

Git

Distributed Version Control System

<http://git-scm.com/>

<http://git-scm.com/doc>

<https://help.github.com/articles/what-are-other-good-resources-for-learning-git-and-github>

<https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud/>

<http://githowto.com/>

<https://github.com/git/git>

Goal of Version Control

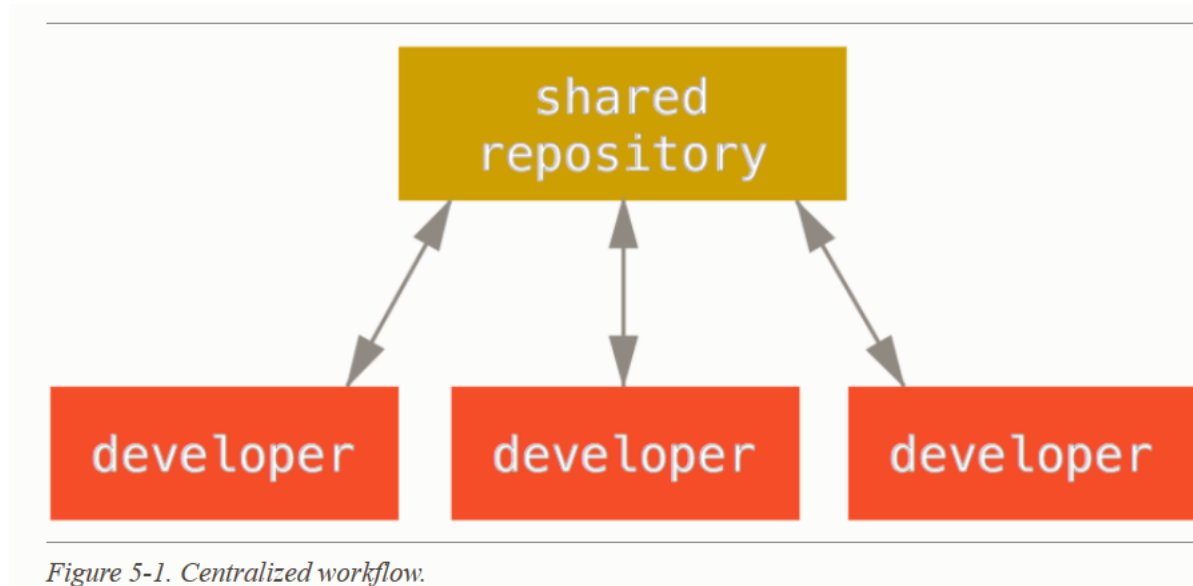
- Other options besides Git:
 - CVS, Subversion, Bazaar, BitKeeper, Team Foundation Server, ClearCase, Mercurial (hg)

History

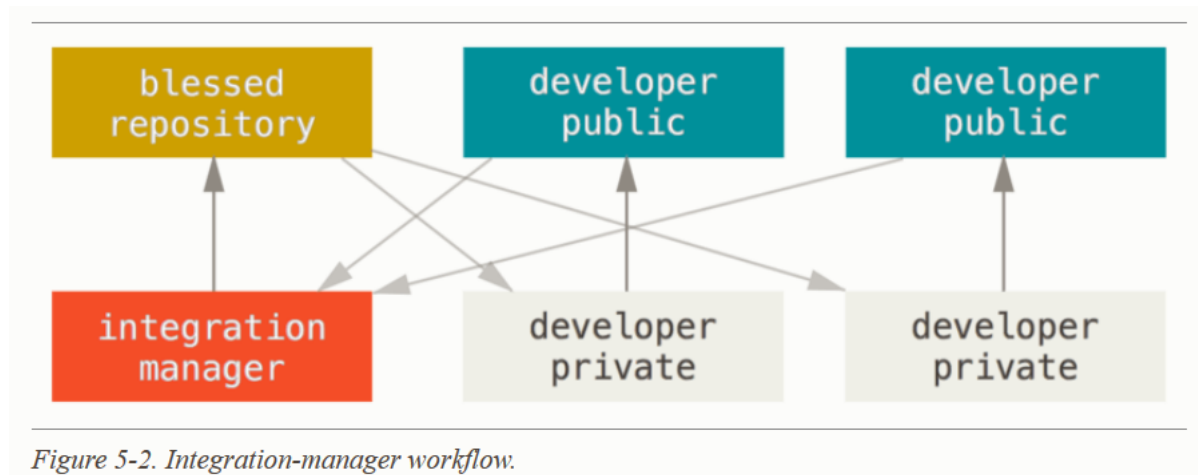
- Allow multiple people to work on the same software easily
- Allow a single user to track all his/her changes
- Developed for use with the Linux Kernel
 - move away from proprietary BitKeeper
- Modeled after Linux Kernel work flow
 - branches
 - distributed
 - data assurance
- Mix of local and remote repositories

Let's first look at using the command line then we'll look at GitHub.

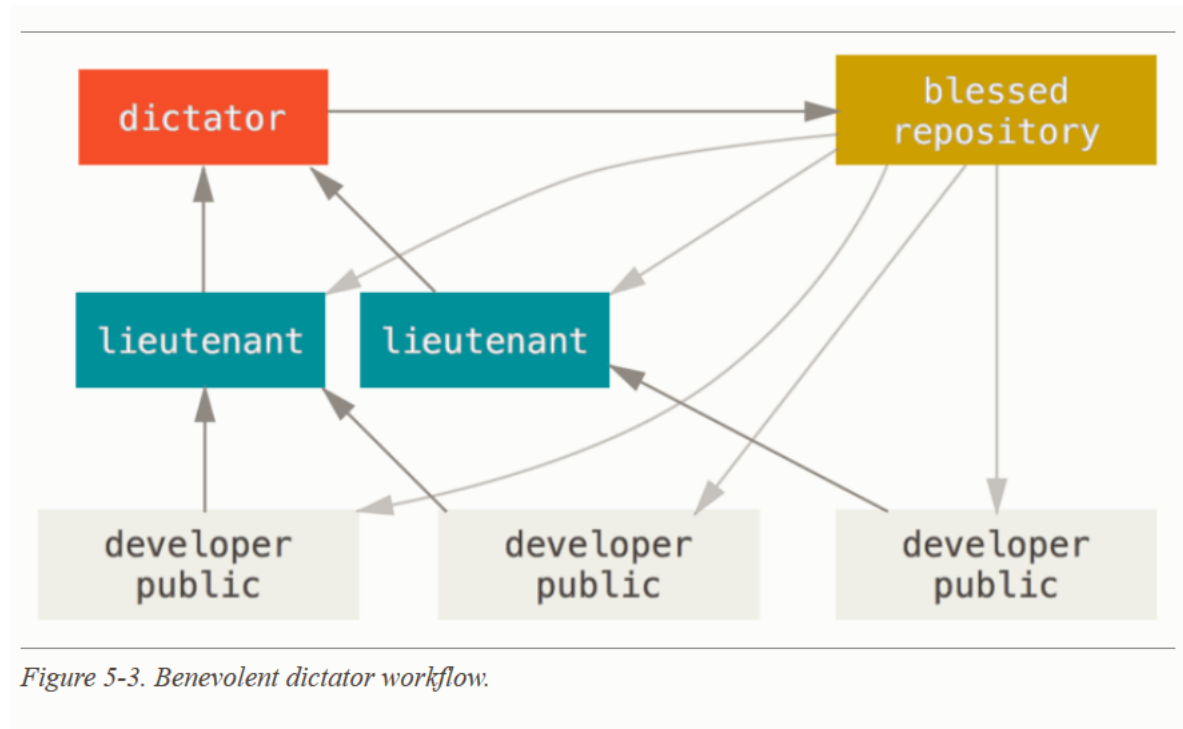
Workflows



Workflows



Workflows



Documentation

- <http://git-scm.com/docs/>
 - link to GitHub cheat sheet (PDF - 2 pages)
 - videos
 - free book (Pro Git)
 - <http://git-scm.com/book/en/Git-Basics-Undoing-Things>
 - <https://git-scm.com/book/en/v2/Git-in-Other-Environments-Git-in-Bash>
 - integrate git support into bash

Setup

- Open a terminal
 - terminator
- Go to Documents

```
script -a --timing=GitIntro.tm GitIntro.txt
```

```
git config --global core.editor "nano"
```

script will terminate when you type exit!

- Open two more terminals, don't start script!
 - double click punetid@linux, change names to One, Two, Three
 - make sure One is running script!

Typical Workflow - Single User

- `git init`
 - builds the repository (.git directory)
 - the repository is in your local working directory
- create .gitignore
 - list types of files to not put into version control
 - any file that is generated: *.obj, *.o, *.class, *.pyc
- Create files! Do work!
- `git add [filenames]`
 - add the files you just created to the index for **staging**
- `git commit -m "commit message"`
 - actually commit changes to the repository
- `git log`

Typical Workflow - Single User

One

- `mkdir MyCoolProject.git; cd MyCoolProject.git`
- `git init`
- `ls -al`
- `geany .gitignore`
 - `*.o`
 - `test`
- create test.c from the next slide
- `gcc -c -o test.o test.c`
- `gcc -o test test.o`
- `git add test.c`
- `git commit -m "initial add of test.c"`
- `git log`

test.c

```
#include <stdio.h>

int main()
{
    printf("HELLO");
    return 0;
}
```

Typical Workflow - Single User

One

- edit test.c to include `printf("-BYE\n");`
- `gcc -c -o test.o test.c`
- `gcc -o test test.o`
- `git add .`
- `git status`
- `git commit -m "updated test.c to say BYE"`
- `git log`

I need to revert!

One

- <http://git-scm.com/book/en/Git-Basics-Undoing-Things>
- `git log --name-status`
- `git diff <commit hash> test.c`
- `git checkout <commit hash> test.c`
- `edit test.c`
- `git add .`
- `git commit`
- `git log`

<http://stackoverflow.com/questions/215718/reset-or-revert-a-specific-file-to-a-specific-revision-using-git/373848#373848>

* two dashes precede a command line option of more than one character

CS360

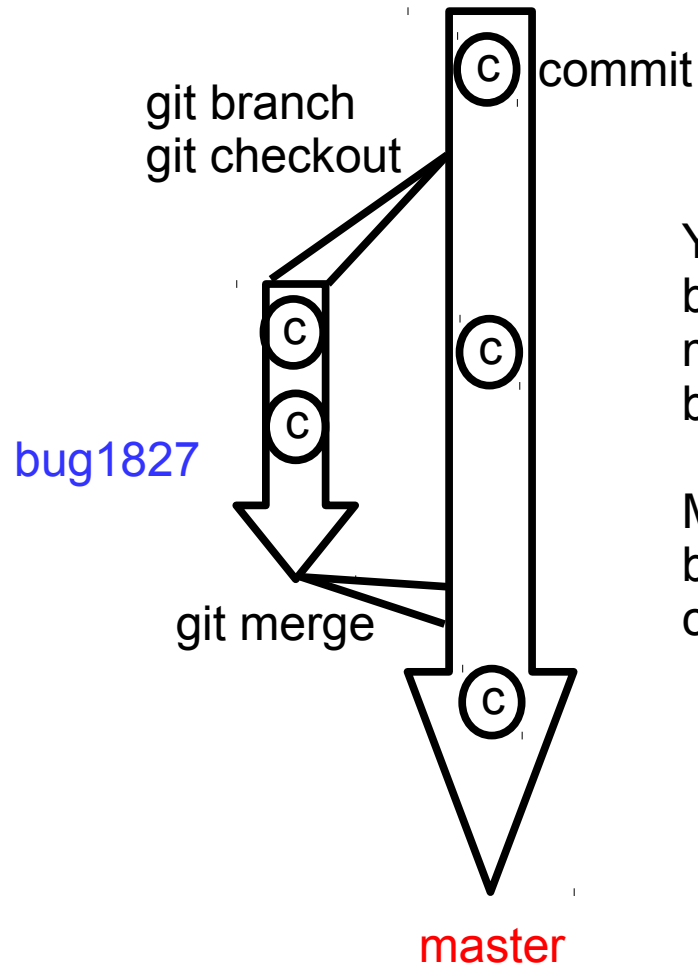
Pacific University

Branches

- Master
 - Main line of development
 - Often this is always kept buildable
- Branches
 - Initially a copy of Master (not always...)
 - Used to build a feature
 - Used to fix a bug
 - Not necessarily always buildable
 - Not necessarily public
 - Maybe local to a developer
 - This is where your organization's culture comes into play.

Typical Workflow

- Single user – bug fix! (or maybe feature add)



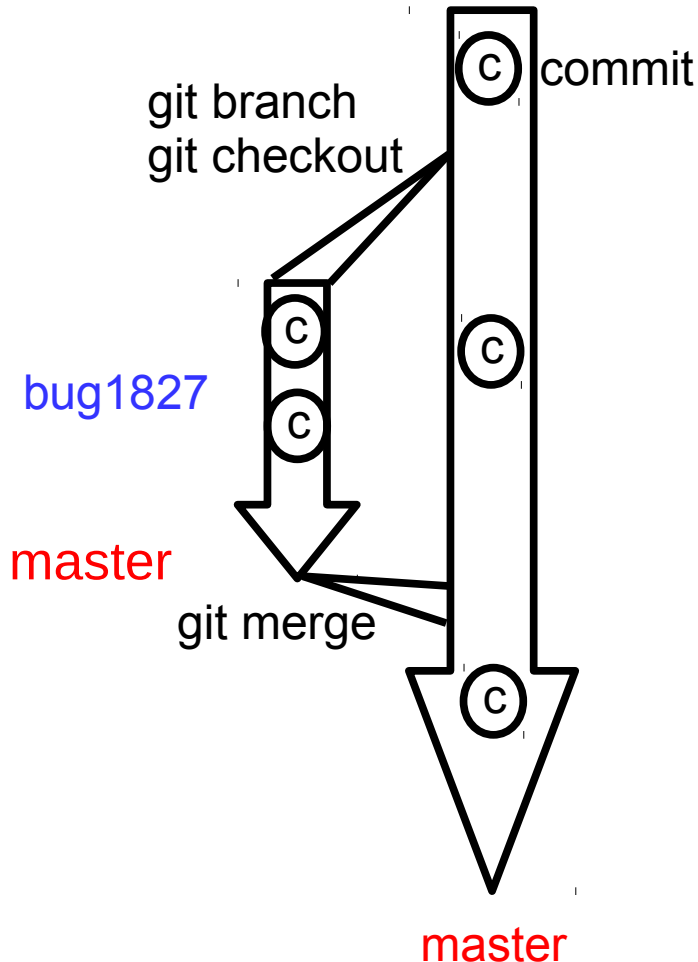
You might merge master into bug1827 if the changes in master are relevant to bug1827 but you are not done with bug1827.

Merges are non-destructive. Both branches continue to exist and can be used/edited/committed/branched.

<https://www.atlassian.com/git/tutorials/merging-vs-rebasing/>

Typical Workflow

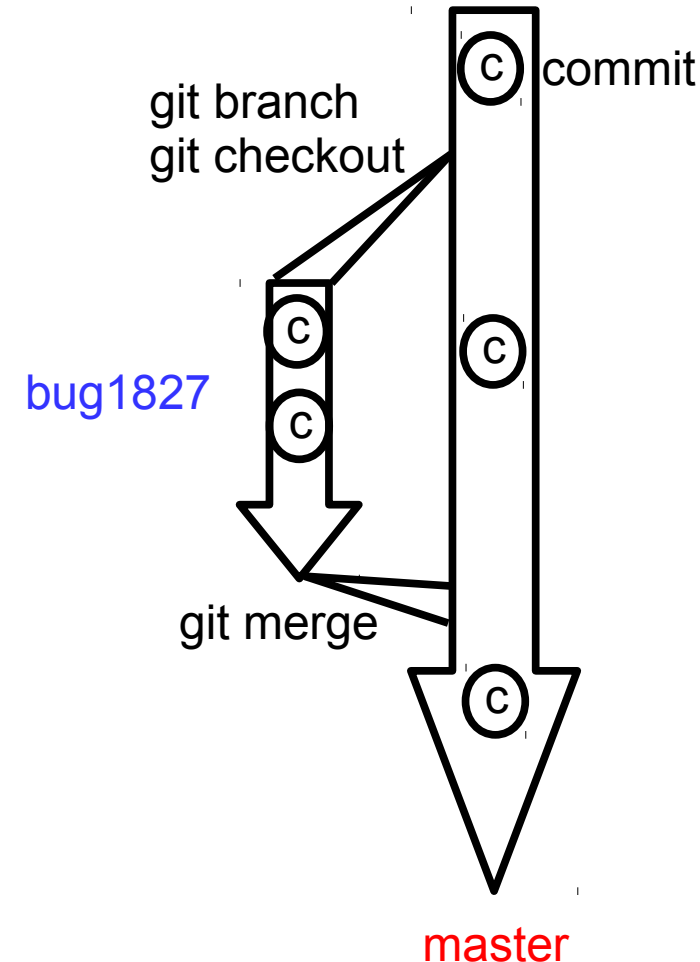
- Single user – bug fix! (or maybe feature add)
 - `git branch bug1827`
 - create a branch to contain all the work for the bug fix
 - `git checkout bug1827`
 - start using that branch
 - Do work (add/commit)
 - `git checkout master`
to work on master again.
 - `git merge --no-ff bug1827`
 - replay the commits on bug1827 into **master**
 - `git log`
 - `git branch -d bug1827`



Typical Workflow

One

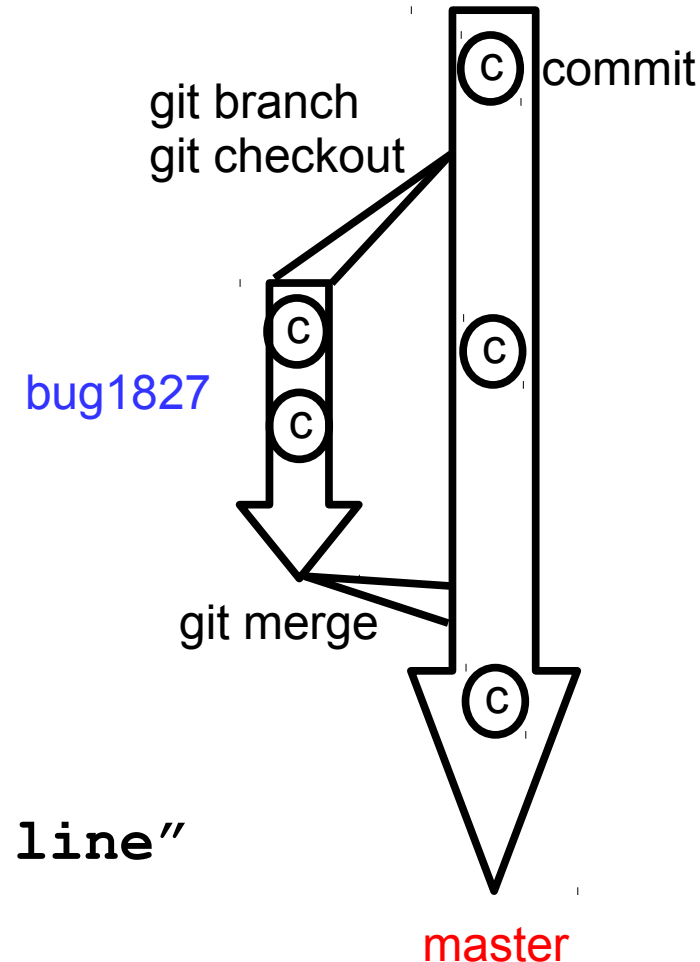
- Single user – bug fix! (or maybe feature add)
 - `git branch bug1827`
 - `git checkout bug1827`
 - Add `printf("CS360\n");` to `test.c`
 - `git add .`
 - `git commit -m "added CS360 line"`
 - `git log`
 - `git checkout master`
 - `cat test.c`
 - `git merge --no-ff bug1827`
 - `git log`
 - `git status`
 - `git branch`
 - `git branch -d bug1827`



Typical Workflow

One

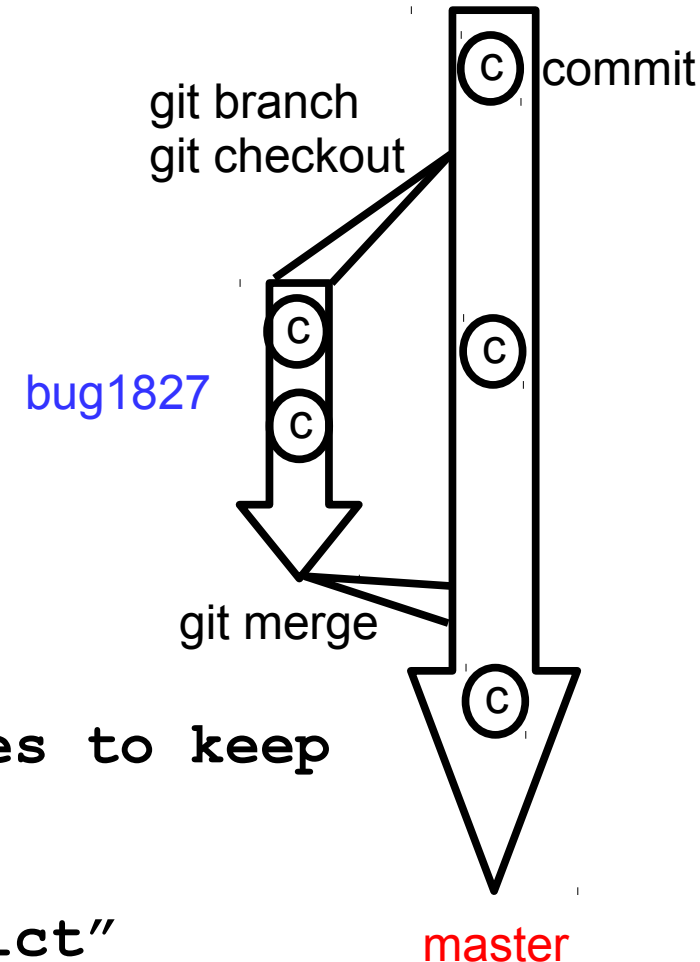
- Single user – bug fix! (or maybe feature add)
 - `git log --graph`
 - `git blame test.c`
 - `add printf("Come back later");`
 - `cat test.c`
 - `git stash`
 - `cat test.c`
 - `git stash list`
 - `git stash show`
 - `git stash apply`
 - `cat test.c`
 - `git add .`
 - `git commit -m "committed stashed line"`
 - `git log`



Typical Workflow

One

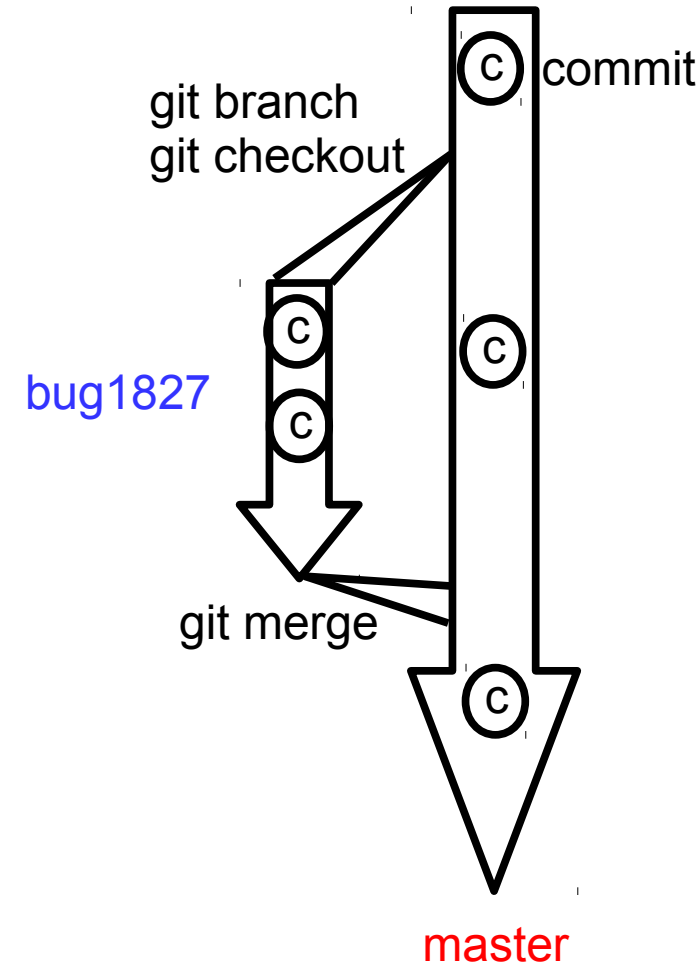
- Single user – bug fix! (or maybe feature add)
 - `git branch MERGE_CONFLICT`
 - `git checkout MERGE_CONFLICT`
 - add ! to "come back later"
 - `git add .`
 - `git commit -m "added bang"`
 - `git checkout master`
 - add <> to "come back later"
 - `git add .`
 - `git commit -m "added angles"`
 - `git merge MERGE_CONFLICT`
 - edit `test.c` and choose which lines to keep
 - `git add .`
 - `git commit -m "fixed merge conflict"`



Typical Workflow

One

- Single user – bug fix! (or maybe feature add)
 - `git log --graph`
 - `git blame test.c`
 - `git branch -d MERGE_CONFLICT`
 - `git log --graph`



More commands

- <http://git-scm.com/docs>
 - `git stash`
 - `git status`
 - what files have uncommitted changes?
 - `git log`
 - show the commits and log messages
 - `git diff`
 - show the differences between local and committed files
 - build a patch you can email to someone else
 - `git apply`
 - apply a patch to your working directory
 - `git blame`
 - who last changed each line of a file?
 - `git bisect`
 - try to determine when a property of your project changed

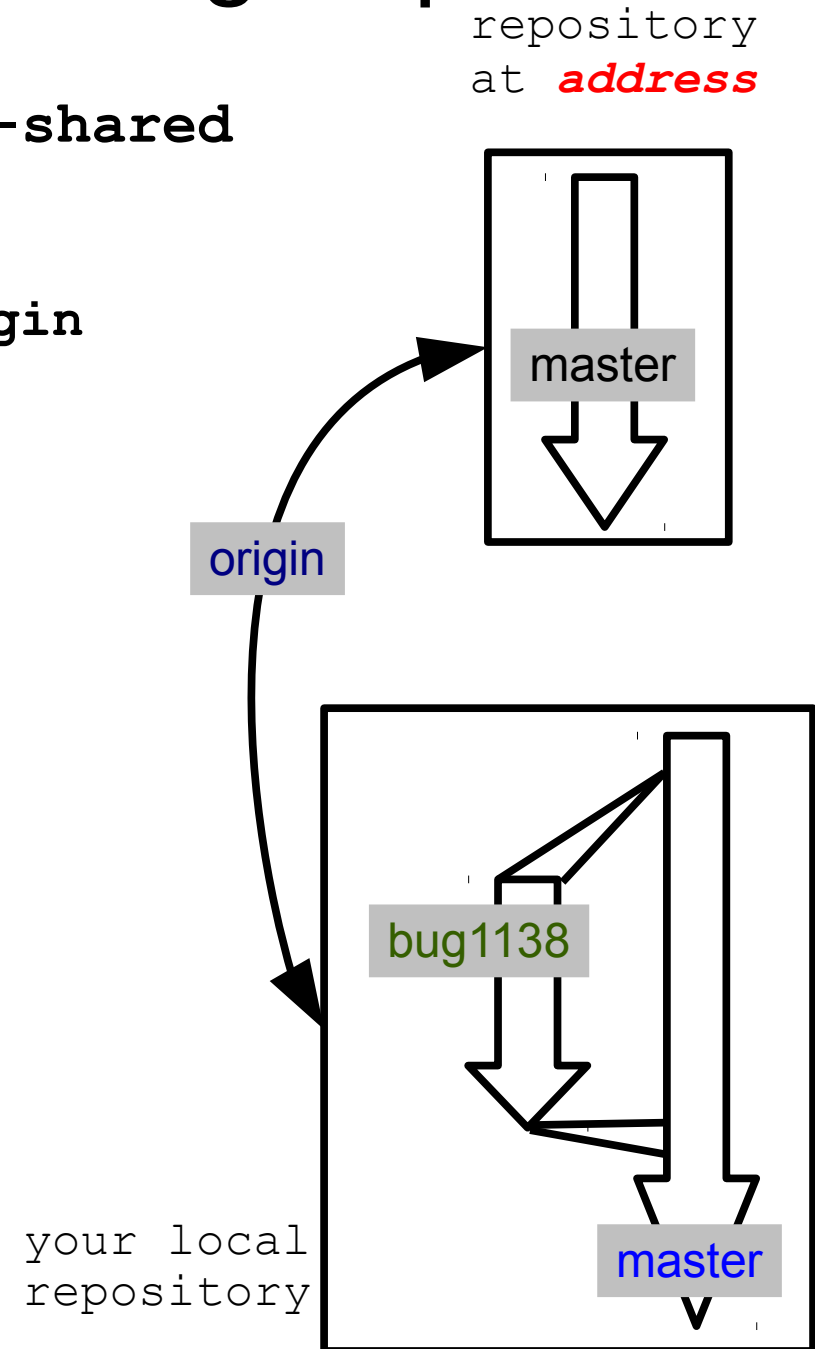
Workflow - Group

- <http://nvie.com/posts/a-successful-git-branching-model/>
- <http://scottchacon.com/2011/08/31/github-flow.html>

Typical Workflow - group

- Group of developers

- someone else: `git init --bare --shared`
- `git clone address`
 - pull down code and setup `origin`
 - `git remote -v`
- `git branch bug1138`
 - only a local branch is created
- `git checkout bug1138`
- do work
- `git add files / git commit`
- `git checkout master`
- `git merge --no-ff bug1138`
- `git push origin master`
- `git branch -d bug1138`

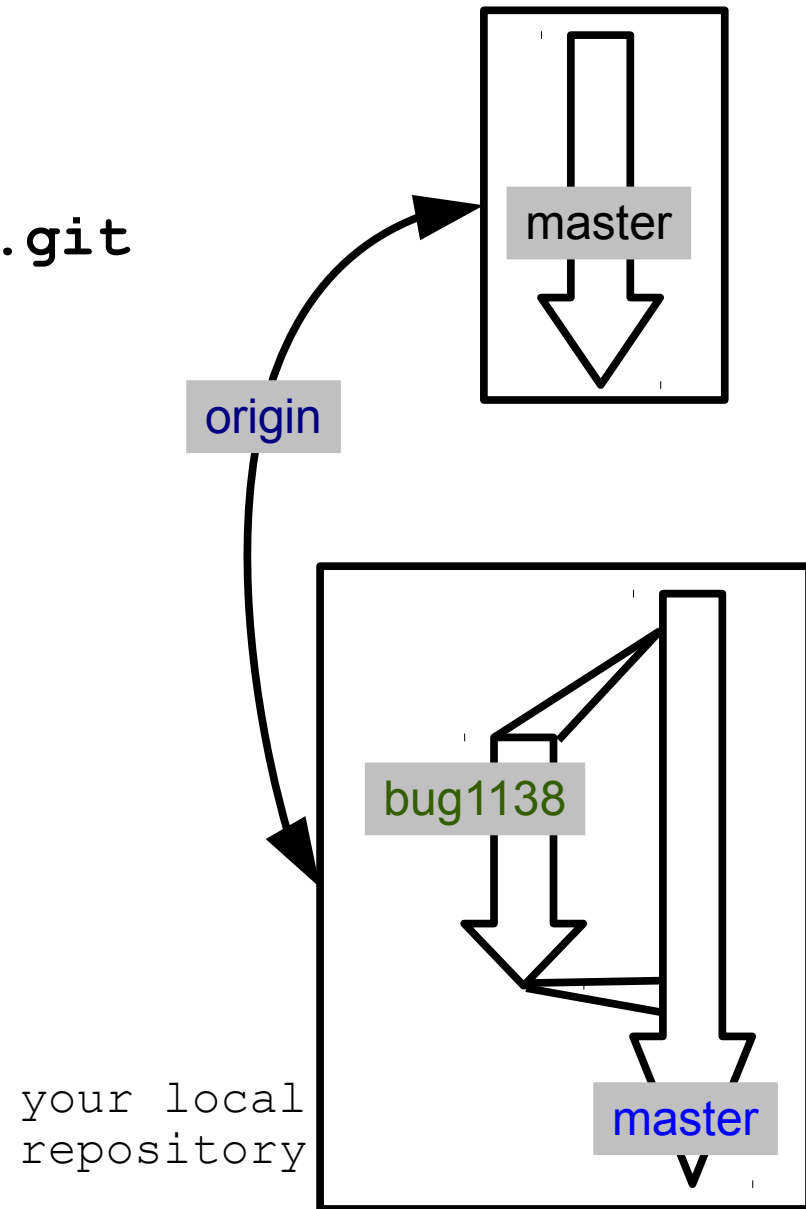


Typical Workflow - group

Two
repository
at *address*

- Group of developers

- setup the remote repos
- `ssh zeus.cs.pacificu.edu`
- `mkdir gitTest.git; cd gitTest.git`
- `git init --bare --shared`



Typical Workflow - group

One

- Group of developers
 - `cd ~/Documents`
 - `git clone`
`ssh://YOU@zeus.cs.pacificu.edu/home/YOU/gitTest.git`
 - `cd gitTest`
 - `git remote -v`
 - `create test.c`
 - `git add .`
 - `git commit -m "initial commit"`
 - `git push origin master`
 - in terminal Two: `git log`

Typical Workflow - group

One

- `git branch bug1138`
- `git checkout bug1138`
- `git branch`
 - in terminal Two: `git branch`
- `edit test.c / git add files / git commit`
- `git log`
 - in terminal Two: `git log`
- `git checkout master`
- `git merge --no-ff bug1138`
- `git branch`
- `git push origin master`
 - in terminal Two: `git log --graph ; cat test.c`
- `git branch -d bug1138`

Typical Workflow - group

```
git fetch
```

```
git log ..origin/master
```

```
git checkout origin/master
```

```
git checkout master
```

```
git merge origin/master
```

Get changes from
origin
(remote repos)

OR

```
git pull
```

- git pull performs lots of magic
- hard to fix things when magic fails.

Typical Workflow - group

Three

- `mkdir ~/Documents/Fetch; cd ~/Documents/Fetch`
- `git clone`
`ssh://YOU@zeus.cs.pacificu.edu/home/YOU/gitTest.git`
- `cd gitTest`
- `git remote -v`
- `cat test.c`
- `git checkout -b FetchMe`
- `edit test.c / git add files / git commit -m "fetch"`
- `git log`
- `git checkout master`
- `git merge --no-ff FetchMe`
- `git branch`
- `git push origin master`
- `git branch -d FetchMe`

Typical Workflow - group

One

- `ls -al`
- `cat test.c`
- `git fetch`
- `git log ..origin/master`
- `git checkout origin/master`
- `git checkout master`
- `git merge origin/master`
- `edit test.c/add /commit -m "again"`
- `git push origin master`
 - in terminal Three:
 - `git pull`
 - `cat test.c`
 - `git log`

Conflict

- edit/add/commit/push

- edit/add/commit/push (ERROR)
- fetch/merge (ERROR)
- edit file (resolve conflict)

- add/commit/push

```
<<<<<<< HEAD  
BUY  
=====  
BYE  
>>>>>> origin/master
```

- fetch/merge

Tag

One

- `git tag -a MarkTwo -m "this is a tag"`
- `git push --follow-tags`
- `git log --decorate=full`

A tag is just one type of ref.

A ref is an alias for a particular commit.

Other types of refs are branch names and remotes.

Branches are easy in Git since they are just an alias for a particular commit. (mostly)

<https://www.atlassian.com/git/tutorials/refs-and-the-reflog/>

Pull Request

One

- `git checkout -b Request-Pull`
- `edit test.c`
- `git add .`
- `git commit -m "email"`
- `git checkout master`
- `git merge --no-ff Request-Pull`
- `git push origin master`
- `git request-pull MarkTwo`
`ssh://YOU@zeus.cs.pacificu.edu/home/YOU/git`
`Test.git master`

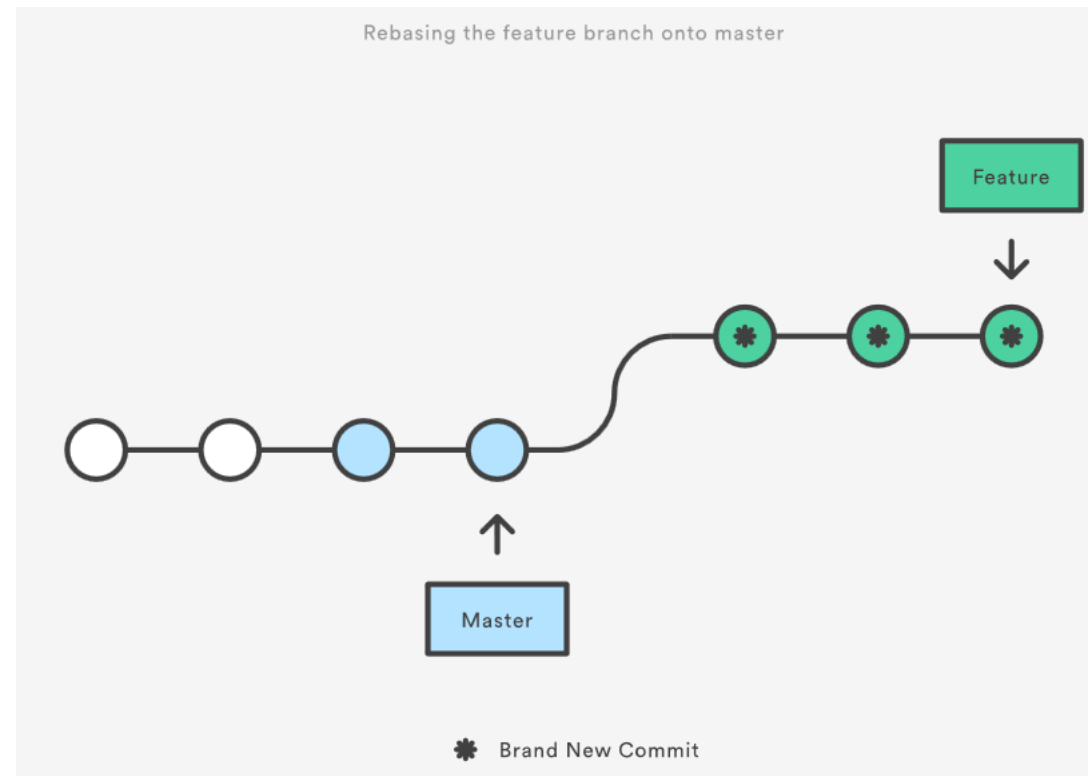
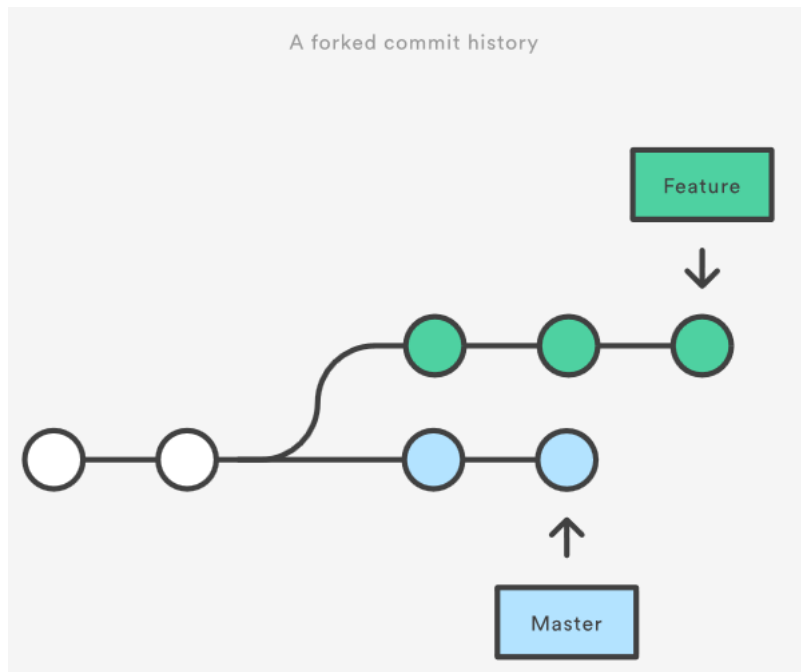
Cherry Pick

- `git cherry-pick <commit-hash>`
- pick a commit from one branch and reapply that one commit to another branch
 - rather than merge all the commits on the branch at the current location

Rebase

- Move a set of commits from one starting point to another
 - “integrate changes from one branch to another”
 - **“The golden rule of git rebase is to never use it on public branches.”**
 - “potentially catastrophic” for collaboration

Automatic
vs interactive



<https://www.atlassian.com/git/tutorials/merging-vs-rebasing/>

<https://medium.freecodecamp.com/git-rebase-and-the-golden-rule-explained-70715eccc372>

Typical Workflow - GitHub

- Individual
 - Create repository at GitHub
 - setup `.gitignore` and license.
 - `git clone git@github.com:USER/REPOS.git`
 - pull down code and setup `origin`
 - `git checkout -b bug1138`
 - do work
 - `git add files`
 - `git commit ...`
 - `git checkout master`
 - `git merge --no-ff bug1138`
 - `git push origin master`

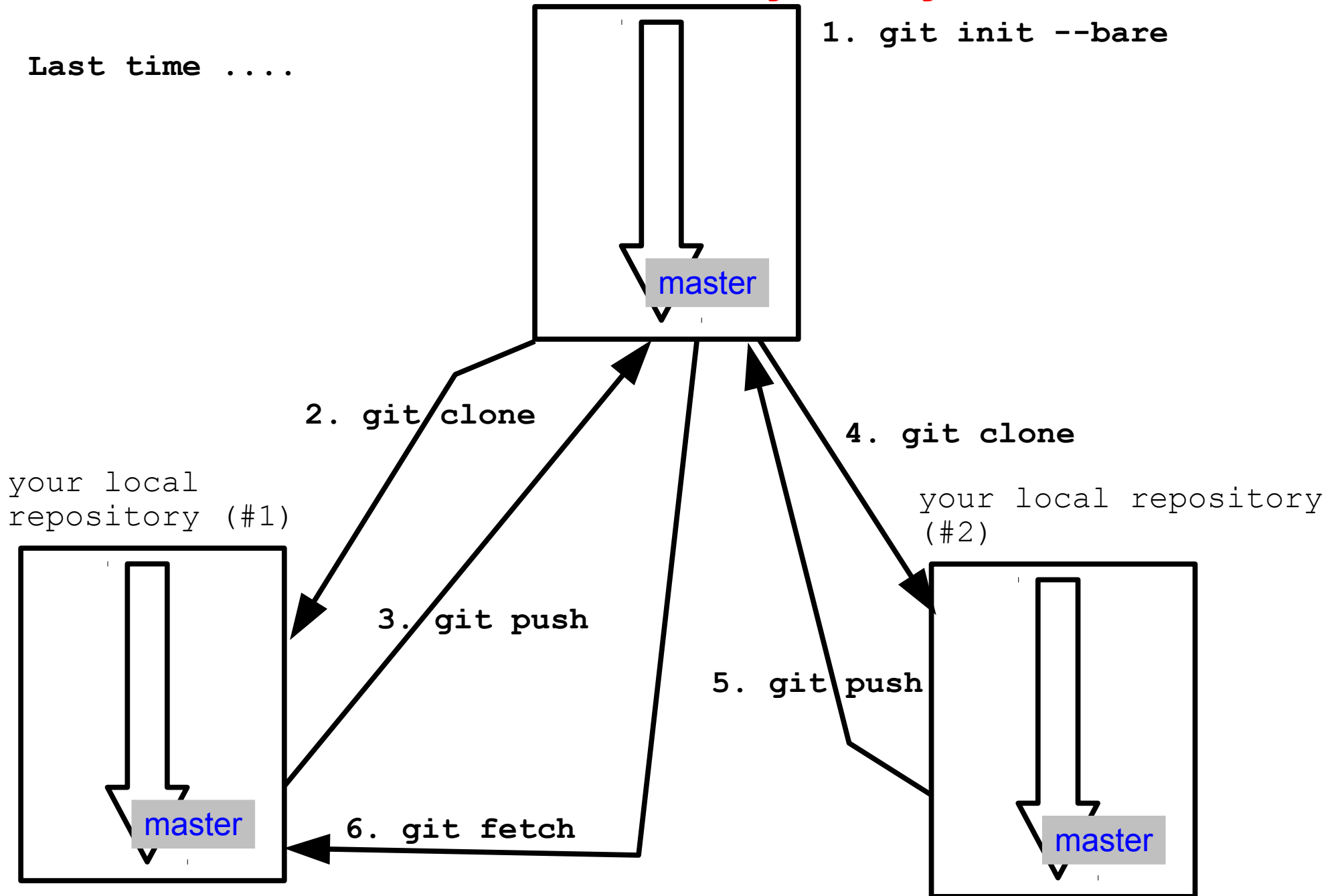
Typical Workflow - GitHub

- Group of developers
 - someone else creates a repository on GitHub (**OTHERUSER**)
 - on your GitHub account, **fork** the repository
 - `git clone git@github.com:USER/repos.git`
 - `git remote add upstream
git@github.com:OTHERUSER/FirstGitPractice.git`
 - `git checkout -b bug1138`
 - do work/add/commit
 - `git push origin bug1138`
 - push to **YOUR** GitHub repository
 - On GitHub, issue a Pull request!
 - `git fetch upstream`
 - `git merge upstream/master`



repository at // public or master
zeus/home/YOU/gitTest.git

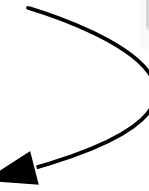
Last time



Forks and branches

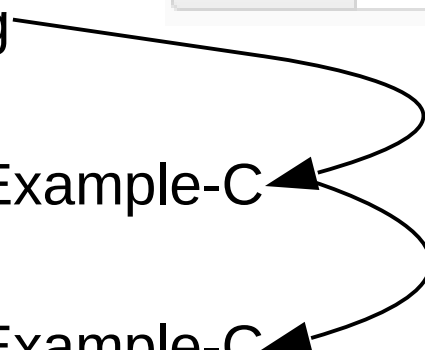
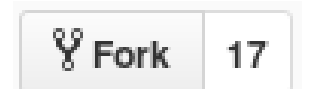
- Correct usage: I used branches

- <https://github.com/chaddcw/pullRequestTesting>
- <https://github.com/cs360f16/pullRequestTesting>

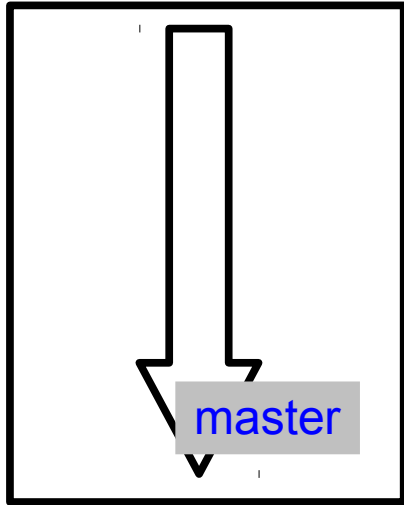


- Horrible mess: I did not use branches :(

- <https://github.com/chaddcw/pullRequestTesting>
- <https://github.com/cs360f14/ContactManager-Example-C>
- <https://github.com/cs360f16/ContactManager-Example-C>

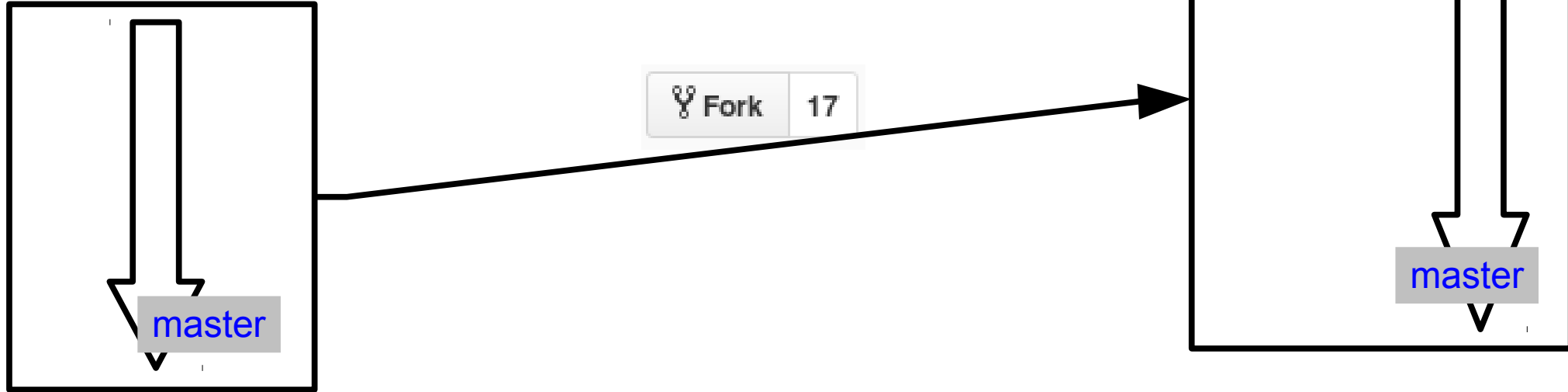


repository at
github.com/GROUP/REPOS



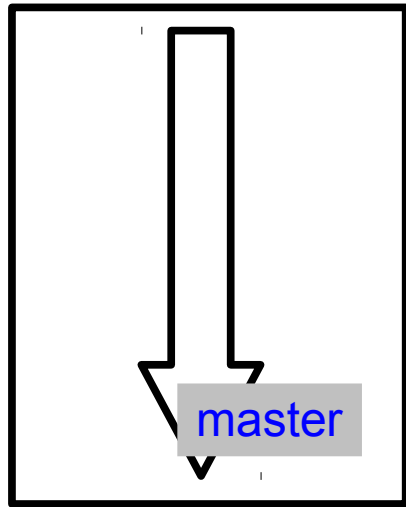
repository at
github.com/GROUP/REPOS

repository at
github.com/USER/REPOS

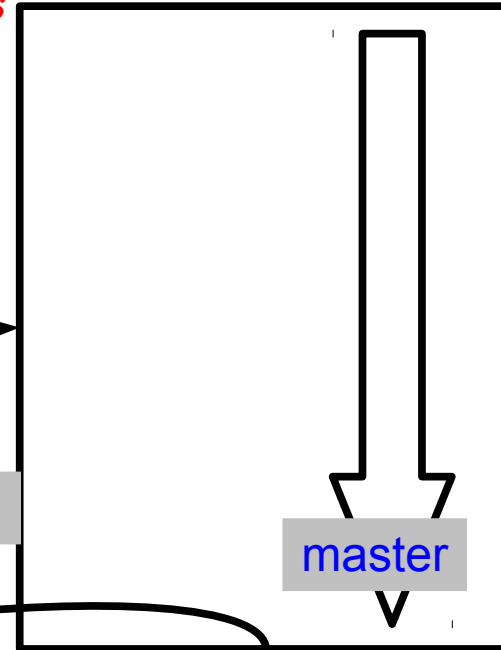


repository at
github.com/GROUP/REPOS

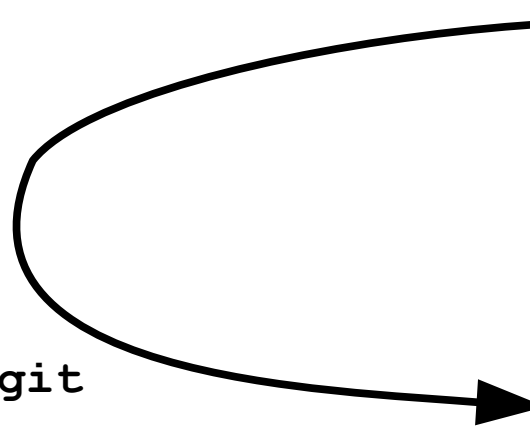
repository at
github.com/USER/REPOS



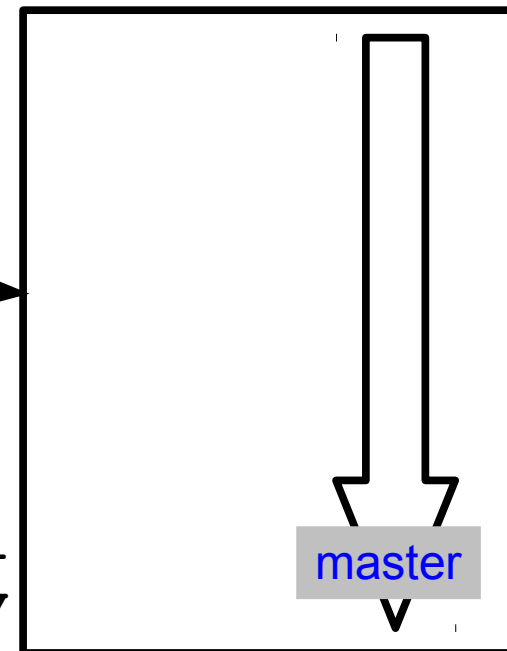
origin



```
git clone git@github.com:USER/REPOS.git
```

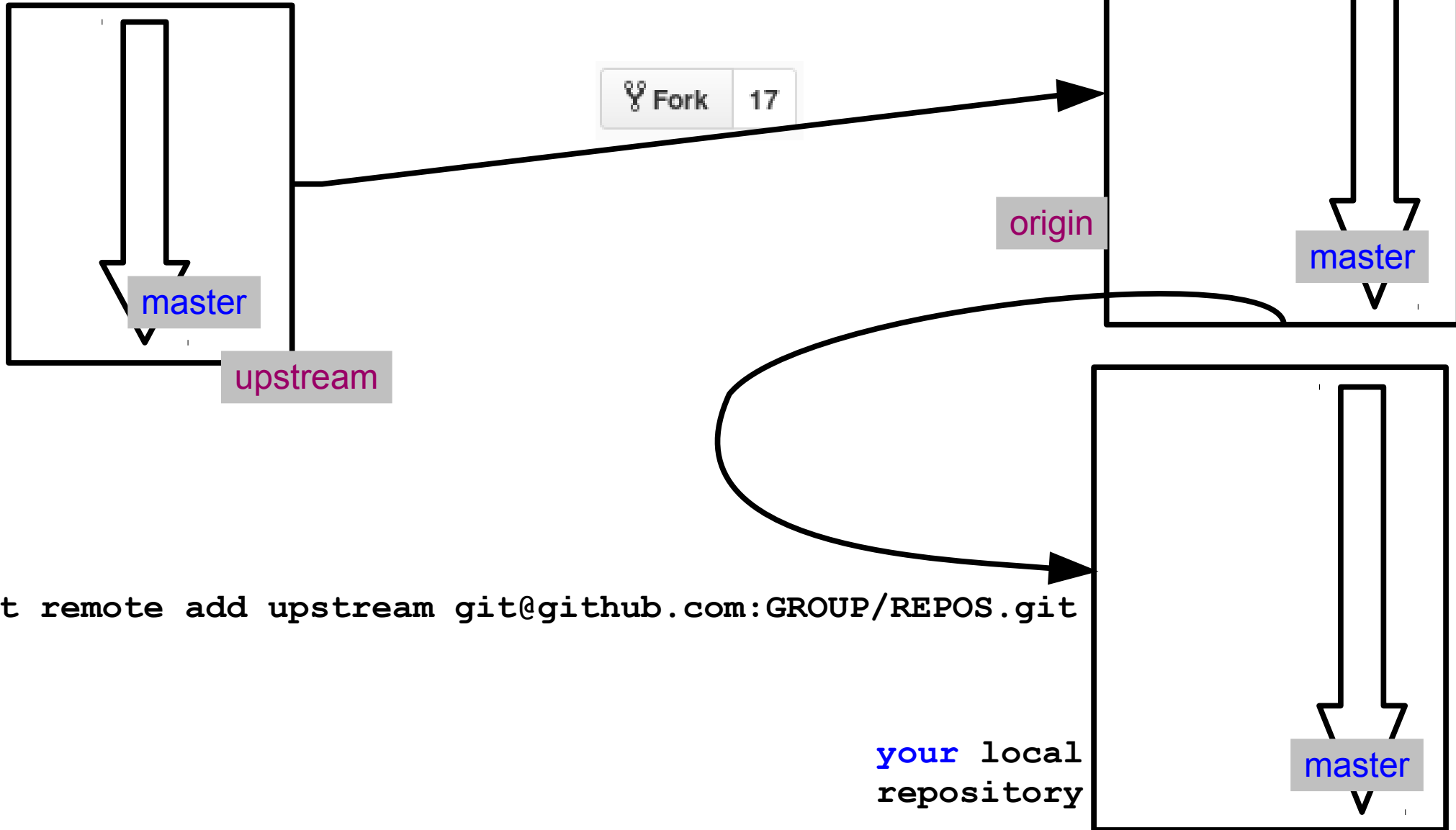


your local
repository

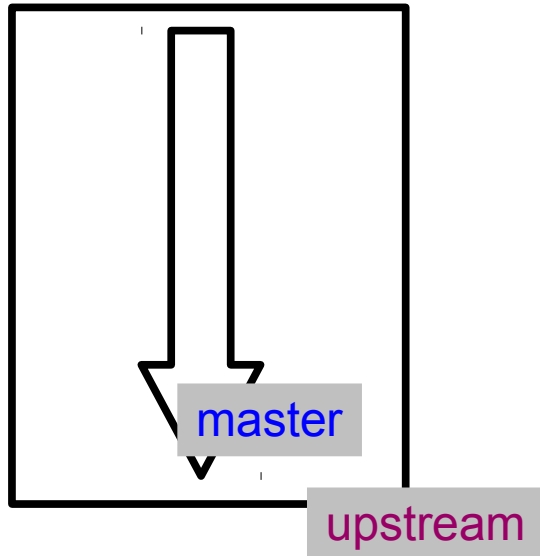


repository at
github.com/USER/REPOS

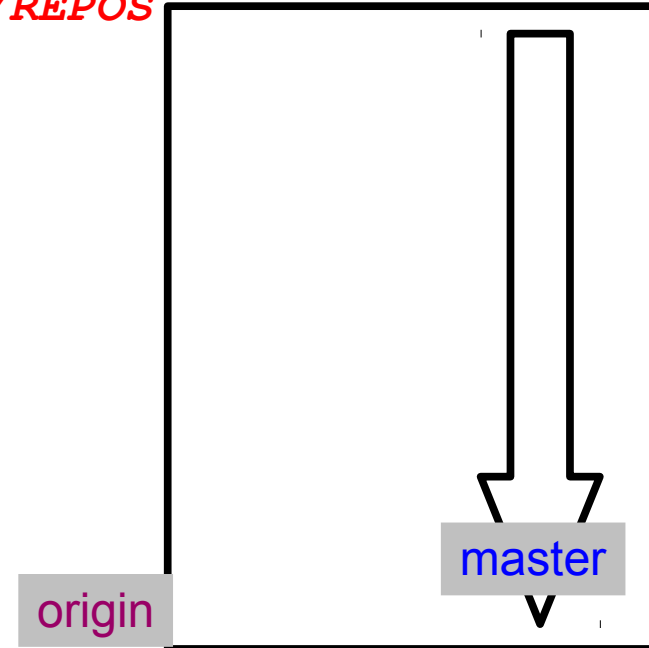
repository at
github.com/GROUP/REPOS



repository at
github.com/GROUP/REPOS

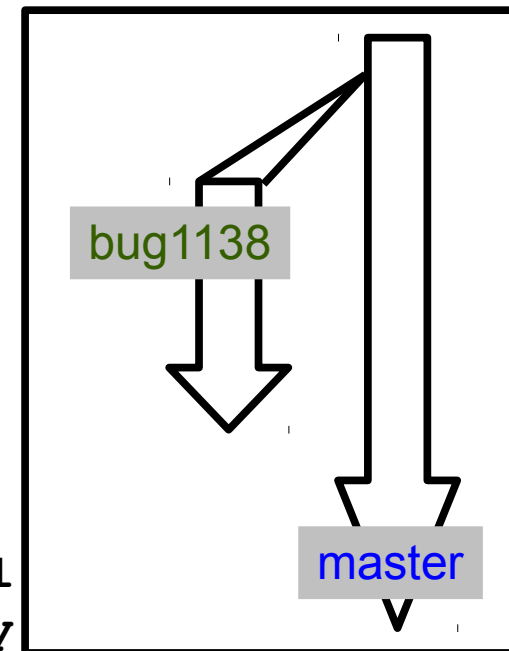


repository at
github.com/USER/REPOS

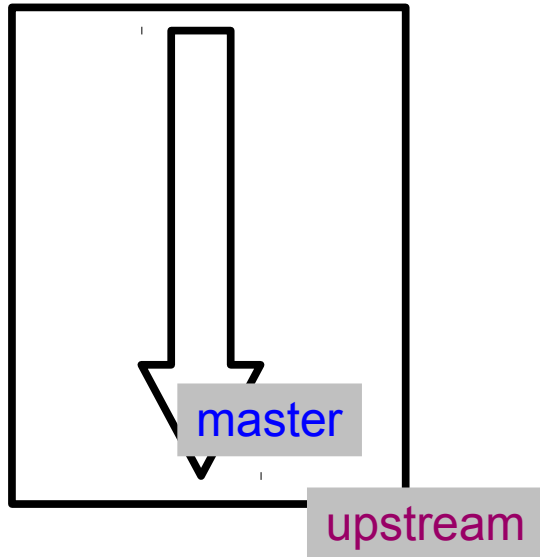


```
git branch
git checkout
# do work
git add
git commit
```

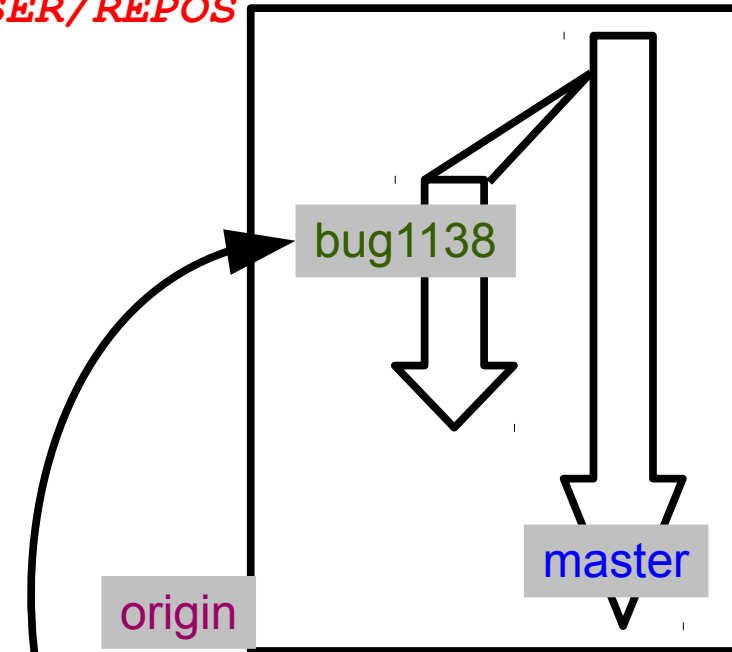
your local
repository



repository at
github.com/GROUP/REPOS

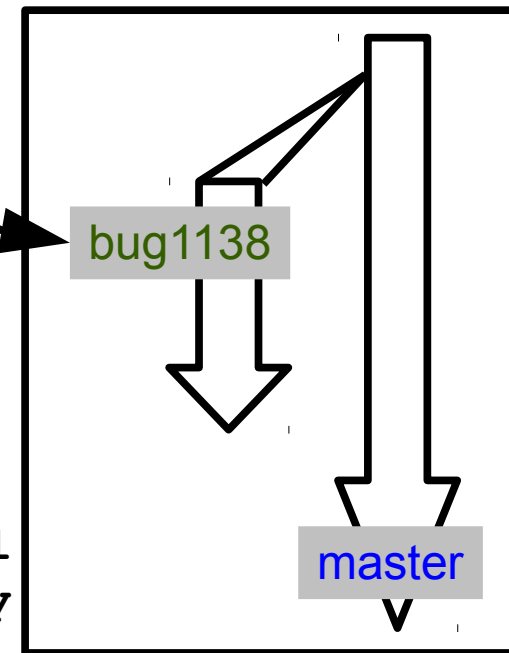


repository at
github.com/USER/REPOS



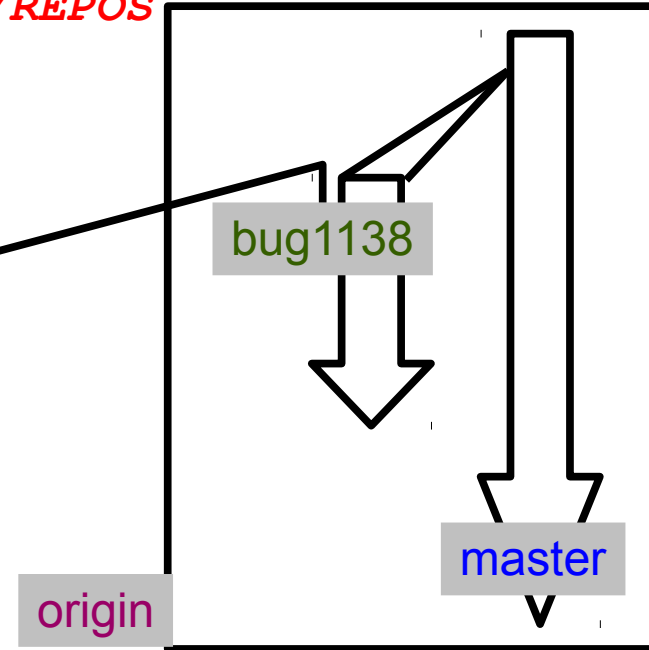
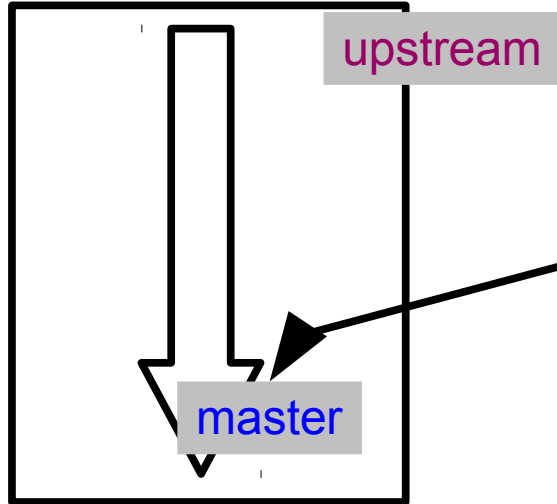
```
git push origin bug1138
```

your local
repository

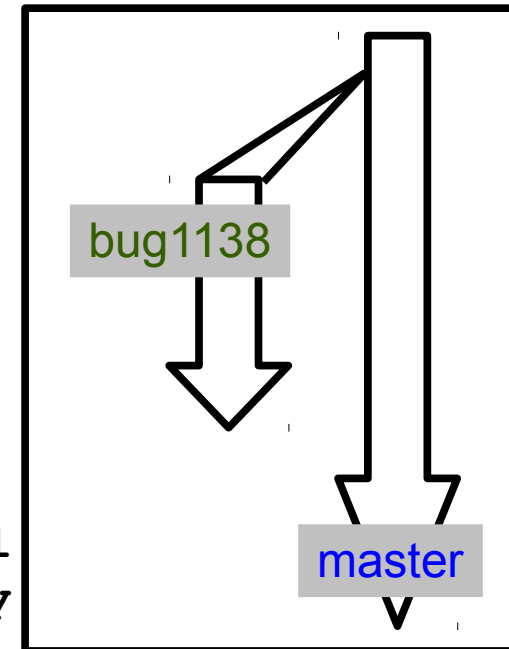


repository at
github.com/USER/REPOS

repository at
github.com/GROUP/REPOS



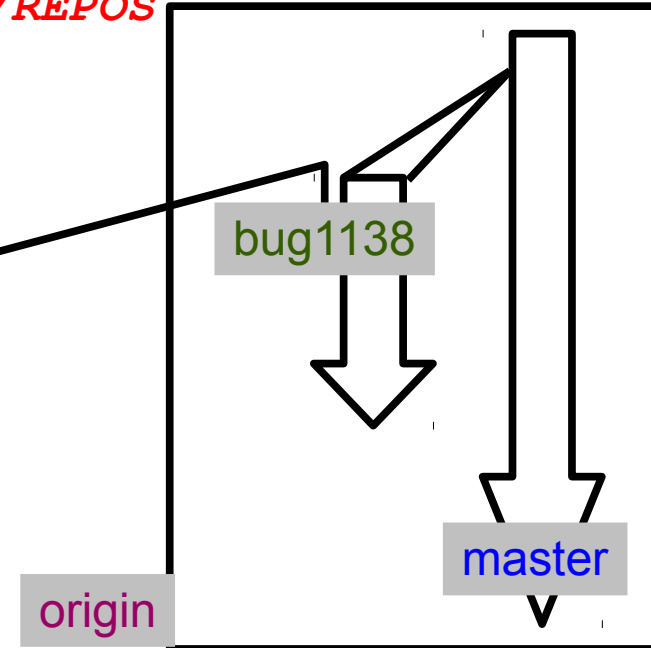
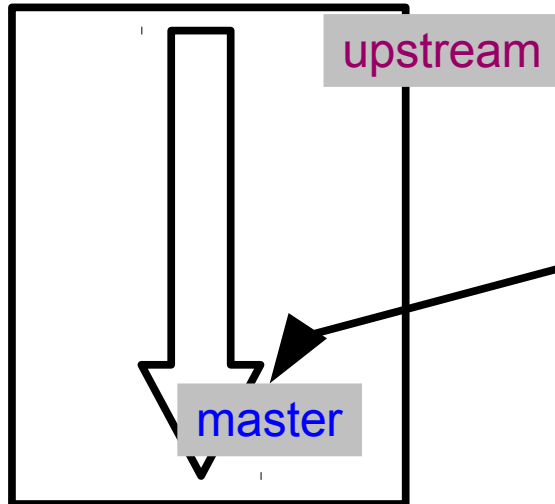
use web browser to merge
(if no conflicts)



your local
repository

repository at
github.com/GROUP/REPOS

repository at
github.com/USER/REPOS

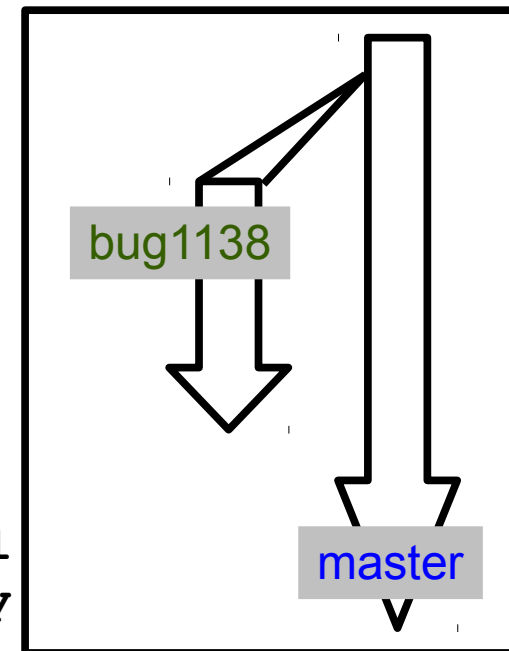


use web browser to merge
(if no conflicts)

OR

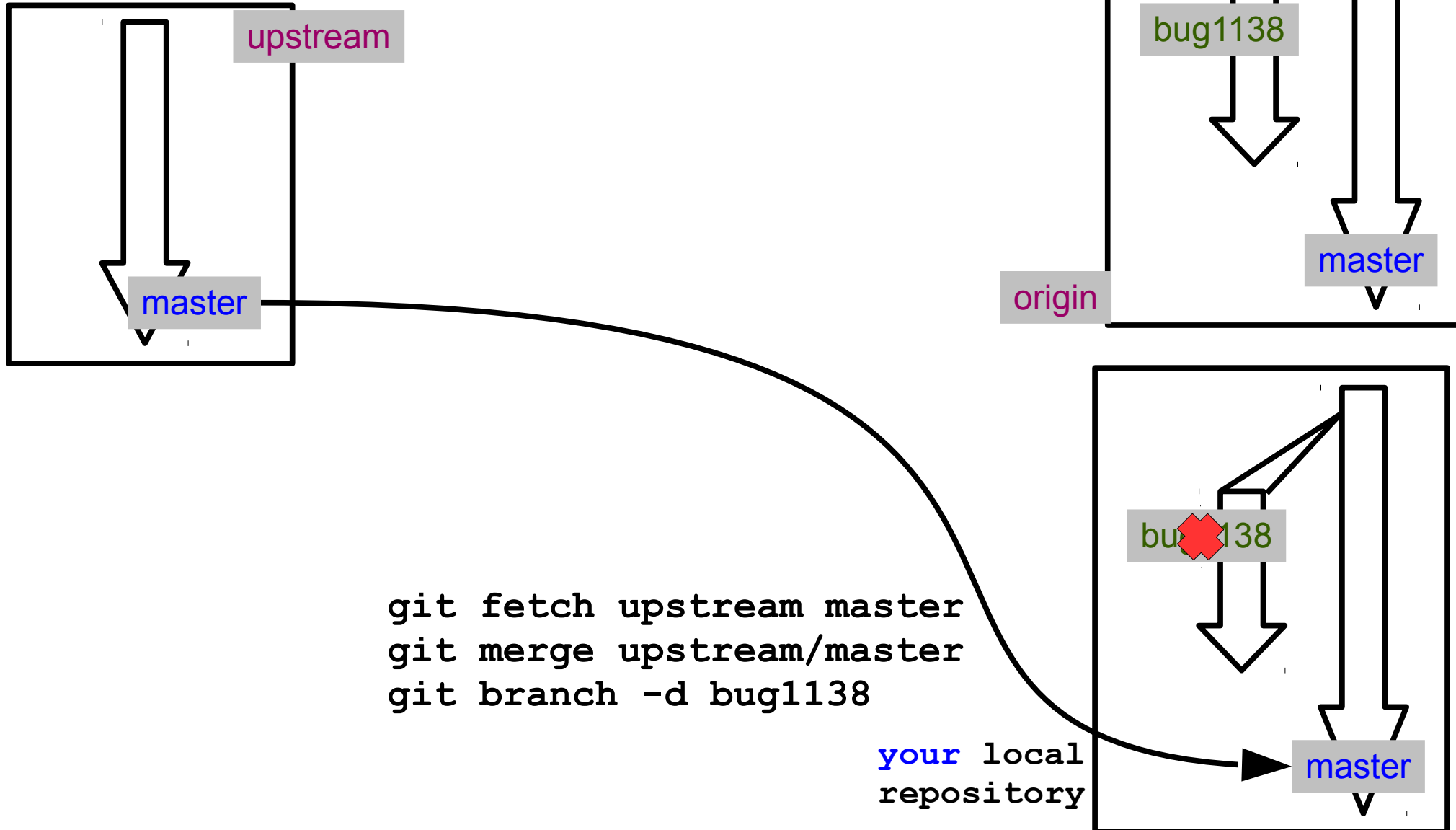
```
git clone # to group owner's local machine
git checkout -b pullreq_bug1138
git pull github.com/USER/REPOS bug1138
# fix merge conflict
git add/git commit
git checkout master
git merge --no-ff pullreq_bug1138
git push origin master
```

your local
repository



repository at
github.com/USER/REPOS

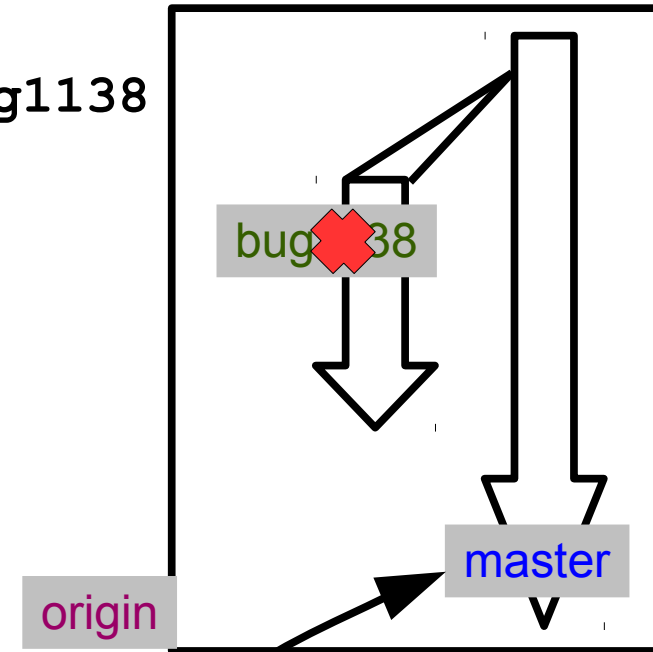
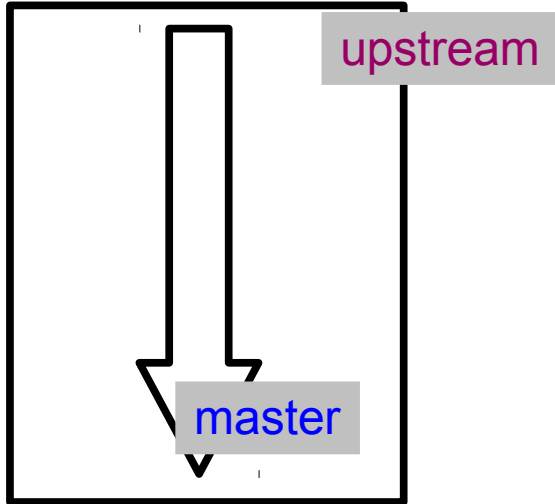
repository at
github.com/GROUP/REPOS



repository at *github.com/USER/REPOS*

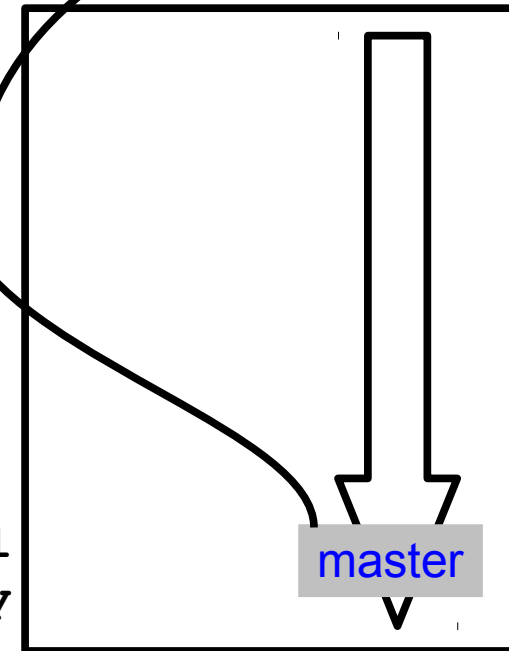
repository at
github.com/GROUP/REPOS

```
git branch -d bug1138
```



```
git push origin master
```

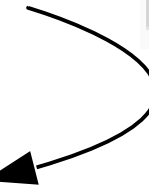
your local
repository



Forks and branches

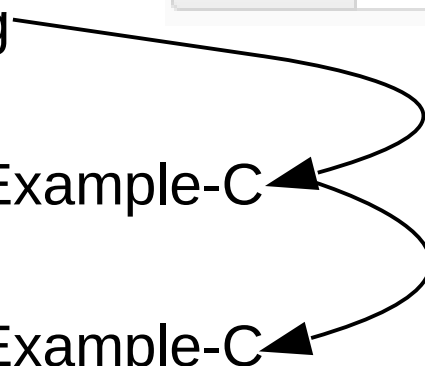
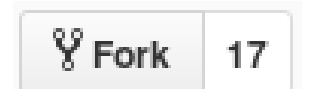
- Correct usage: I used branches

- <https://github.com/chaddcw/pullRequestTesting>
- <https://github.com/cs360f16/pullRequestTesting>



- Horrible mess: I did not use branches :(

- <https://github.com/chaddcw/pullRequestTesting>
- <https://github.com/cs360f14/ContactManager-Example-C>
- <https://github.com/cs360f16/ContactManager-Example-C>





Create newFileForPullRequest.txt #2



Edit

 Open **chaddcw** wants to merge 1 commit into `cs360f14:master` from `chaddcw:master`



 Conversation 0  Commits 1  Files changed 1 +1 -0

 **chaddcw** commented just now Owner 



This file was added to demonstrate what a pull request looks like.

  Create newFileForPullRequest.txt ... e3eae4e

Add more commits by pushing to the **master** branch on **chaddcw/FirstGitPractice**.


 **This pull request can be automatically merged.**  **Merge pull request**


You can also merge branches on the [command line](#).


 **Write** Preview M Markdown  Edit in Fullscreen

Leave a comment


Attach images by dragging & dropping or [selecting them](#).

 **ProTip** Add comments to specific lines under Files changed. Close **Comment**

Milestone 
No milestone


Assignee 
No one assigned

Notifications

 **Unsubscribe**

You're receiving notifications because you authored the thread.

1 participant



http

Providers

- GitHub
 - free (\$\$\$), proprietary (not open)
 - Go get a GitHub account and email me your username
- GitLab
 - open source
 - <https://about.gitlab.com/>
 - <https://gitlab.com/gitlab-org/gitlab-ce/>

CI

- <http://docs.travis-ci.com/user/getting-started/>
- <http://computer-vision-talks.com/articles/2014-02-23-using-travis-ci/>

Practice: DUE: Monday, noon

- Build a new local git repository in
~/Documents/CS360Practice_PUNetID
- Add the file test.c that prints Hello World
- Add the file test.txt that states “test.c prints Hello World”
- Add a .gitignore file that ignores *.o files and the executable named test
- Commit all the files.
- Tag this revision as “INIT_COMMIT”
- Make a branch for a feature add: FA_Name
 - add code to test.c that prints your name
 - update test.txt to document your change
 - commit all the files to the branch

Practice

- Switch back to master and make a branch for a feature add: FA_Git
 - add code to test.c that prints I love git
 - update test.txt to document your change
 - commit all the files to the branch
- Switch back to master
- Merge the FA_Name branch to master
- Tag this commit as ADDED_NAME
- Merge the FA_Git branch to master
 - resolve any merge conflicts!
- Tag this commit as ADDED_GITLOVE
- Don't delete the branches

Submit your practice!

- `git log --decorate=full --name-status > punetid_git_log.txt`
- `cd ..`
- `tar czf CS360_Git_PUNetID.tar.gz CS360Practice_PUNetID`
- `scp CS360_Git_PUNetID.tar.gz YOU@zeus.cs.pacificu.edu:`
- `ssh YOU@zeus.cs.pacificu.edu`
- `submit cs360f16 CS360_Git_PUNetID.tar.gz`

- Leave clear, useful commit messages.
- You can work on
 - your own machine
 - a lab machine
 - ssh to zeus.cs.pacificu.edu
 - note: zeus likely does not have a Documents folder!

commit 2a50c7dffddffb8c59356945bf77f881f78f4145 (HEAD -> refs/heads/master,
tag: refs/tags/ADDED_GITLOVE)

Merge: 68099dd 8bce123

fixed merge conflicts

commit 68099dd6347be782f7af2c40c6fbe6bb64b3d32a (tag: refs/tags/ADDED_NAME)

Merge: f75bfa4 cb0828d

Merge branch 'FA_NAME'

commit 8bce1233f6f9be1673a2ec840dcdd7a518a20633 (refs/heads/FA_Git)

added I love git

M test.c
M test.txt

commit cb0828d1e773c40c81171666517c1402e03488e0 (refs/heads/FA_NAME)

prints my name

M test.c
M test.txt

commit f75bfa4892eb8fc8dc534c71ccbd931a75c6bbeb (tag: refs/tags/INIT_COMMIT)

initial commit

A .gitignore
A test.c
A test.txt

commit 7b78d4ede35747f2b108390c3d5a0c5178fef0b0 (HEAD -> refs/heads/master)
Merge: eccc9ea 007f0f5

ADDED_GITLOVE

commit 007f0f548929096e0db791071e642bcd4d976247 (refs/heads/FA_git)

ADDED_GITLOVE

M test.c
M test.txt

commit eccc9ea2246ab2f8abfb890cb99e0c04a17fd54d (refs/heads/FA_Name)

ADDED_NAME

M test.c
M test.txt

commit 1e63953f391a17b389005de635c53c3263b68a5c

INIT_COMMIT

A .gitignore
A test.c
A test.txt