Table Of Contents

Rejoice, for this is the only slide with bullet points!

Part 1: Basic Concepts
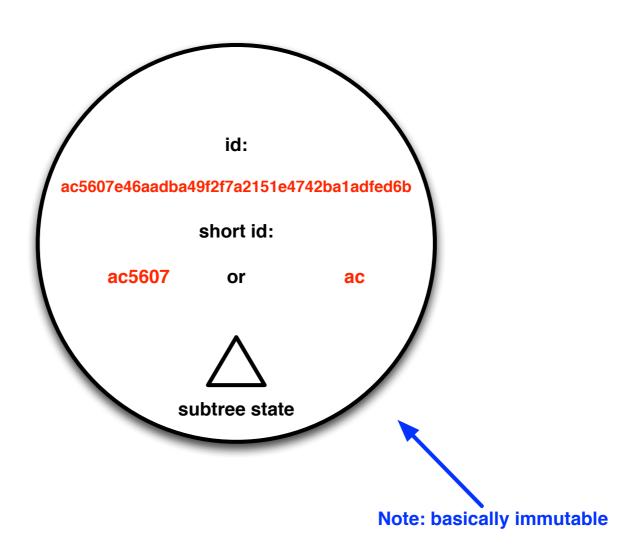
(Where we think on an abstract level and we use only three of the following shapes, unfortunately)
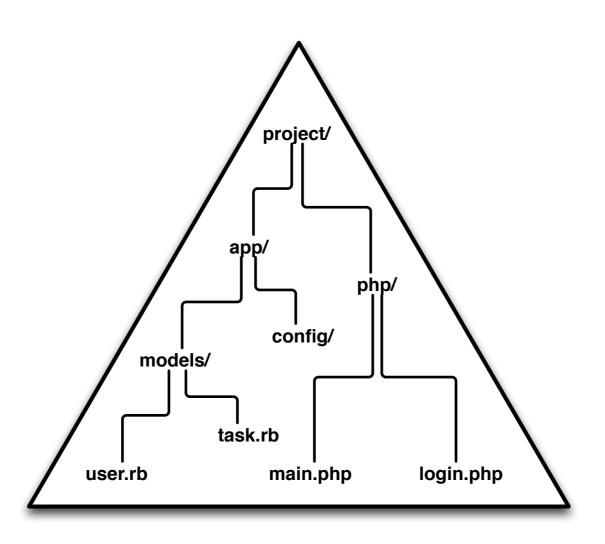
Triangle: a directory tree

Circle: a commit

id:

ac5607e46aadba49f2f7a2151e4742ba1adfed6b

short id:

ac5607   or   ac

△

subtree state

Note: basically immutable

Commit graph

Time
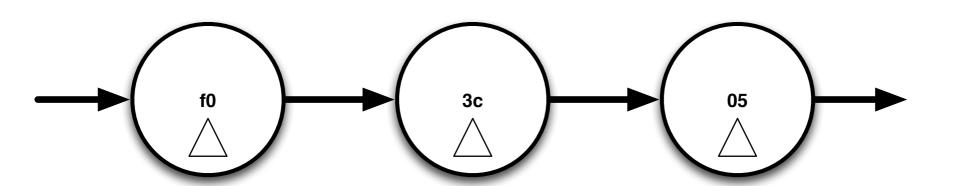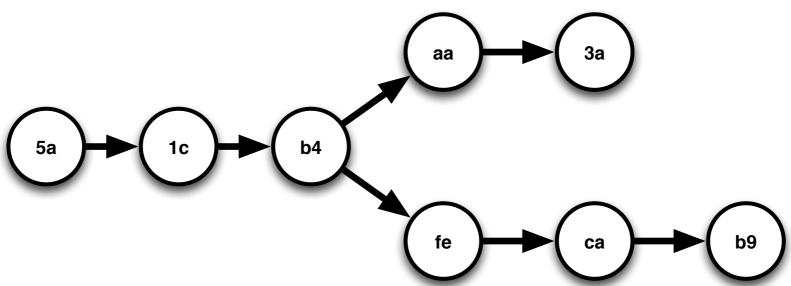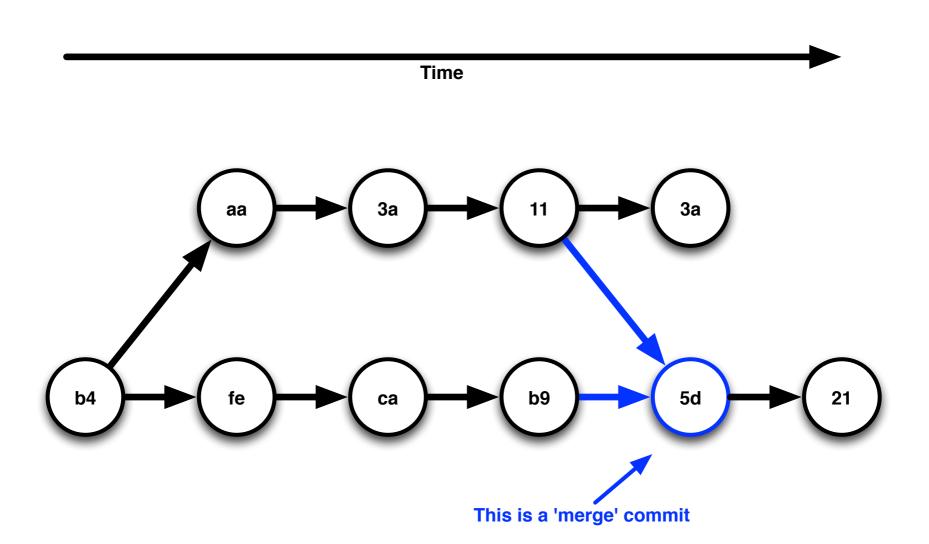
Commit graph branching
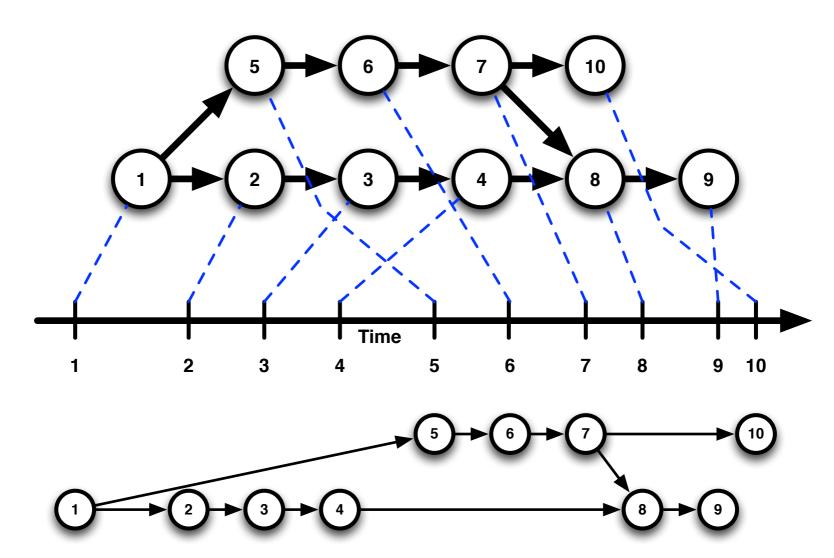
Commit graph merging

Time

This is a 'merge' commit
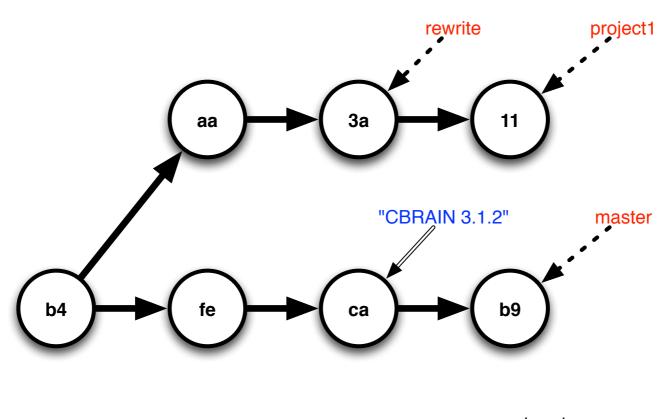
Commit graph ordering
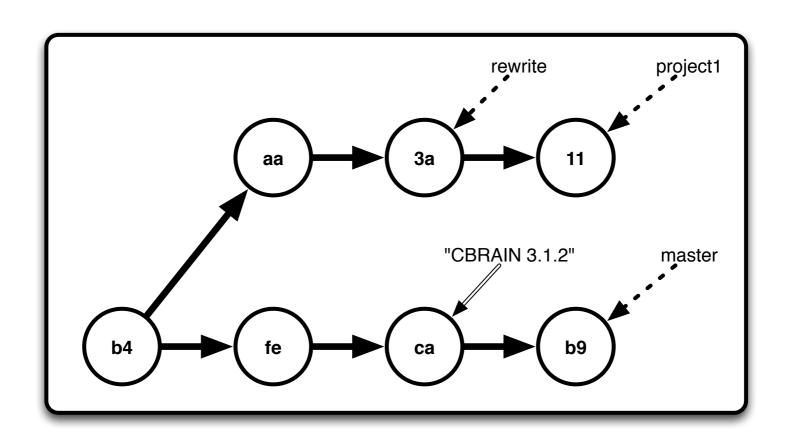
References
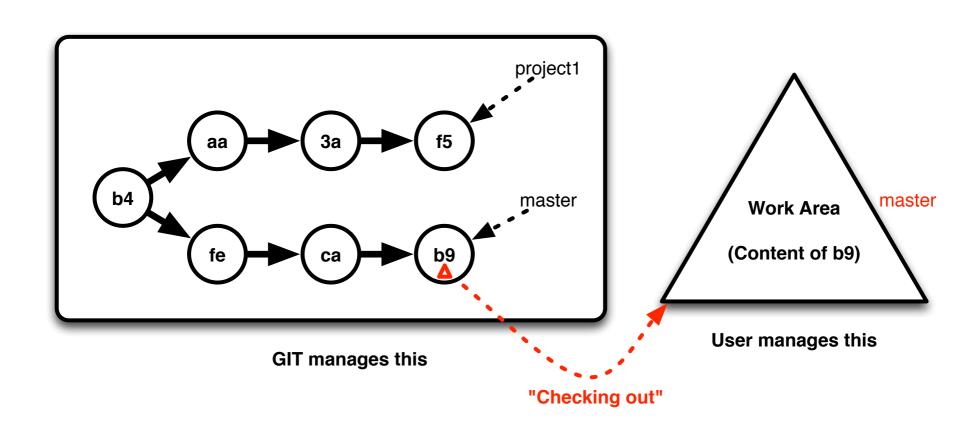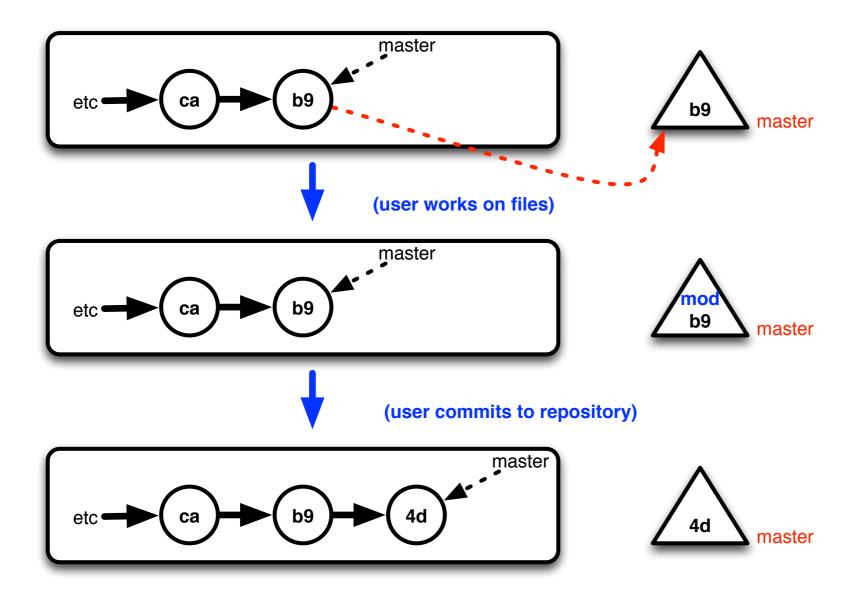
Rounded Rectangle: Bare repository
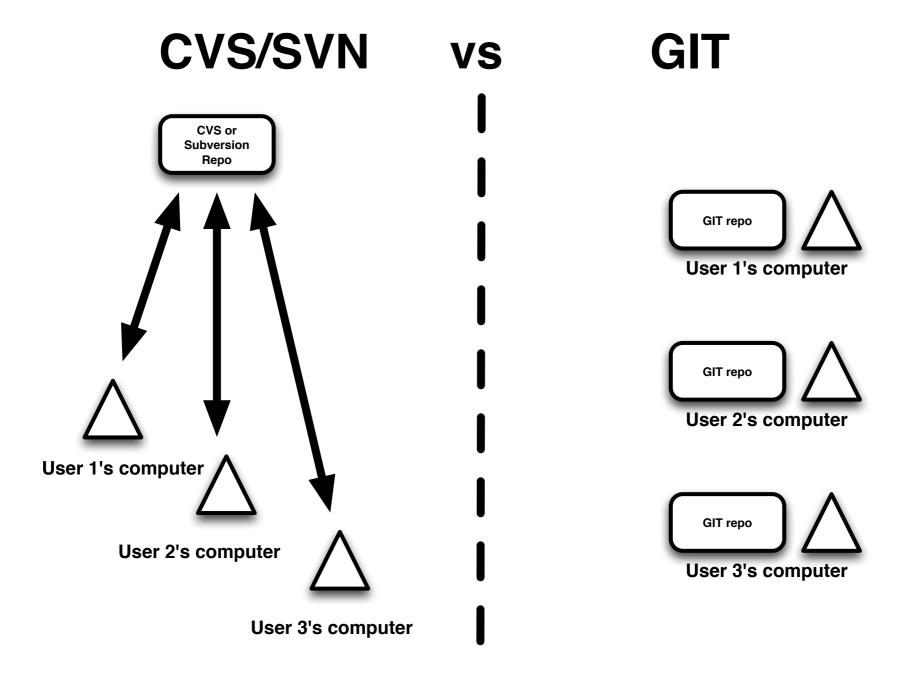
Repository + Work area
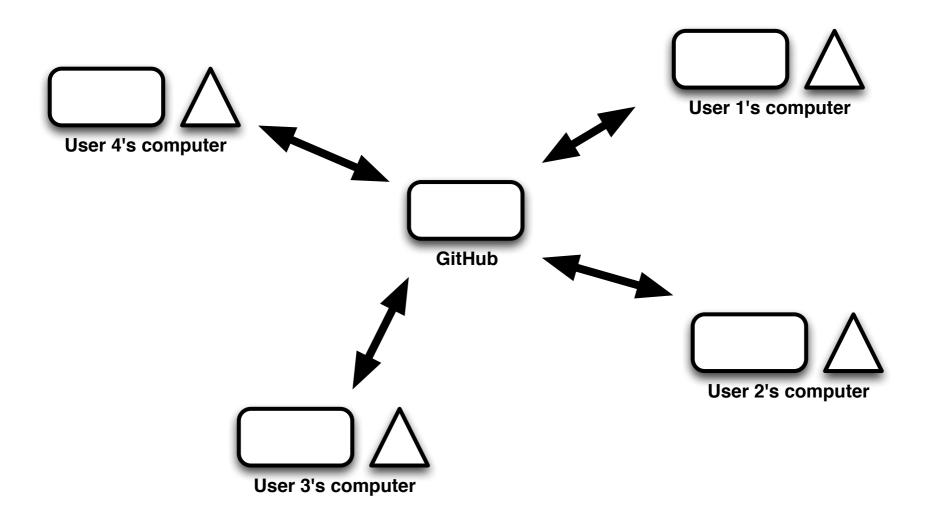
Commit progress (simplified view)

CVS/SVN vs GIT

CVS or Subversion Repo

User 1's computer

User 2's computer

User 3's computer

GIT repo

User 1's computer

GIT repo

User 2's computer

GIT repo

User 3's computer
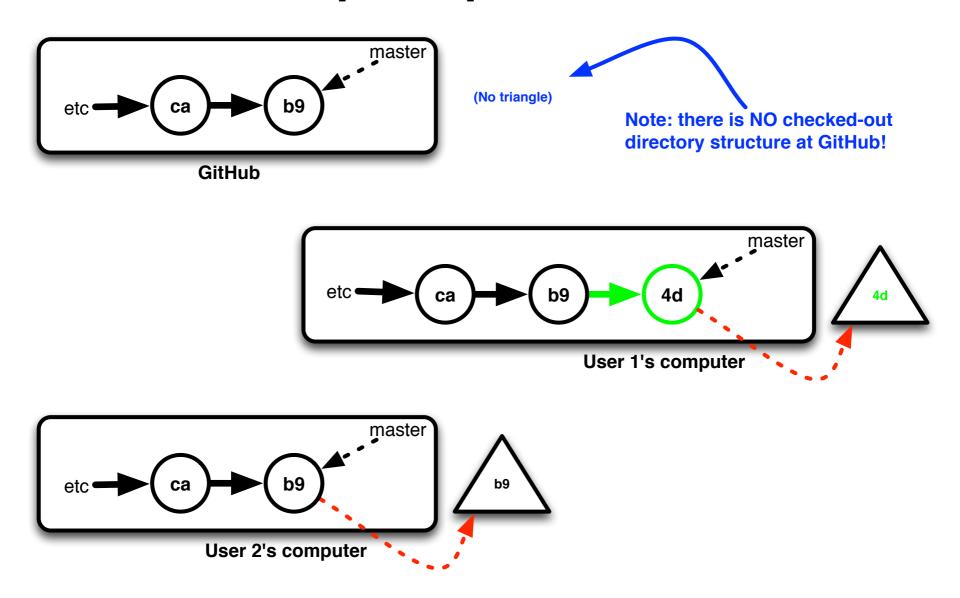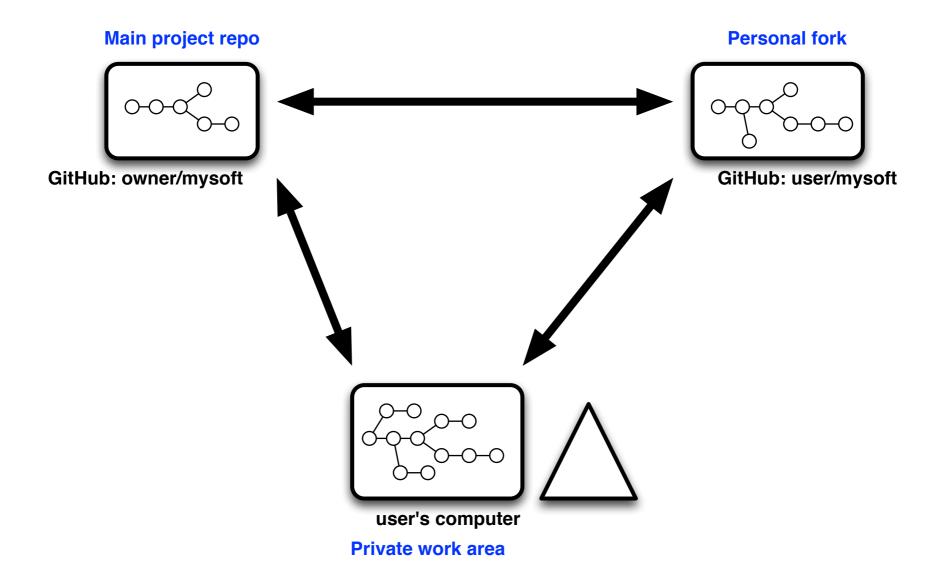
"Main Repo": purely a *convention*

User 4's computer

User 1's computer

GitHub

User 2's computer

User 3's computer

Multiple repositories

GitHub

master

etc → ca → b9

(No triangle)

Note: there is NO checked-out
directory structure at GitHub!

User 1's computer

master

etc → ca → b9 → 4d

4d

User 2's computer

master

etc → ca → b9

b9

GitHub Forks

Main project repo

Personal fork

GitHub: owner/mysoft

GitHub: user/mysoft

user's computer

Private work area

Typical Fork Workflow

Main project repo

Personal fork

(user *makes pull requests* to main repo)

GitHub: owner/mysoft

GitHub: user/mysoft

(user *pulls* from main repo)

(user *pushes* to fork repo)

user's computer

Private work area

Commit races lead to merging
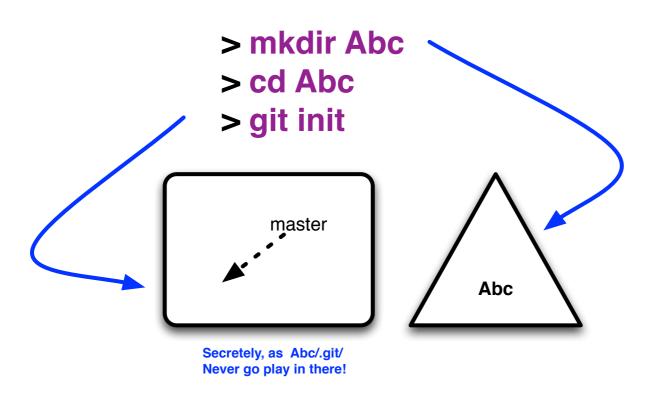
Part 2: Working With GIT

**(Where we run UNIX commands and do not even
introduce a single new shape, except for a dotted one)**

Oh yeah!

Me.
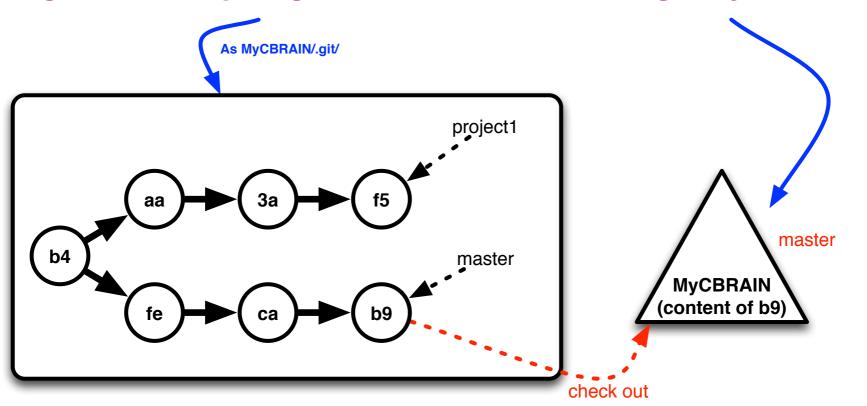
Creating your own blank GIT repository

> **mkdir Abc**
> **cd Abc**
> **git init**

master

Abc

Secretely, as Abc/.git/
Never go play in there!

You can also run git init in a directory that
already has files; they will not be touched.
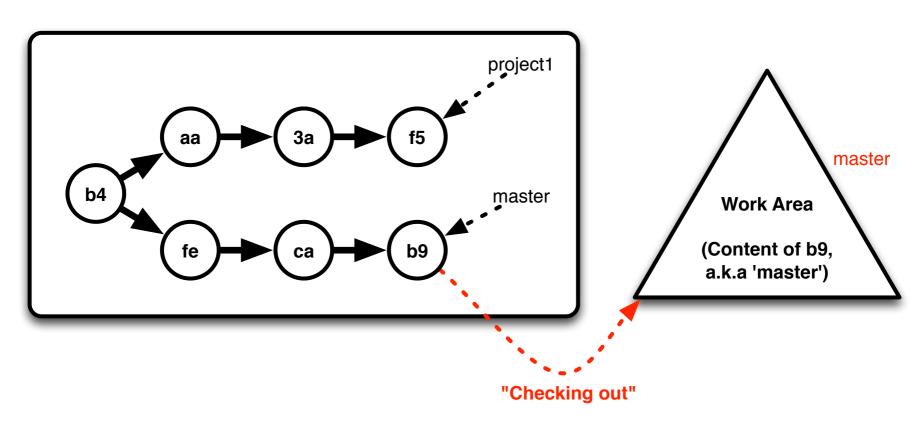
Locally cloning another repository

> **git clone https://github.com/aces/cbrain.git MyCBRAIN**

As MyCBRAIN/.git/

project1

aa → 3a → f5

b4

fe → ca → b9

master

check out

MyCBRAIN
(content of b9)

master
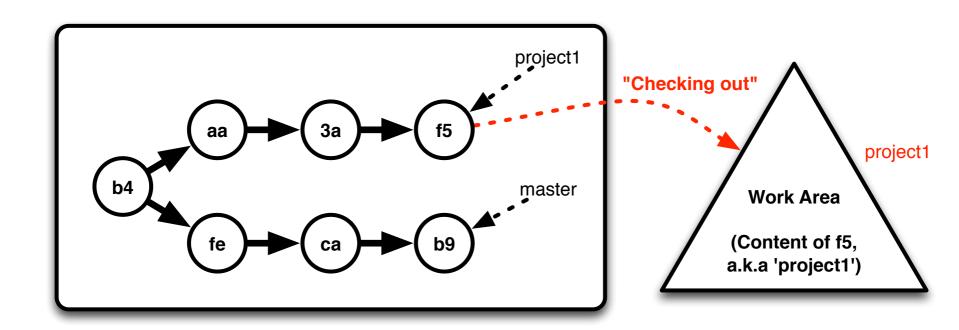
This also sets up the configuration for pulling and pushing to the remote repository on GitHub.

Checking out (1/2)

> **git checkout master**
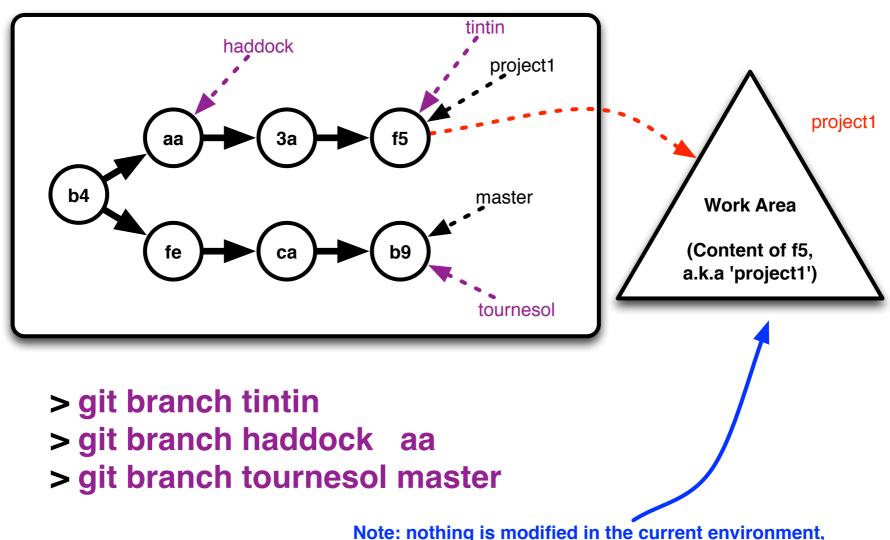
Checking out (2/2)

> **git checkout project1**
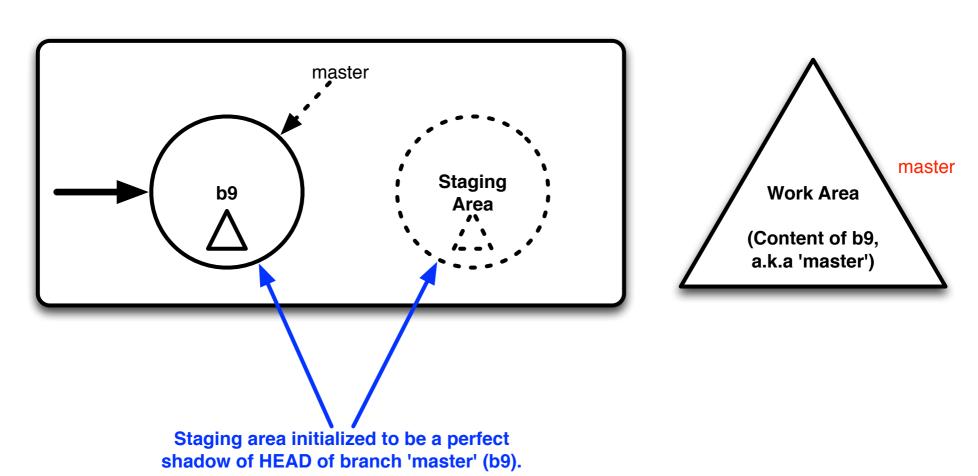
Note: there is **no** such concept as '**checking in**' !

Branching out

> **git branch tintin**
> **git branch haddock   aa**
> **git branch tournesol master**
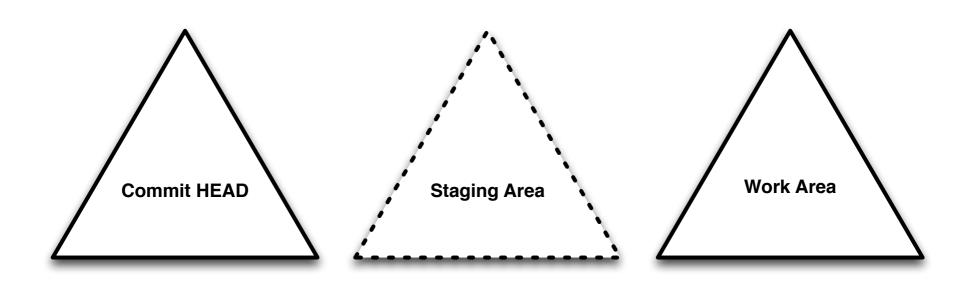
Note: nothing is modified in the current environment, not even the currently checked out branch.

Dotted Circle: Staging area ('index')

master

b9

Staging
Area

Work Area

(Content of b9,
a.k.a 'master')

master

Staging area initialized to be a perfect
shadow of HEAD of branch 'master' (b9).

Three trees

Commit HEAD

Staging Area

Work Area

Immutable

MUST be changed using GIT commands

Can be changed using UNIX commands: editor, cp, rm, mv, etc (with caveats)

GIT operations on individual files

> **git checkout -- file**

> **git rm file**
> **git mv file**

> **git reset -- file**

Staging Area

> **git add file**

(For new and modified files)

Work Area

Commit HEAD

> **git checkout HEAD file**

> **editor**

GIT status

```
Bianca > vi README
Bianca > vi Makefile
Bianca > git add Makefile
Bianca > git rm Gemfile.lock
Bianca > vi NEW_FILE.txt
Bianca > git add NEW_FILE.txt
Bianca > git mv config.ru config.ru.renamed
Bianca > touch howdy.txt
Bianca > git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   deleted:    Gemfile.lock
#   new file:   NEW_FILE.txt
#   modified:   Makefile
#   renamed:    config.ru -> config.ru.renamed
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#   modified:   README
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   howdy.txt
```
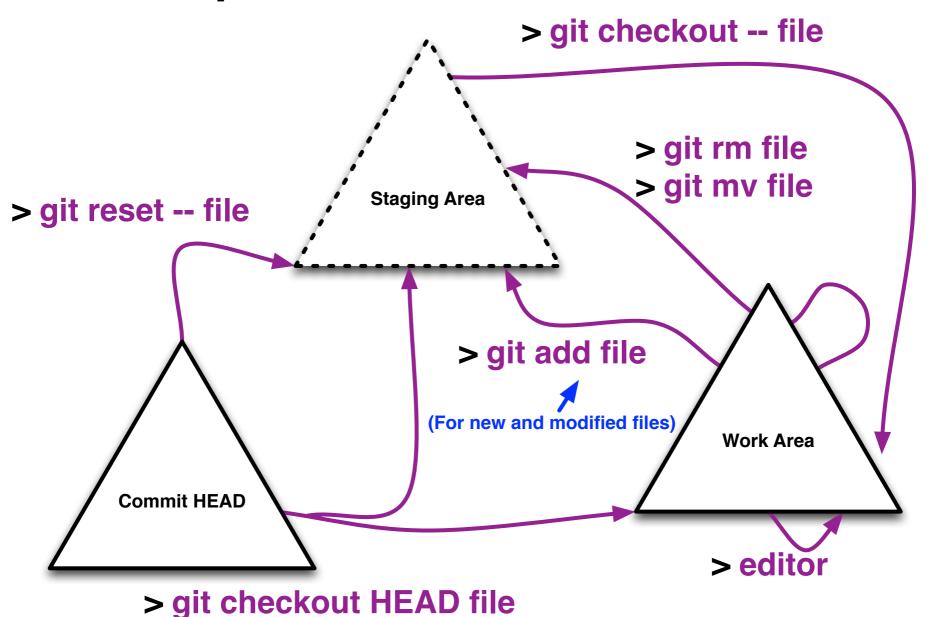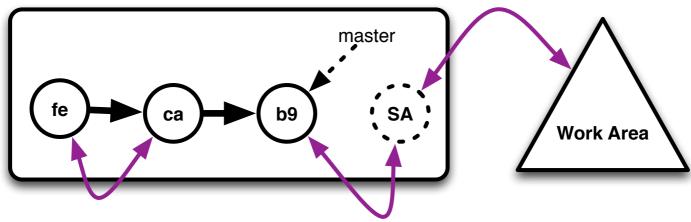
What would go into the next commit.

What is changed in the work area, but WON'T go in the next commit.

New files in work area, not registered to be in GIT repository at all.

GIT status -s

```
Bianca > git status -s
D   Gemfile.lock
A   NEW_FILE.txt
 M  README
M   Rakefile
R   config.ru -> config.ru.renamed
??  howdy.txt
```

Status of the files in the Staging Area

Status of the files in the Work Area

GIT diff

> **git diff [file]**
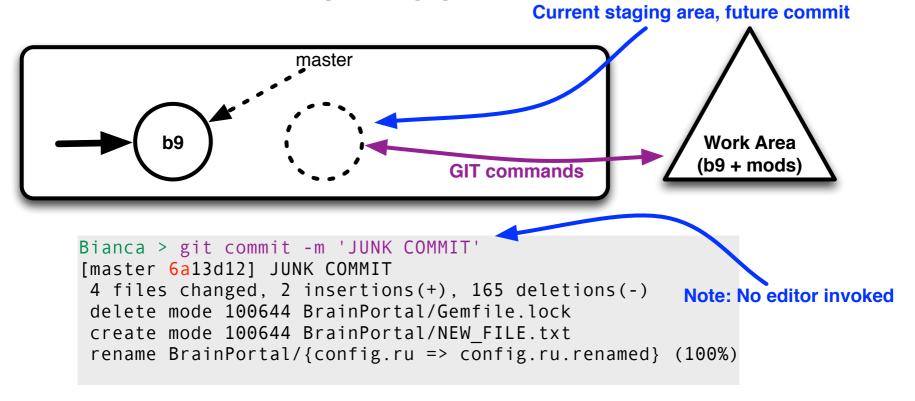
> **git diff fe ca [file]**   > **git diff --cached [file]**
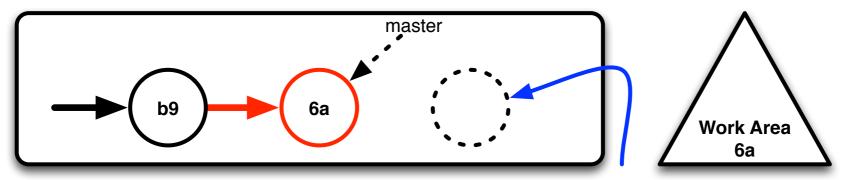
```
Bianca > git diff
diff --git a/BrainPortal/Rakefile b/BrainPortal/Rakefile
index 02a5494..dfd0c36 100644
--- a/BrainPortal/Rakefile
+++ b/BrainPortal/Rakefile
@@ -23,14 +25,11 @@
 # Add your own tasks in files placed in lib/tasks ending in .rake,
 # for example lib/tasks/capistrano.rake, and they will automatically be available to Rake.

-require File.expand_path('../config/application', __FILE__)
-require 'rake'
-

-CbrainRailsPortal::Application.load_tasks
+CbrainRailsPortal::Application.load_tasks.2
```

GIT commit

Current staging area, future commit

master

b9

GIT commands

**Work Area
(b9 + mods)**

```
Bianca > git commit -m 'JUNK COMMIT'
[master 6a13d12] JUNK COMMIT
 4 files changed, 2 insertions(+), 165 deletions(-)
 delete mode 100644 BrainPortal/Gemfile.lock
 create mode 100644 BrainPortal/NEW_FILE.txt
 rename BrainPortal/{config.ru => config.ru.renamed} (100%)
```

Note: No editor invoked

master

b9   **6a**

**Work Area
6a**
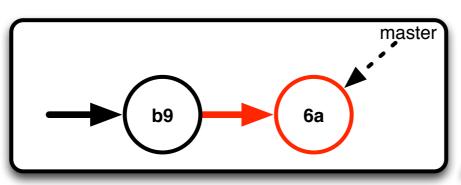
New 'blank' staging area, shadow of 6a

GIT push (simple case)

User's Computer

master

b9 → 6a

Work Area
(6a + mods)

Note: nothing is modified on this side

```
Bianca > git push
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 287 bytes, done.
Total 3 (delta 2), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
   b9d9623..6a13d12  master -> master
```
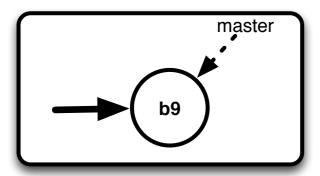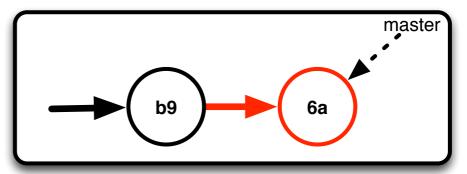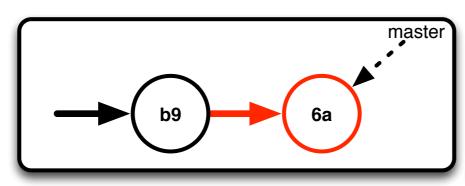
GitHub

master

b9

Before push

master

b9 → 6a
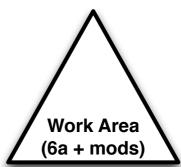
After push

GIT push (merge needed)

User's Computer

master

b9 → 6a

Work Area
(6a + mods)

```
Bianca > git push
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs
To prevent you from losing history, non-fast-forward updates were rejected
Merge the remote changes (e.g. 'git pull') before pushing again.  See the
'Note about fast-forwards' section of 'git push --help' for details.
```

GitHub

master

b9 → d4
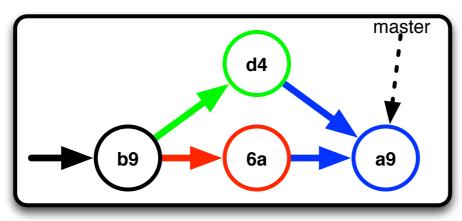
**Before push;
this prevents it**

GIT pull (with merge)


```
Bianca > git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
   b9d9623..a93a987  master     -> origin/master
Merge made by recursive.
         port.rb |    3 +++
         hanged, 3 insertions(+), 0 deletions(-)
```
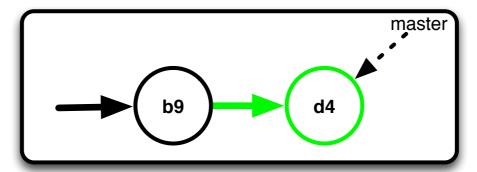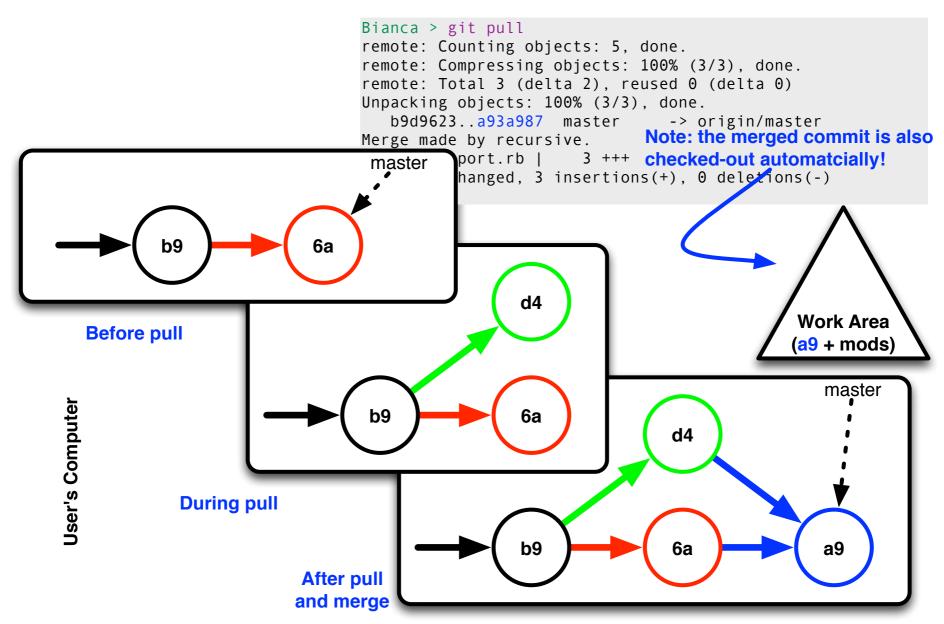
Note: the merged commit is also checked-out automatcially!

User's Computer

Before pull

During pull

After pull and merge

master

b9 → 6a

d4

b9 → 6a

d4

b9 → 6a → a9

master

Work Area (a9 + mods)

GIT push of merged commits

User's Computer

master

d4

master

b9 → 6a → a9

Work Area
(a9 + mods)

```
Bianca > git push
Counting objects: 8, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 539 bytes, done.
Total 5 (delta 3), reused 0 (delta 0)
Unpacking objects: 100% (5/5), done.
   b9d9623..a93a987  master -> master
```
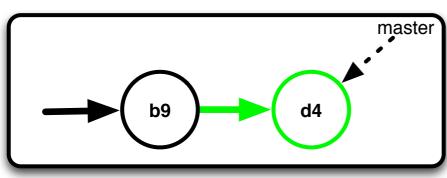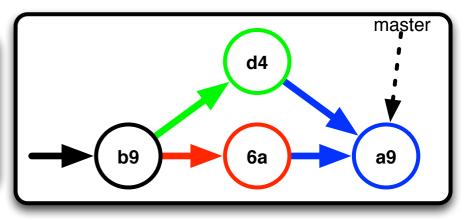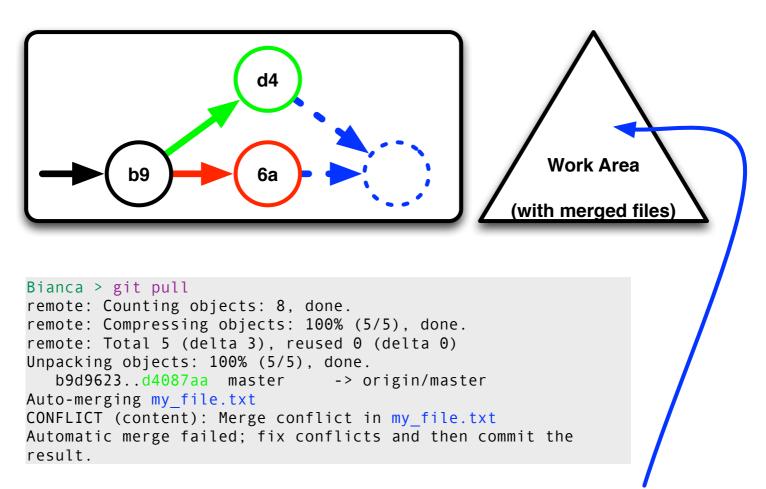
GitHub

master

b9 → d4

master

d4

b9 → 6a → a9

Before push

After push

Merge conflicts


```
Bianca > git pull
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 3), reused 0 (delta 0)
Unpacking objects: 100% (5/5), done.
   b9d9623..d4087aa  master      -> origin/master
Auto-merging my_file.txt
CONFLICT (content): Merge conflict in my_file.txt
Automatic merge failed; fix conflicts and then commit the
result.
```
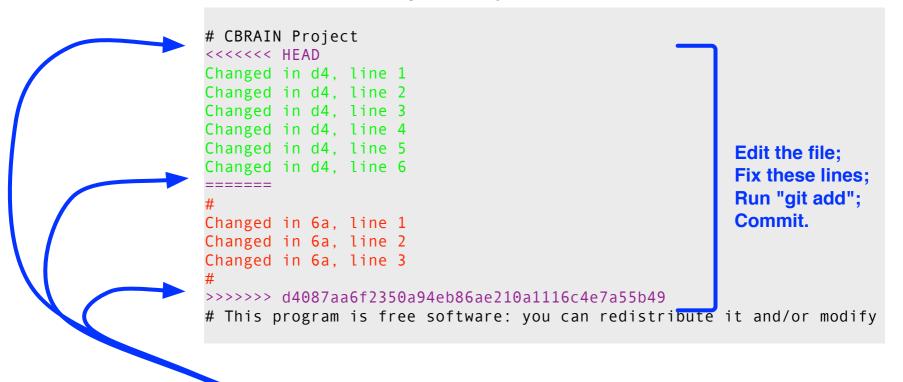
Work Area

(with merged files)

my_file.txt contains conflict tokens

Note: the entire system is now in a special 'merge mode'.

Conflict resolution

Excerpt from my_file.txt

```
# CBRAIN Project
<<<<<<< HEAD
Changed in d4, line 1
Changed in d4, line 2
Changed in d4, line 3
Changed in d4, line 4
Changed in d4, line 5
Changed in d4, line 6
=======
#
Changed in 6a, line 1
Changed in 6a, line 2
Changed in 6a, line 3
#
>>>>>>> d4087aa6f2350a94eb86ae210a1116c4e7a55b49
# This program is free software: you can redistribute it and/or modify
```

Edit the file;
Fix these lines;
Run "git add";
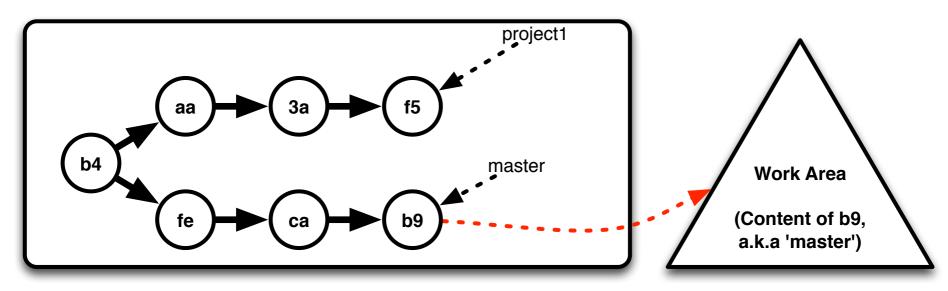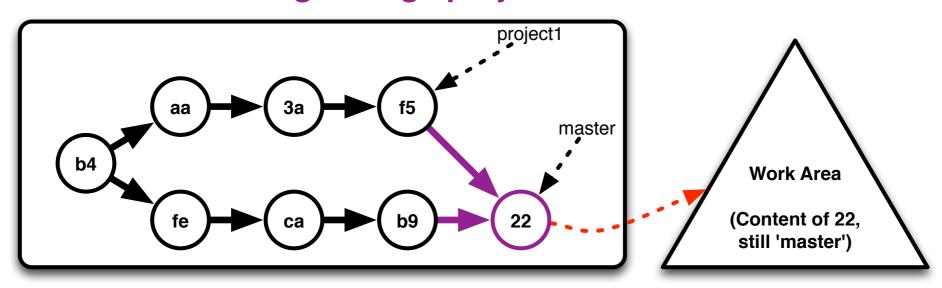Commit.

These three lines were added by GIT

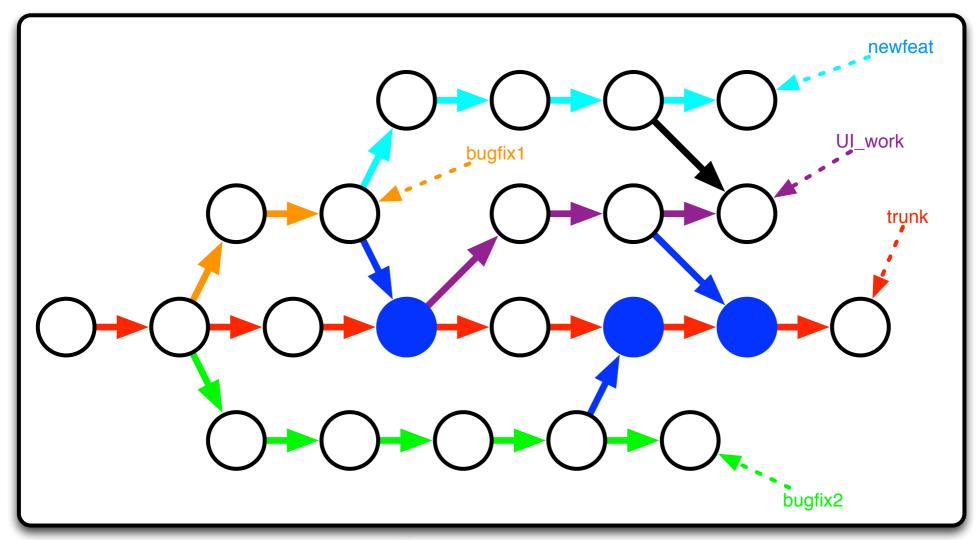**You must resolve all conflicts in all files and then commit before you can proceed to use GIT as usual.**
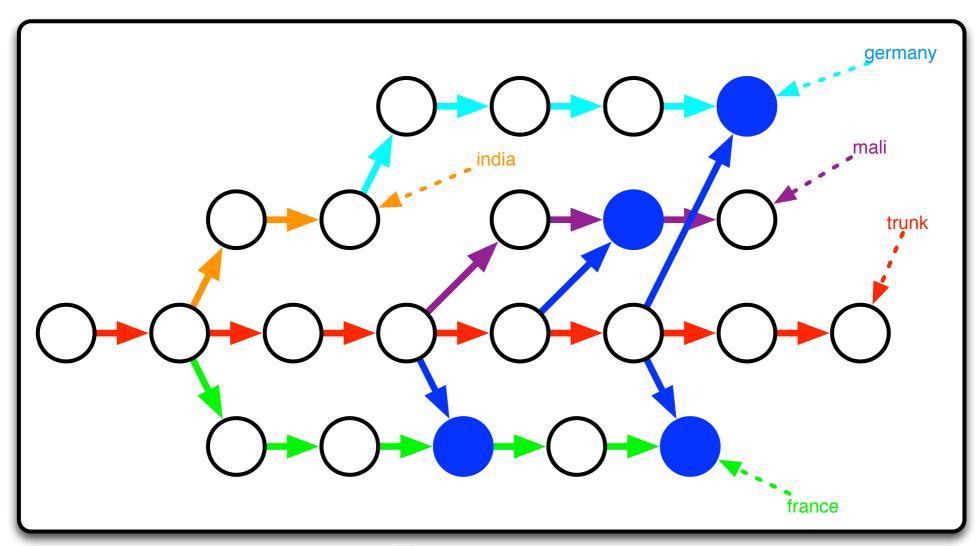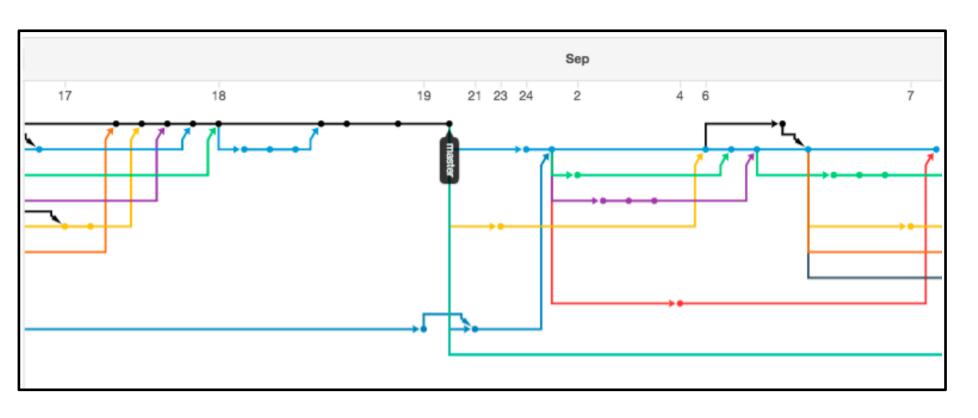
Merging branches
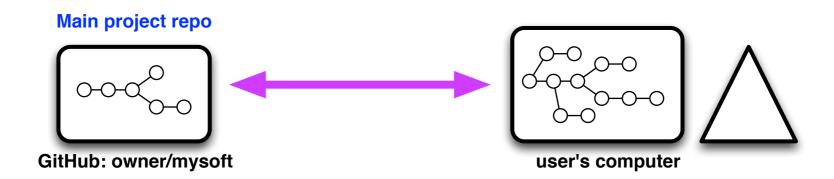

> **git merge project1**

Branching towards a trunk

newfeat

bugfix1

UI_work

trunk

bugfix2

Merge events FROM side branches TO trunk

Branching towards subprojects

Merge events FROM trunk TO side branches

An example: CBRAIN weaving

Typical Work Cycle: No Forks

Main project repo

GitHub: owner/mysoft   **user's computer**
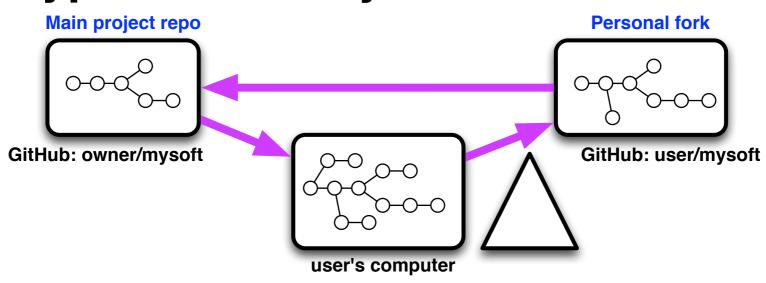
> **git checkout # optional**
> **edit; git add/rm/mv/reset etc**
> **git commit**

Repeat: Make progress in your local repository.

> **git pull**
> **(resolve conflicts; commit)**
> **git push**

Just once: Integrate other people's work.

Repeat.

Typical Work Cycle: With Forks

Main project repo

Personal fork

GitHub: owner/mysoft

user's computer

GitHub: user/mysoft

> **git checkout somebranch**
> **git pull github somebranch**
> **git checkout -b newstuff**

Just once: select a base starting point from other people's work.

> **edit; git add/rm/mv/reset etc**
> **git commit**

Repeat: Make progress in your local repository.

Repeat.

> **git push myfork newstuff**
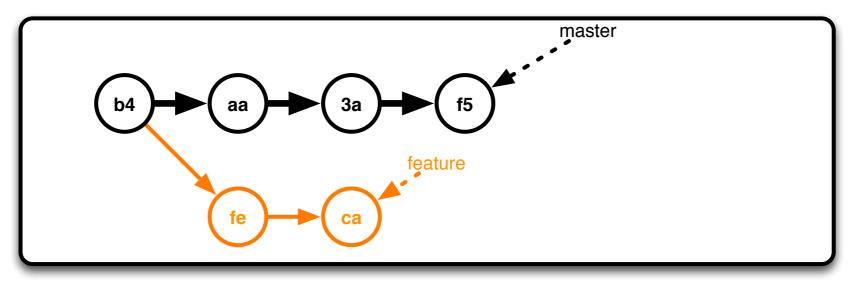> **(use GitHub's interface to make pull request)**

Just once: send modifications to fork

The End

Addendum: Rebasing

> **git checkout feature**
> **git rebase f5**