

# Git & Github

## pt1

**Andrea Fornaia, Ph.D.**

Department of Mathematics and Computer Science

University of Catania

Viale A.Doria, 6 - 95125 Catania Italy

fornaia@dmi.unict.it

<http://www.cs.unict.it/~fornaia/>

# Track changes in your Code



GITHUB

---

DROPBOX

---

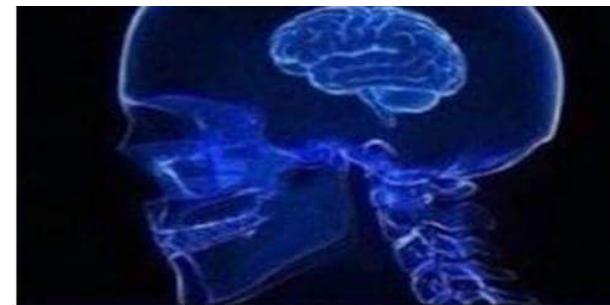
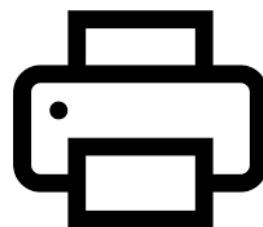
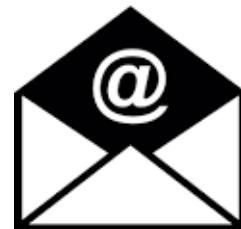
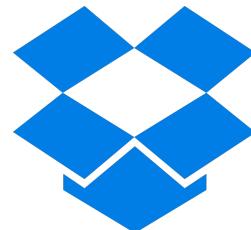


v1.0.0



v1.0.1

# Sharing your Code

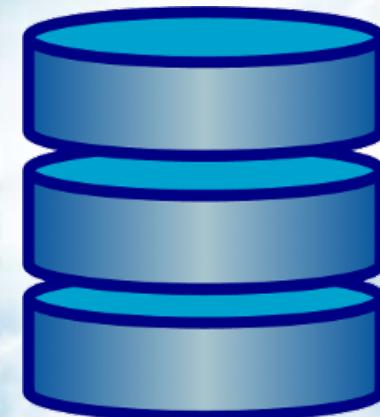


# Version Control System

- Tracks file changes over time (**history**)
  - **revert** files (or entire project) to a previous state
  - **compare** changes over time
  - **document** change history
  - see **who** last modified a file and **when**
- Allows developers to **share the code** and **work simultaneously** on it
  - branch oriented
  - avoid overwriting each other's changes

# Also Known As

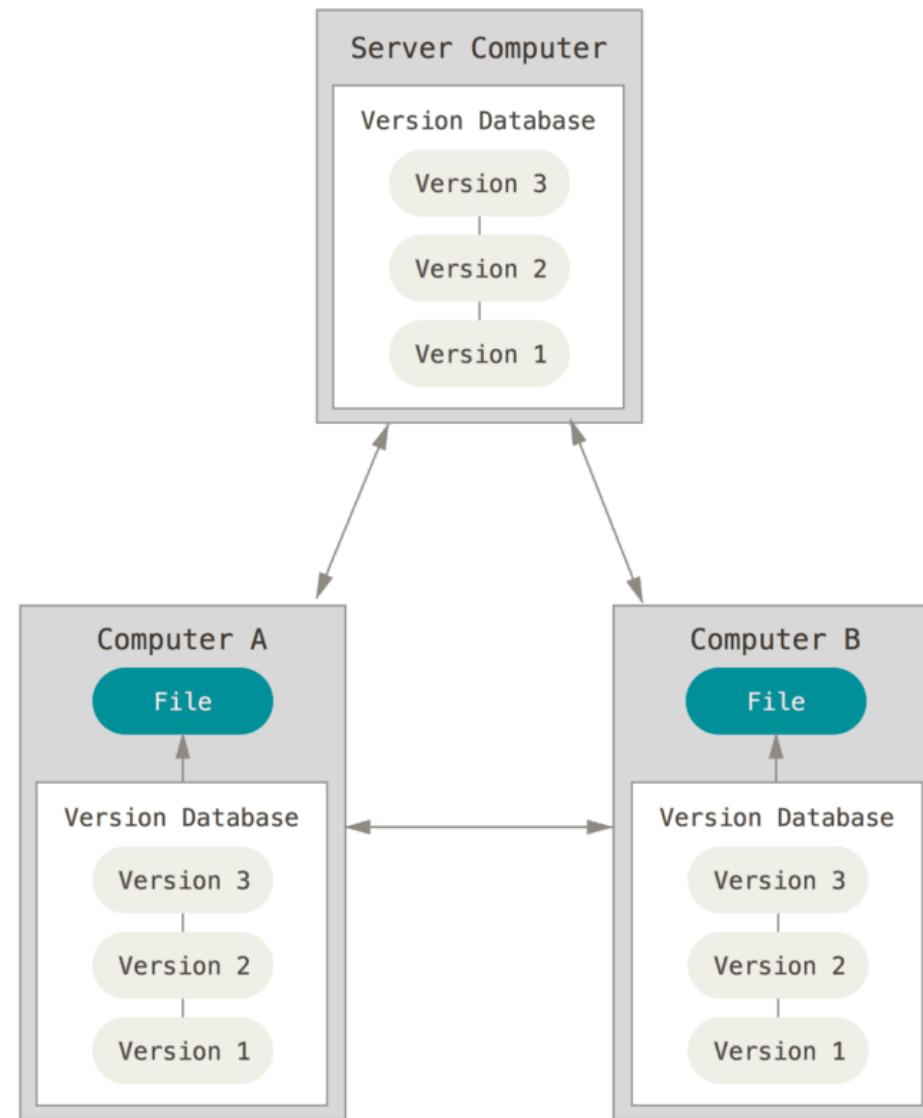
- VCS: **Version Control System**
- SCM: **Source Code Manager**
- **Code Base**
- Or (if you are old as me) **Repository!**





- Its origins are tied with the **Linux Kernel** maintenance:
  - Changes stored as patches and archived files (1991)
  - Start using a proprietary DVCS called BitKeeper (2002)
  - **Linus Torvalds** and Linux dev community create their own DVCS: Git

# Git is a Distributed VCS



# Git Goals

- Strong support for non-linear development
  - supports thousands of parallel **branches**
  - **branching made easy!**
- Fully distributed
- We can use git just locally or sharing one (or more) remote
- Storing **entire history locally** most operations can be done offline
  - no latency
  - free to try new solutions and make experiments
- Efficient for large projects (e.g. Linux kernel)
  - speed
  - data size

# Contents

- Basics
- Brancing & Marging
- Conflicts
- Branching Workflow
- Go Remote
- GitHub

# Basics

# Install Git

```
$ git --version  
git version 2.9.2
```

- Mac
  - via Xcode.
  - Installer:
    - <https://sourceforge.net/projects/git-osx-installer/files/>
  - Homebrew:
    - \$ sudo brew install git
  - MacPort:
    - \$ sudo port install git
- Windows
  - Installer:
    - <https://git-for-windows.github.io>
- Linux
  - Package manager:
    - \$ sudo apt-get install git

# First time Git setup

```
$ git config --global user.name "Andrea Fornaia"  
$ git config --global user.email fornai@example.com  
$ git config --global core.editor vim  
  
$ git config --list  
user.name=Andrea Fornaia  
user.email=fornai@example.com  
core.editor=vim  
  
...  
  
$ git config user.name  
Andrea Fornaia
```



# Mac & Windows

docs/themes/sourcetree/templates/base.html

```
1 1 Hunk 1 : Lines 1-7
2 2
3 3
4 4 + <title>SourceTree Help</title>
5 5 <link rel="stylesheet" href="{{ SITEURL
6 6 </head>
7 7 <body>
```

14 15 Hunk 2 : Lines 15-24
15 16 <footer>
16 17 <a href="http://blog.sourcetreeapp.com/">Blog</a> | <a href="http://w
17 17 - Copyright © 2014
18 18 - <a href="http://www.atlassian.com/"><img href="{{ SITEURL }}/{{ THEME
18 18 + <div id="footer-right">
19 19 + Copyright © 2014

Graph	Commit	Author	Description	Date	
b7358c7	Rahul Chhab...	master	origin/master	origin/HEAD Removing ol...	Mar 3, 2016, 11:...
bdb8bef	Rahul Chhab...	Merged in update-google-verification (pull request #14)		Feb 18, 2016, 1:3...	
dfe975d	Tyler Tadej...	origin/update-google-verification	Update google verificati...	Feb 11, 2016, 2:2...	
3bc3290	Tyler Tadej...	Replace outdated Atlassia...	logo in footer with base-64 en...	Feb 11, 2016, 2:1...	
dba47f9	Tyler Tadej...	Add gitignore		Feb 11, 2016, 1:3...	
ff67b45	Mike Minns...	Updated Mac min-spec to 10.10		Feb 15, 2016, 11:...	
72d32a8	Michael Min...	Merged in hero_images (pull r...	#13)	Feb 15, 2016, 10:...	
246c4ff	Joel Unger...	origin/hero_images	hero_images Used Tinypng to c...	Feb 11, 2016, 3:3...	
9d9438c	Joel Unger...	Replacing hero images with ne...	w version of SourceTree	Feb 9, 2016, 2:59...	
ce75b63	Michael Min...	Merged in bug/date-https (pu...	ll request #12)	Feb 15, 2016, 10:...	
85367bb	Patrick Tho...	origin/bug/date-https	fixed date and https errors	Jan 7, 2016, 12:2...	
4f9b557	Joel Unger...	New Favicon		Feb 8, 2016, 3:55...	
384e6d5	Rahul Chhab...	origin/search-console-access	search console google ver...	Feb 3, 2016, 2:09...	
6fa47a9	Mike Minns...	updated to move supported ve...	n to OSX 10.9+	Dec 15, 2015, 2:0...	
8dd87bb	Mike Minns...	remove extra , when a line i...	s skipped due to empty ser...	Nov 23, 2015, 2:2...	
faa195e	Mike Minns...	Skip records with empty ser...	ver rejects them	Nov 23, 2015, 2:1...	
0cdfe96	Mike Minns...	corrected paths after merge		Nov 23, 2015, 2:0...	
051ab1b	Mike Minns...	corrected column counting		Nov 23, 2015, 1:5...	
a723bc2	Mike Minns...	Merge branch 'au2gex'		Nov 23, 2015, 1:5...	
65fd580	Mike Minns...	deal with invalid instanceid...	s	Nov 23, 2015, 1:5...	
500a892	Michael Min...	Merged in au2gex (pull reque...	s #11)	Nov 23, 2015, 1:0...	



Eclipse Marketplace

Select solutions to install. Press Finish to proceed with installation.  
Press the information button to see a detailed overview and a link to more information.

Search Recent Popular Installed Ma

Find: git All Markets All Categories Go

**gitignore file association** by Nodeclipse organization, MIT  
gitignore file association text editor  
Installs: 10.1K (75 last month) Install

**EGit - Git Team Provider** by Eclipse.org, EPL  
EGit is an Eclipse Team provider for the Git version control system. Git is distributed SCM, which means every developer has a full copy of all history of every... more info  
Installs: 456K (1,513 last month) Update Uninstall

**Gittflow Nightly** by null, EPL  
Nightly build of the Gittflow integration for Eclipse EGit. more info  
Installs: 1.30K (165 last month) Install

Marketplaces

Associations, Gitflow Nightly, Gittflow

Git Repositories

org.eclipse.core.commands - C:\git\org.eclipse.core.commands\.git [master]

- Branches
  - Local Branches
    - master
  - Remote Branches
    - origin
      - master
      - origin- Tags
- Symbolic References
- Remotes
  - origin
    - git://dev.eclipse.org/org.eclipse.core/org.eclipse.core.commands.git
- Working directory - C:\git\org.eclipse.core.commands

New Go Into

Show In ⌘W ▶

Copy ⌘C ▶

Copy Qualified Name ▶

Paste ⌘V ▶

Delete ▶

Remove from Context ⌘D ▶

Build Path ▶

Refactor ⌘T ▶

Import... ▶

Export... ▶

Build Project F5

Refresh ▶

Close Project ▶

Close Unrelated Projects ▶

Validate ▶

Profile As ▶

Debug As ▶

Run As ▶

Team ▶

Compare With ▶

Replace With ▶

Restore from Local History... ▶

Maven ▶

Configure ▶

Source ▶

Properties ⌘I ▶

Commit... ⌘3 ▶

Stashes ▶

Push to Upstream ▶

Fetch from Upstream ▶

Push Branch 'master'... ▶

Pull ▶

Remote ▶

Switch To ▶

Advanced ▶

Synchronize Workspace ▶

Merge Tool ▶

Merge... ▶

Rebase... ▶

Reset... ▶

Create Patch... ▶

Apply Patch... ▶

Add to Index + ▶

Remove from Index ▶

Ignore ▶

Show in History ▶

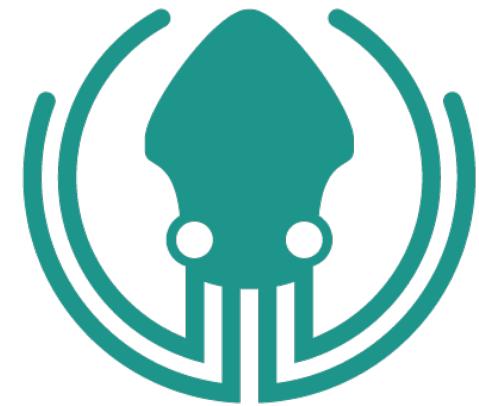
Show in Repositories View ▶

Upgrade Projects... ▶

Disconnect ▶

[<http://www.eclipse.org/egit/>]

# Other solutions



axosoft  
**GitKraken**

Tig

# Tig

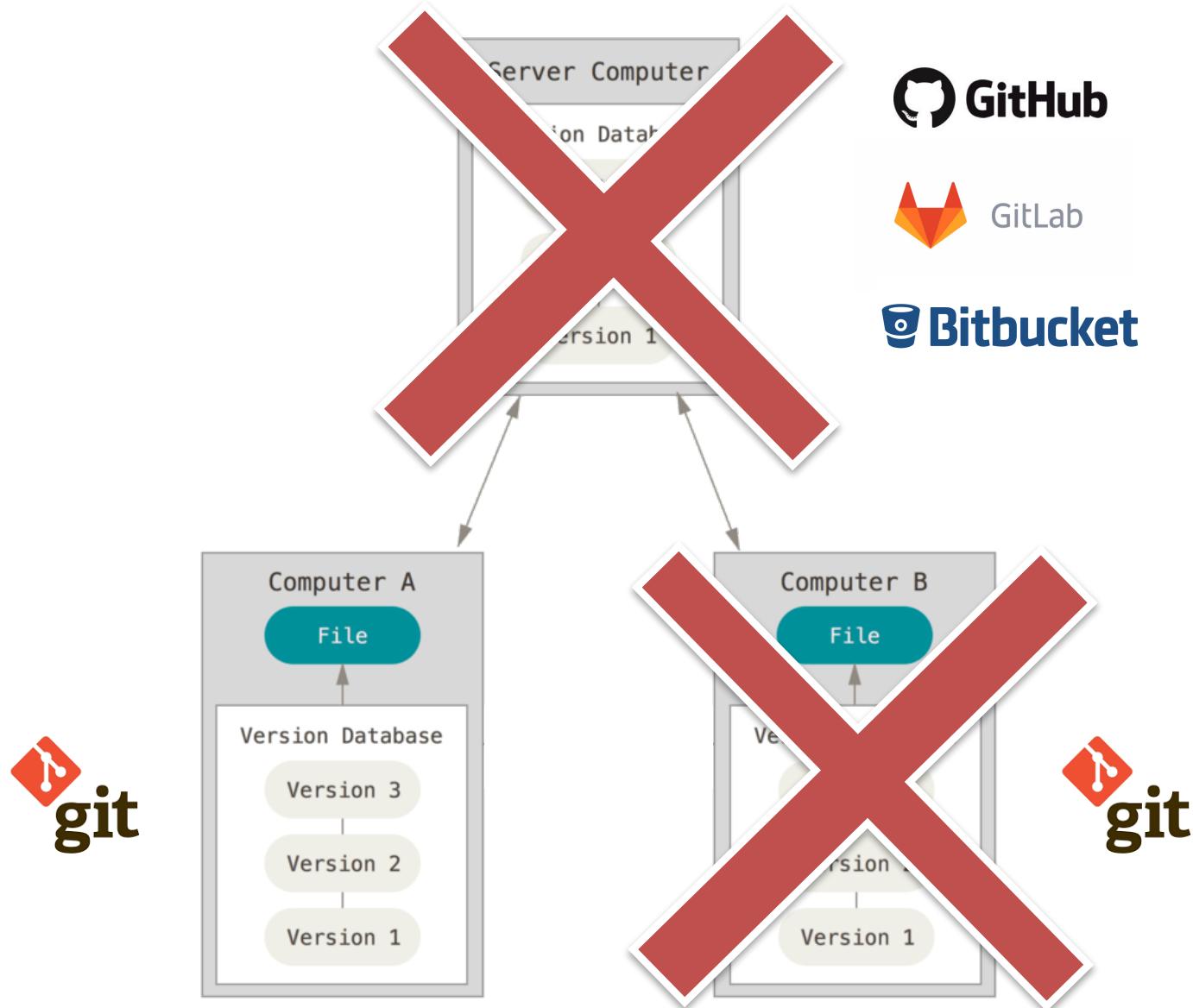
```
1. tig
2012-08-10 16:59 Robert Curth      o Refactoring of todo_view
2012-08-10 16:29 Robert Curth      o Added code-coverage
2012-08-10 16:11 Robert Curth      o Cleanup
2012-08-09 20:24 Robert Curth      o [v0.1.1] Version bump to 0.1.1
2012-08-09 11:21 Robert Curth      M- Merge pull request #1 from shostakovich-
2012-08-09 20:18 Robert Curth      | o [tag_support] [origin/tag_support] Ads ~
2012-08-08 20:05 Robert Curth      | o Adds parameter parsing for tag support
[main] a9963940f5998426e0e97eb41f33d56c1919f557 - commit 26 of 89 (25%)
commit a9963940f5998426e0e97eb41f33d56c1919f557
Author:    Robert Curth <robert@rocu.de>
AuthorDate: Mon Aug 6 19:35:09 2012 +0200
Commit:   Robert Curth <robert@rocu.de>
CommitDate: Mon Aug 6 19:35:09 2012 +0200

    Minor refactoring in todo_store
---
 lib/todo_store.rb |  23 ++++++-----+
 1 file changed, 13 insertions(+), 10 deletions(-)

diff --git a/lib/todo_store.rb b/lib/todo_store.rb
index 56682fd..5740c13 100644
[diff] a9963940f5998426e0e97eb41f33d56c1919f557 - line 1 of 49 (26%)
```

[<https://rocu.de/tig-a-nice-git-repository-browser>]

# Typical Git Scenario



# Initialising a **New Local** Repository

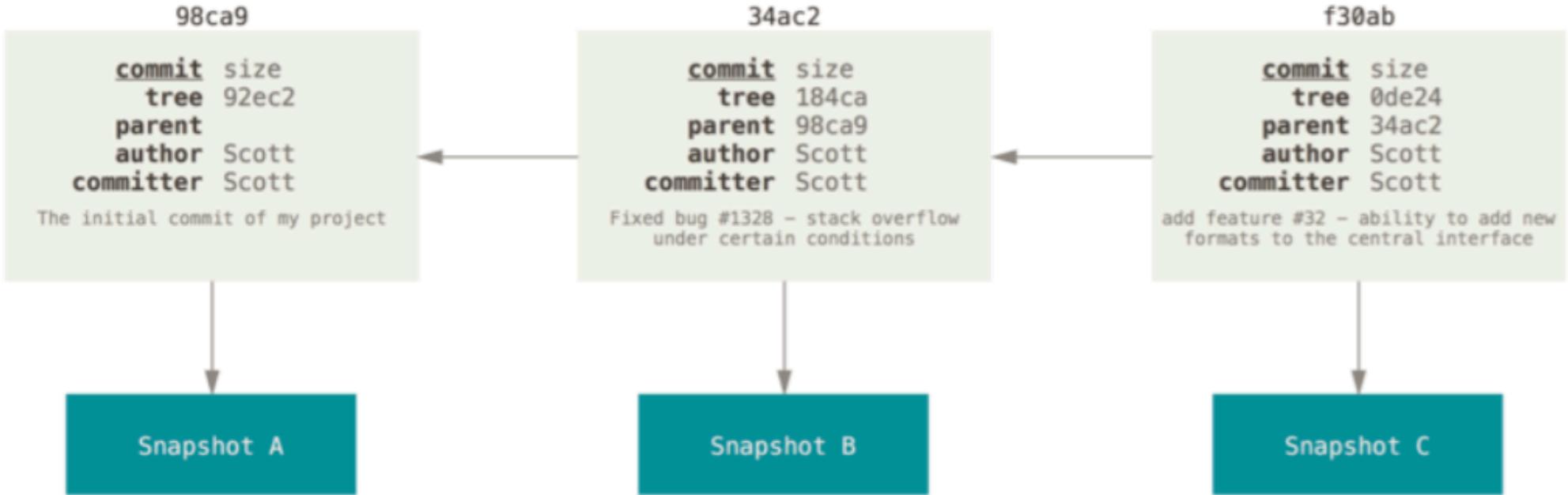
- Setup a repository to track an existing project (some \*.txt files): move to project dir and type:

```
$ git init
```

- This creates a new subdirectory named **.git**
- Contains all of your necessary repository files (Git repository skeleton)
- **Nothing in your project is tracked yet!**

```
$ git add *.txt  
$ git add README  
$ git commit -m 'first commit'
```

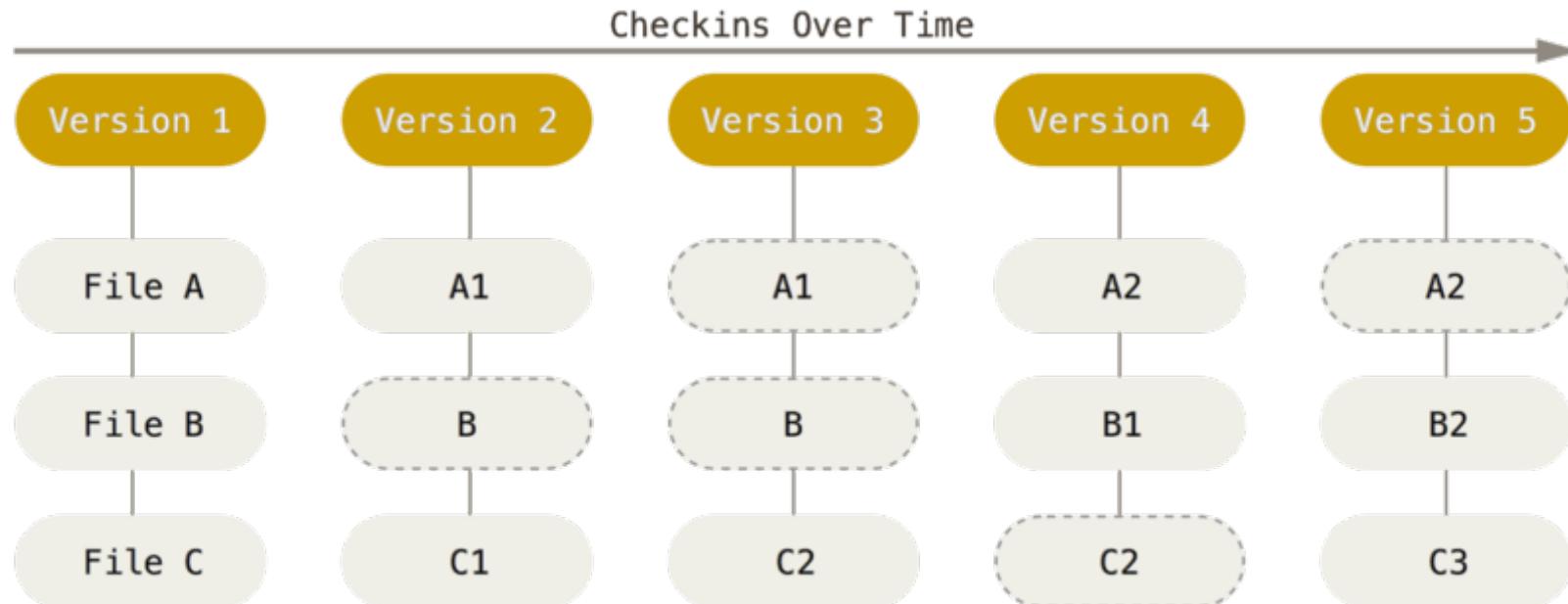
# Commit



- **Snapshot (tree):** a structure of nested files and directories (complete state of the project)
- **Author:** name, email address, and date/time (or “timestamp”) of who made the commit
- **Committer:** who added this commit to the repository (may differ from author)
- **Message:** text used to comment on the changes made by this commit
- **List of parent commits (0, 1 or 2):** immediately preceding states of the project content

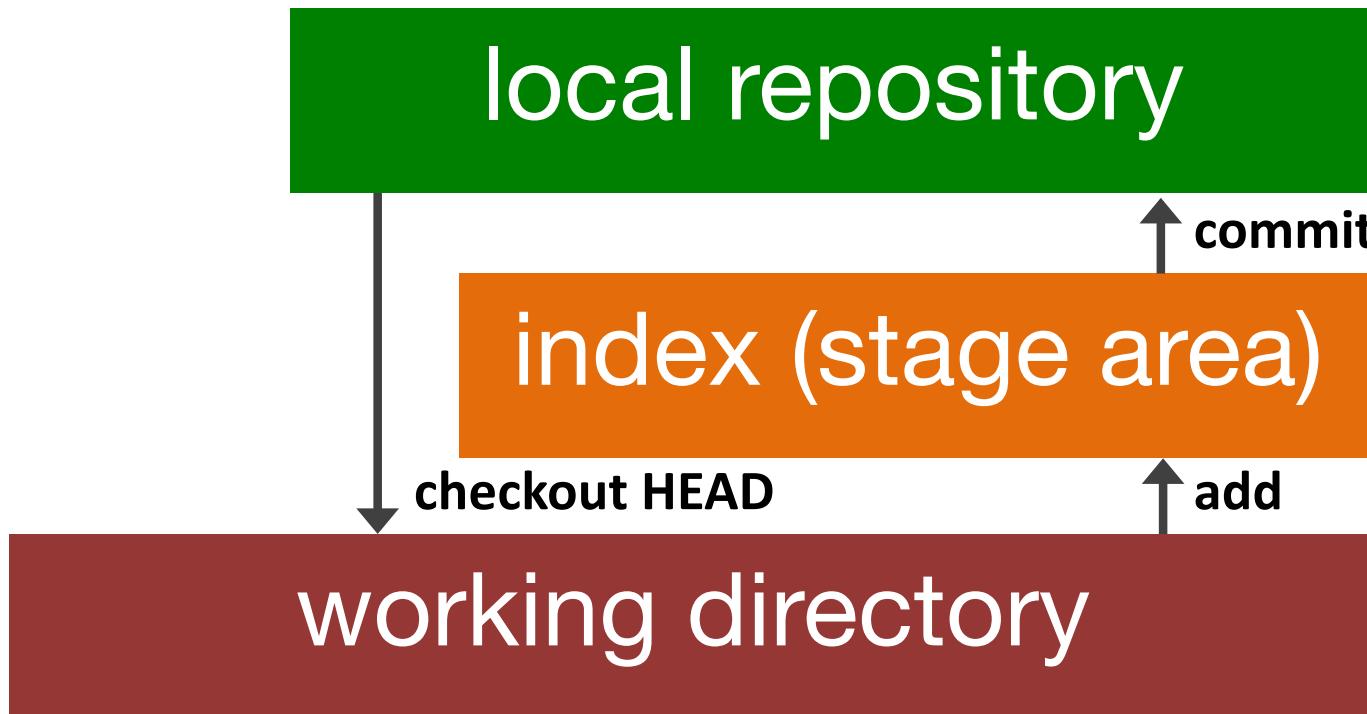
# Snapshots, Not Differences

- Git thinks of its data more like a set of **snapshots** of a **miniature filesystem** than a sequence of diffs
- At every commit, Git takes a **picture** of what all files look like at that moment (**snapshot**)
- To be efficient, if files have **not changed**, Git doesn't store the file again, just a **link** to the **previous identical file** it has already stored.
- Makes branching operations fast and easy.



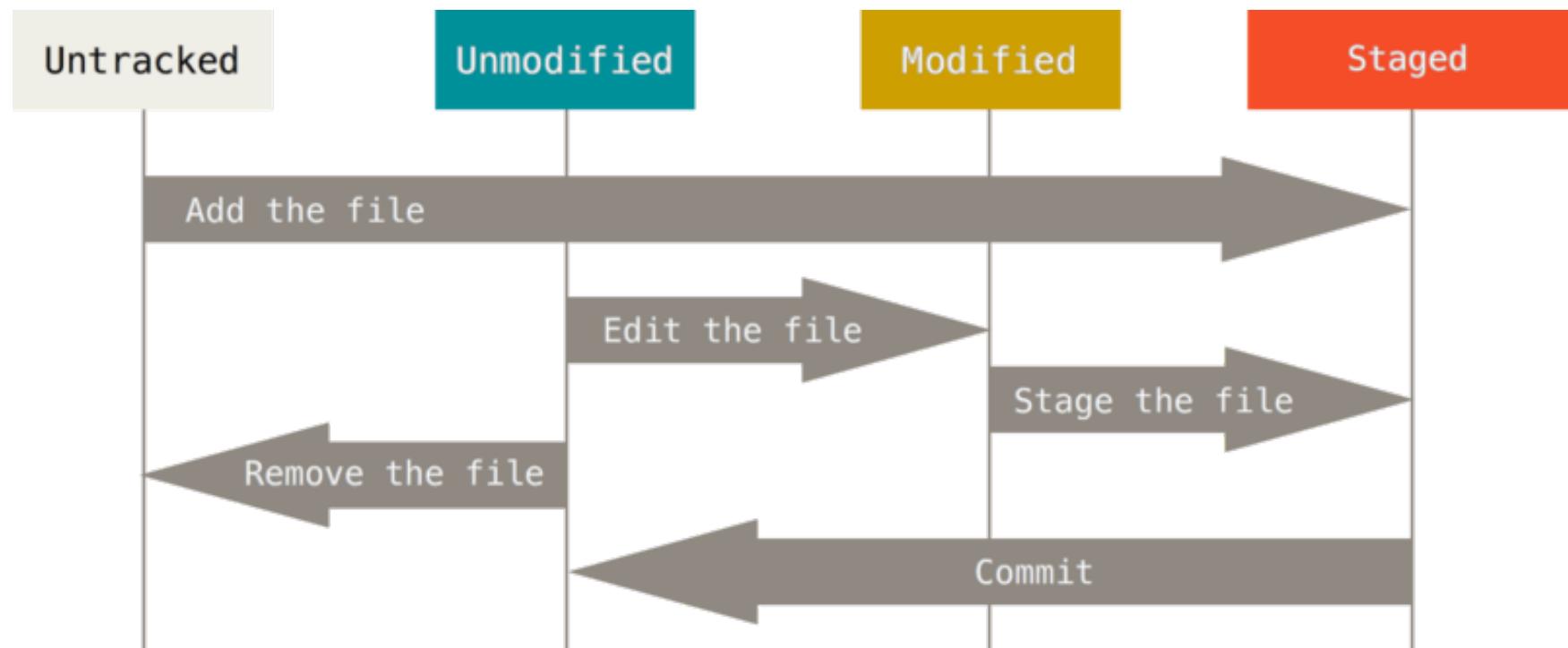
# Basic Git Workflow

- Git has three main states that files can reside in:
  - **Modified:** file changed but not committed yet
  - **Staged:** file marked to be included into the next commit
  - **Committed:** file stored in your **local** database



# Checking the status of your file

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```



# git add & git commit

```
$ echo "file content" > file.txt          (untracked)
$ git add file.txt                         (new file: staged for commit)
$ echo "added new content" >> file.txt    (changes not staged for commit)
$ git add file.txt                         (changes staged for commit)

$ git commit -m "second commit"           (it's now unmodified)
```

# Log

```
$ git log  
commit d921970aadf03b3cf0e71becdaab3147ba71cdef  
Author: Andrea Fornaia <mrossi@email.com>  
Date:   Thu May 07 15:08:43 2020 -0800
```

second commit

```
commit 1c002dd4b536e7479fe34593e72e6c6c1819e53b  
Author: Andrea Fornaia <mrossi@email.com>  
Date:   Thu May 07 14:58:32 2020 -0800
```

first commit

# Basic Git Workflow

1. Modify files in your working tree.
2. **Selectively stage** just those changes you want to be part of your next commit, which adds only those changes to the staging area.
3. Do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

# A few points on commits

- Use **frequent, small, cohesive commits**
  - No need to wait, they are local at first
- Don't get out of sync with your collaborators
- Commit the sources, **not the derived files**
- Use a **.gitignore** file to indicate files to be ignored

```
$ cat .gitignore
*.class
*.log
build/
doc/*.pdf
doc/**/*.*txt
```

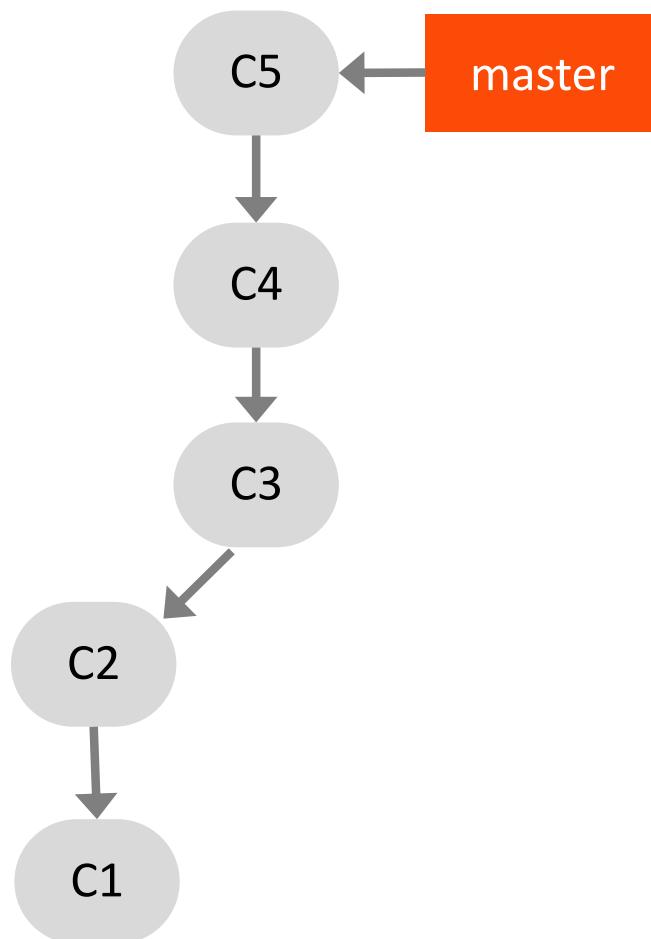
# Good comments

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSOKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

# Branching & Merging

# Branch – Linear History



Every repository starts with one commit (root) that has no predecessors (C1)

Every repository starts with a branch called “master”

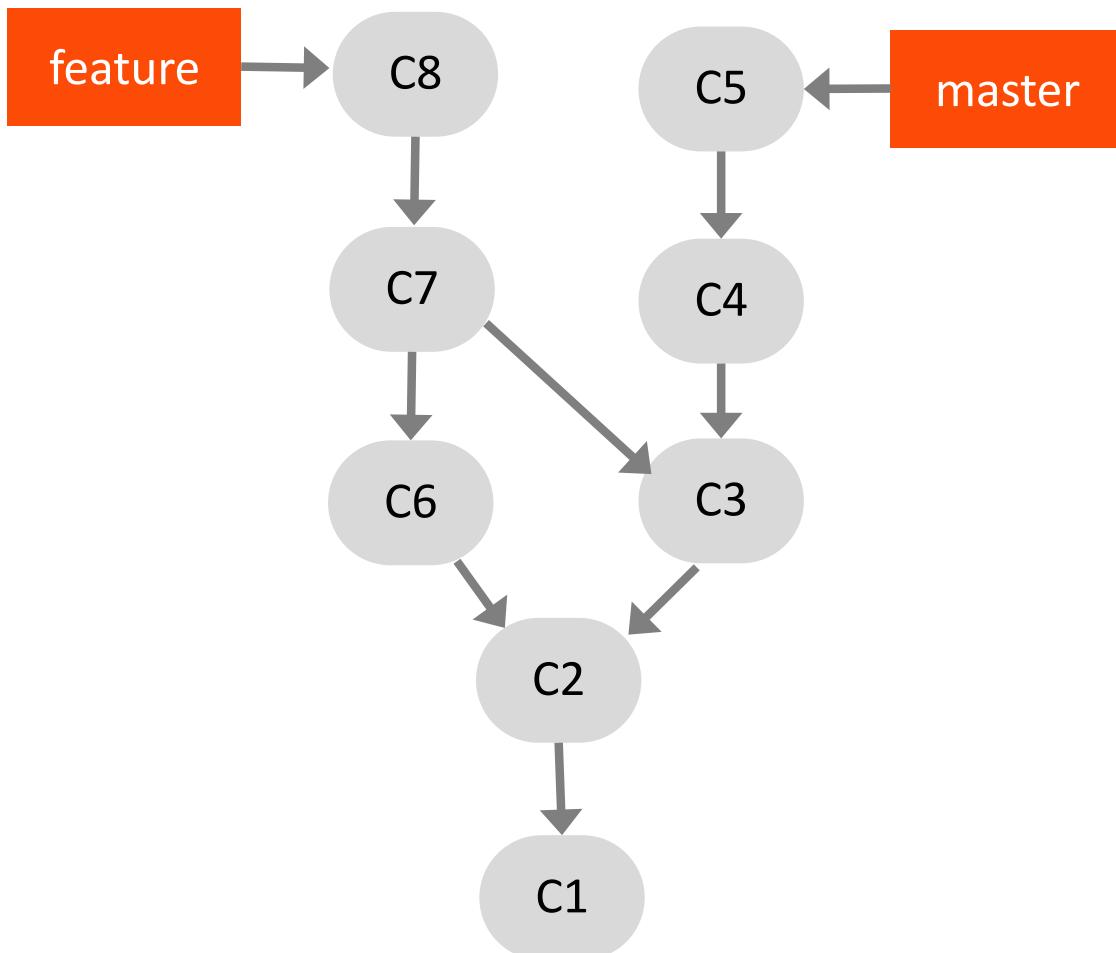
The **branch** label points to the last commit on that branch, called “branch tip” (e.g. master -> C5)

A branch is the **collection** of all commits reachable from the **tip** down **to the root** (e.g. master = {C1 .. C5})

A **new commit** is added to the branch as a node pointing to the branch tip (**on top of branch**), then moving the branch label to the new tip

# Branch – Non Linear History

Git supports and encourages non-linear development. Commits can be represented as a **DAG**



A commit may have:

- no predecessor (first commit, **root**)
- one predecessor (**normal** commit)
- two predecessors (**merge** commit)

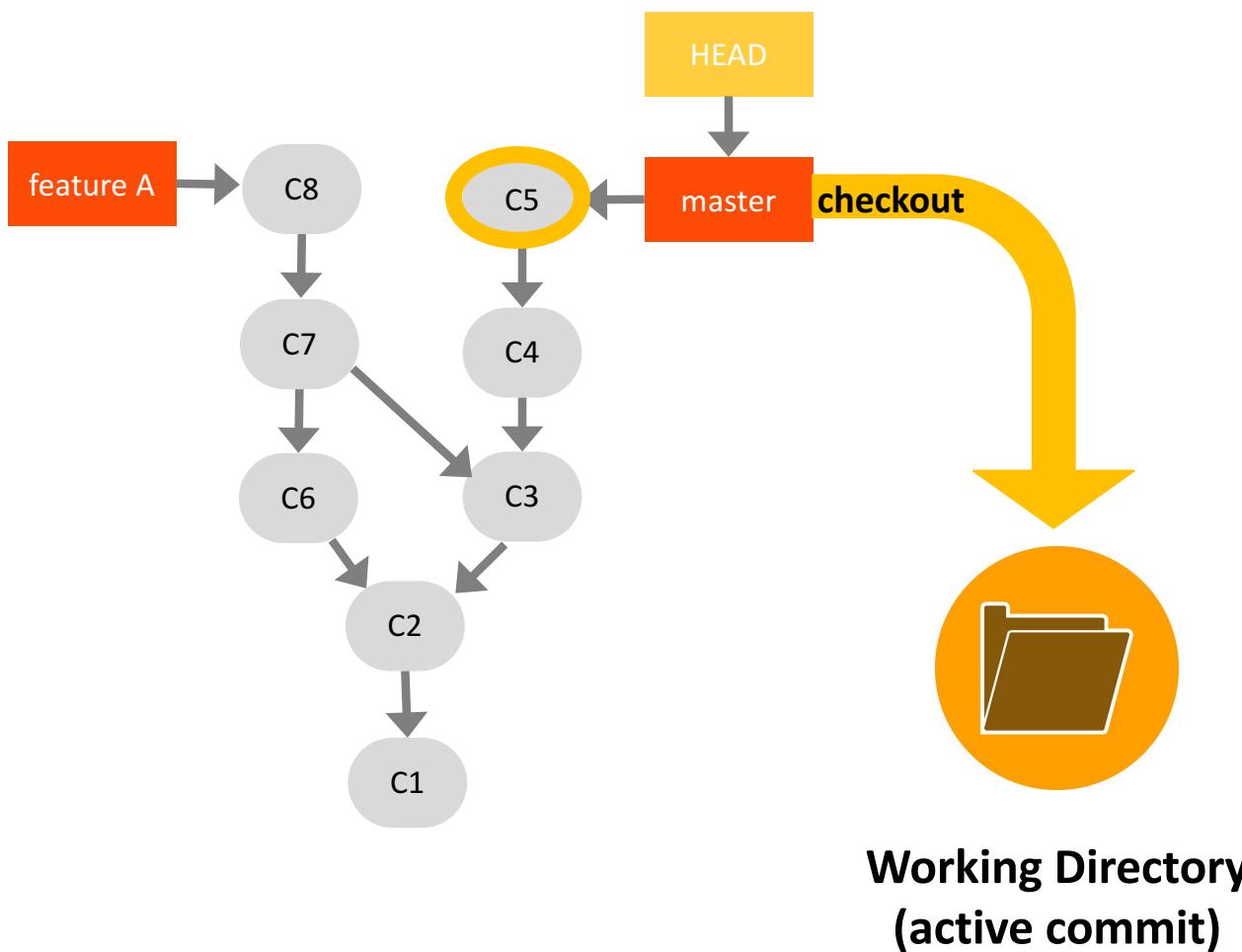
We may have several branches; in this example:

- master {C1; C2; C3; C4; C5}
- feature {C1; C2; C3; C6; C7; C8}

Note that branches **can overlap**

References are **not bidirectional**: by deleting a branch you may lose access to commits (garbage collector)

# Only one active branch at a time



We can **switch** from one branch to another without loosing our work

HEAD is a special label that points to the **active branch**

Working directory in your file system will contain the snapshot (files and dirs) of the **active commit**

# Git checkout

```
$ git checkout <branch>
```

- If branch doesn't exist both on local and remote repositories, will rise an **error**. To create a **new branch** use **-b** option

```
$ git checkout -b <new-branch>
# same as:
$ git branch <new-branch> && git checkout <new-branch>
```

- If branch doesn't exist into local repository, but exists into exactly one remote repository, will create a **tracking branch**

# Git graph

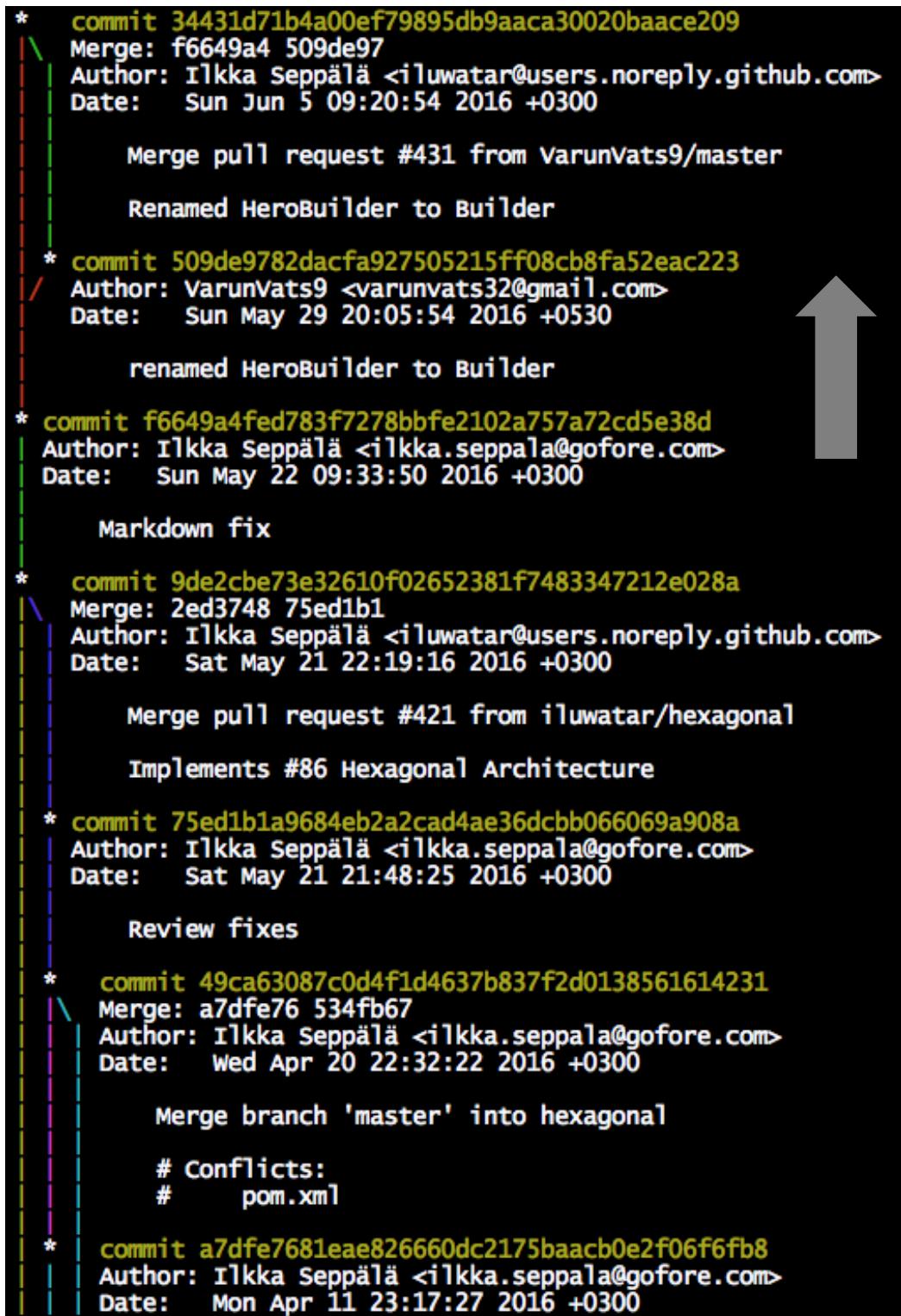
```
$ git log  
--graph  
--decorate  
--oneline  
--all
```

Possiamo creare un alias

```
$ git config --global  
alias.graph "log  
--graph  
--decorate  
--oneline"
```

Creando un nuovo comando

```
$ git graph  
$ git graph --all
```



```
* commit 34431d71b4a00ef79895db9aaca30020baace209  
Merge: f6649a4 509de97  
Author: Ilkka Seppälä <iluwatar@users.noreply.github.com>  
Date: Sun Jun 5 09:20:54 2016 +0300  
  
    Merge pull request #431 from VarunVats9/master  
  
    Renamed HeroBuilder to Builder  
  
* commit 509de9782dacfa927505215ff08cb8fa52eac223  
Author: VarunVats9 <varunvats32@gmail.com>  
Date: Sun May 29 20:05:54 2016 +0530  
  
    renamed HeroBuilder to Builder  
  
* commit f6649a4fed783f7278bbfe2102a757a72cd5e38d  
Author: Ilkka Seppälä <ilkka.seppala@gofore.com>  
Date: Sun May 22 09:33:50 2016 +0300  
  
    Markdown fix  
  
* commit 9de2cbe73e32610f02652381f7483347212e028a  
Merge: 2ed3748 75ed1b1  
Author: Ilkka Seppälä <iluwatar@users.noreply.github.com>  
Date: Sat May 21 22:19:16 2016 +0300  
  
    Merge pull request #421 from iluwatar/hexagonal  
  
    Implements #86 Hexagonal Architecture  
  
* commit 75ed1b1a9684eb2a2cad4ae36dcbb066069a908a  
Author: Ilkka Seppälä <ilkka.seppala@gofore.com>  
Date: Sat May 21 21:48:25 2016 +0300  
  
    Review fixes  
  
* commit 49ca63087c0d4f1d4637b837f2d0138561614231  
Merge: a7dfe76 534fb67  
Author: Ilkka Seppälä <ilkka.seppala@gofore.com>  
Date: Wed Apr 20 22:32:22 2016 +0300  
  
    Merge branch 'master' into hexagonal  
  
    # Conflicts:  
    #   pom.xml  
  
* commit a7dfe7681eae826660dc2175baacb0e2f06f6fb8  
Author: Ilkka Seppälä <ilkka.seppala@gofore.com>  
Date: Mon Apr 11 23:17:27 2016 +0300
```

# Git merge

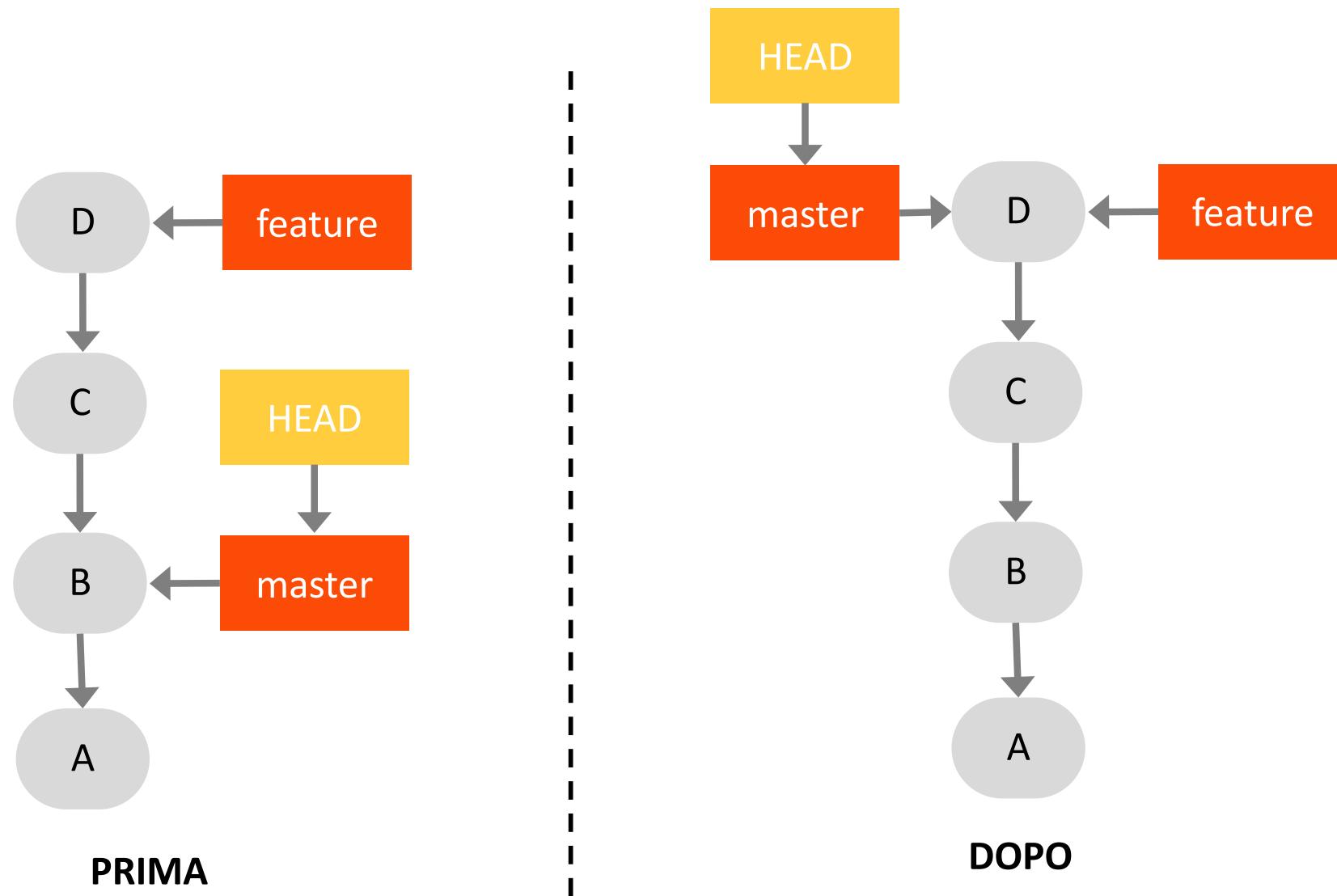
- Merge a source branch **into** the active branch

```
$ git merge <branch>
```

- Merge the **file modifications** made in the target branch into to the active branch
- It can be seen as a **set operation**:
  - After the merge, the active branch must contain **all the commits** of the source branch

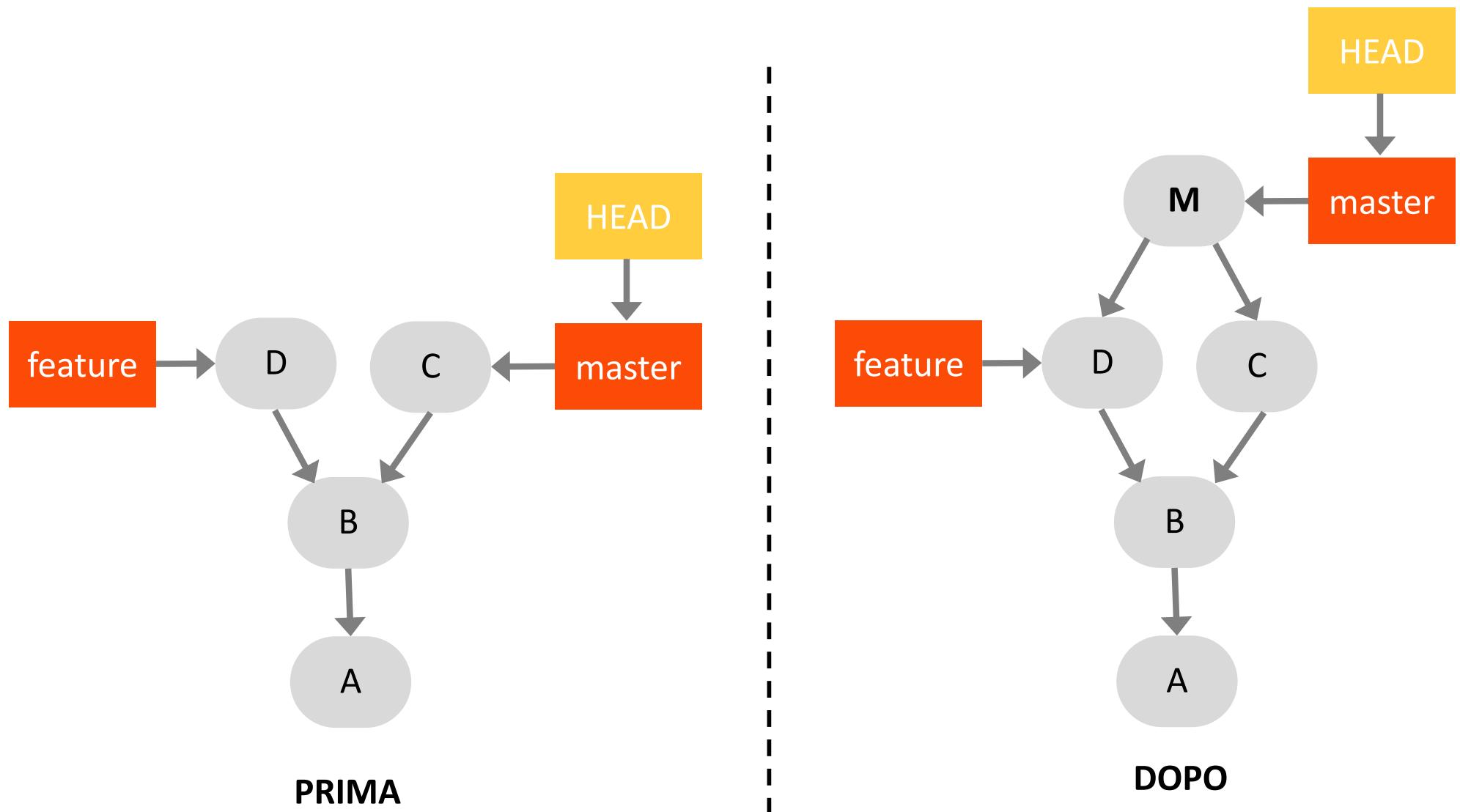
# Fast Forward Merge

master = {A,B} feature = {A,B,C,D} ==> master = {A,B,C,D}



# Commit Merge

master = {A,B,C} feature = {A,B,D} ==> master = {A,B,C,D,M}



# Conflicts

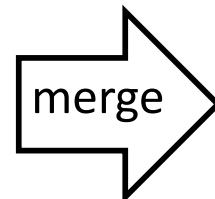
# Auto-merging differences on separate lines

```
0 xxx  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

<master>

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10 xxx
```

<feature>



```
0 xxx  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10 xxx
```

<master>

```
$ git checkout master  
$ git merge feature  
Auto-merging long.txt  
Merge made by the 'recursive' strategy.  
 long.txt | 2 +-  
 1 file changed, 1 insertion(+), 1  
deletion(-)
```

recursive  
strategy

# Auto-merging rocks... but?

- Occasionally, this process doesn't go smoothly...
- Did you change the **same part of the same file?**
  - Git won't be able to merge them cleanly!
  - Git will ask you what to do to solve the **conflict**

**PUSH REJECTED, REBASE OR  
MERGE**

**GIT PUSH --FORCE**

# Conflict example

**master**

**file3.txt <master>**

added file3 for bug fix

**file1.txt <master>**

original file1 content  
bug fixed directly on master!

**feature**

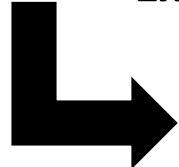
**file2.txt <feature>**

file2 content for feature

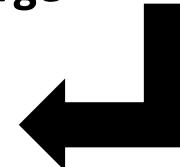
**file1.txt <feature>**

original file1 content  
added new content for feature

**Expected merge result: file1.txt <master after merge>**



original file1 content  
bug fixed directly on master!  
added new content for feature1



# Try to merge master ← feature

```
$ git merge feature
Auto-merging file1.txt
CONFLICT (content): Merge conflict in file1.txt
Automatic merge failed; fix conflicts and then commit the result.
```

```
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

```

Changes to be committed:

```
  new file:  file2.txt
```

Unmerged paths:

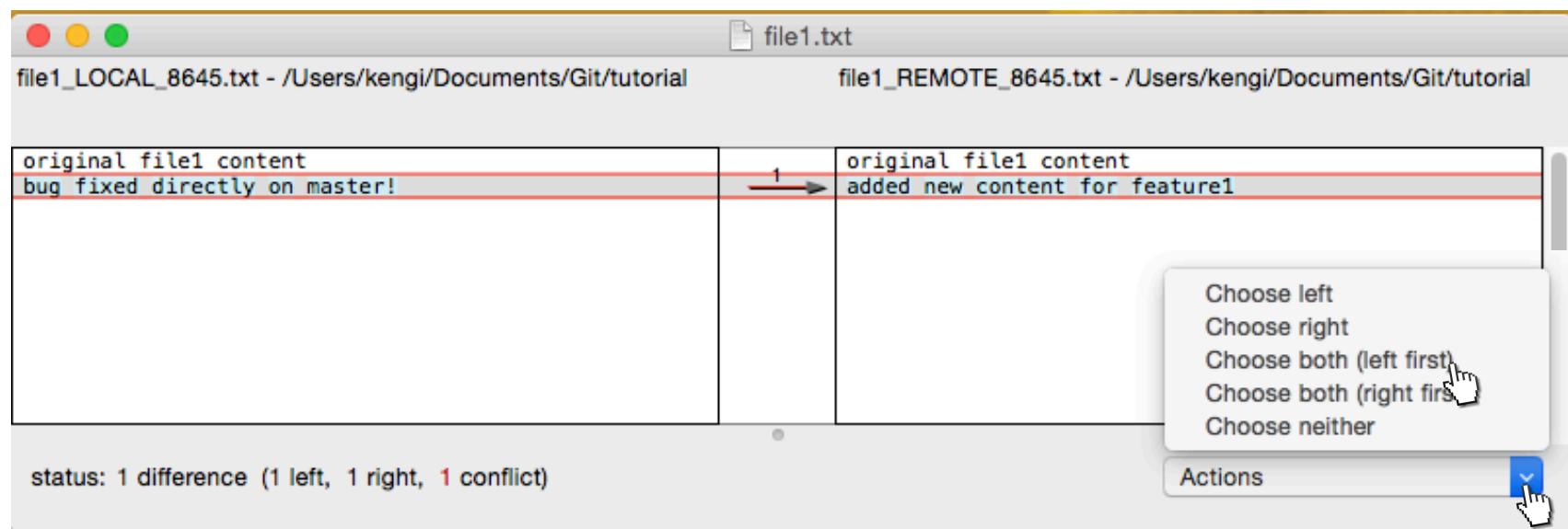
(use "git add <file>..." to mark resolution)

both modified: file1.txt

# Solve conflicts on file1.txt

```
$ vim file1.txt
original file1 content
<<<<< HEAD
bug fixed directly on master!
=====
added new content for feature1
>>>>> feature1

$ git mergetool
```



# Merge with Visual Studio Code

The screenshot shows the Visual Studio Code interface during a merge operation. The Explorer sidebar on the left lists files: file1.txt (marked C), file1.txt (marked M, currently selected), file2.txt (marked A), and file3.txt. The main editor area displays the content of file1.txt with a merge conflict:

```
1 original file1 content
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
2 <<<< HEAD (Current Change)
3 bug fixed directory on master
4 =====
5 added new content for feature1
6 >>>> feature1 (Incoming Change)
7
```

The status bar at the bottom indicates the current branch is 'master'.

The terminal at the bottom shows the command-line output of the merge process:

```
[master 489335e] bug fixed on master
2 files changed, 2 insertions(+)
create mode 100644 file3.txt
MacBook-Pro-di-Andrea-4:merge kengi$ git merge feature1
Auto-merging file1.txt
CONFLICT (content): Merge conflict in file1.txt
Automatic merge failed; fix conflicts and then commit the result.
MacBook-Pro-di-Andrea-4:merge kengi$ ls
file1.txt file2.txt file3.txt
MacBook-Pro-di-Andrea-4:merge kengi$ ls -la
total 24
drwxr-xr-x  6 kengi  staff  204 Oct 10 15:47 .
drwxr-xr-x  4 kengi  staff  136 Oct 10 15:39 ..
drwxr-xr-x 17 kengi  staff  578 Oct 10 15:47 .git
-rw-r--r--  1 kengi  staff  122 Oct 10 15:47 file1.txt
-rw-r--r--  1 kengi  staff   27 Oct 10 15:47 file2.txt
-rw-r--r--  1 kengi  staff   24 Oct 10 15:45 file3.txt
MacBook-Pro-di-Andrea-4:merge kengi$
```

The status bar at the bottom right shows the file is in 'Plain Text' mode with tab size 4, and there are 0 changes (0 additions, 0 deletions).

# Merge with Visual Studio Code

The screenshot shows the Visual Studio Code interface during a merge operation. The title bar indicates "file1.txt: Current Changes ↔ Incoming Changes – merge".

**EXPLORER** sidebar:

- OPEN EDITORS:** file1.txt (C)
- MERGE:** file1.txt (C), file2.txt (A), file3.txt

**EDITOR** pane:

file1.txt: Current Changes ↔ Incoming Changes

```
1 - bug fixed directory on master
2
1 + added new content for feature1
2
```

**PROBLEMS**, **OUTPUT**, **DEBUG CONSOLE**, **TERMINAL** tabs:

TERMINAL output:

```
[master 489335e] bug fixed on master
2 files changed, 2 insertions(+)
create mode 100644 file3.txt
MacBook-Pro-di-Andrea-4:merge kengi$ git merge feature1
Auto-merging file1.txt
CONFLICT (content): Merge conflict in file1.txt
Automatic merge failed; fix conflicts and then commit the result.
MacBook-Pro-di-Andrea-4:merge kengi$ ls
file1.txt file2.txt file3.txt
MacBook-Pro-di-Andrea-4:merge kengi$ ls -la
total 24
drwxr-xr-x  6 kengi  staff  204 Oct 10 15:47 .
drwxr-xr-x  4 kengi  staff  136 Oct 10 15:39 ..
drwxr-xr-x 17 kengi  staff  578 Oct 10 15:47 .git
-rw-r--r--  1 kengi  staff  122 Oct 10 15:47 file1.txt
-rw-r--r--  1 kengi  staff   27 Oct 10 15:47 file2.txt
-rw-r--r--  1 kengi  staff   24 Oct 10 15:45 file3.txt
MacBook-Pro-di-Andrea-4:merge kengi$
```

Bottom status bar:

master+1 0 ▲ 0 Ln 1, Col 1 Tab Size: 4 Plain Text

# Commit. New merge node created

```
$ git commit
Merge branch 'feature1'
# Conflicts:
#   file1.txt
#
# It looks like you may be committing a merge.
# If this is not correct, please remove the file
#   .git/MERGE_HEAD
# and try again.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# All conflicts fixed but you are still merging.
#
# Changes to be committed:
#   modified:   file1.txt
#   new file:   file2.txt
```