



Git Workflows

James M. Willenbring
Sandia National Laboratories

Jared O'Neal
Argonne National Laboratory

Software Productivity Track, ATPESC 2020



See slide 2 for
license details

License, Citation and Acknowledgements



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).
- **The requested citation the overall tutorial is: David E. Bernholdt, Anshu Dubey, Mark C. Miller, Katherine M. Riley, and James M. Willenbring, Software Productivity Track, in Argonne Training Program for Extreme Scale Computing (ATPESC), August 2020, online. DOI: [10.6084/m9.figshare.12719834](https://doi.org/10.6084/m9.figshare.12719834)**
- Individual modules may be cited as *Speaker, Module Title*, in Software Productivity Track...

Acknowledgements

- Additional contributors include: Patricia Grubel, Rinku Gupta, Mike Heroux, Alicia Klinvex, Jared O'Neal, David Rogers, Deborah Stevens
- 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.
- This work was performed in part at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.
- This work was performed in part at the Lawrence Livermore National Laboratory, which is managed by Lawrence Livermore National Security, LLC for the U.S. Department of Energy under Contract No. DE-AC52-07NA27344.
- This work was performed in part at Sandia National Laboratories. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND2020-7985 PE

Goals

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

- Briefly cover version control basics
- Introduce a workflow for the heat equation example work
 - Branches
 - Forks
 - Pull requests
- Exposure to workflows of different complexity
- What to think about when evaluating different workflows
- Motivate continuous integration

Our First Workflow

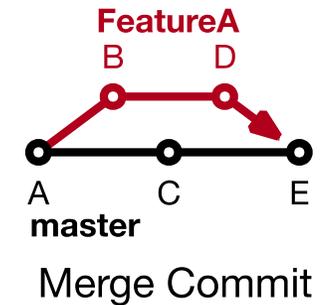
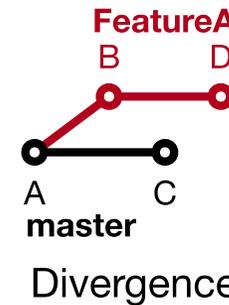
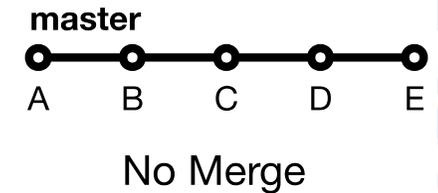
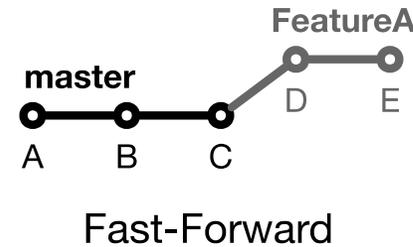
This process of collaborating *via* Git is called the **Centralized Workflow**

- 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?
 - Lengthy development efforts without integrating
 - Occasional contributors
- What if team members works on different parts of the code?
- Working directly on master

Branches

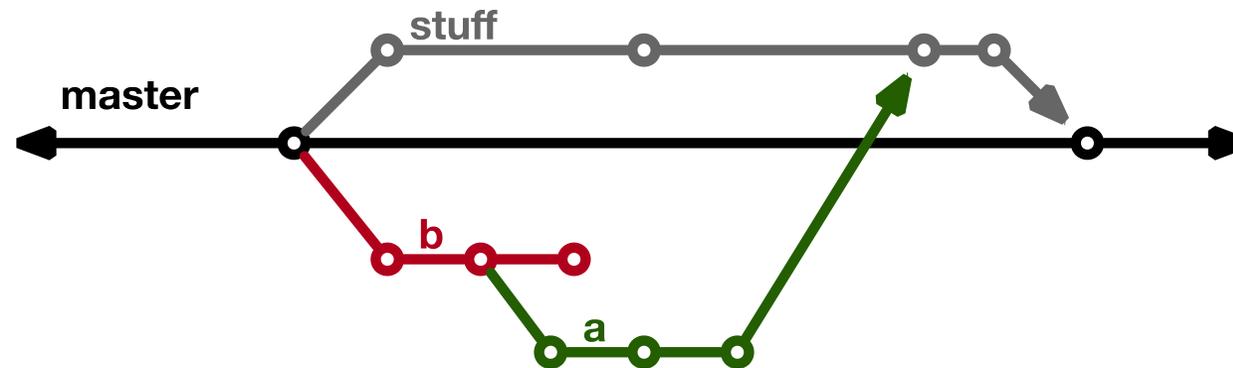
Branches are independent lines of development

- 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



Control Project Branch Complexity

- Workflow policy is needed
 - Project supported branches and workflows should not be unnecessarily complex
 - Individuals and sub-teams can leverage more complex models when advantageous
 - Descriptive names or linked to issue tracking system
 - Where do branches start and end?

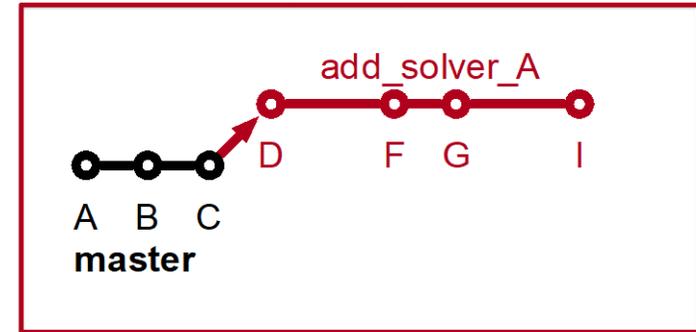


Feature Branches

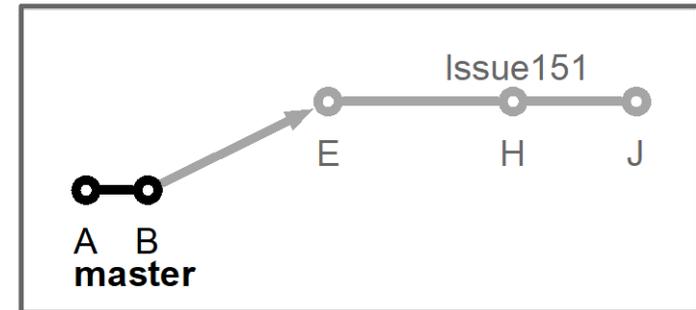
Extend Centralized Workflow

- 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)

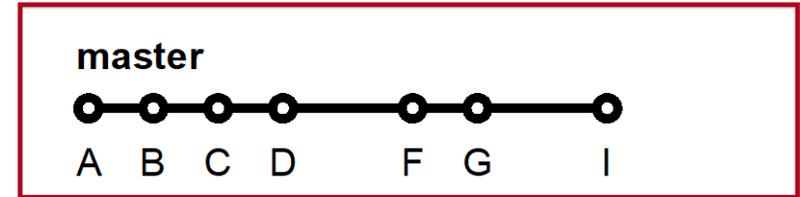


Feature Branch Divergence

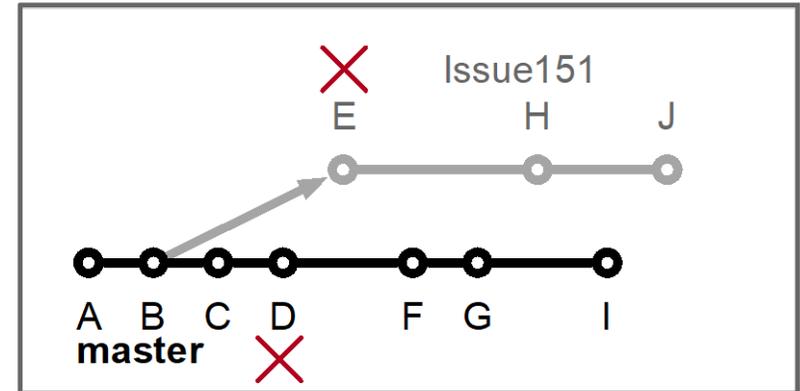
Alice integrates first without issue

- 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

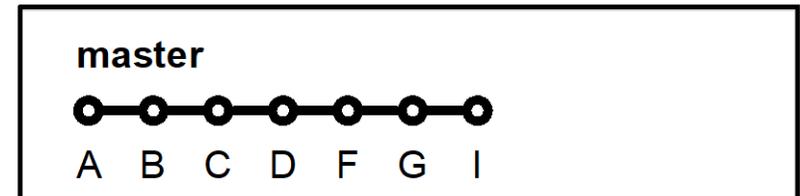
Alice's Local Repository



Bob's Local Repository



Main Remote Repository (origin)

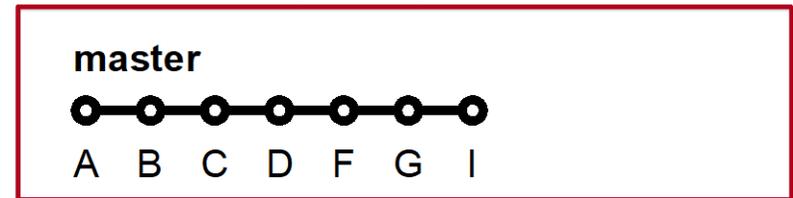


Feature Race Condition

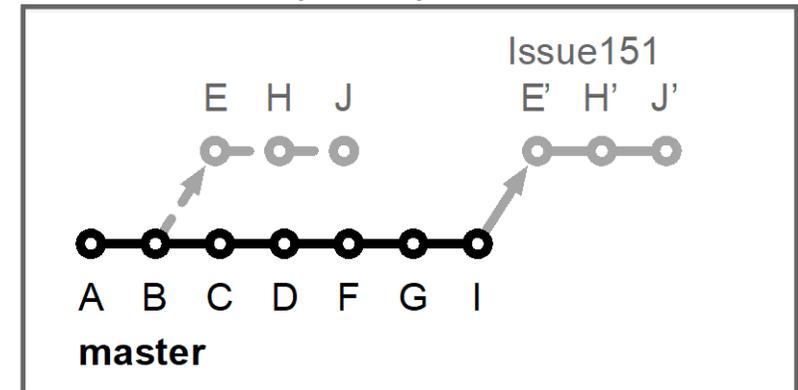
Integration occurs on Bob's local repo

- 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
- See [Atlassian/BitBucket](#) for a richer Feature Branch Workflow

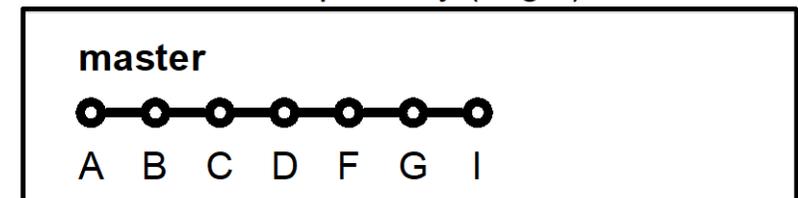
Alice's Local Repository



Bob's Local Repository



Main Remote Repository (origin)



GitHub Forks

- A “fork” of a repository is a complete copy of another repository, inside a different GitHub account.
 - Forking requires read access to the main (often referred to as “upstream”) repository
 - Forks of public repositories are public
 - Other users can be granted write access to your fork
 - You cannot fork a fork
 - Does not copy issues or pull requests
 - Use branches within your fork (do not modify master)
 - A pull request (GitLab uses “merge request”) can be used to suggest changes to the upstream repository
 - Added benefit: pull requests from forks prevent huge numbers of branches on the upstream repository

Git Workflow for the Heat Equation Example

- Developers
 - Work on feature branches in their forks
 - Using forks requires contributors to have only read-access to primary repository
 - Issue pull requests for changes
 - Natural opportunity to review and test all changes
- Pull requests
 - Are reviewed by at least one developer (not the author)
 - Undergo CI testing prior to merging

Demo for Heat Equation Example Workflow

- Fork repository (once)
- Clone the fork (once)
- Create and checkout branch
 - Base branch on current development or other appropriate version for each feature
- Modify and commit code
- Push change to fork
- Issue pull request to upstream repository
- Review pull request
- CI testing (covered in upcoming module)

Fork the Repository

The screenshot shows the GitHub interface for the repository `betterscientificsoftware/hello-numerical-world`. The `Fork` button in the top right corner is circled in red. A modal dialog titled "Fork hello-numerical-world" is open, asking "Where should we fork hello-numerical-world?". A red arrow points to the user `jwillenbring` in the list of suggestions. The repository's file tree is visible in the background, showing files like `main`, `tools`, `.codecov.yml`, `.travis.yml`, `Double.H`, `README.md`, `args.C`, `check.sh`, `crankn.C`, `exact.C`, `ftcs.C`, `heat.C`, `heat.H`, `makefile`, and `upwind15.C`.

Find the Path to Clone

Search or jump to... Pull requests Issues Marketplace Explore

jwillenbring / hello-numerical-world
forked from betterscientificsoftware/hello-numerical-world

Watch 0 Star 0 Fork 5

Code Pull requests Actions Projects Wiki Security Insights Settings

main 2 branches 0 tags

This branch is even with betterscientificsoftware:main.

markcmiller86 Update README.md

tools simplify shell math

.codecov.yml adjust how coverage computed

.travis.yml resetting back to normal

Double.H adding initial heatapp code 19 days ago

README.md Update README.md 15 days ago

args.C adding initial heatapp code 19 days ago

check.sh making bash not sh 19 days ago

crankn.C adding initial heatapp code 19 days ago

exact.C adding comments 18 days ago

ftcs.C adding initial heatapp code 19 days ago

heat.C adding initial heatapp code 19 days ago

heat.H adding initial heatapp code 19 days ago

makefile adjust how coverage computed 17 days ago

upwind15.C adding initial heatapp code 19 days ago

Clone with SSH
Use a password protected SSH key.
git@github.com:jwillenbring/hello-n

Open with GitHub Desktop

Download ZIP

About
No description, website, or topics provided.

Readme

Releases
No releases published
Create a new release

Packages
No packages published
Publish your first package

Languages

- C 59.5%
- C++ 21.9%
- Shell 13.4%
- Makefile 5.2%

Clone the fork; Create and Checkout a New Branch

```
[s988335:repos jmwille$ git clone git@github.com:jwillenbring/hello-numerical-world.git
Cloning into 'hello-numerical-world'...
[Enter passphrase for key '/Users/jmwille/.ssh/id_rsa':
remote: Enumerating objects: 102, done.
remote: Counting objects: 100% (102/102), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 102 (delta 54), reused 94 (delta 50), pack-reused 0
Receiving objects: 100% (102/102), 21.69 KiB | 120.00 KiB/s, done.
Resolving deltas: 100% (54/54), done.
[s988335:repos jmwille$
```

```
[s988335:repos jmwille$ cd hello-numerical-world/
[s988335:hello-numerical-world jmwille$ git checkout -b issue-1000
Switched to a new branch 'issue-1000'
s988335:hello-numerical-world jmwille$ █
```

Modify and Commit Code

```
[s988335:hello-numerical-world jmwille$ vi README.md
[s988335:hello-numerical-world jmwille$ git diff
diff --git a/README.md b/README.md
index 3cd1a3c..b44c57e 100644
--- a/README.md
+++ b/README.md
@@ -22,7 +22,7 @@ is known as the _Diffusion Equation_ and also the [_Heat Equation_](https://en.w

### Simplifying Assumptions

-To make the problem tractable for this lesson, we make some simplifying assumptions...
+To make the problem tractable for this lesson, we make some simplifying assumptions:

1. The thermal diffusivity,  $\alpha$ (http://latex.codecogs.com/gif.latex?%5Calpha),
   is constant for all _space_ and _time_.
s988335:hello-numerical-world jmwille$ █
```

```
[s988335:hello-numerical-world jmwille$ git add README.md
[s988335:hello-numerical-world jmwille$ git commit -m "replaced ... with :"
[issue-1000 1c3a901] replaced ... with :
1 file changed, 1 insertion(+), 1 deletion(-)
s988335:hello-numerical-world jmwille$ █
```

Push Change to Fork

```
[s988335:hello-numerical-world jmwille$ git remote -vv
origin git@github.com:jwillenbring/hello-numerical-world.git (fetch)
origin git@github.com:jwillenbring/hello-numerical-world.git (push)
[s988335:hello-numerical-world jmwille$ git branch
* issue-1000
  main
[s988335:hello-numerical-world jmwille$ git push origin issue-1000
[Enter passphrase for key '/Users/jmwille/.ssh/id_rsa':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 310 bytes | 310.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'issue-1000' on GitHub by visiting:
remote:   https://github.com/jwillenbring/hello-numerical-world/pull/new/issue-1000
remote:
To github.com:jwillenbring/hello-numerical-world.git
 * [new branch]      issue-1000 -> issue-1000
s988335:hello-numerical-world jmwille$ █
```

Issue Pull Request to Upstream Repository

 [jwillenbring / hello-numerical-world](#) Watch 0 Star 0 Fork 5

forked from [betterscientificsoftware/hello-numerical-world](#)

[Code](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

 issue-1000 had recent pushes less than a minute ago [Compare & pull request](#)

[main](#) [2 branches](#) [0 tags](#) [Go to file](#) [Add file](#) [Code](#)

This branch is even with [betterscientificsoftware:main](#). [Pull request](#) [Compare](#)

 [markcmiller86](#) Update README.md c591164 15 days ago 26 commits

tools	simplify shell math	17 days ago
.codecov.yml	adjust how coverage computed	17 days ago
.travis.yml	reseting back to normal	17 days ago
Double.H	adding initial heatapp code	19 days ago
README.md	Update README.md	15 days ago
args.C	adding initial heatapp code	19 days ago
check.sh	making bash not sh	19 days ago
crankn.C	adding initial heatapp code	19 days ago
exact.C	adding comments	18 days ago
ftcs.C	adding initial heatapp code	19 days ago
heat.C	adding initial heatapp code	19 days ago
heat.H	adding initial heatapp code	19 days ago
makefile	adjust how coverage computed	17 days ago

About 

No description, website, or topics provided.

[Readme](#)

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Languages



- C 59.5%
- C++ 21.9%
- Shell 13.4%
- Makefile 5.2%

Issue Pull Request to Upstream Repository

The screenshot shows the GitHub interface for a pull request. At the top, the repository name is 'betterscientificsoftware/hello-numerical-world'. Below the navigation bar, the title 'Open a pull request' is followed by a description: 'Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).' The main area is a form for creating the pull request. A red box highlights the top section of the form, which includes dropdown menus for 'base repository: betterscientificsoftware/hell...', 'base: main', 'head repository: jwillenbring/hello-numerical-...', and 'compare: issue-1000'. Below these is a green checkmark and the text 'Able to merge. These branches can be automatically merged.' The main text area contains the text 'replaced ... with :', 'Issue 1000', and 'I really like using : a lot better.' A red arrow points to this text area. At the bottom of the form, there is a checkbox for 'Allow edits by maintainers' and a green 'Create pull request' button, which is circled in red. To the right of the form is a sidebar with sections for 'Reviewers' (markcmiller86), 'Assignees' (jwillenbring), 'Labels' (enhancement), 'Projects' (None yet), 'Milestones', 'Linked issues', and 'Helpful resources' (GitHub Community Guidelines). Three red arrows point to the settings icons for Reviewers, Assignees, and Labels. At the bottom of the page, a summary bar shows '1 commit', '1 file changed', '0 comments', and '1 contributor'.

Review Pull Request

replaced ... with : #4

Open jwillenbring wants to merge 1 commit into `betterscientificsoftware:main` from `jwillenbring:issue-1000`

Conversation 1 Commits 1 Checks 1 Files changed 1

Changes from all commits File filter... Jump to... 0 / 1 files viewed Finish your review 1

25 - To make the problem tractable for this lesson, we make some simplifying assumptions...

25 + To make the problem tractable for this lesson, we make some simplifying assumptions...

markcmiller86 Member Pending
Are you sure you really wanna do that?

Reply...

Write Preview H B I $\code \link \list \checkbox \@ \reply \undo$

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

- Comment**
Submit general feedback without explicit approval.
- Approve**
Submit feedback and approve merging these changes.
- Request changes**
Submit feedback that must be addressed before merging.

Submit review 1 pending comment

ProTip! Use `n` and `p` to navigate between commits in a pull request.

CI Testing for PR

[EXTERNAL] Passed: jwillenbring/hello-numerical-world#1 (issue-1000 - 1c3a901)



Travis CI <builds@travis-ci.com>

Today at 2:37 PM

To: Willenbring, James M

jwillenbring / hello-numerical-world

 issue-1000

 **Build #1 passed** >

 20 secs



James M. Willenbring

[1c3a901 CHANGESSET](#) →

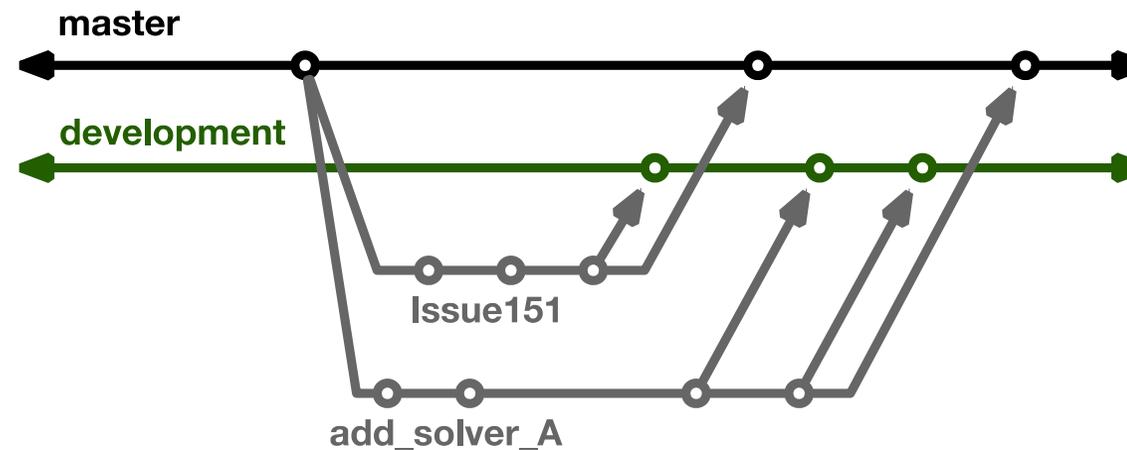
replaced ... with :

This will be covered in the CI module

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



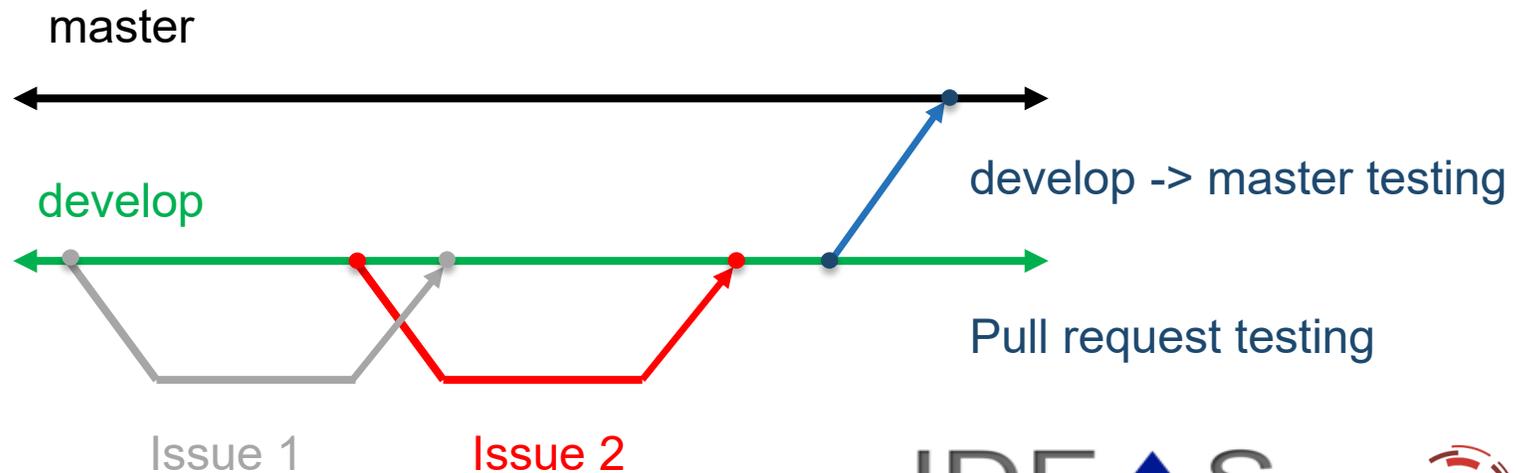
Current Trilinos Workflow

Test-driven workflow

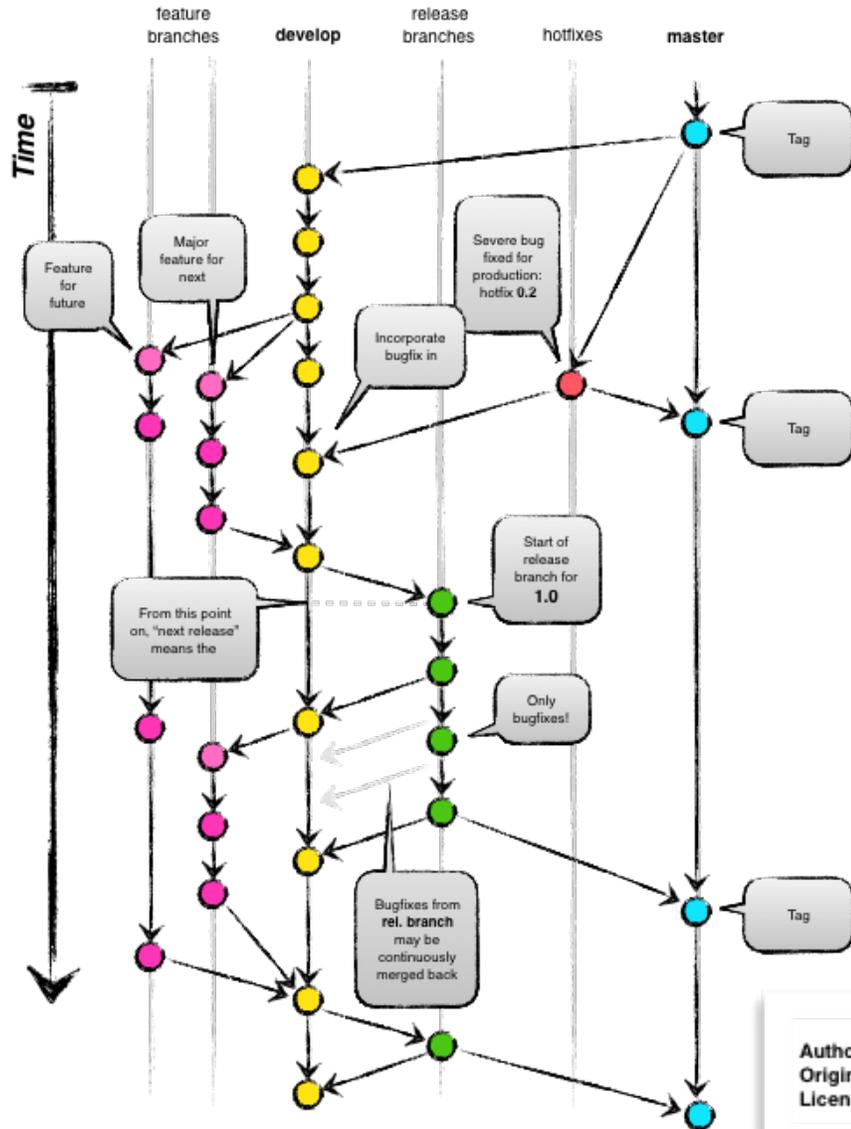
- Feature branches start and end with develop
- All changes to develop must come from GitHub pull requests
- Feature branches are merged into develop only after passing pull request test suite
- Change sets from develop are tested daily for integration into master

Workflow designed so that

- All commits in master are in develop
- Merge conflicts exposed when integrating into develop
- Merge conflicts never occur when promoting to master



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

- 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

Considerations for 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