Motivate continuous integration

# Distributed Version Control System (DVCS)

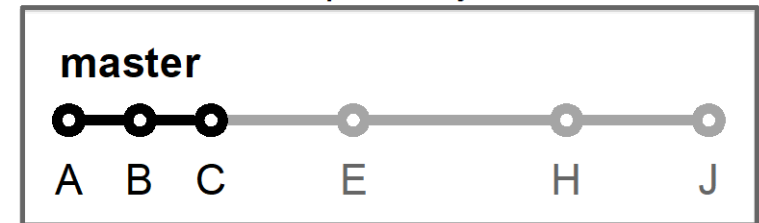
Two developers collaborating *via* Git

- Local copies of master branch synched to origin
- Each develops on **local** copy of master branch
- All copies of master immediately diverge
- How to **integrate** work on origin?

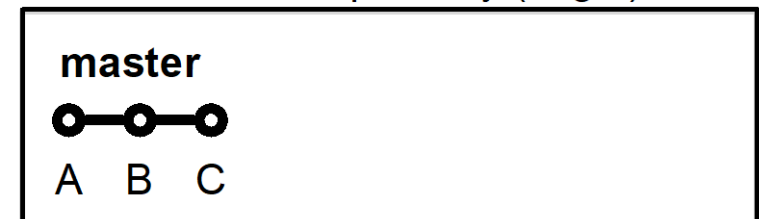
Alice's Local Repository



Bob's Local Repository



Main Remote Repository (origin)



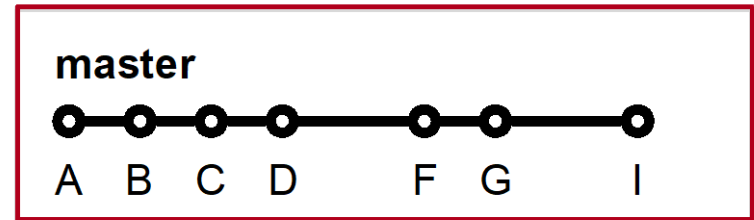
● = commit      — = branch  
X = commit ID

# DVCS Race Condition

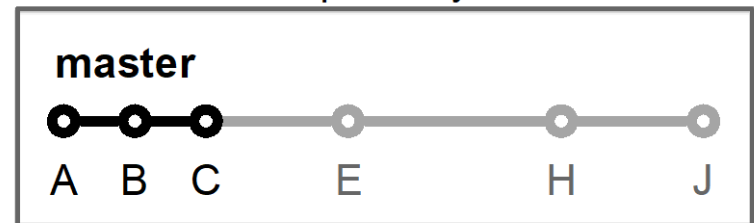
Integration of independent work occurs when local repos interact with remote repo

- Alice pushes her local commits to remote repo first
- No integration conflicts
- No risk
- Alice's local repo identical to remote repo

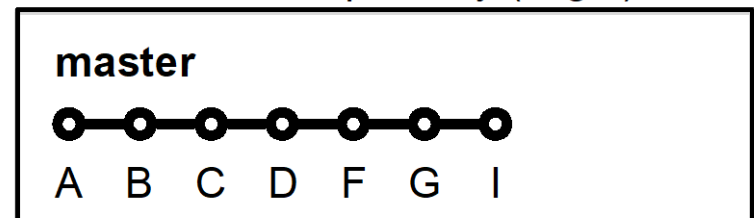
Alice's Local Repository



Bob's Local Repository



Main Remote Repository (origin)



● = commit      — = branch  
X = commit ID

# Integration Conflicts Happen

Bob's push to remote repo is rejected

- Alice updated code in commit D
- Bob updated same code in commit E
- Alice and Bob need to study conflict and decide on resolution at pull (time-consuming)
- Possibility of introducing bug on master branch (risky)

loops.cpp (commit C)

```
36
37 // TODO: Code very important loop here ASAP
38
39 ...
40
41
42
43 // TODO: Code other very important loop here ASAP
44
```

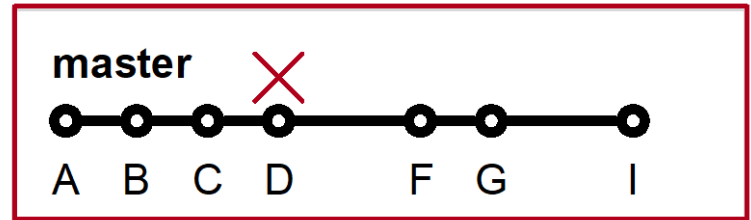
loops.cpp (commit D)

```
36
37 // Very important loop
38 for (int i=0; i<N; ++i) {
39     ...
40     ...
41
42 // Another very important loop
43 for (int i=1; i<=N; ++i) {
44     foo[i] = bar[i] * i;
45 }
```

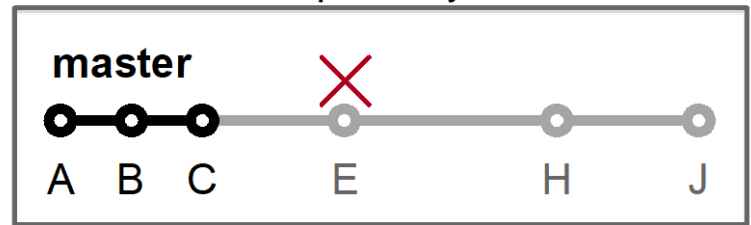
loops.cpp (commit E)

```
36
37 // Very important loop
38 for (int i=0; i<N; i++) {
39     ...
40     ...
41
42 // Another very important loop
43 for (int i=0; i<N; i++) {
44     foo[i] = bar[i] * i;
45 }
```

Alice's Local Repository



Bob's Local Repository



# Our First Workflow

This process of collaborating *via* Git is called the **Centralized Workflow**

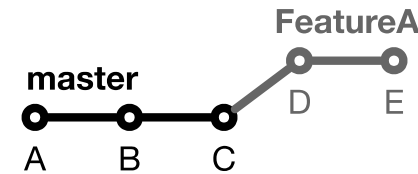
- See [Atlassian/BitBucket](#) for more information
- “Simple” to learn and “easy” to use
- Leverages local vs. remote repo dimension
  - Integration in local repo when local repos interact with remote repo
- What if you have many team members?
- What if developers only push once a month?
- What if team members works on different parts of the code?
- Working directly on master



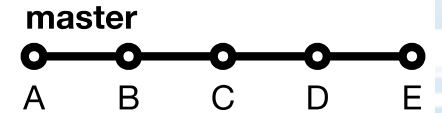
# Branches

Branches are independent lines of development

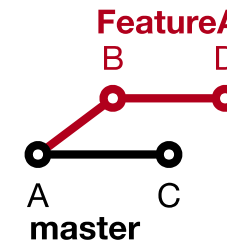
- Use branches to protect master branch
- Feature branches
  - Organize a new feature as a sequence of related commits in a branch
- Branches are usually combined or **merged**
- Develop on a branch, test on the branch, and merge into master
- Integration occurs at merge commits



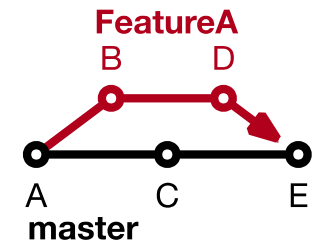
Fast-Forward



No Merge



Divergence

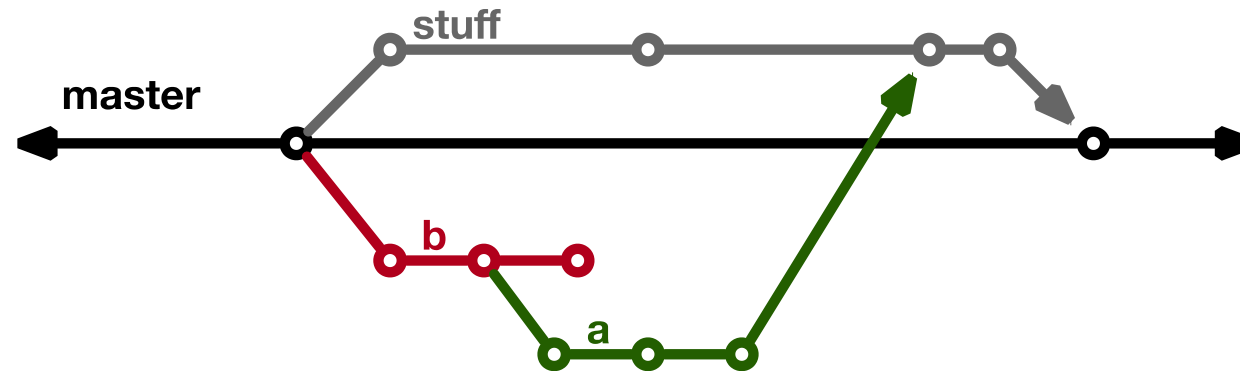


Merge Commit

# Control Branch Complexity

Workflow policy is needed

- Descriptive names or linked to issue tracking system
- Where do branches start and end?
- Can multiple people work on one branch?

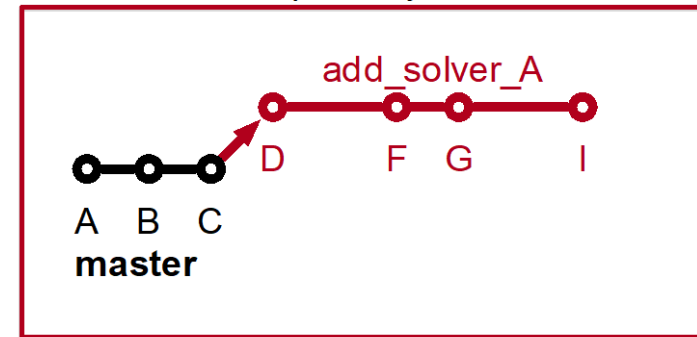


# Feature Branches

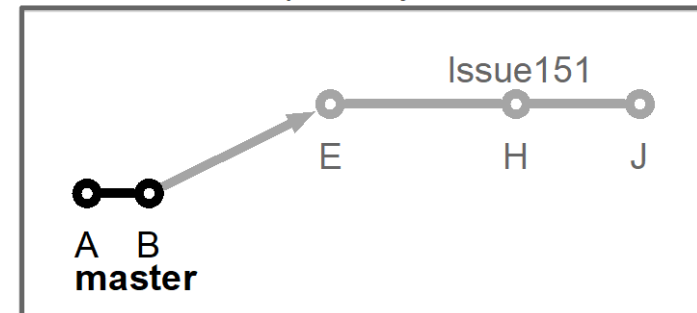
## Extend Centralized Workflow

- Remote repo has commits A & B
- Bob pulls remote to synchronize local repo to remote
- Bob creates local feature branch based on commit B
- Commit C pushed to remote repo
- Alice pulls remote to synchronize local repo to remote
- Alice creates local feature branch based on commit C
- Both develop independently on local feature branches

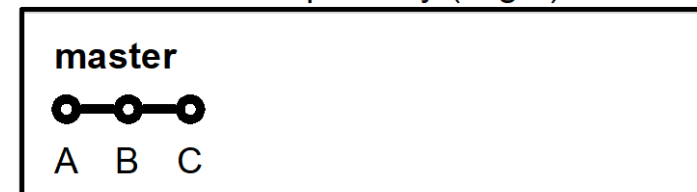
Alice's Local Repository



Bob's Local Repository



Main Remote Repository (origin)

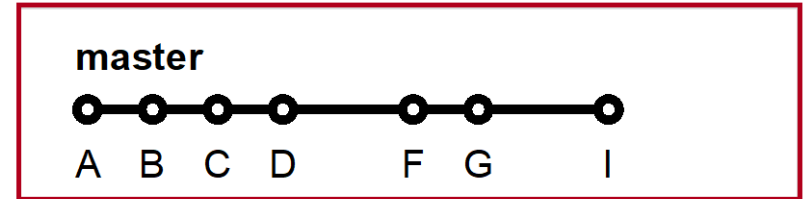


# Feature Branch Divergence

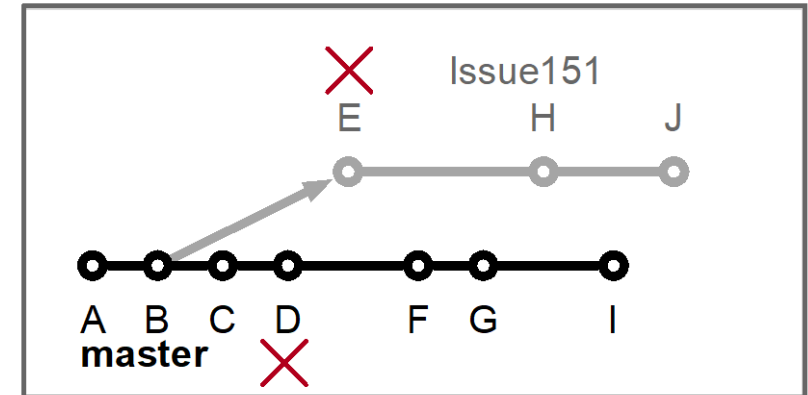
Alice integrates first without issue

- Alice does fast-forward merge to local master
- Alice deletes local feature branch
- Alice pushes master to remote
- Meanwhile, Bob pulls master from remote and finds Alice's changes
- Merge conflict between commits D and E

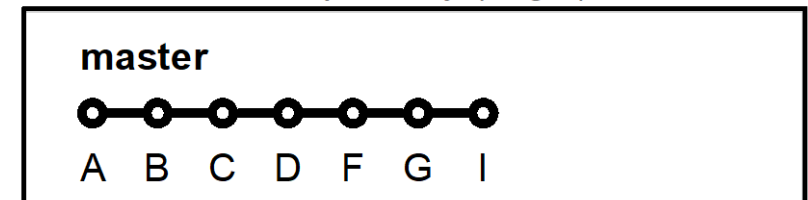
Alice's Local Repository



Bob's Local Repository



Main Remote Repository (origin)

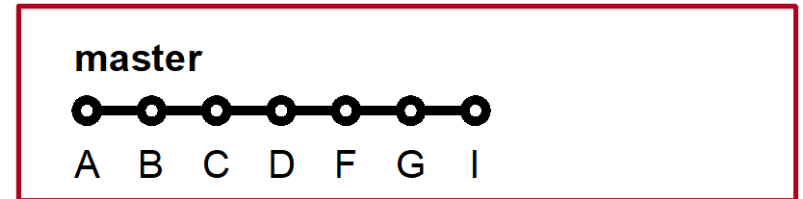


# Feature Race Condition

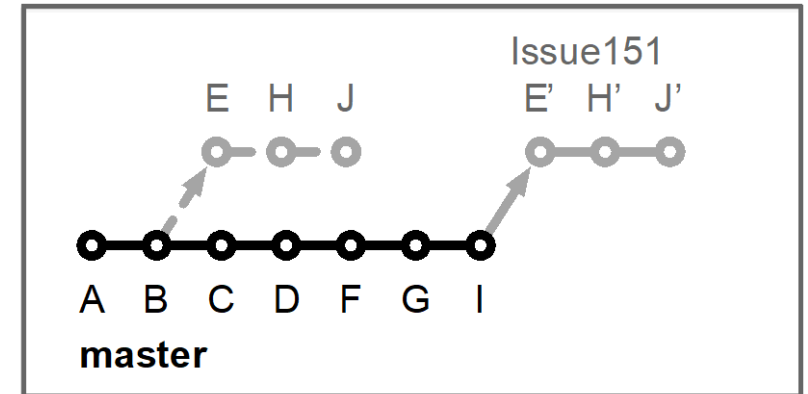
Integration occurs on Bob's local repo

- Bob laments not having fast-forward merge
- Bob **rebases** local feature branch to latest commit on master
  - E based off of commit B
  - E' based off of Alice's commit I
  - E' is E integrated with commits C, D, F, G, I
- Merge conflict resolved by Bob & Alice on Bob's local branch when converting commit E into E'
- Can test on feature branch and merge easily and cleanly

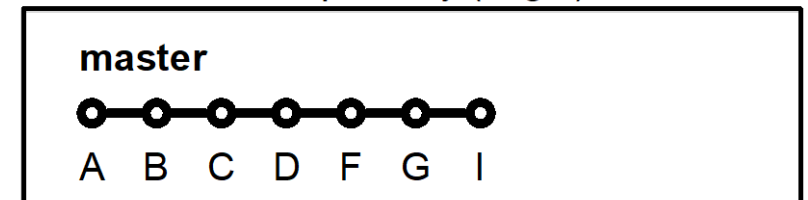
Alice's Local Repository



Bob's Local Repository



Main Remote Repository (origin)



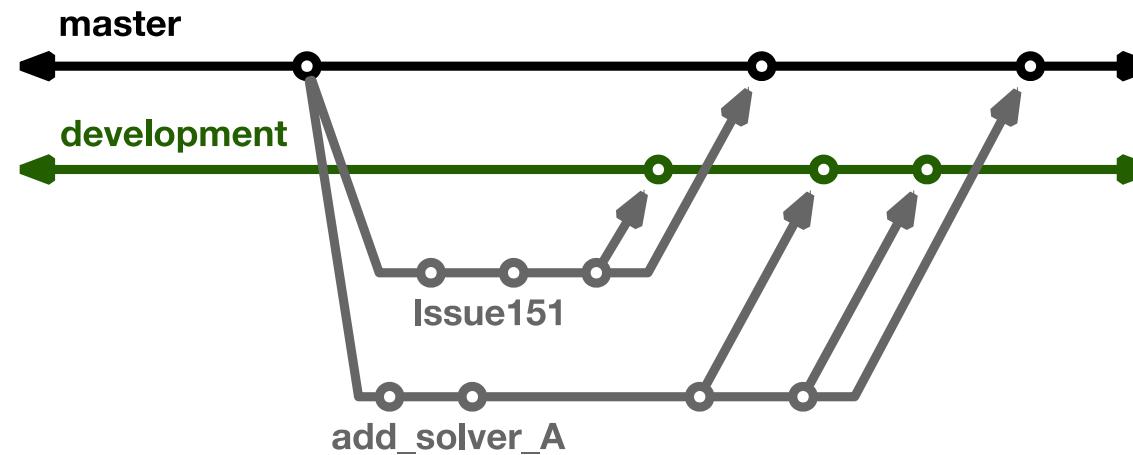
# Feature Branches Summary

- Multiple, parallel lines of development possible on single local repo
- Easily maintain local master up-to-date and useable
- Integration with rebase on local repo is safe and can be aborted
- Testing before updating local and remote master branches
- Rebase is advanced Git command
  - Rebase can cause complications and should be used carefully.
- Hide actual workflow
  - History in repo is not represent actual development history
  - Less communication
  - Fewer back-ups using remote repo
- Does it scale with team size? What if team integrates frequently?
- Commits on master can be broken
- See [Atlassian/BitBucket](#) for a richer Feature Branch Workflow

# More Branches

Branches with infinite lifetime

- Base off of master branch
- Exist in all copies of a repository
- Each provides a distinct **environment**
  - Development vs. pre-production



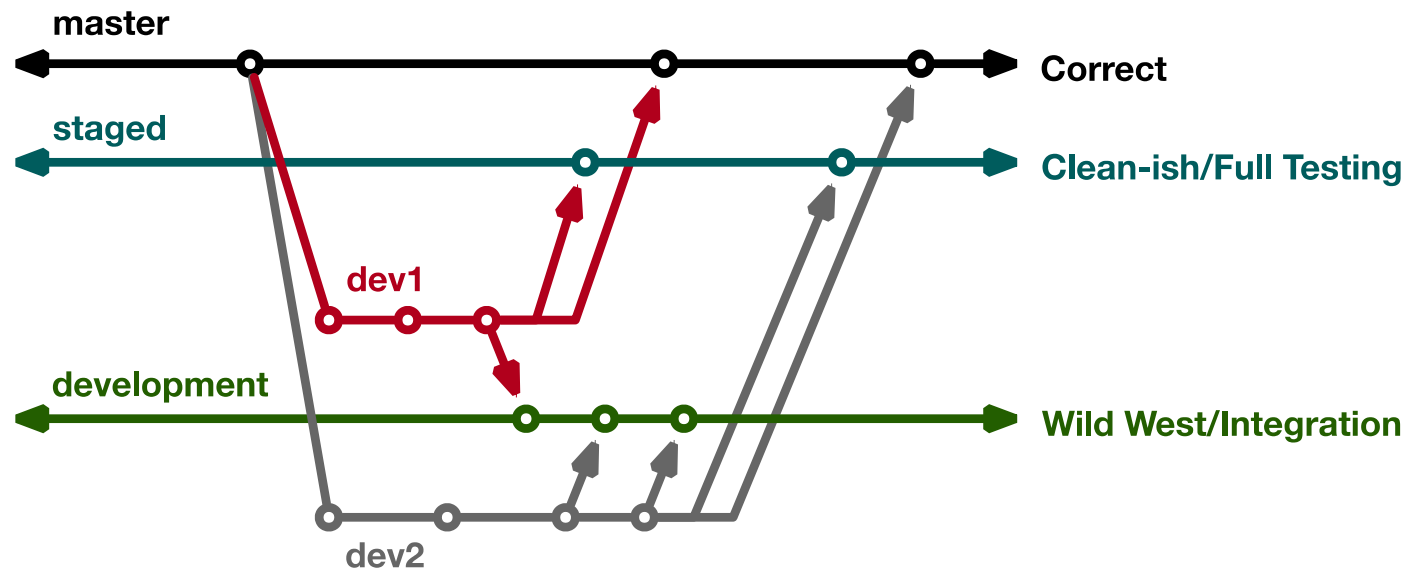
# Current FLASH5 Workflow

## Test-driven workflow

- Feature branches start and end with master
- All feature branches are merged into development for integration & manual testing
- All feature branches are then merged into staged for full, automated testing

Workflow designed so that

- All commits in master are in staged & development
- infinite branches don't diverge
- Merge conflicts first exposed on development





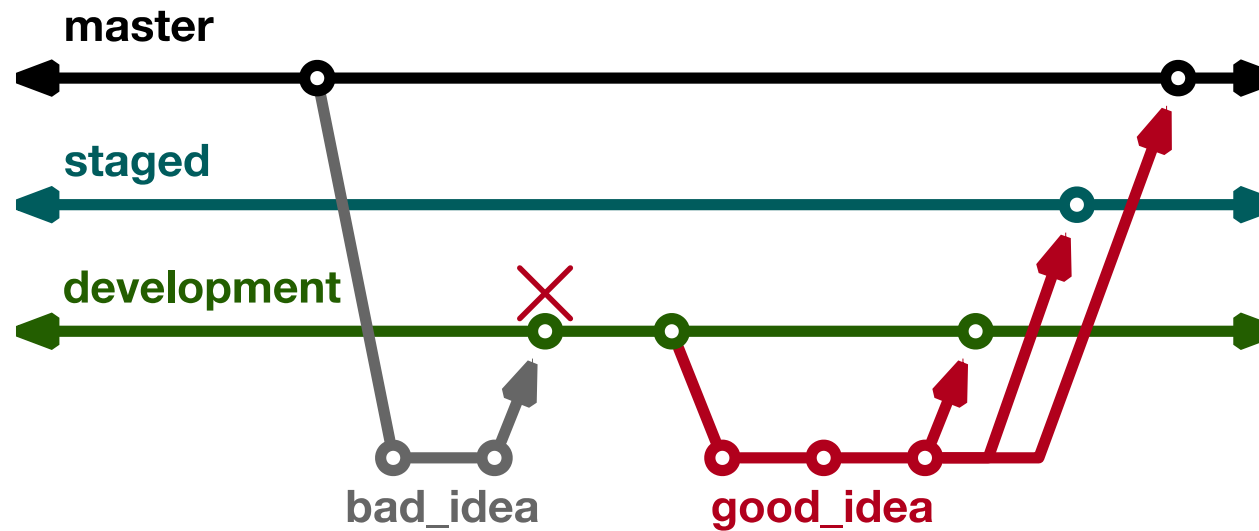
# Branch Rules

Why base feature branches off master?

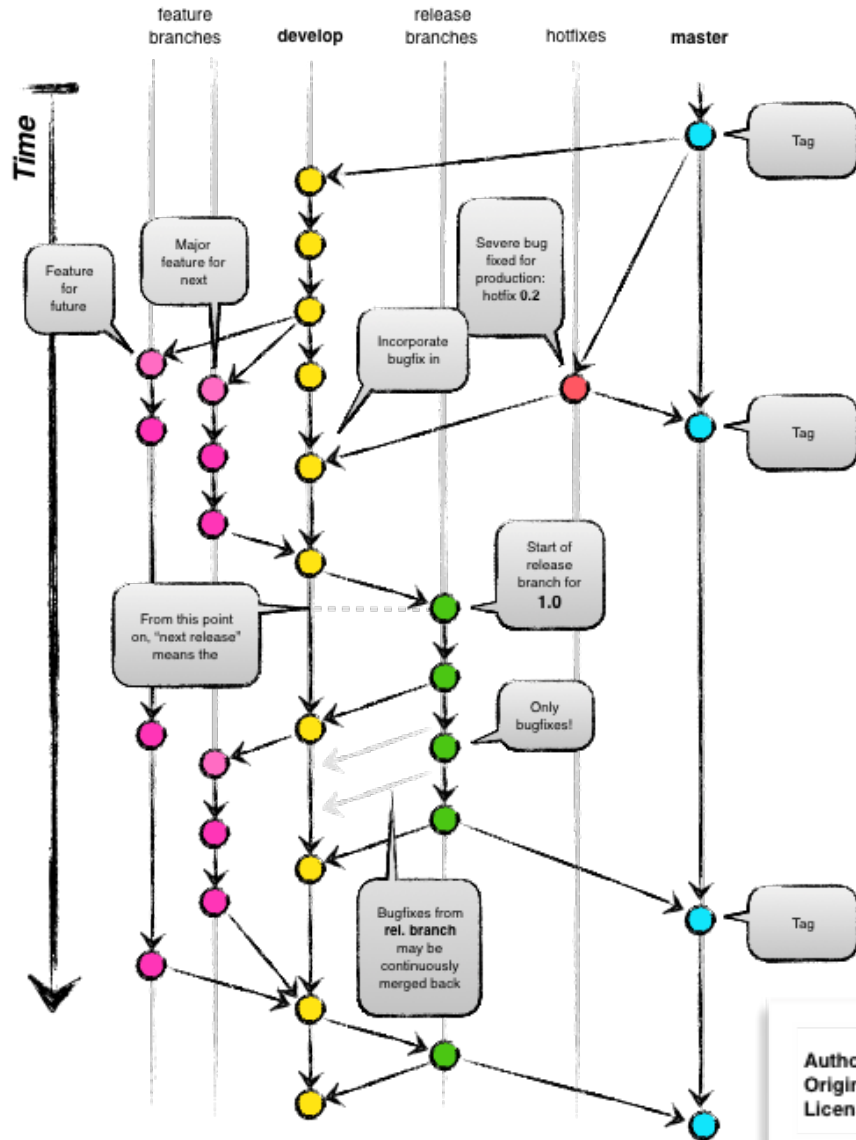
- Start from correct, verified commit
- Clean and simple to learn/enforce
- Isolate master from integration environment

Motivates more rules

- Development never merged into another branch
- Staged never merged into another branch



# Git Flow



- Full-featured workflow
- Increased complexity
- Designed for SW with official releases
- Feature branches based off of develop
- Git extensions to enforce policy
- How are develop and master synchronized?
- Where do merge conflicts occur and how are they resolved?

Author: Vincent Driessen  
Original blog post: <http://nvie.com/archives/323>  
License: Creative Commons



# GitHub Flow

<http://scottchacon.com/2011/08/31/github-flow.html>

- Published as viable alternative to Git Flow
- No structured release schedule
- Continuous deployment & continuous integration allows for simpler workflow

## Main Ideas

1. All commits in master are **deployable**
2. Base feature branches off of master
3. Push local repository to remote constantly
4. Open Pull Requests early to start dialogue
5. Merge into master after Pull Request review

# GitLab Flow

[https://docs.gitlab.com/ee/workflow/gitlab\\_flow.html](https://docs.gitlab.com/ee/workflow/gitlab_flow.html)

- Published as viable alternative to Git Flow & GitHub Flow
- Semi-structured release schedule
- Workflow that simplifies difficulties and common failures in synchronizing infinite lifetime branches

## Main Ideas

- Master branch is staging area
- Mature code in master flows downstream into pre-production & production infinite lifetime branches
- Allow for release branches with downstream flow
  - Fixes made upstream & merged into master.
  - Fixes cherry picked into release branch

# Things to Think About When Choosing a Git Workflow

Want to establish a clear set of policies that

- results in correct code on a particular branch (usually master),
- ensures that a team can develop in parallel and communicate well,
- minimizes difficulties associated with parallel and distributed work, and
- minimizes overhead associated with learning, following, and enforcing policies.

## **Adopt what is good for your team**

- Consider team culture and project challenges
- Assess what is and isn't feasible/acceptable
- Start with simplest and add complexity where and when necessary

# Continuous Integration

# The Short & Sweet of Continuous Integration

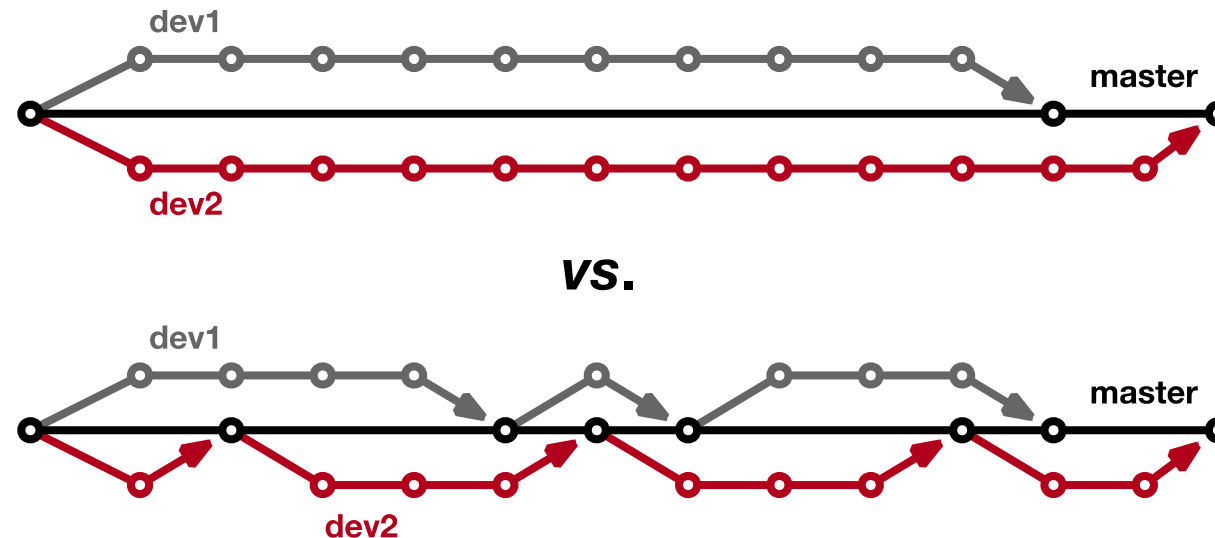
## A master branch that always works

- DVCS workflow isolate master from integration environment
- Extend workflow to address difficulties of integrating
  - Minimize likelihood of merge conflict
  - Detect bugs immediately
  - Make debugging process quick and easy

# Work Decomposition

Commit and integrate often

- Limit divergence between feature and master branches
- Decreased probability of conflict
- Conflict resolution is simpler and less risky





# Error detection

Test at integration to identify failures immediately

- Control quality of code
- Isolate failure to few commits
- No context switching for programmer

We want a system that

- triggers automated builds/tests on target environments when code changes and
- ideally tests on proposed merge product without finalizing merge.

# Test Servers

## Servers that

- automate the execution of a test suite or a subset of a test suite,
- allow for running tests on different environments,
- host an interface for viewing results, and
- allows for configuring when the tests are run.

## Examples

- CTest/CDash
- Jenkins
- Travis CI and GitLab CI

# Cloud-based Test Servers

- Linked to VCS hosts
  - GitHub & Travis CI
  - GitLab CI
  - BitBucket Pipelines
- Automated builds/tests triggered *via* pushes and pull requests
- Builds/tests can be run on cloud systems
- Test results are reported in repository's web interface
- Can trigger code coverage analysis & documentation build

# Continuous integration (CI)

- Has existed for some time and interest is growing
- ECP working to adapt CI for HPC machines
  - **ECP Continuous Integration Framework:** Tue Jan 15 from 3:00 PM to 4:00 PM
  - **Working Lunch - Plenary Continuous Integration:** Wed Jan 16 from 12:00 PM to 1:30 PM
- Setup, maintenance, and monitoring required
- Prerequisites
  - A reasonably automated build system
  - An automated test system with significant test coverage & useful feedback
  - Builds/tests must finish in reasonable amount of time
  - Ability to bundle subset of tests

# CI Hello World

## **Simplest CI example**

[https://github.com/jrdoneal/CI\\_HelloWorld](https://github.com/jrdoneal/CI_HelloWorld)

[https://travis-ci.org/jrdoneal/CI\\_HelloWorld](https://travis-ci.org/jrdoneal/CI_HelloWorld)

## **CI example w/ multiple platforms and specific compiler versions**

[https://github.com/jrdoneal/CI\\_Multiplatform](https://github.com/jrdoneal/CI_Multiplatform)

## **Code coverage, testing and CI tutorial (C++)**

<https://github.com/amklinv/morpheus>

## **Code coverage, testing, and CI example (Fortran, C++)**

<https://github.com/jrdoneal/infrastructure>

# Agenda

Time	Module	Topic	Speaker
9:00am-9:30am	01	Overview of Best Practices in HPC Software Development	Anshu Dubey, ANL
9:30am-10:00am	02	Better (Small) Scientific Software Teams	David E. Bernholdt, ORNL
10:00am-10:30am	03	Improving Reproducibility through Better Software Practices	David E. Bernholdt, ORNL
<i>10:30am-11:00am</i>		<i>Break</i>	
11:00am-11:45am	04	Verification & Refactoring	Anshu Dubey, ANL
11:45am-12:30pm	05	Git Workflow & Continuous Integration	Jared O'Neal, ANL

# CI Hello World – Backup Slides

# GitHub Repository Page

[https://github.com/jrdoneal/CI\\_HelloWorld](https://github.com/jrdoneal/CI_HelloWorld)

jrdoneal / CI\_HelloWorld

Unwatch

1

Star

0

Fork

0

<> Code

Issues 0

Pull requests 0

Projects 0

Wiki

Insights

Settings

No description, website, or topics provided.

Edit

Manage topics

5 commits

1 branch

0 releases

0 contributors

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

Developer D. Develop

This change should lead to a correct build environment for the purpos...

Latest commit 93a75c4 2 days ago

.travis.yml	This change should lead to a correct build environment for the purpos...	2 days ago
README.md	Add README file to explain the intent and eventual content of this tu...	2 days ago
hello_world.sh	Add the script that tests that the build environment is correctly con...	2 days ago



# Travis CI Configuration File

## .travis.yml

```
env:
- TRAVIS_CI_ENV="Hello, World"

#before_install:
#- Put commands here to prepare for executing builds/installs
#- Examples would be using apt-get to install dependencies not
#  included in the Travis CI build environment by default.

#install:
#- Put build commands here
#- In each phase, you can execute multiple commands
#- Travis CI stops if any single command fails in this phase

before_script:
- echo $TRAVIS_CI_ENV

script:
- $TRAVIS_BUILD_DIR/hello_world.sh
#- Travis CI will run each command in this phase even if a previous command
#  terminated in failure

after_success:
- echo "You should see that Hello, World was printed by before_script"

after_failure:
- echo "Hello, World should not have been printed by before_script"
```

# The Script Phase

## hello\_world.sh

```
#!/bin/bash

if [ -z "${TRAVIS_CI_ENV}" ]; then
    echo "Please set the TRAVIS_CI_ENV environment variable"
    exit 1
elif [ "${TRAVIS_CI_ENV}" != "Hello, World" ]; then
    echo "TRAVIS_CI_ENV value is ill-suited for this tutorial"
    exit 2
fi
```

# Connecting GitHub & Travis CI

## MY ACCOUNT



jrdoneal

Sync account

## ORGANIZATIONS

You are not currently a member of any organization.

## MISSING AN ORGANIZATION?

Review and add your authorized organizations.



jrdoneal

@jrdoneal

Repositories

Settings

We're only showing your public repositories. You can find your private projects on [travis-ci.com](https://travis-ci.com).

## Legacy Services Integration



Filter repositories



CI\_HelloWorld



Settings



CI\_Multiplatform



Settings



infrastructure



Settings

# Repository in Travis CI

[https://travis-ci.org/jrdoneal/CI\\_HelloWorld](https://travis-ci.org/jrdoneal/CI_HelloWorld)

 jrdoneal / CI\_HelloWorld  

Current Branches Build History Pull Requests

More options 

✓ **master** This change should lead to a correct build environment for the pu 🔗 #3 passed

tutorial. Travis CI builds should now be successful.

🔗 Commit 93a75c4 

🔗 Compare ff52718...93a75c4 

🔗 Branch master 

 jrdoneal

 Restart build

🕒 Ran for 18 sec


📅 27 a day ago

 </> Ruby

📦 TRAVIS\_CI\_ENV="Hello, World"


# Commit History

.travis.yml  
added →


 [jrdoneal](#) / [CI\\_HelloWorld](#)

[Code](#) [Issues 0](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#) [Insights](#)


Branch: **master** ▼

 Commits on Nov 3, 2018


**This change should lead to a correct build environment for the purpos...** [...](#)

 Developer D. Develop committed 2 days ago ✓


**Update Travis CI configuration file so that it is a step closer to se...** [...](#)

 Developer D. Develop committed 2 days ago ✗


**Add Travis CI configuration file. With the present content, the build** [...](#)

 Developer D. Develop committed 2 days ago ✗

**Add the script that tests that the build environment is correctly con...** [...](#)

 Developer D. Develop committed 2 days ago

**Add README file to explain the intent and eventual content of this tu...** [...](#)

 Developer D. Develop committed 2 days ago

# Travis CI Build History

Add Travis CI configuration file. With the present content, the build ...


 Developer D. Develop committed 2 days ago ✖

```
▶ 1 Worker information worker_info
▶ 6 Build system information system_info
413
414
415 Setting APT mirror in /etc/apt/sources.list: http://us-east-1.ec2.archive.ubuntu.com/ubuntu/
416
▶ 417 $ git clone --depth=50 --branch=master https://github.com/jrdoneal/CI_HelloWorld.git jrdoneal/CI_HelloWorld git.checkout 0.54s
▶ 427 $ rvm use default rvm 5.27s
▶ 434 $ ruby --version ruby.versions
442 No Gemfile found, skipping bundle install
▼ 443 $ echo $TRAVIS_CI_ENV before_script 0.00s
444
445
446 $ $TRAVIS_BUILD_DIR/hello_world.sh 0.00s
447 Please set the TRAVIS_CI_ENV environment variable
448
449
450 The command "$TRAVIS_BUILD_DIR/hello_world.sh" exited with 1.
▶ 451 $ echo "Hello, World should not have been printed by before_script" after_failure 0.00s
454
455 Done. Your build exited with 1.
```

Top ▲

# Travis CI Build History

Update Travis CI configuration file so that it is a step closer to se... ...

 Developer D. Develop committed 2 days ago ✖

```
▶ 1 Worker information worker_info
▶ 6 Build system information system_info
413
414
415 Setting APT mirror in /etc/apt/sources.list: http://us-east-1.ec2.archive.ubuntu.com/ubuntu/
416
▶ 417 $ git clone --depth=50 --branch=master https://github.com/jrdoneal/CI_HelloWorld.git jrdoneal/CI_HelloWorld git.checkout 0.52s
427
428 Setting environment variables from .travis.yml
429 $ export TRAVIS_CI_ENV="This content will result in failure"
430
▶ 431 $ rvm use default rvm 4.53s
▶ 438 $ ruby --version ruby.versions
446 No Gemfile found, skipping bundle install
▼ 447 $ echo $TRAVIS_CI_ENV before_script 0.00s
448 This content will result in failure
449
450 $ $TRAVIS_BUILD_DIR/hello_world.sh 0.00s
451 TRAVIS_CI_ENV value is ill-suited for this tutorial
452
453
454 The command "$TRAVIS_BUILD_DIR/hello_world.sh" exited with 2.
▶ 455 $ echo "Hello, World should not have been printed by before_script" after_failure 0.00s
458
459 Done. Your build exited with 1.
```

# Travis CI Build History

This change should lead to a correct build environment for the purpos... ...

 Developer D. Develop committed 2 days ago ✓

```
▶ 1 Worker information worker_info
▶ 6 Build system information system_info
413
414
415 Setting APT mirror in /etc/apt/sources.list: http://us-east-1.ec2.archive.ubuntu.com/ubuntu/
416
▶ 417 $ git clone --depth=50 --branch=master https://github.com/jrdoneal/CI_HelloWorld.git jrdoneal/CI_HelloWorld git.checkout 0.53s
427
428 Setting environment variables from .travis.yml
429 $ export TRAVIS_CI_ENV="Hello, World"
430
▶ 431 $ rvm use default rvm 4.69s
▶ 438 $ ruby --version ruby.versions
446 No Gemfile found, skipping bundle install
▼ 447 $ echo $TRAVIS_CI_ENV before_script 0.00s
448 Hello, World
449
450 $ $TRAVIS_BUILD_DIR/hello_world.sh 0.00s
451
452
453 The command "$TRAVIS_BUILD_DIR/hello_world.sh" exited with 0.
▶ 454 $ echo "You should see that Hello, World was printed by before_script" after_success 0.00s
457
458 Done. Your build exited with 0.
```

! →

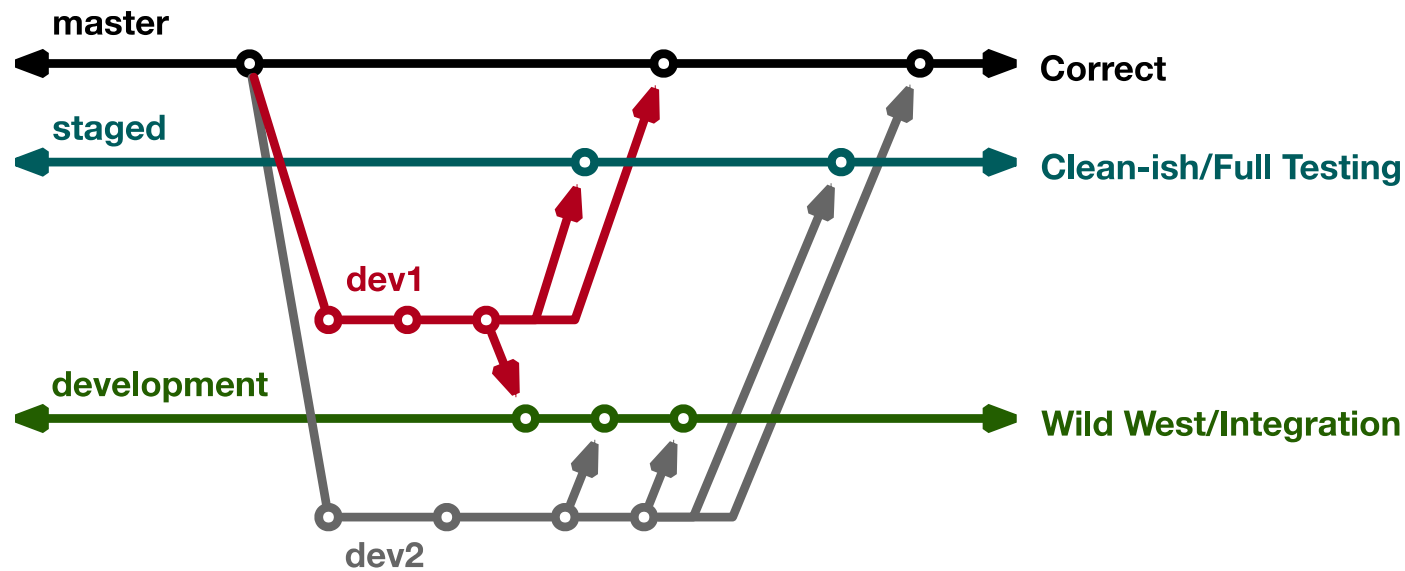


# Extra Slides

# More Branch Rules

Is staged really necessary?

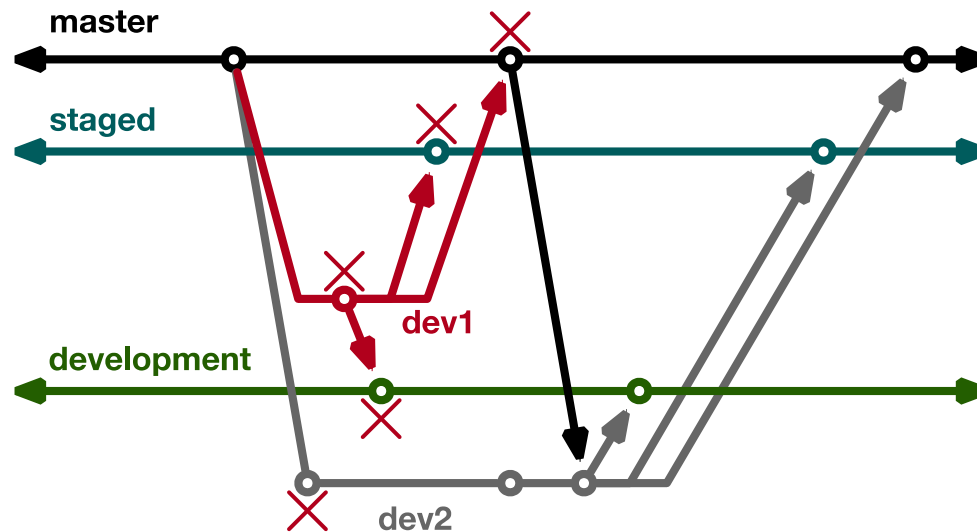
- Contains only changes intended for master
- No integration means cleaner branch
- Allows for extra stage of testing with more tests
- Extra buffer for protecting master branch



# Merge Conflicts

How are merge conflicts resolved in FLASH5 Workflow?

- Merge conflict with master means merge conflict with staged and development
- We want to avoid conflict resolution when merging into master
- Directly on feature branch if resolution is there
- One idea is to merge master into feature branch



# How do we determine what other tests are needed?

## Code coverage tools

- Expose parts of the code that aren't being tested
- gcov
  - standard utility with the GNU compiler collection suite
  - Compile/link with `-coverage` & turn off optimization
  - counts the number of times each statement is executed
- lcov
  - a graphical front-end for gcov
  - available at <http://ltp.sourceforge.net/coverage/lcov.php>
- Hosted servers (e.g. coveralls, codecov)
  - graphical visualization of results
  - push results to server through continuous integration server

# Code coverage output

## Overall Analysis

SOURCE FILES ON BUILD 45					
LIST 2	CHANGED 0	SOURCE CHANGED 0	COVERAGE CHANGED 0		
▲ COVERAGE	Δ	FILE	LINES	RELEVANT	COVERED
— 74.39		src/functions/linear_fcn_class.f90	301	82	61
— 100.0		src/general/modulo_mod.f90	52	3	3

## Detailed Analysis

```
265      ! Error distribution same for all x values
266      delta = S*Sxx - Sx*Sx
267      if (delta == 0.0_wp) then
268          ERRORMSG("Cannot do linear least-sqrs. Divide by zero.")
269          stop
270      end if
271      delta_inv = 1.0_wp / delta
```

<https://github.com/jrdoneal/infrastructure>

# Code coverage is popular

- gcov also works for C and Fortran
- Other tools exist for other languages
  - Jcov for Java
  - Coverage.py for python
  - Devel::Cover for perl
  - profile for MATLAB
  - *etc.*

# Special Notes for Morpheus Tutorial

- A code coverage and testing tutorial can be found at the Morpheus repository doxygen pages
  - <https://amklinux.github.io/morpheus/index.html>
- **STEP 1:** These exercises must be run on your own local machine or on a remote machine that you have access to.
- If you cannot generate your own gcov output, the associated lcov output is online
  - <https://amklinux.github.io/morpheus/lcovFiles/index.html>