Introduction to Git

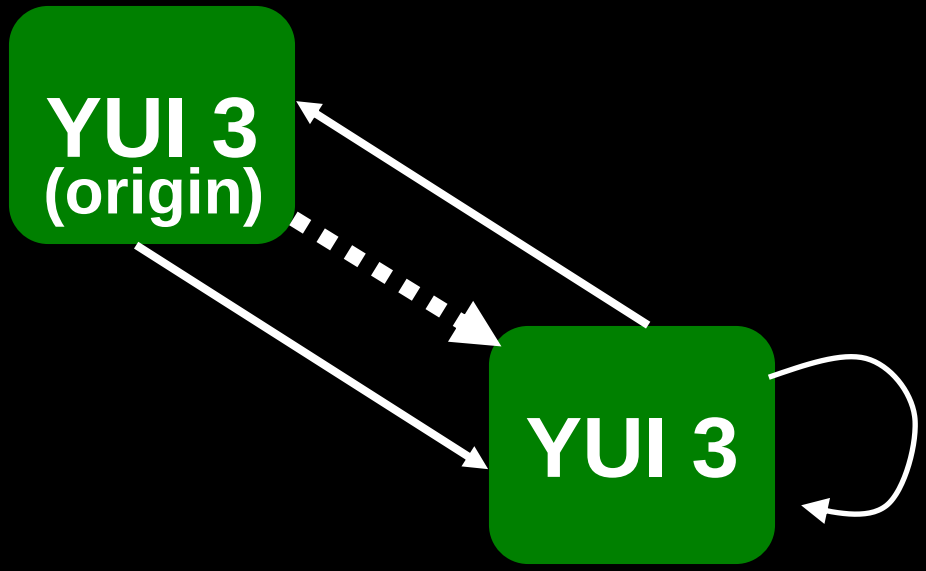
Introduction to Git

- Git overview
- Basic usage
- Git, YUI, and GitHub
- Real world tips and tricks

Git is a version control system

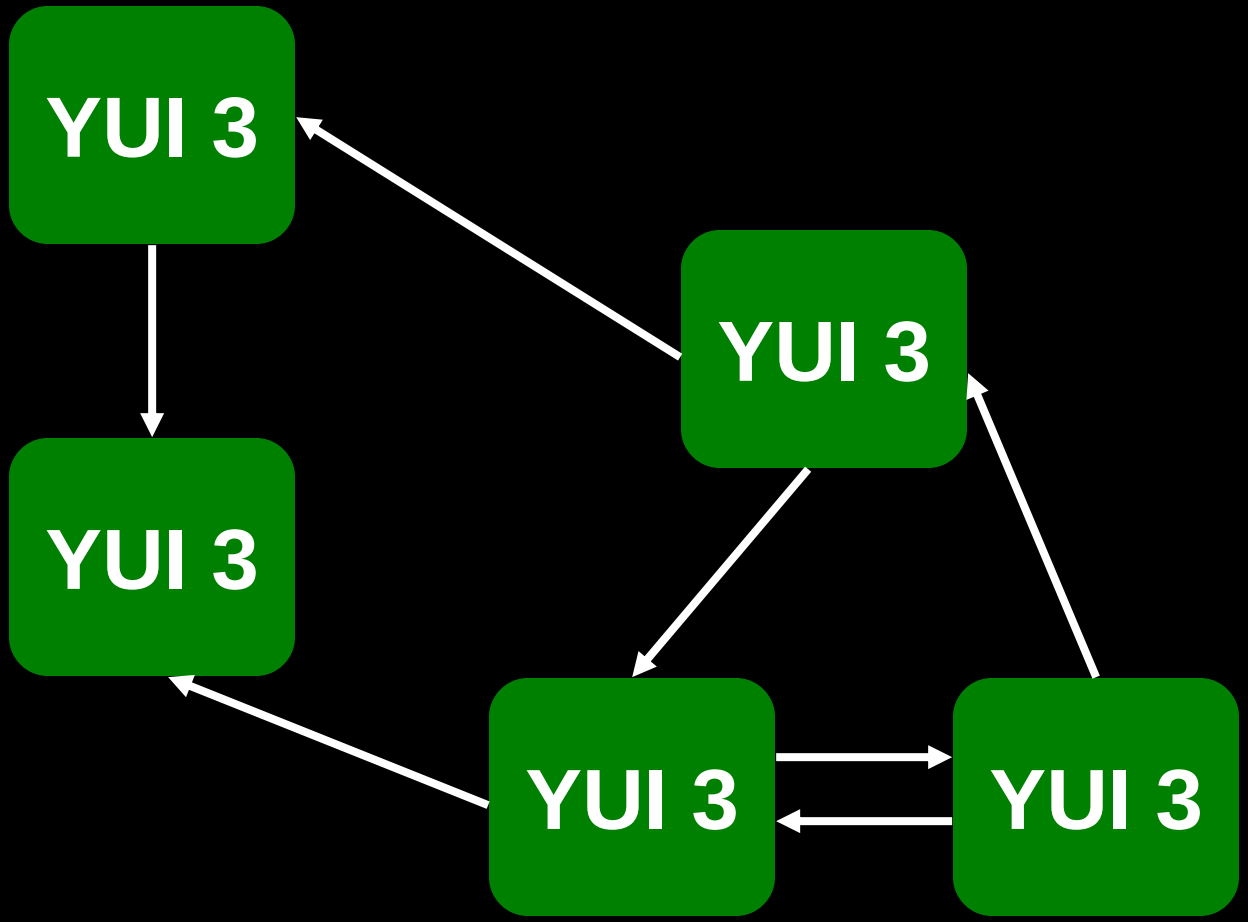
- Track changes (who/what/when)
- Compare versions
- Revert changes
- Perform merges
- Implement tags and branches
- http://en.wikipedia.org/wiki/Revision_control

Git is distributed

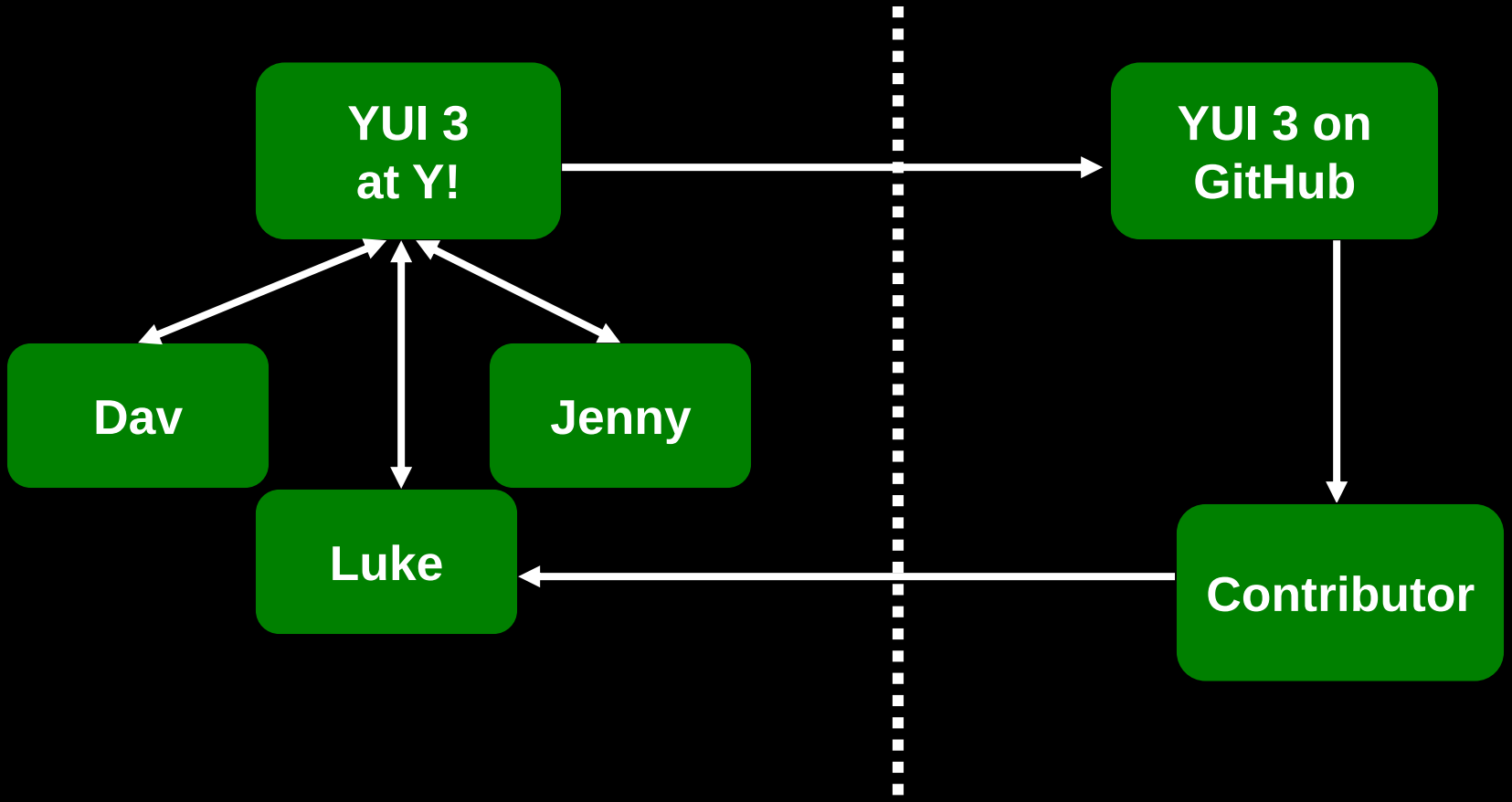


> git **clone**
> git **commit**
> git **pull**
> git **push**

Git is distributed

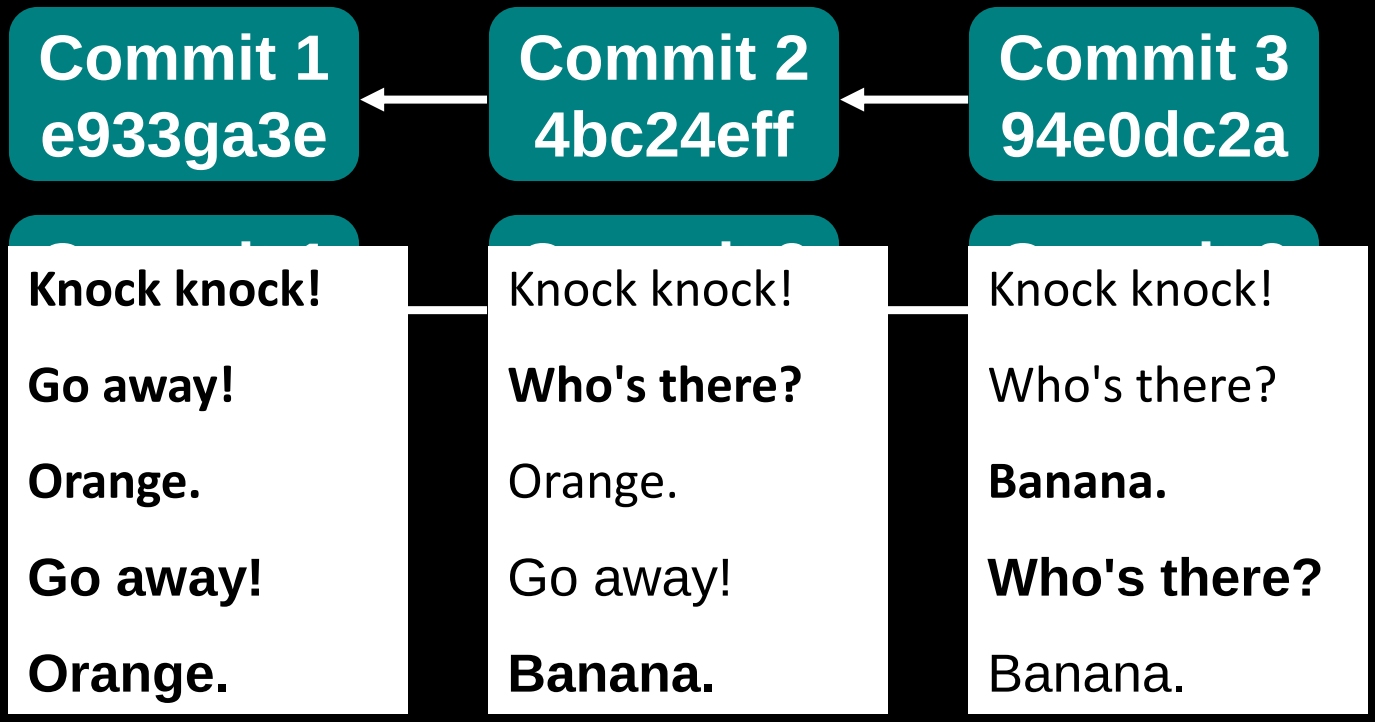


The YUI Workflow

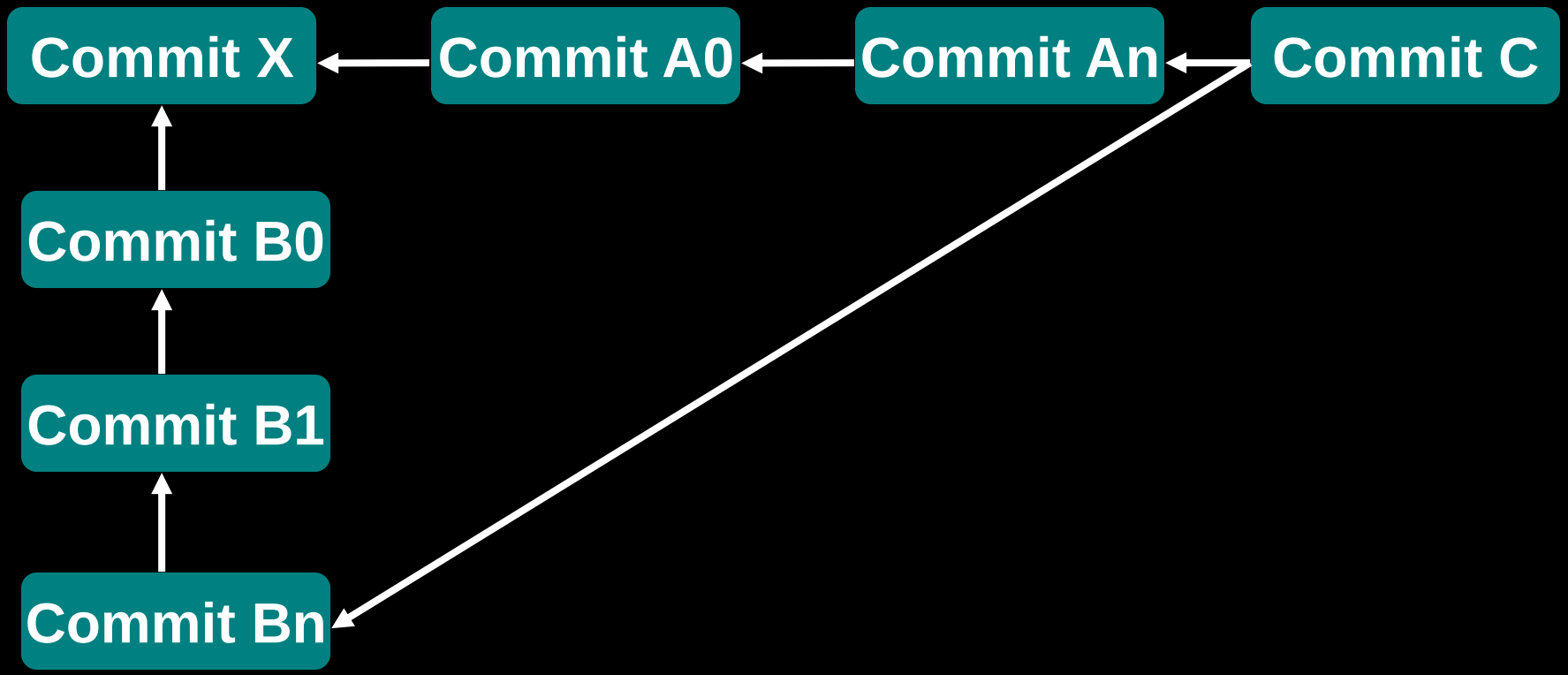


Git is a series of snapshots

Git snapshots have integrity



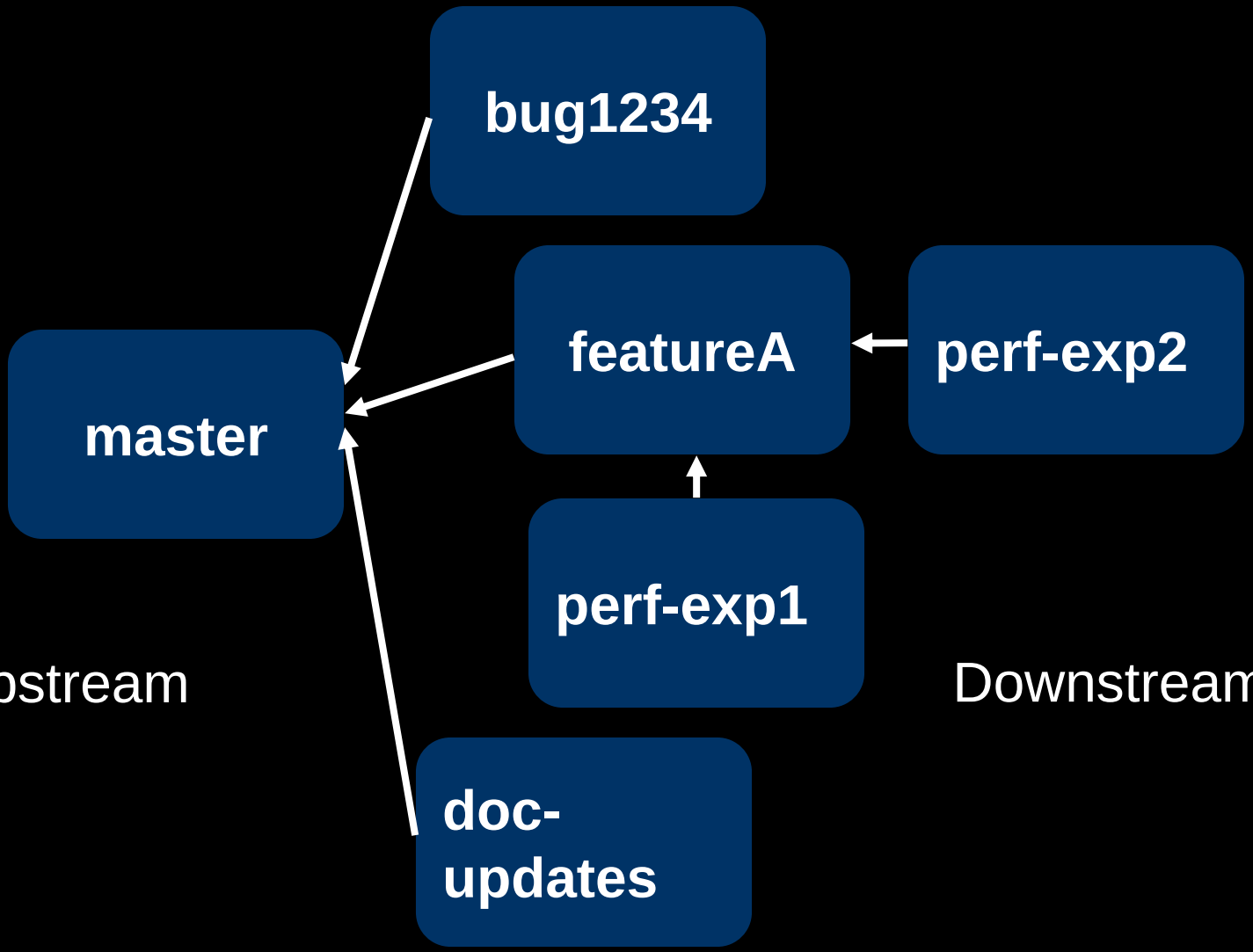
Git makes branches easy



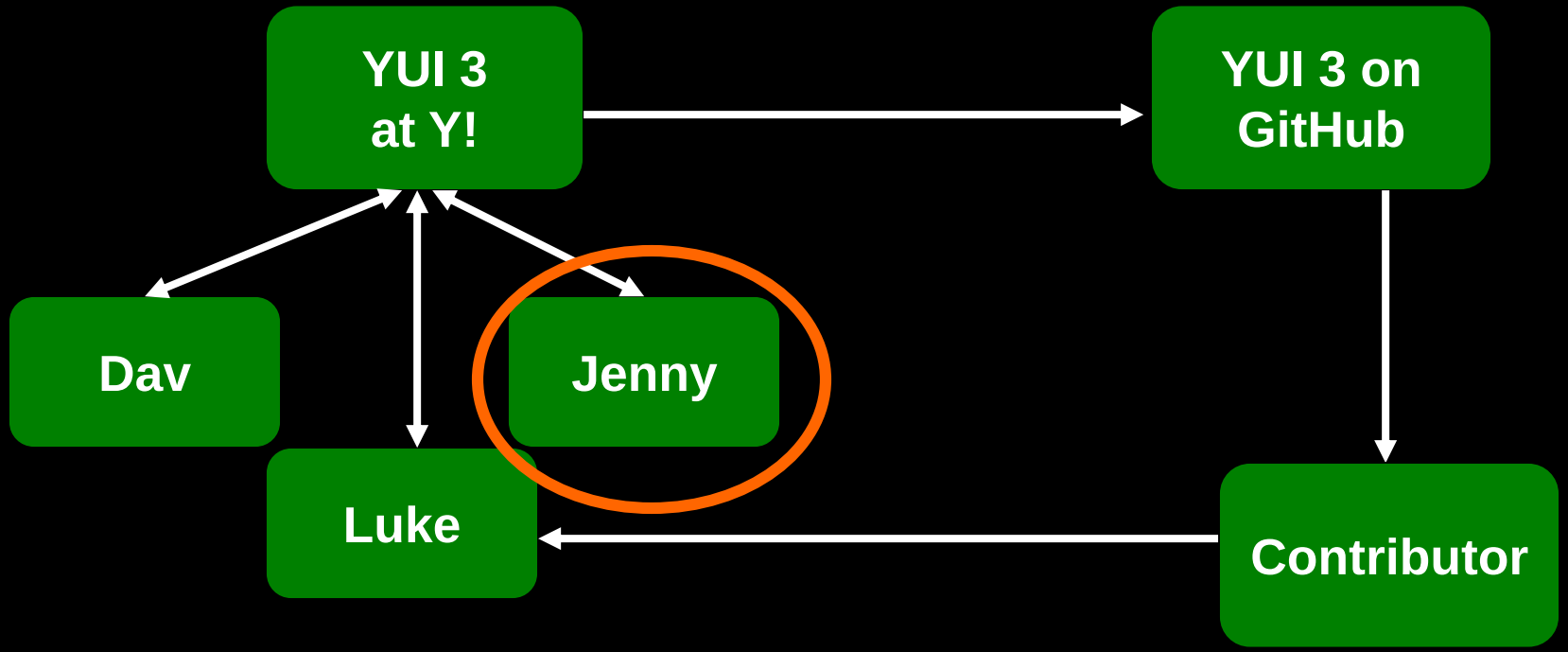
When to branch?

- For each version
- For each fix
- Explore multiple solutions
- Collaborate
- All the time!

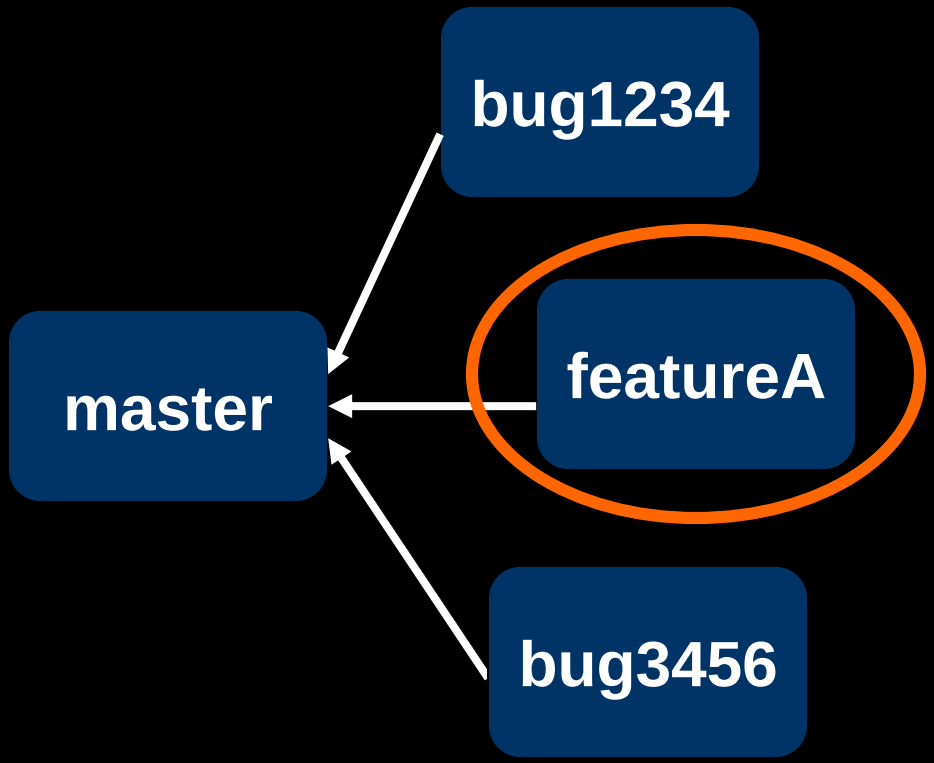
Branch workflow



Recap



Recap



Recap

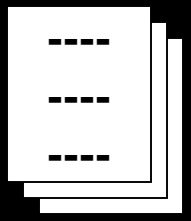
diff + parent = hash

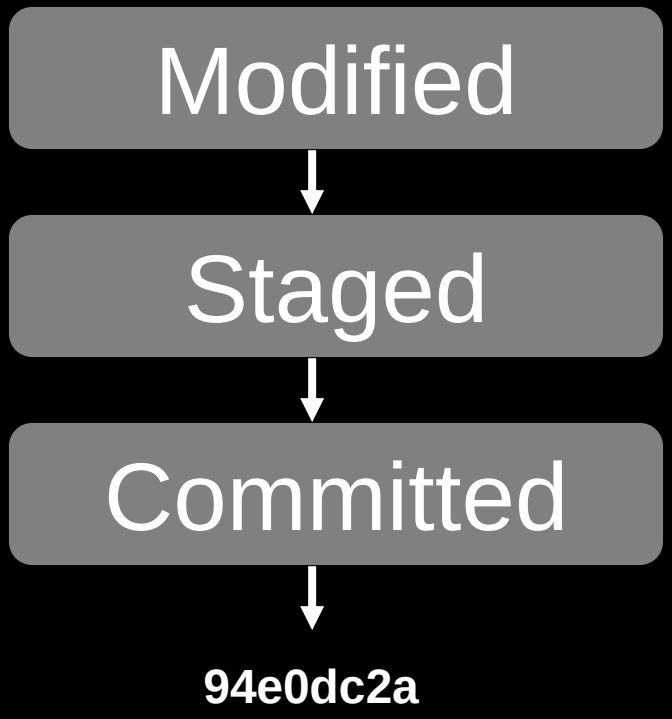
File states

> vi file.txt

> git **add** file.txt

> git **commit** file.txt





Edit files

```
> vi joke.txt
> git diff
diff --git a/joke.txt b/joke.txt
index 7f172de..4aaf7a3 100644
--- a/joke.txt
+++ b/joke.txt
@@ -1,4 +1,4 @@
 Knock knock!
-Go away!
+Who's there?
 Orange.
```

Edit files

```
> git status
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working
    directory)
#
#       modified:   joke.txt
#
no changes added to commit (use "git add" and/or "git commit -
    a")
```

Edit files

```
> git add joke.txt
> git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   joke.txt
#
```

Edit files

> git **commit** -m "Reply with a question."

Edit files

> git **commit**

Anatomy of a proper commit message

```
1
2 # Please enter the commit message for your changes. Lines starting
3 # with '#' will be ignored, and an empty message aborts the commit.
4 # On branch master
5 # Changes to be committed:
6 #   (use "git reset HEAD <file>..." to unstage)
7 #
8 #   modified:   joke.txt
9 #
```

Anatomy of a proper commit message

```
 1 Reply with a question.
 2
 3 I thought of a better reply. If I ask a question, it continues the
 4 dialog. Then maybe knocker can respond with something funny._
 5 # Please enter the commit message for your changes. Lines starting
 6 # with '#' will be ignored, and an empty message aborts the commit.
 7 # On branch master
 8 # Changes to be committed:
 9 #   (use "git reset HEAD <file>..." to unstage)
10 #
11 #   modified:   joke.txt
12 #
```

http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html

Edit files

> git **commit**

[master 77dcfbc] Reply with a question.

1 files changed, 1 insertions(+), 1 deletions(-)

Edit files

> git **log**

commit 77dcfbcc318be5ef5b9af5f1865647db3566fd4a

Author: Jenny Donnelly <jennydonnelly@yahoo-inc.com>

Date: Thu Mar 24 17:03:36 2011 -0700

    Reply with a question.

    I thought of a better reply. If I ask a question, it continues the dialog. Then maybe knocker can respond with something funny.

...

New files

```
> vi new.txt
> git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       new.txt
nothing added to commit but untracked files present (use "git add" to track)
```

New files

```
> git add new.txt
> git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   new.txt
#
> git commit
```

Remove files

> git **rm** new.txt

> git **commit**

Rename files

> git **mv** foo.txt bar.txt

> git **commit**

The “modified” state

- Diff from the last commit
- Follows you from branch to branch
- Moved, not copied
- Merge conflicts prevent checkout to another branch
- > git **diff**

The “staged” state

- Very similar to "modified" state
- Diffs have been staged with

  > git **add** .

  > git **add** file.txt
- > git **diff --cached**
- Skip with > git **commit -a**

The “committed” state

- Commits are snapshots
- Uniquely identified by SHA-1 hashes
- Hashes are diffs + parent
- Commits are sticky to branch
- > git **whatchanged** master..branch

Recap: Commit workflow

> vi joke.txt
> git **diff**
> git **commit –a**
> git **status**

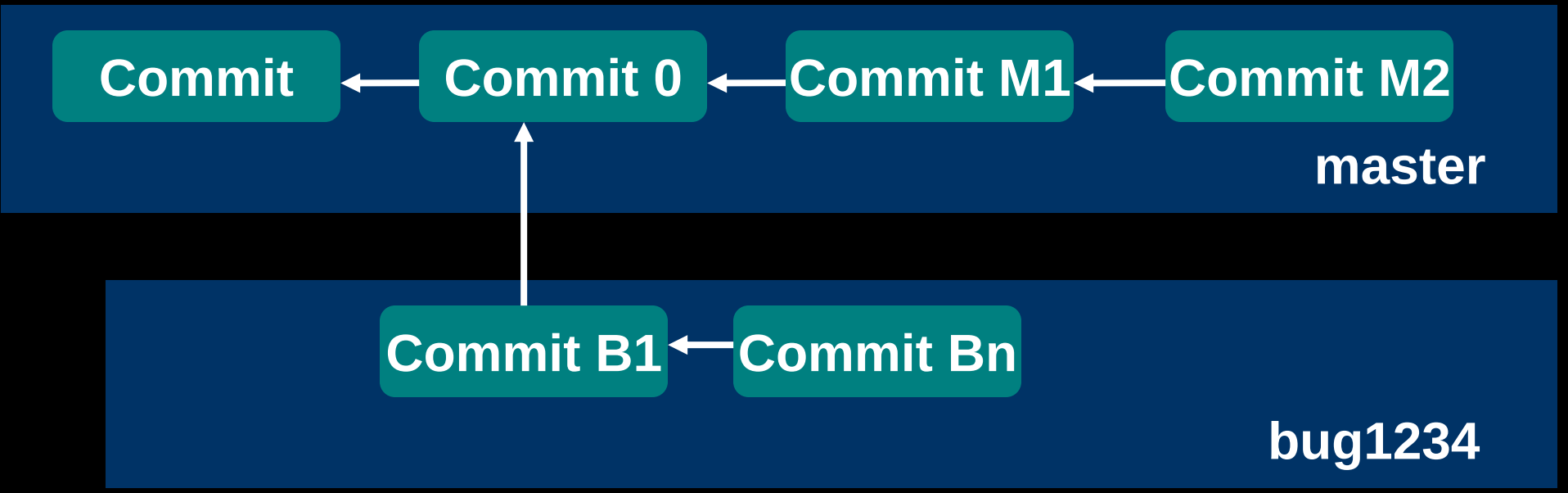
> vi new.txt
> git **add** new.txt
> git **commit**
> git **status**

> git **rm** new.txt
> git **commit**
> git **status**

Syncing branches workflow

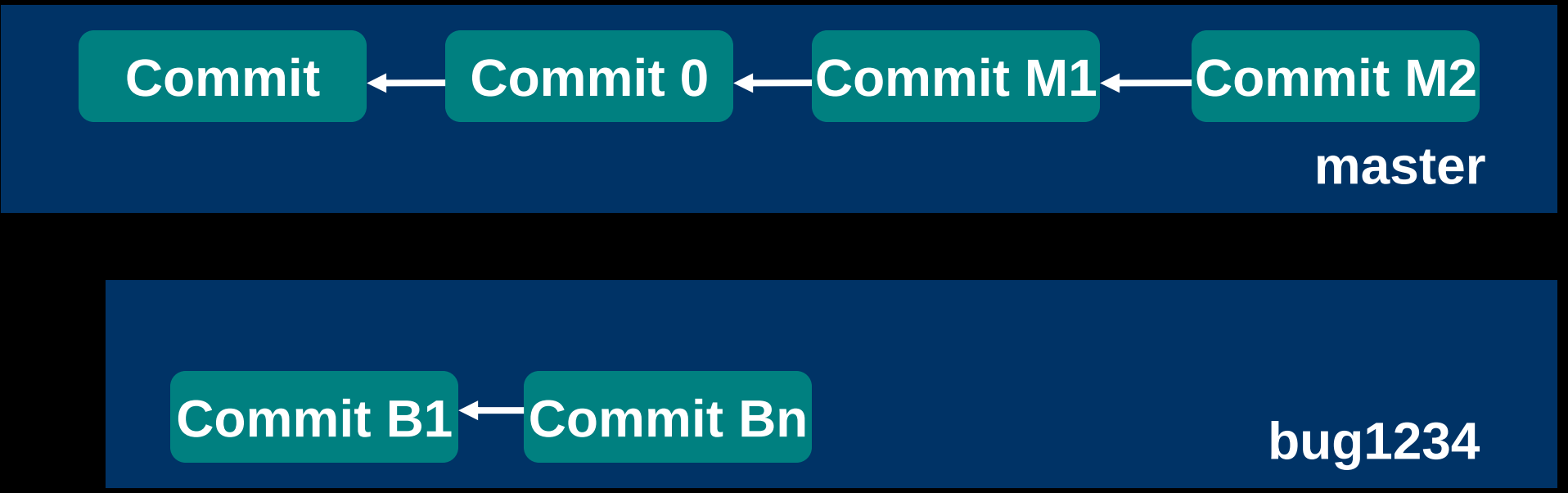
```
> git checkout master # start fresh
> git pull
> git checkout –b bug1234
> vi bugfix.txt
> git commit –a
> git checkout master
> git pull # new commits come down
```

Syncing branches

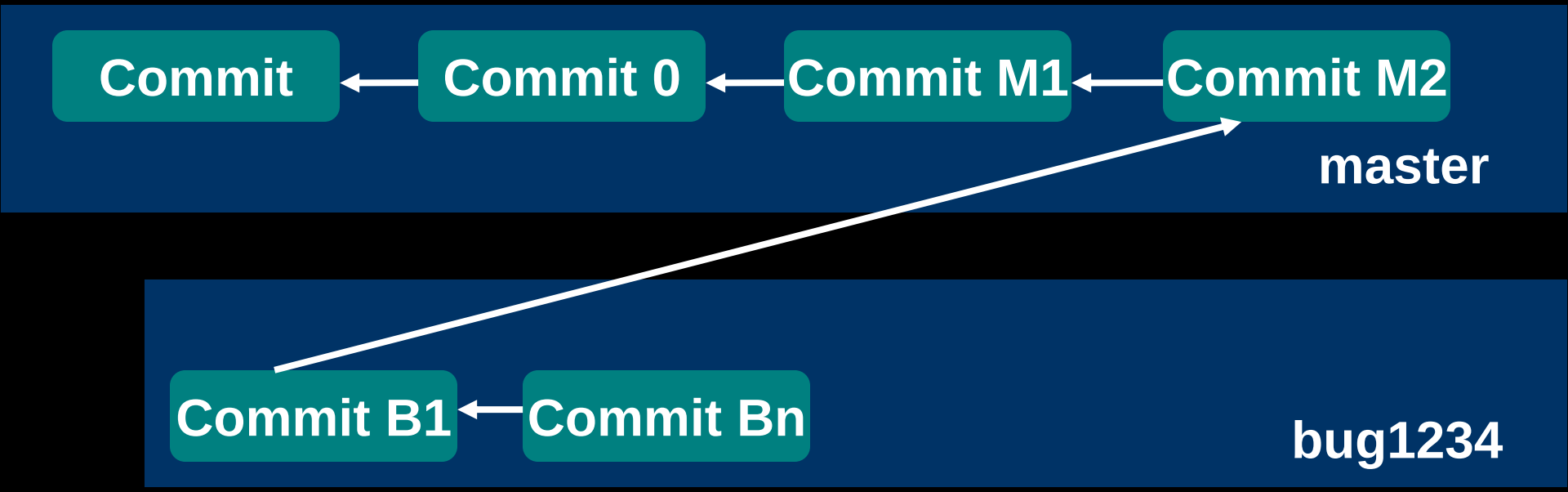


> git **checkout** bug1234
> git **rebase** master

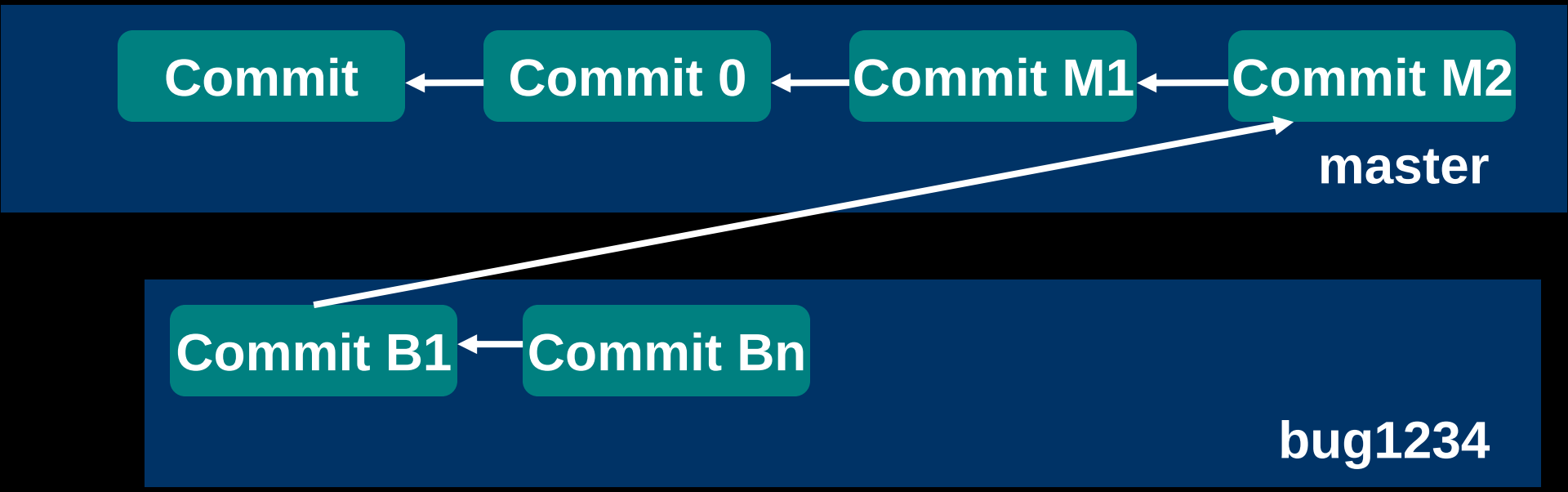
Syncing branches



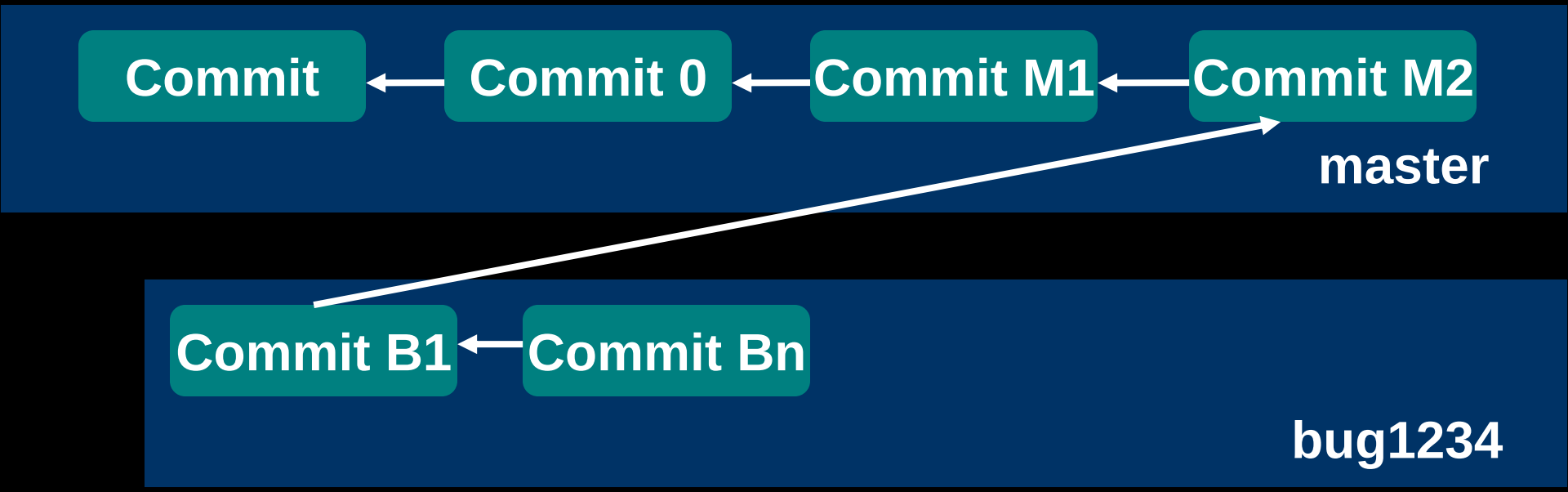
Syncing branches



Syncing branches



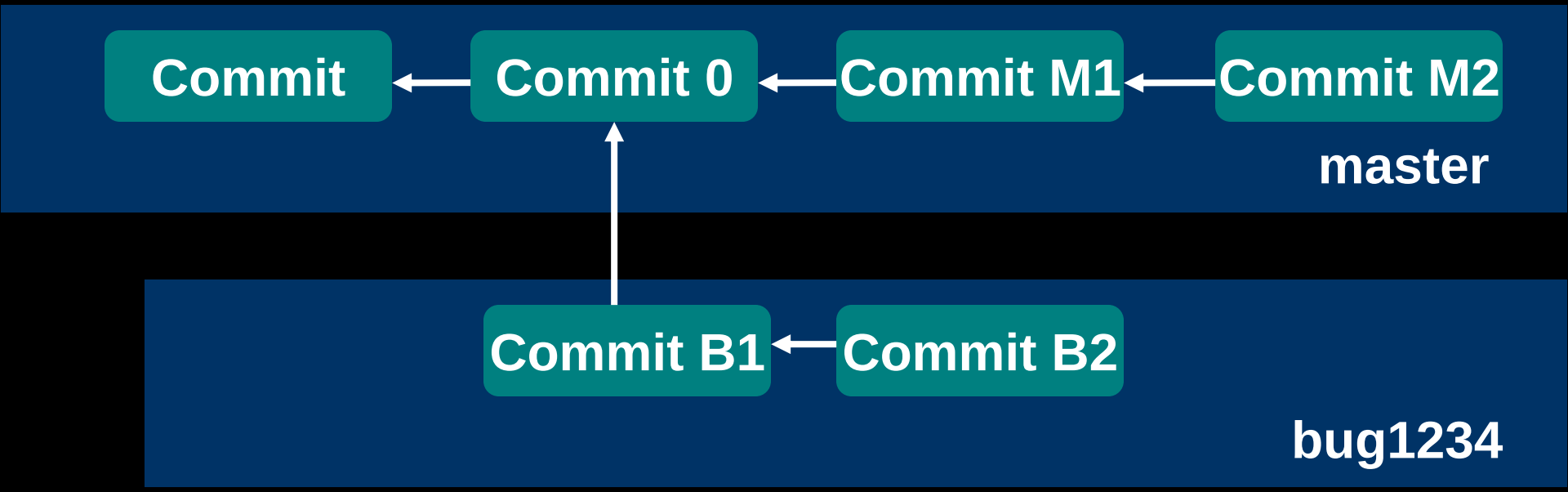
Syncing branches



Recap: Rebase

- Rewinds branchB
- Fast-forwards branchB to new HEAD of master
- Replays branchB commits
- Assumes upstream is vetted
- Rearranges branchB's history
- Branch B's commits receive new hashes
- Great for keeping downstream branches in sync with upstream changes
- Do NOT rebase once commits have been pushed upstream!

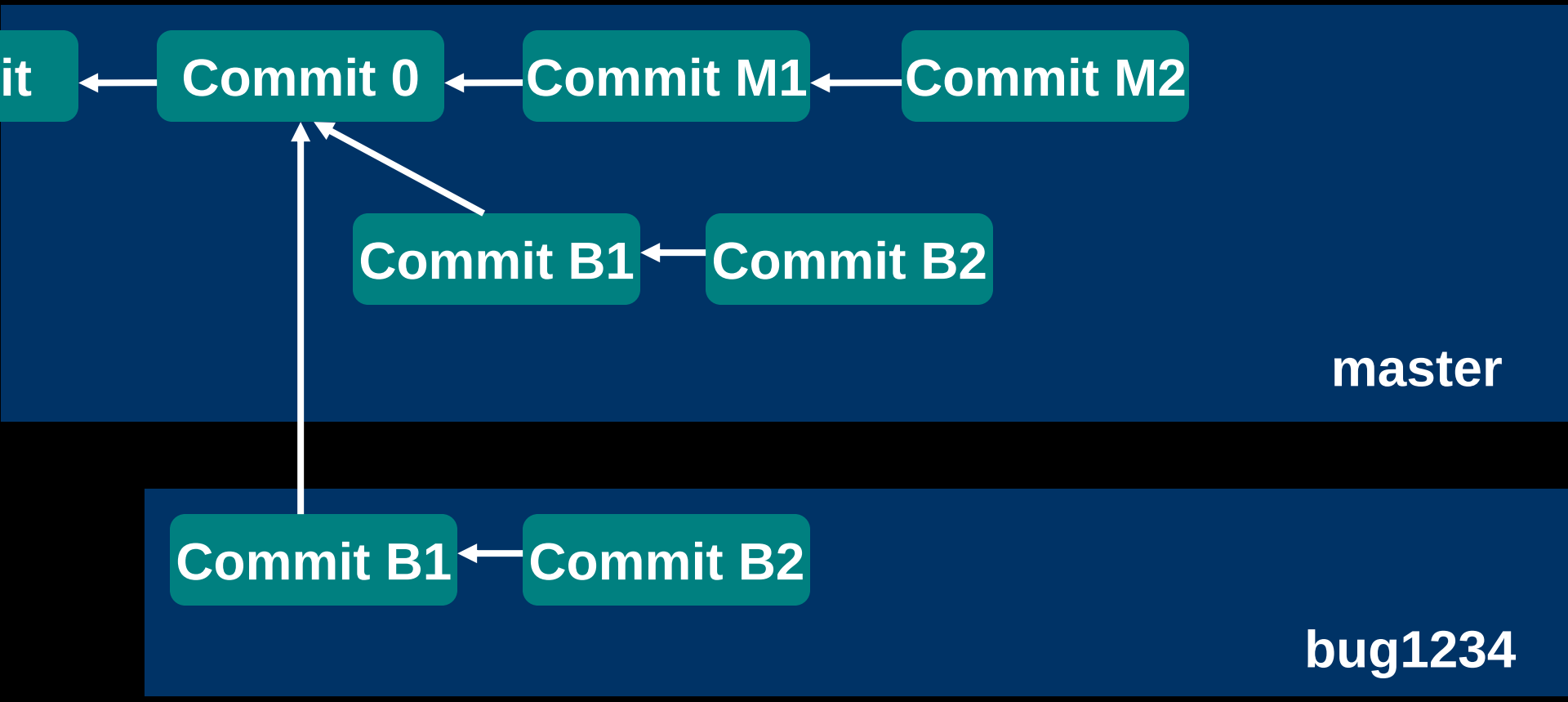
Sending changes upstream



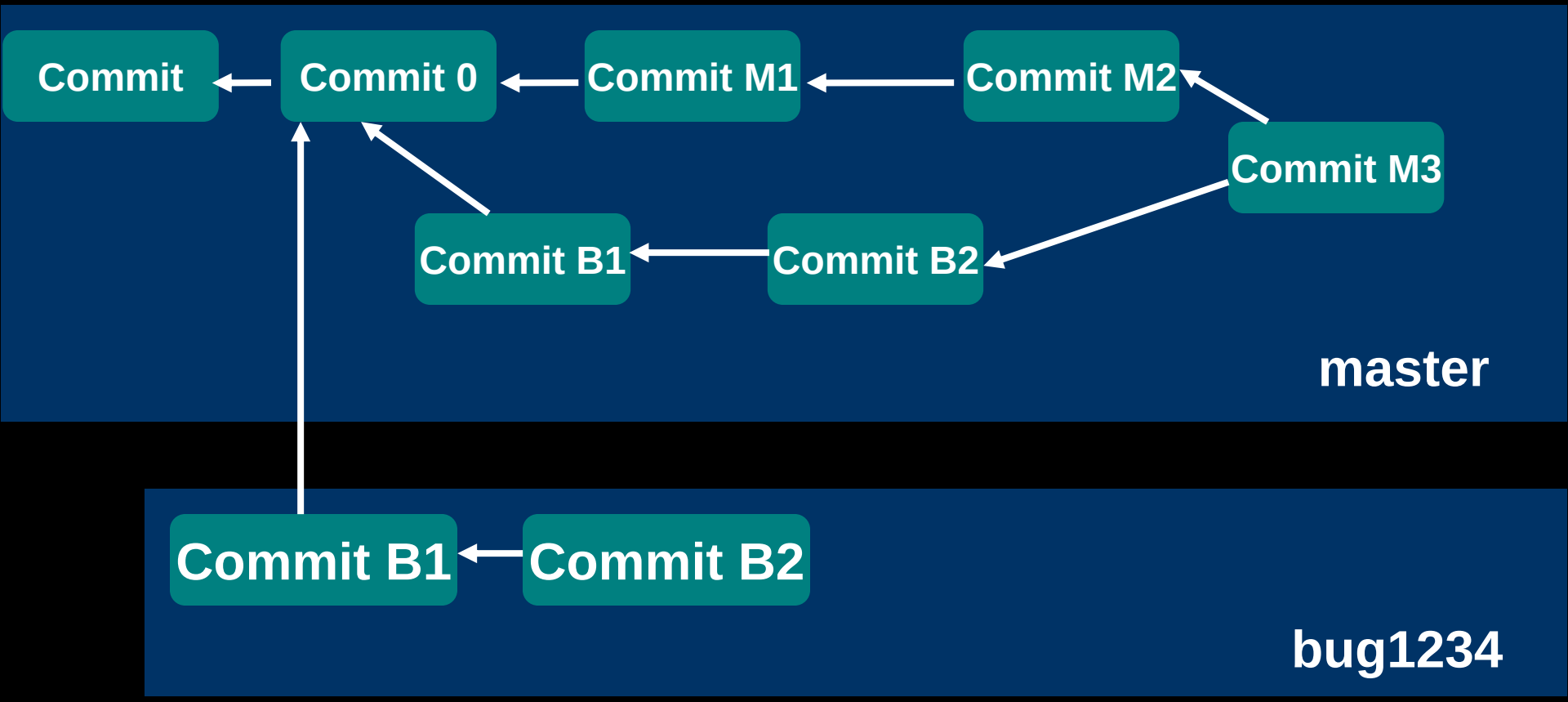
> git **checkout** master

> git **merge** bug1234

Sending changes upstream



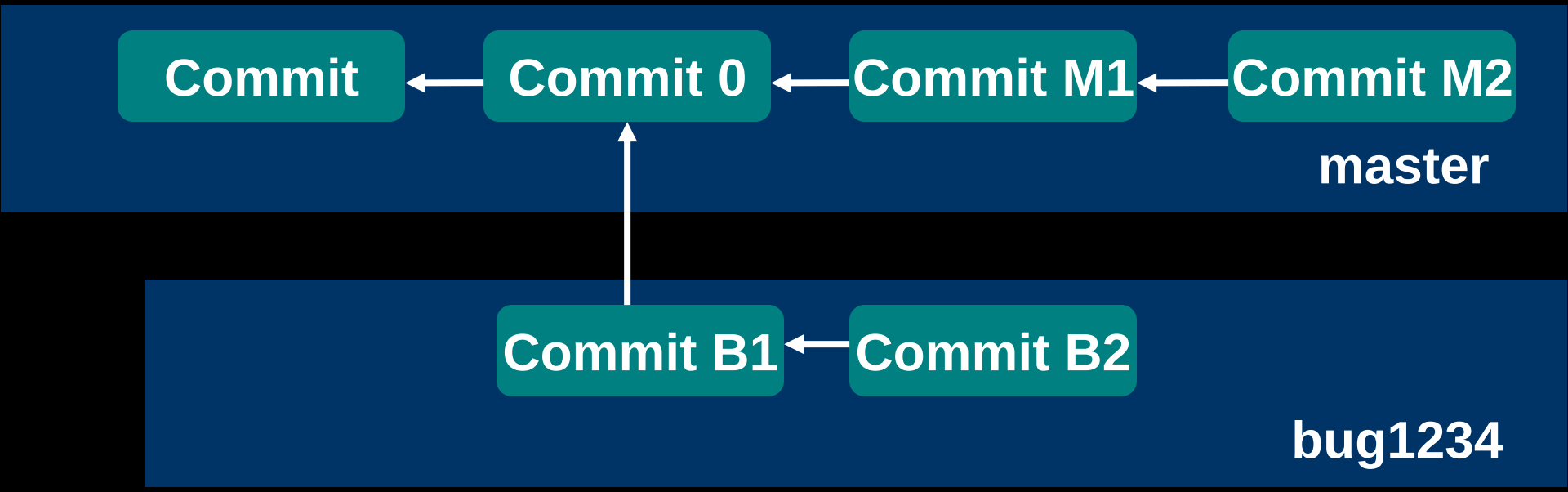
Sending changes upstream



Merging updates

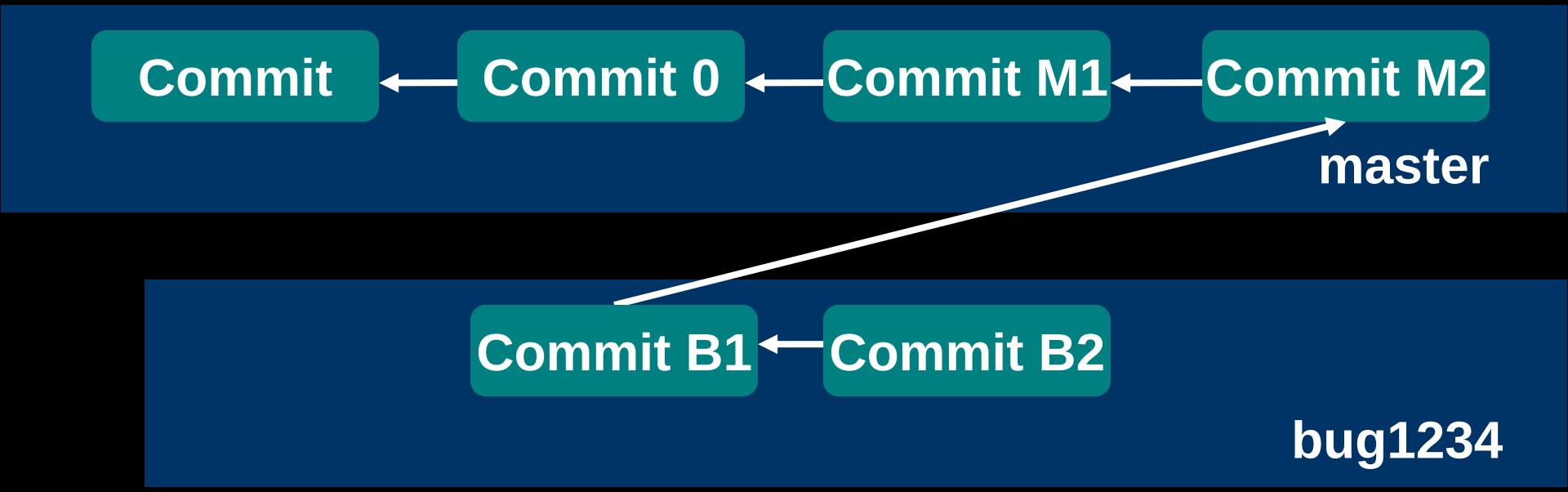
- Unifies 2 or more histories
- All branches treated as equals
- Preserves all existing hashes
- Creates extra merge commit pointing to multiple parents
- Great for updating upstream branches with downstream changes

Tip: Rebase, then merge



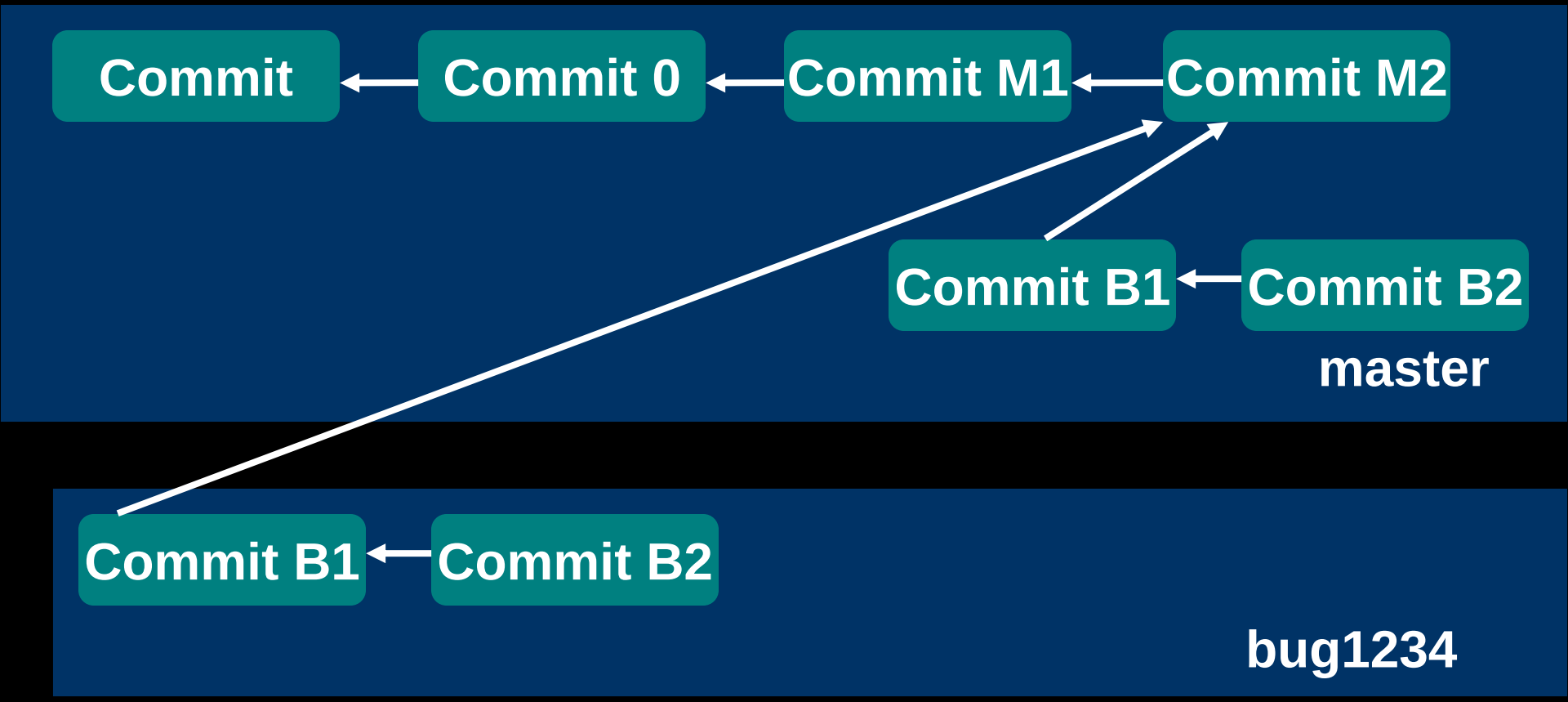
> git **checkout** bug1234
> git **rebase** master

Tip: Rebase, then merge



> git **checkout** master
> git **merge** bug1234

Sending changes upstream



Recap: Branch workflow

> git **checkout** master
> git **pull**
> git **checkout –b** bug1234
> vi bugfix.txt
> git **commit –a**
> git **checkout** master
> git **pull**
> git **checkout** bug1234
> git **rebase** master
> run unittest.txt
> git **checkout** master
> git **merge** bug1234
> git **push**
> git **branch –d** bug1234

More resources

Git Basics

- http://book.git-scm.com/index.html
- http://progit.org/book/
- http://www.kernel.org/pub/software/scm/git/docs/

Contributing to YUI

- http://developer.yahoo.com/yui/theater/video.php?v=glass-yuiconf2009-contributing
- http://yuilibrary.com/gitfaq/

More Git

- http://gitready.com/
- http://www.eecs.harvard.edu/~cduan/technical/git/
- http://gitster.livejournal.com/tag/git