

# Deep dive into { git }

---

Michael Kaufmann

# Michael Kaufmann

Founder & Managing Director, Xebia Germany



@mike\_kaufmann



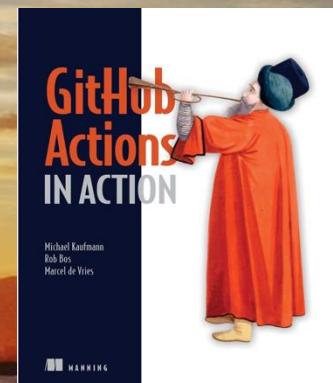
@wulfland



<https://writeabout.net>



- 25 years software developer
- 15 years ALM & DevOps
- 10 years Git and GitHub
- Microsoft RD & MVP



# Agenda

- › Getting started
- › The 3 principles
  - › Hash / SHA
  - › Snapshots
  - › Trees (DAG)
- › Branches and tags
- › Referencing commits
- › Diffs and patches
- › Merging
- › Resolving merge conflicts
- › Remotes
- › Branching workflows
- › Git aliases
- › Reverting changes
- › Ammending changes
- › Resetting your repository
- › Rebase
- › Cherry-picking and the reflog
- › The stash

THIS IS GIT. IT TRACKS COLLABORATIVE WORK  
ON PROJECTS THROUGH A BEAUTIFUL  
DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL  
COMMANDS AND TYPE THEM TO SYNC UP.  
IF YOU GET ERRORS, SAVE YOUR WORK  
ELSEWHERE, DELETE THE PROJECT,  
AND DOWNLOAD A FRESH COPY.



# Getting started

# Getting started

## › Local git configs

- ─  --system
- ─  --global
- ─  -- local

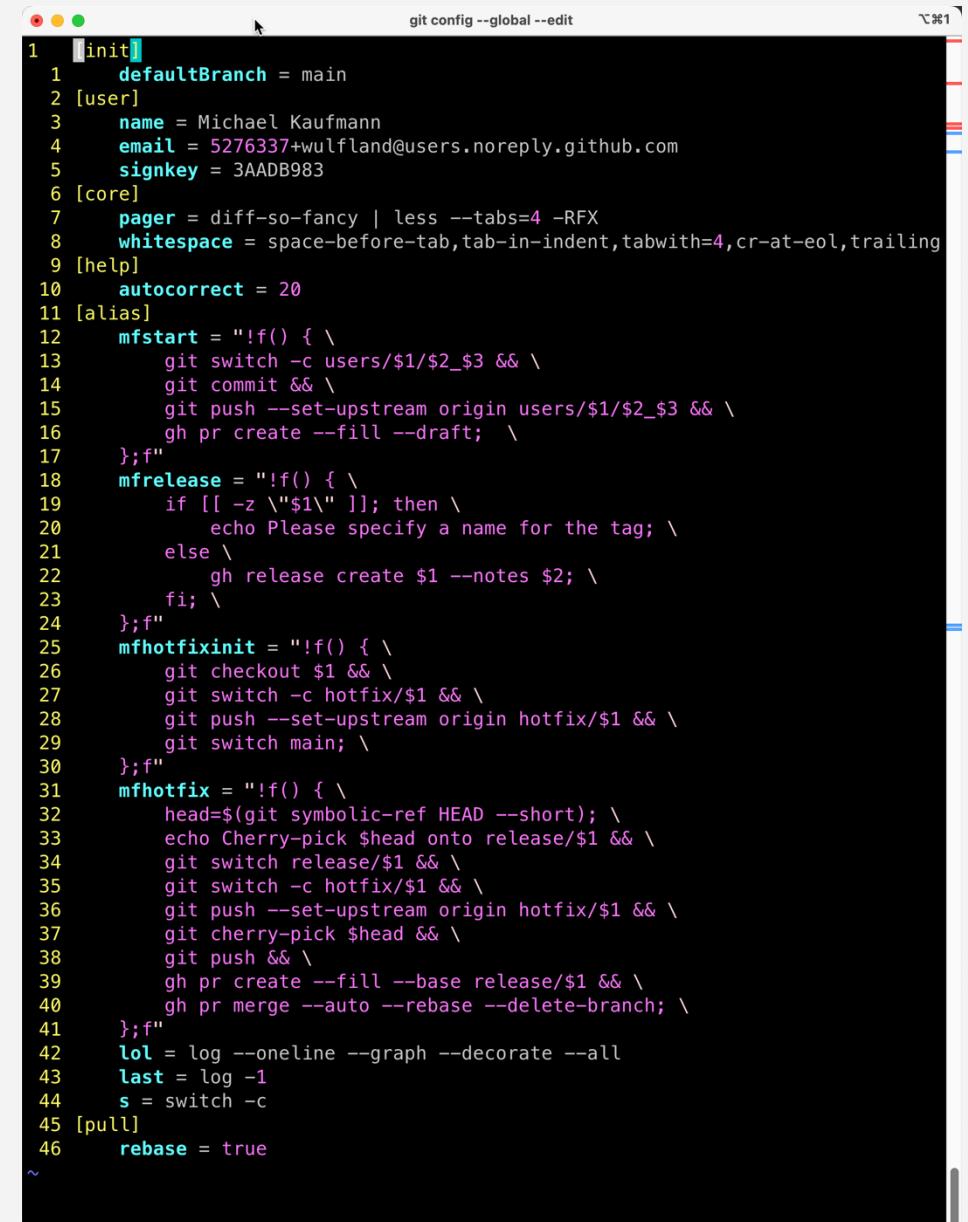
```
$ git config --global user.email foo@bar
$ git config --global user.name "Foo Bar"
$ git config --global init.defaultBranch main
$ git config --global help.autocorrect 20
$ git config --global core.editor "code --wait"
$ git config --global alias.last "log -1"
$ git config --global --edit
```

## › Functions "!f(){ ... };f "

## › \$1, \$2, \$3, ... \$@

## › Line endings: \

## › && or ; to separate commands



The screenshot shows a terminal window titled "git config --global --edit". The window displays a large block of configuration code. The code includes sections for [init], [user], [core], [help], [alias], and [pull]. It defines several functions: `mfstart`, `mfrelease`, `mhotfixinit`, `mhotfix`, and `lol`. These functions perform tasks like switching branches, committing changes, pushing to upstream, creating GitHub releases, and performing hotfixes. The code also includes aliases for `last` and `s`, and settings for `pull.rebase`.

```
1 [init]
1   defaultBranch = main
2 [user]
3   name = Michael Kaufmann
4   email = 5276337+wulfland@users.noreply.github.com
5   signkey = 3AADB983
6 [core]
7   pager = diff-so-fancy | less --tabs=4 -RFX
8   whitespace = space-before-tab,tab-in-indent,tabwith=4,cr-at-eol,trailing
9 [help]
10  autocorrect = 20
11 [alias]
12  mfstart = "!f() { \
13      git switch -c users/$1/$2_\$3 && \
14      git commit && \
15      git push --set-upstream origin users/$1/$2_\$3 && \
16      gh pr create --fill --draft; \
17  };f"
18  mfrelease = "!f() { \
19      if [[ -z \"\$1\" ]]; then \
20          echo Please specify a name for the tag; \
21      else \
22          gh release create \$1 --notes \$2; \
23      fi; \
24  };f"
25  mhotfixinit = "!f() { \
26      git checkout \$1 && \
27      git switch -c hotfix/\$1 && \
28      git push --set-upstream origin hotfix/\$1 && \
29      git switch main; \
30  };f"
31  mhotfix = "!f() { \
32      head=$(git symbolic-ref HEAD --short); \
33      echo Cherry-pick \$head onto release/\$1 && \
34      git switch release/\$1 && \
35      git switch -c hotfix/\$1 && \
36      git push --set-upstream origin hotfix/\$1 && \
37      git cherry-pick \$head && \
38      git push && \
39      gh pr create --fill --base release/\$1 && \
40      gh pr merge --auto --rebase --delete-branch; \
41  };f"
42  lol = log --oneline --graph --decorate --all
43  last = log -1
44  s = switch -c
45 [pull]
46  rebase = true
~
```

## Check your git installation

Make sure git is installed and the version is > 2.23:

```
$ git --version  
> git version 2.35.1
```

If not, [download](#) and install git.

## Check your git config

1. Check your name and email address:

```
$ git config --global user.name  
$ git config --global user.email
```

If this does not return the desired values, set the config using these commands:

```
$ git config --global user.name '<your_name>'  
$ git config --global user.email '<your_email_address>'
```

2. Set the default branch to `main`:

```
$ git config --global init.defaultBranch main
```

3. Check your default editor

Check your default editor (i.e. by running `git config --global --edit`). If you like the editor, then you are good. If you're stuck in vim (click `ESC` : `q` ! `Enter` to exit), then configure the editor of your choice - for example CSCode:

```
$ git config core.editor 'code --wait'
```

# The 3 principles

# The 3 principles

- › Everything in git is a file – and changes are tracked using SHA hash values (SHA-1 or SHA-256)
- › Git saves snapshots – and not deltas
- › Everything is a rooted tree
  - › Commits
  - › Files and folders

```
..UnterDerHaube (-zsh) ~/source/repos/UnterDerHaube 10:56:10 ⓘ
> git init
Initialized empty Git repository in /Users/m.kaufmann/source/repos/UnterDerHaube/.git/
Apple ~/source/repos/UnterDerHaube git ⌂ main ?1 10:56:36 ⓘ
> git hash-object txt/Doc.txt
f20acb769ad757b2058c8c3c96e7eac2e1258503
Apple ~/source/repos/UnterDerHaube git ⌂ main ?1 10:57:58 ⓘ
> vim txt/Doc.txt
Apple ~/source/repos/UnterDerHaube git ⌂ main ?1 10:58:21 ⓘ
14s ✘ 10:58:21 ⓘ
> git hash-object txt/Doc.txt
f85de0e273f9ed77d2a37b1dd7397e667b0ad335
Apple ~/source/repos/UnterDerHaube git ⌂ main ?1 10:58:24 ⓘ
> git add txt/Doc.txt
Apple ~/source/repos/UnterDerHaube git ⌂ main +1 10:59:03 ⓘ
> cat .git/HEAD
ref: refs/heads/main
Apple ~/source/repos/UnterDerHaube git ⌂ main +1 11:00:27 ⓘ
> cat .git/Index
DIRC_0*00
0_*0g000000>0]00s00WH{09~f{
05
txt/Doc.txt0({0700030>0pC0k0J0%
Apple ~/source/repos/UnterDerHaube git ⌂ main +1 11:00:43 ⓘ
> git ls-files --stage
100644 f85de0e273f9ed77d2a37b1dd7397e667b0ad335 0      txt/Doc.txt
Apple ~/source/repos/UnterDerHaube git ⌂ main +1 11:01:05 ⓘ
> cat .git/objects/f8/5de0e273f9ed77d2a37b1dd7397e667b0ad335
xK00R03b0K0M0Rp0Sp00M-QPr00VRp*0/000/0L00K0Q(0)2000000L0â000000%
Apple ~/source/repos/UnterDerHaube git ⌂ main +1 11:03:05 ⓘ
> git cat-file -t f85de
blob
Apple ~/source/repos/UnterDerHaube git ⌂ main +1 11:05:52 ⓘ
> git cat-file -p f85de
Name: Dr. Emmet "Doc" Brown
Original time: 1985
Nickname: Doc
Apple ~/source/repos/UnterDerHaube git ⌂ main +1 11:05:57 ⓘ
>
```

watch (watch) Andoria.local: Wed Nov 25 11:06:00 2020

```
.git/
  HEAD
  config
  description
  index
  info/
  exclude
  objects/
    f8/
      5de0e273f9ed77d2a37b1dd7397e667b0ad335
        info/
        pack/
      refs/
        heads/
        tags/
    txt/
      Doc.txt

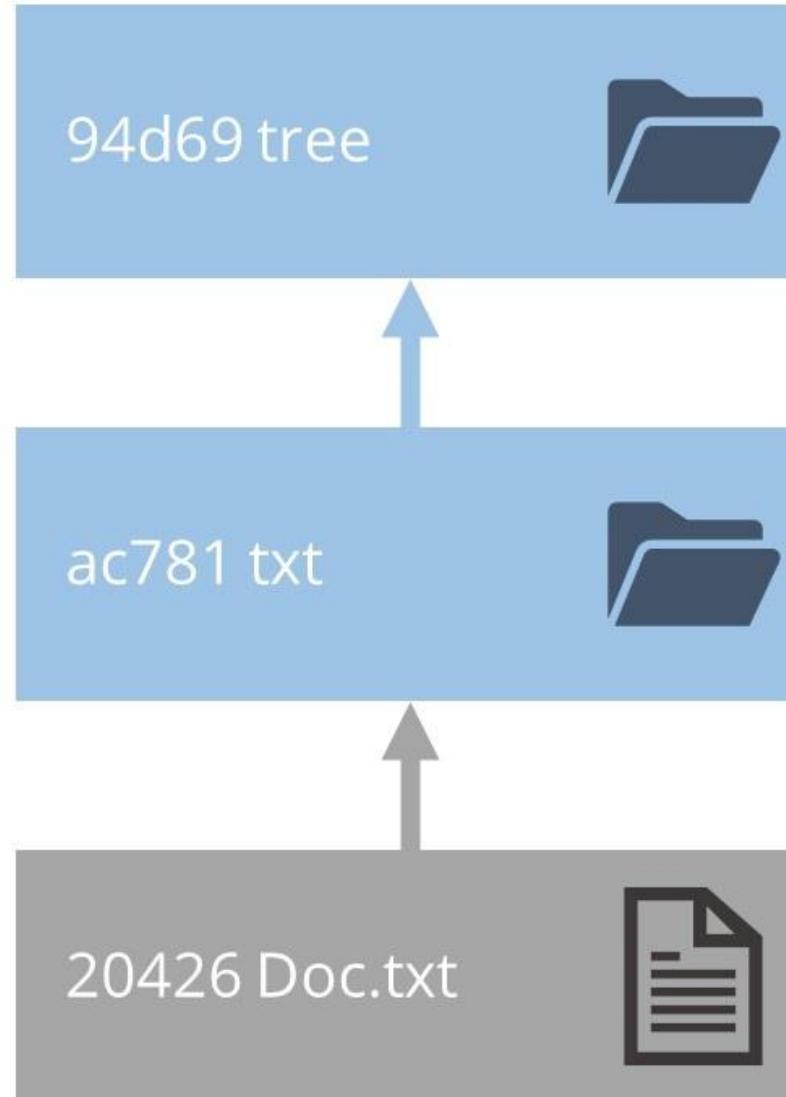
10 directories, 7 files
```

```
..UnterDerHaube (-zsh) 11:09:37 ⓘ watch (watch) Every 2.0s... Andoria.local: Wed Nov 25 11:13:11 2020
❯ git cat-file -t fa79
commit
apple ~ /source/repos/UnterDerHaube git ⚡ main 11:09:42 ⓘ
❯ git cat-file -p fa79
tree 70f8d78e3776ac011b60926fc52e30d76789de59
author Michael Kaufmann <wulfland@hotmail.com> 1606298912 +0100
committer Michael Kaufmann <wulfland@hotmail.com> 1606298912 +0100

ErsterCommit
apple ~ /source/repos/UnterDerHaube git ⚡ main 11:09:46 ⓘ
❯ git cat-file -t 70f8
tree
apple ~ /source/repos/UnterDerHaube git ⚡ main 11:10:57 ⓘ
❯ git cat-file -p 70f8
040000 tree ba105a49862bcf36b8ffff1dae703fc5678f7358a txt
apple ~ /source/repos/UnterDerHaube git ⚡ main 11:11:01 ⓘ
❯ git ls-tree ba105
100644 blob f85de0e273f9ed77d2a37b1dd7397e667b0ad335 Doc.txt
apple ~ /source/repos/UnterDerHaube git ⚡ main 11:11:19 ⓘ
❯ cat .git/refs/heads/main
fa794f016773b66bed29596a0cec389295a5d758
apple ~ /source/repos/UnterDerHaube git ⚡ main 11:12:48 ⓘ
❯

.
.git/
  COMMIT_EDITMSG
  HEAD
  MERGE_RR
  config
  description
  index
  info/
    exclude
  logs/
    HEAD
    refs/
      heads/
        main
  objects/
    70/
      f8d78e3776ac011b60926fc52e30d76789de59
    ba/
      105a49862bcf36b8ffff1dae703fc5678f7358a
    f8/
      5de0e273f9ed77d2a37b1dd7397e667b0ad335
    fa/
      794f016773b66bed29596a0cec389295a5d758
    info/
    pack/
  refs/
    heads/
      main
    tags/
    rr-cache/
  txt/
    Doc.txt

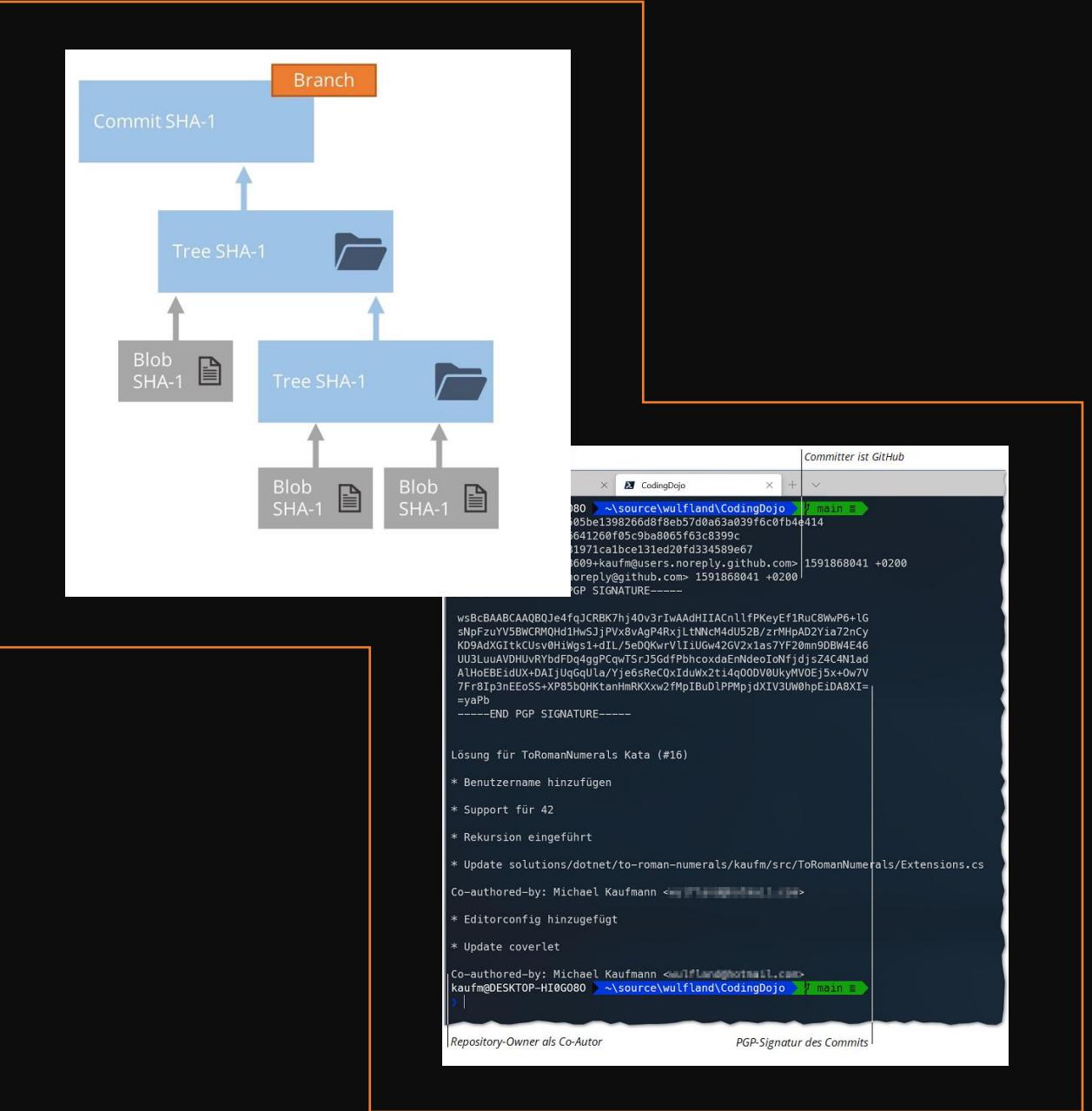
17 directories, 15 files
```



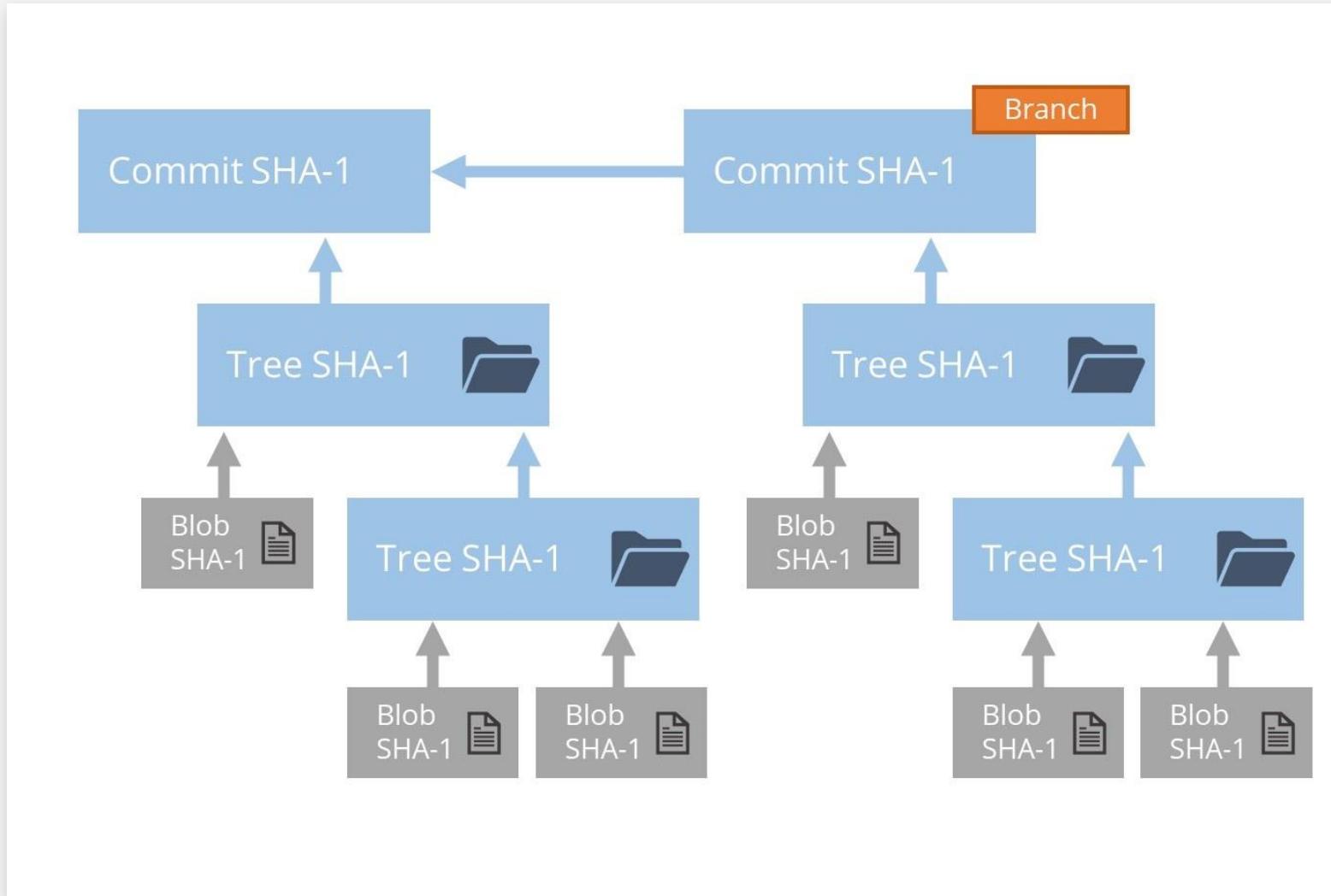
# The structure of a tree

---

# The structure of a commit



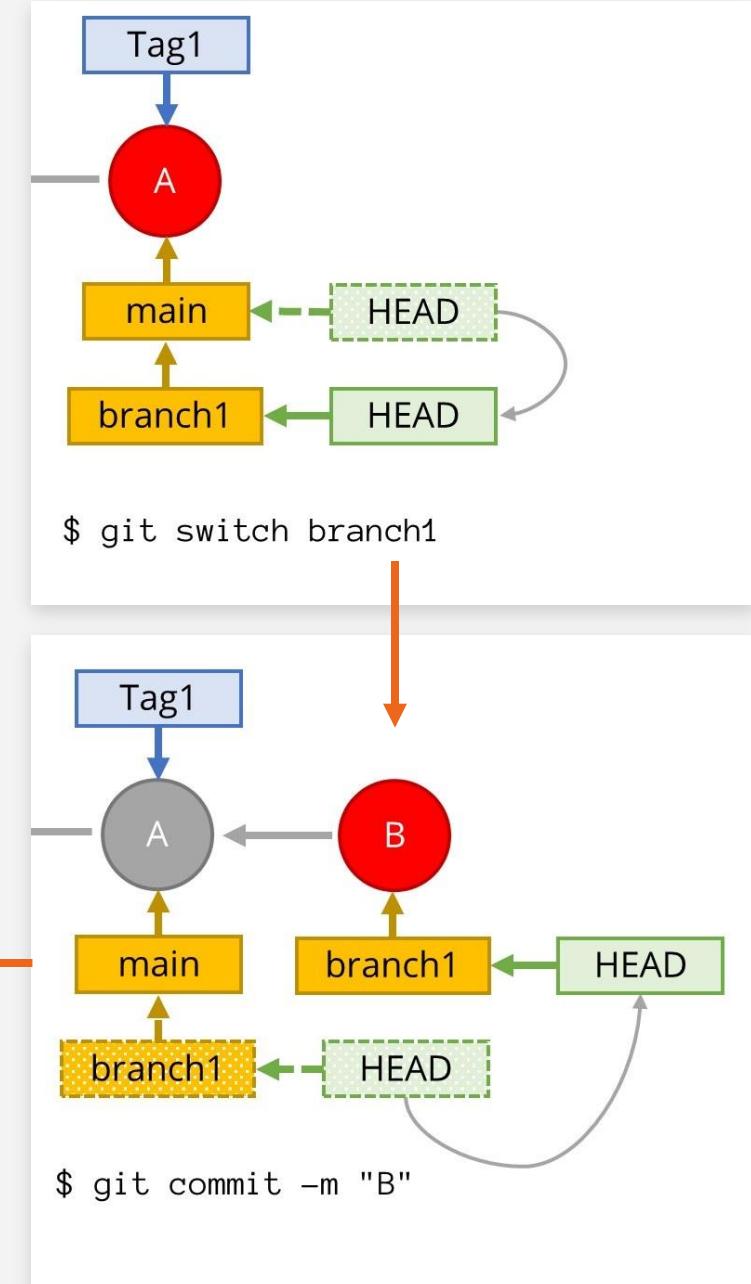
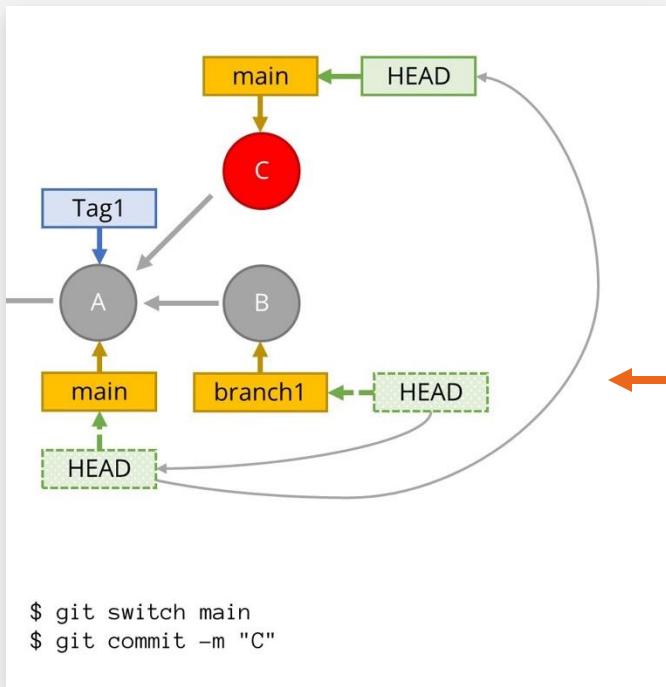
# Everything is a rooted tree



# Branches and tags

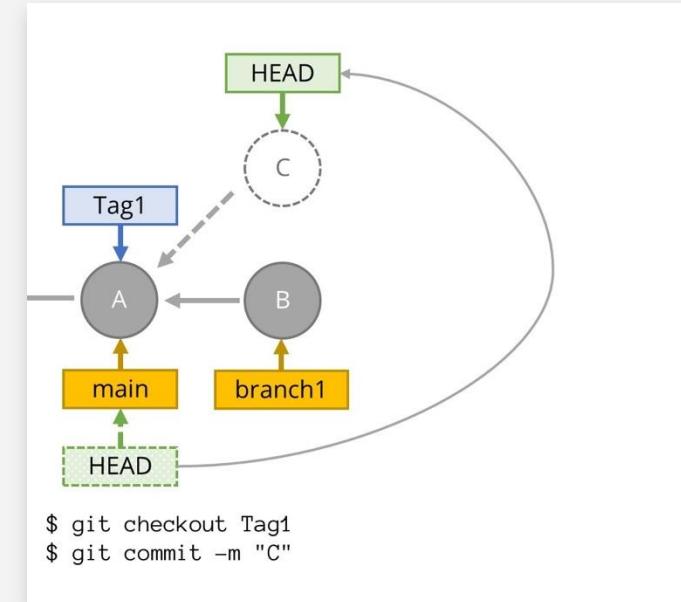
# Branches

- › Pointers to a commit
- › The branch that HEAD points to
  - › will be the parent commit for a new commit
  - › Will point to the new commit (git update-ref)



# Tags

- › Fix pointers to a commit
- › Annotated tags – object of type tag that points to a commit (`git tag -a`)
- › Detached HEAD mode



# Referencing commits

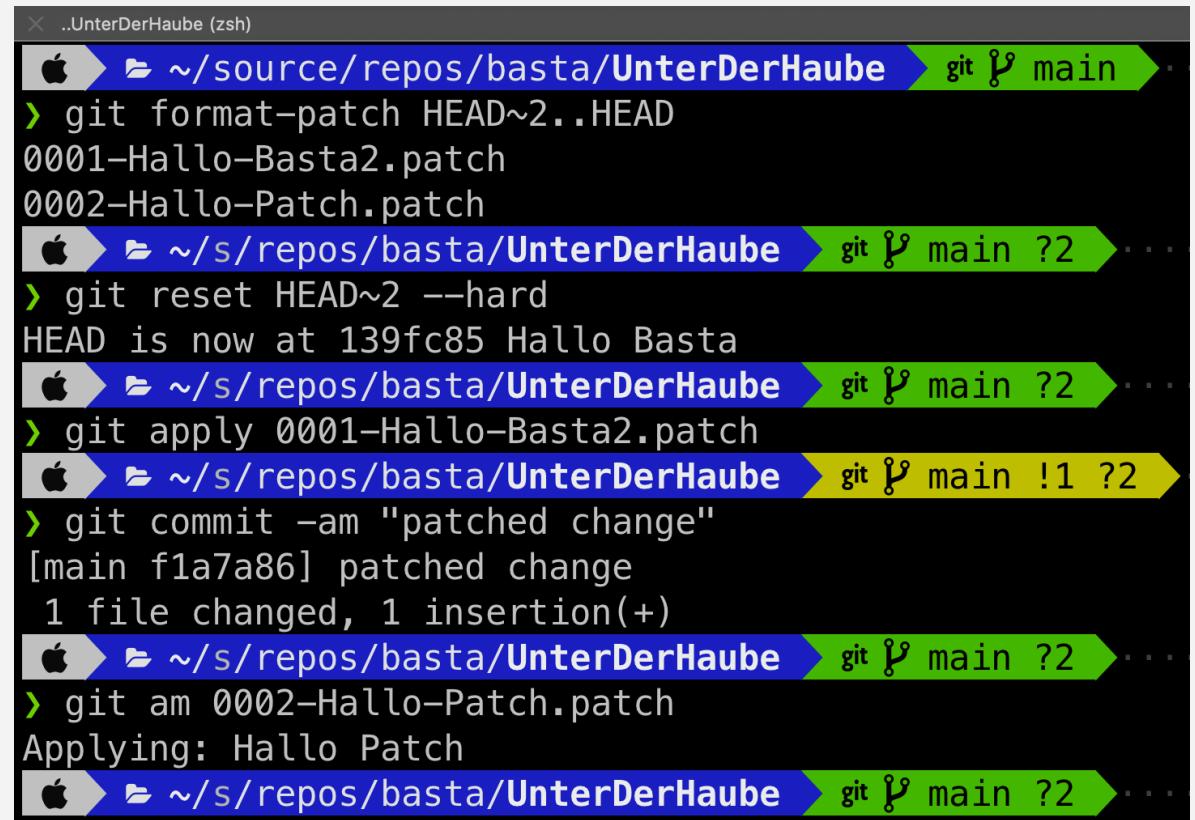
# Referencing commits

- › **Absolute references**
  - › The SHA-1 value  
`c36554656ee78362ced084e5114b0a51001e2f57`
  - › The first part (at least 4 – must be unique)  
`c365`
  - › `HEAD`
  - › `Tag`
  - › `Branch`
- › **Relative in time**
  - › `HEAD@{3}`
  - › `main@{two.weeks.ago}`
  - › `HEAD@{last.month}`
  - › `main@{yesterday}`
- › **Relative to another reference**
  - › `^` = points to the parent commit
  - › `^^` = points to the parent of the parent
  - › `^2` = points to the second parent of the same commit (**merge commits**)
  - › `~` = points to the first parent of the commit
  - › `~2` = `~~` = points to the parent of the parent
- › **Ranges**
  - › Double dot: `main..experiment` (reachable from experiment but not main)
  - › `experiment..main`
  - › Triple dot: `main...experiment` (either of the two references but not both `--left-right`)

# Diff and patches

# Diffs and patches

```
$ git diff  
$ git diff --staged  
$ git diff <REF-1> <REF-2>  
$ git diff HEAD^!  
  
$ git format-patch HEAD~2..HEAD  
$ git reset --hard HEAD~2  
  
$ git apply 0001*.patch  
  
$ git commit -am  
$ git am 0002*.patch
```

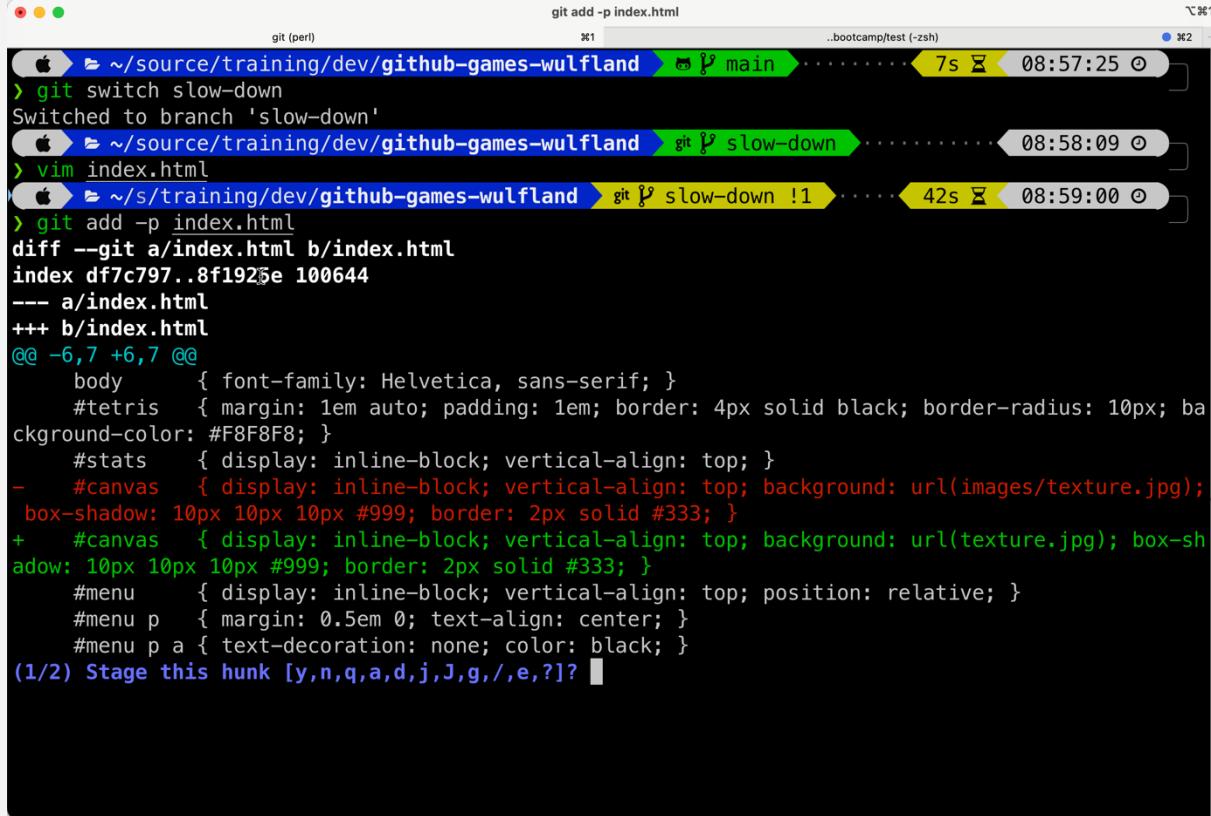


A terminal session showing the creation and application of patches. The session starts in the `UnterDerHaube` repository. It formats two patches from the last two commits into files `0001-Hallo-Basta2.patch` and `0002-Hallo-Patch.patch`. Then, it resets the repository to the state of the second commit, applies the first patch, and commits the changes. Finally, it applies the second patch.

```
x ..UnterDerHaube (zsh)  
apple ~ /source/repos/basta/UnterDerHaube git ⌂ main  
> git format-patch HEAD~2..HEAD  
0001-Hallo-Basta2.patch  
0002-Hallo-Patch.patch  
apple ~ /s/repos/basta/UnterDerHaube git ⌂ main ?2  
> git reset HEAD~2 --hard  
HEAD is now at 139fc85 Hallo Basta  
apple ~ /s/repos/basta/UnterDerHaube git ⌂ main ?2  
> git apply 0001-Hallo-Basta2.patch  
apple ~ /s/repos/basta/UnterDerHaube git ⌂ main !1 ?2  
> git commit -am "patched change"  
[main f1a7a86] patched change  
1 file changed, 1 insertion(+)  
apple ~ /s/repos/basta/UnterDerHaube git ⌂ main ?2  
> git am 0002-Hallo-Patch.patch  
Applying: Hallo Patch  
apple ~ /s/repos/basta/UnterDerHaube git ⌂ main ?2
```

# Diffs and patches

```
$ git add -p | --patch
```



The screenshot shows a terminal window with several tabs. The active tab is titled 'git (perl)' and contains the command:

```
git add -p index.html
```

Below the command, the terminal shows the following session:

```
> git switch slow-down
Switched to branch 'slow-down'
> vim index.html
> git add -p index.html
diff --git a/index.html b/index.html
index df7c797..8f1925e 100644
--- a/index.html
+++ b/index.html
@@ -6,7 +6,7 @@
     body { font-family: Helvetica, sans-serif; }
     #tetris { margin: 1em auto; padding: 1em; border: 4px solid black; border-radius: 10px; background-color: #F8F8F8; }
     #stats { display: inline-block; vertical-align: top; }
-    #canvas { display: inline-block; vertical-align: top; background: url(images/texture.jpg); box-shadow: 10px 10px 10px #999; border: 2px solid #333; }
+    #canvas { display: inline-block; vertical-align: top; background: url(texture.jpg); box-shadow: 10px 10px 10px #999; border: 2px solid #333; }
     #menu { display: inline-block; vertical-align: top; position: relative; }
     #menu p { margin: 0.5em 0; text-align: center; }
     #menu p a { text-decoration: none; color: black; }
```

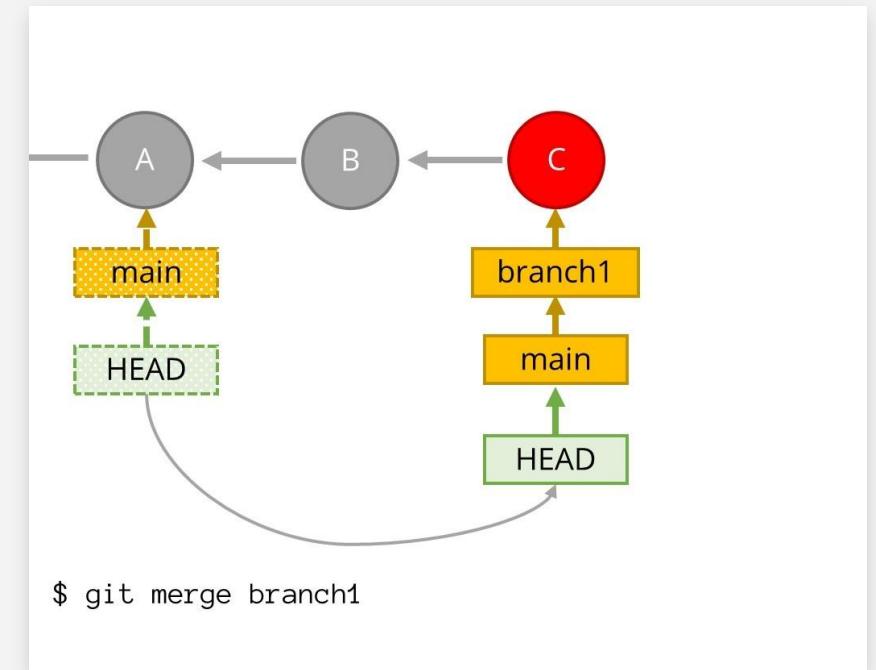
At the bottom of the terminal window, there is a prompt:

```
(1/2) Stage this hunk [y,n,q,a,d,j,J,g/,e,?]?
```

# Merging

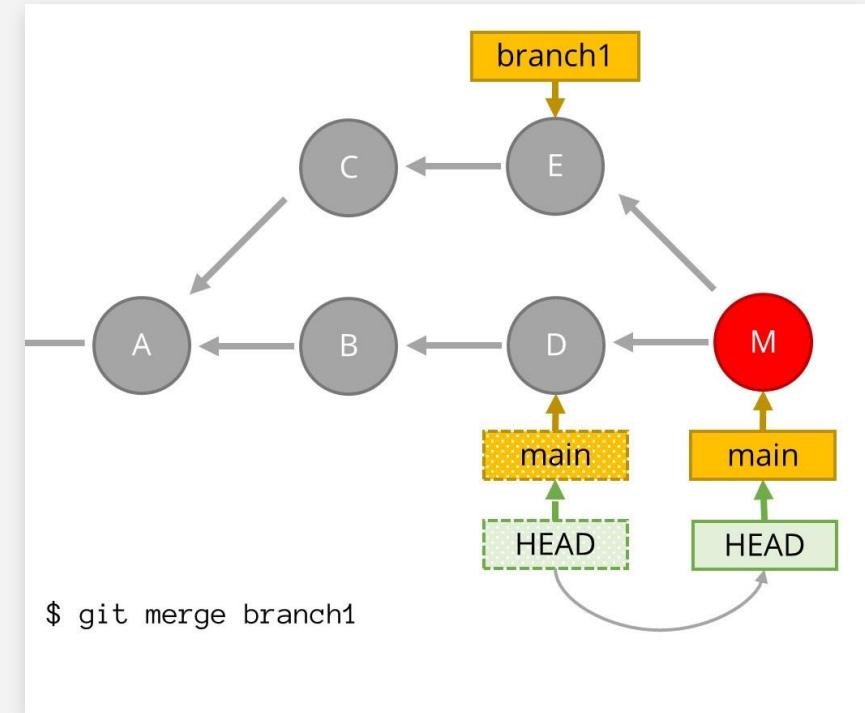
# Fast-forward merge

```
$ git switch main  
$ git merge [--ff] branch1  
> Updating 5efe25f..02fdfff  
> Fast-forward  
> C.txt | 1 +  
> 1 file changed, 1 insertion(+)  
> create mode 100644 C.txt  
  
$ git merge --ff-only  
$ git merge --no-ff
```



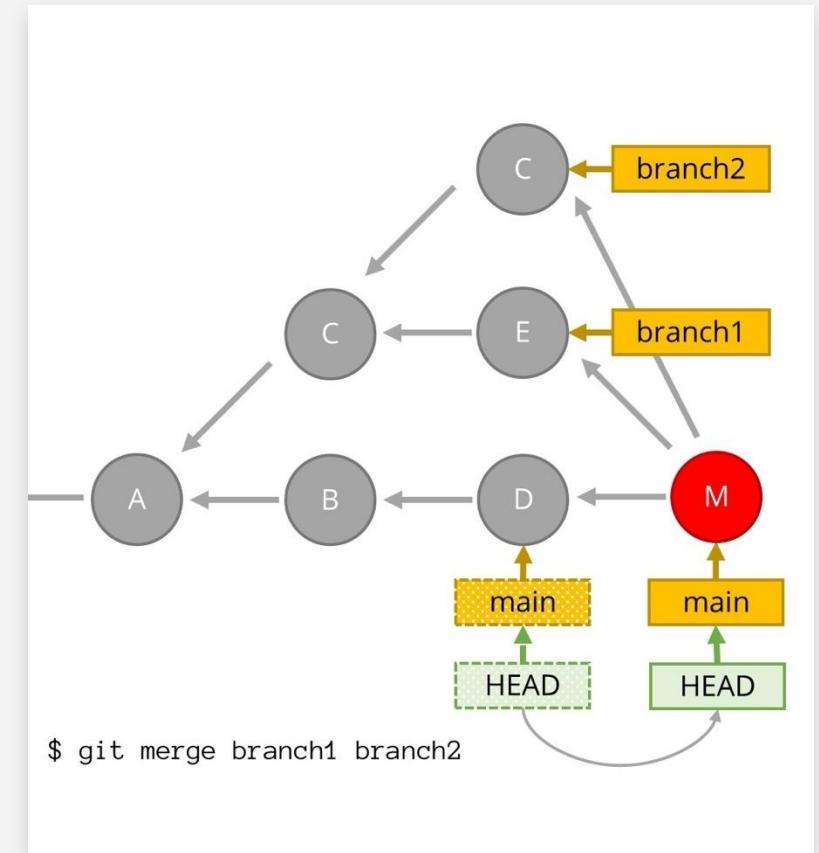
# Merge commit

```
$ git merge branch1
> Merge made by the 'recursive' strategy.
> C.txt | 2 ++
> 1 file changed, 2 insertions(+)
> create mode 100644 C.txt
```



# Octopus merge

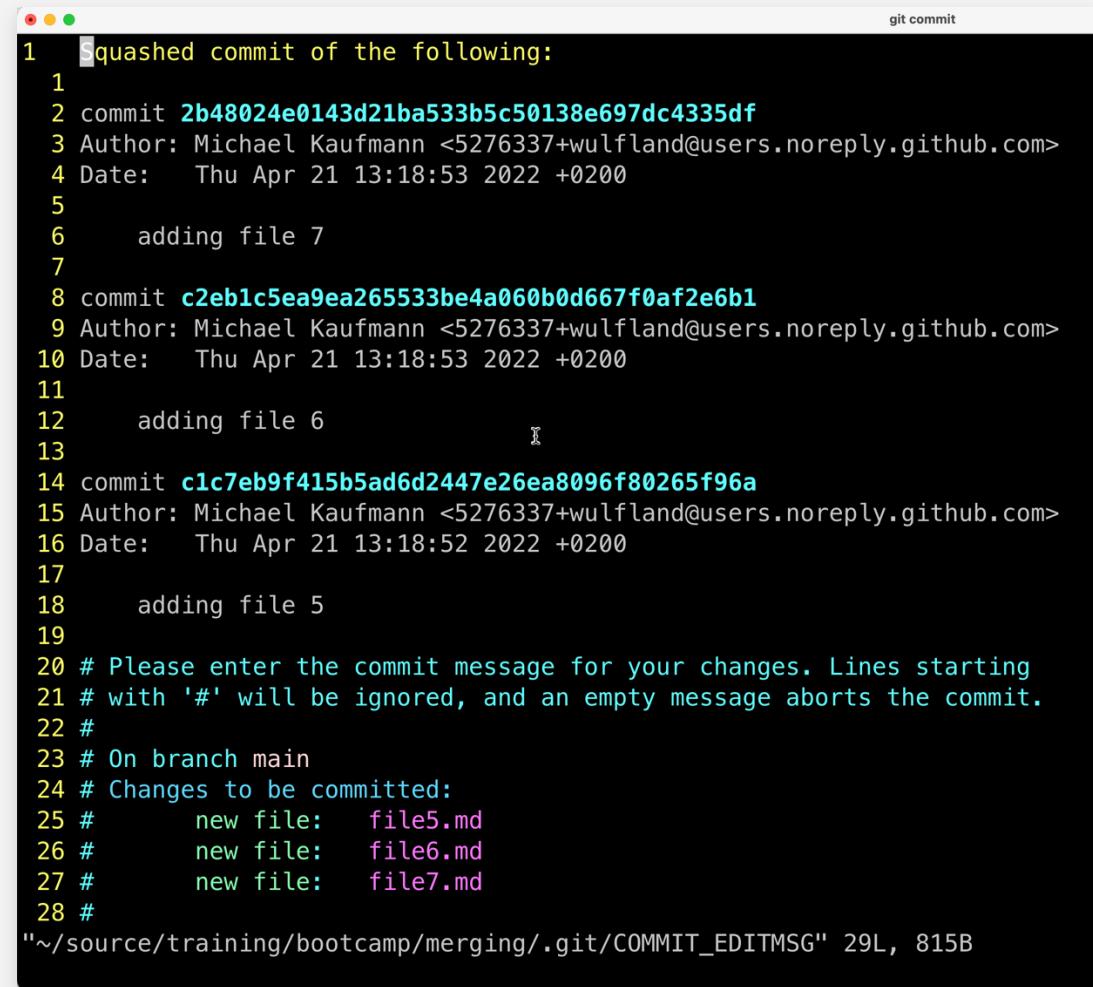
```
$ git merge branch1 branch2
> Trying simple merge with branch1
> Trying simple merge with branch2
> Merge made by the 'octopus' strategy.
>   C.txt | 2 ++
>   F.txt | 1 +
> 2 files changed, 3 insertions(+)
> create mode 100644 C.txt
> create mode 100644 F.txt
```



```
m.kaufmann@Michaels-MacBook-Pro: ~/source/repos/Merging
$ git log --oneline --graph
*-- 6fc5c3a (HEAD -> main) Merge branches 'branch1' and 'branch2' into main
|\ \
| * 6ecc1b9 (branch2) F
| * 1f7deeb (branch1) E
| /
| * eb3f7cf C
* | 48cffc6 D
* | 2bb6ada B
|/
* d781579 (tag: A) A
$ git log --oneline --graph
```

# Squash merge

```
$ git merge --squash <branch>  
$ git commit
```

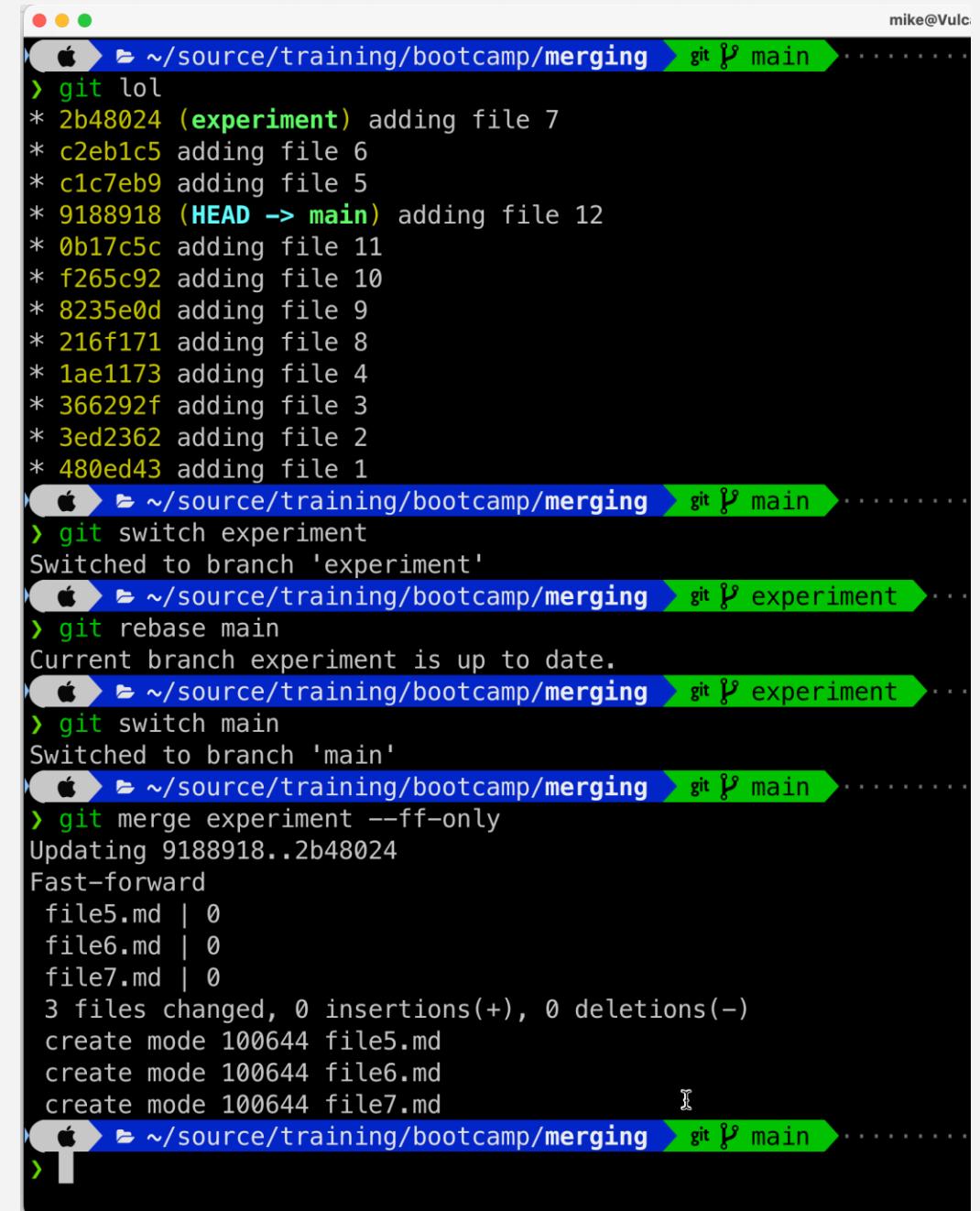


The screenshot shows a terminal window titled "git commit". The message content is as follows:

```
1 Squashed commit of the following:  
2 commit 2b48024e0143d21ba533b5c50138e697dc4335df  
3 Author: Michael Kaufmann <5276337+wulfland@users.noreply.github.com>  
4 Date: Thu Apr 21 13:18:53 2022 +0200  
5  
6     adding file 7  
7  
8 commit c2eb1c5ea9ea265533be4a060b0d667f0af2e6b1  
9 Author: Michael Kaufmann <5276337+wulfland@users.noreply.github.com>  
10 Date: Thu Apr 21 13:18:53 2022 +0200  
11  
12     adding file 6  
13  
14 commit c1c7eb9f415b5ad6d2447e26ea8096f80265f96a  
15 Author: Michael Kaufmann <5276337+wulfland@users.noreply.github.com>  
16 Date: Thu Apr 21 13:18:52 2022 +0200  
17  
18     adding file 5  
19  
20 # Please enter the commit message for your changes. Lines starting  
21 # with '#' will be ignored, and an empty message aborts the commit.  
22 #  
23 # On branch main  
24 # Changes to be committed:  
25 #     new file:   file5.md  
26 #     new file:   file6.md  
27 #     new file:   file7.md  
28 #  
"~/source/training/bootcamp/merging/.git/COMMIT_EDITMSG" 29L, 815B
```

# Rebase and fast-forward merge

```
$ git switch experiment  
$ git rebase main  
$ git switch main  
$ git merge experiment --ff-only  
  
$ git pull --rebase
```



The screenshot shows a terminal window with a dark background and light-colored text. It displays a sequence of git commands and their execution results. The commands are:

- git lol
- \* 2b48024 (experiment) adding file 7
- \* c2eb1c5 adding file 6
- \* c1c7eb9 adding file 5
- \* 9188918 (HEAD -> main) adding file 12
- \* 0b17c5c adding file 11
- \* f265c92 adding file 10
- \* 8235e0d adding file 9
- \* 216f171 adding file 8
- \* 1ae1173 adding file 4
- \* 366292f adding file 3
- \* 3ed2362 adding file 2
- \* 480ed43 adding file 1

Switched to branch 'experiment'

Current branch experiment is up to date.

Switched to branch 'main'

Updating 9188918..2b48024

Fast-forward

file5.md | 0

file6.md | 0

file7.md | 0

3 files changed, 0 insertions(+), 0 deletions(-)

create mode 100644 file5.md

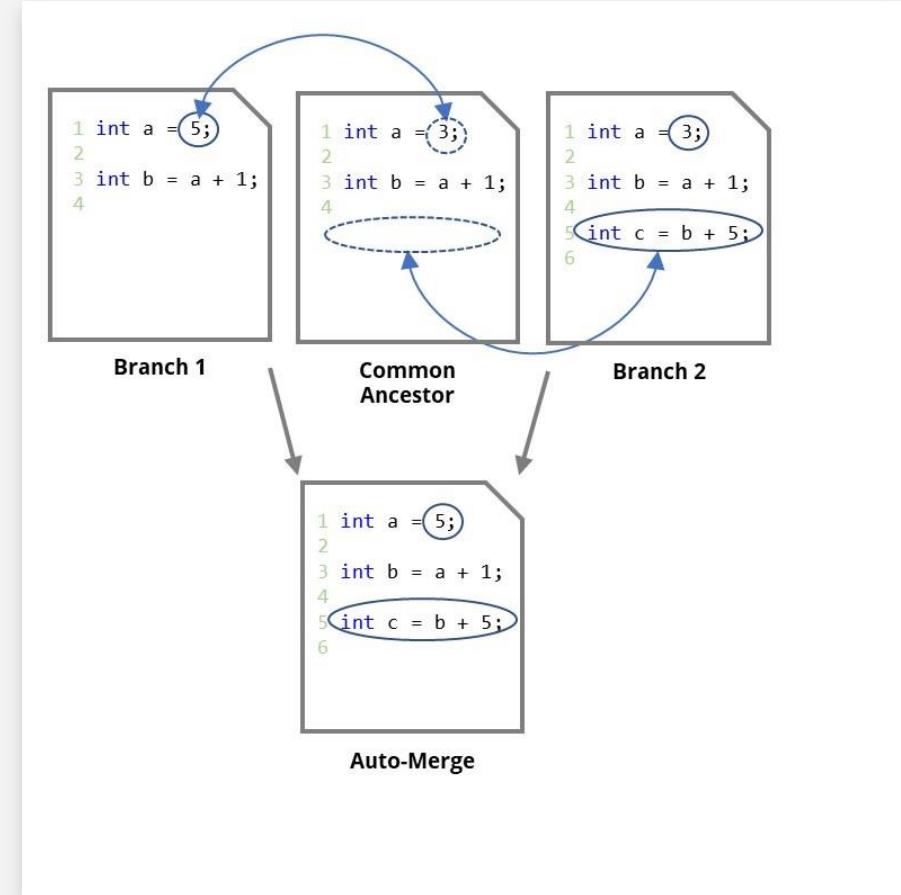
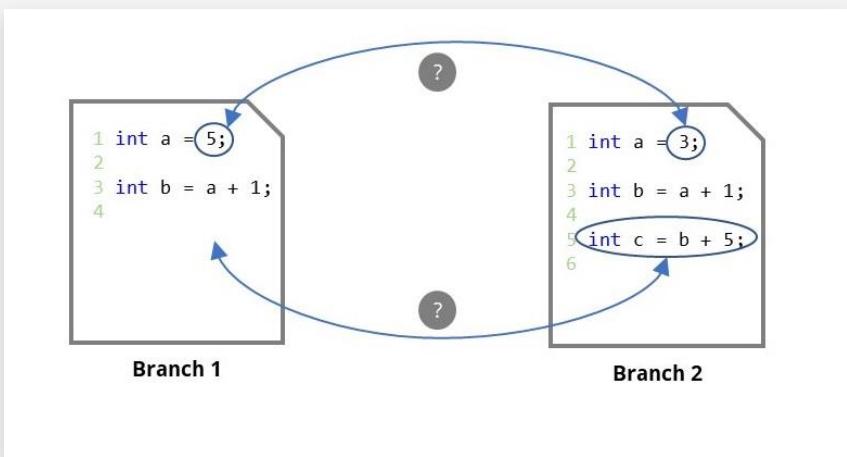
create mode 100644 file6.md

create mode 100644 file7.md

# Resolving merge conflicts

# Three-way merge

- Three-way merge = using the common ancestor



# Resolving merge conflicts

- > Auto-merging Merge.txt
  - > CONFLICT (content): Merge conflict in Merge.txt
  - > Automatic merge failed; fix conflicts and then commit the result.
- > <<<<<, =====, and >>>>>
- <<<<<
- HEAD (ours)
- =====
- MERGE\_HEAD (incoming, theirs)
- >>>>>

```
mike@Vulcan:~/source/training/bootcamp/conflicts git [?] experiment
```

```
> git merge main
Auto-merging Merge.txt
CONFLICT (content): Merge conflict in Merge.txt
Automatic merge failed; fix conflicts and then commit the result.
<<<<< HEAD
It is easy to solve merge conflicts in git
=====
In git it is easy to solve merge conflicts
>>>>> main
> git diff
```

---

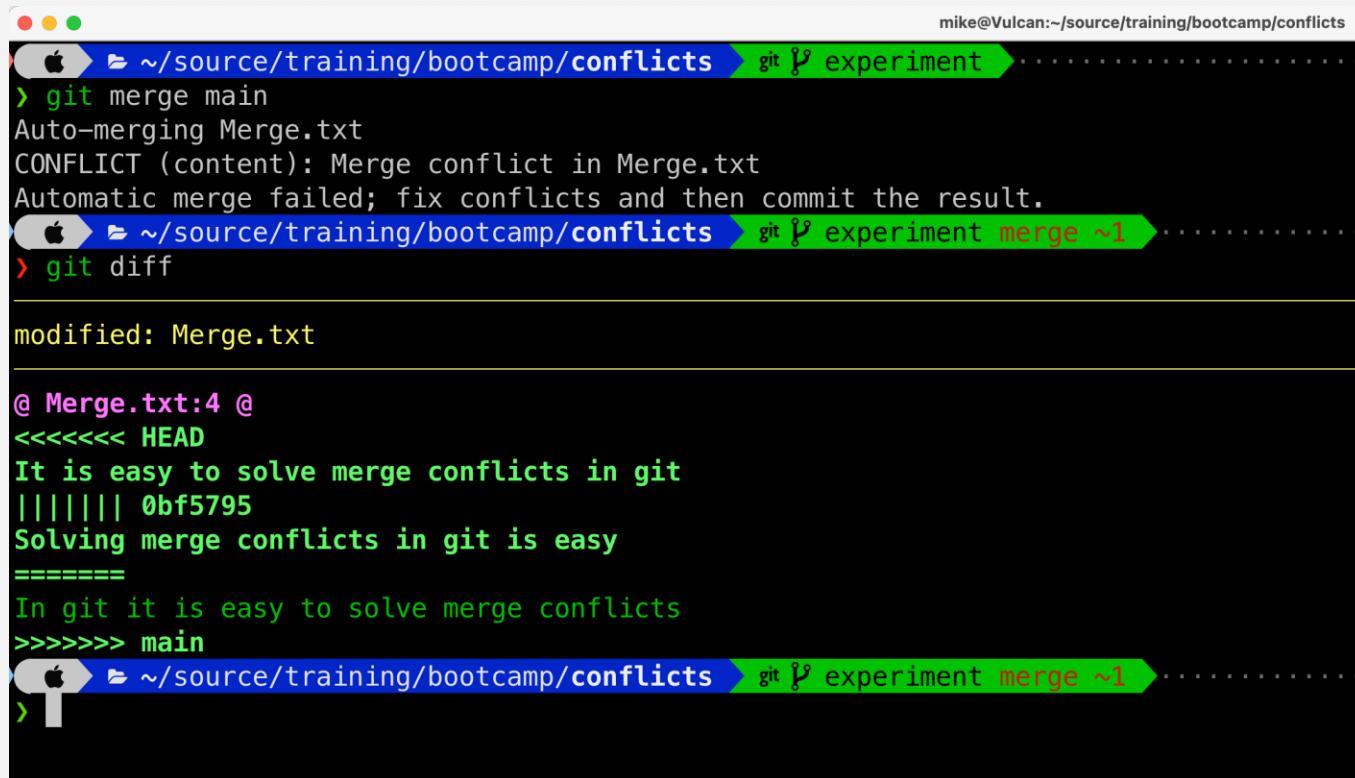
```
modified: Merge.txt
```

```
@ Merge.txt:2 @
<<<<< HEAD
It is easy to solve merge conflicts in git
=====
In git it is easy to solve merge conflicts
>>>>> main
> git diff
```

```
mike@Vulcan:~/source/training/bootcamp/conflicts git [?] experiment merge ~1
```

# Resolving merge conflicts

```
$ git config --global merge.conflictStyle diff3
```



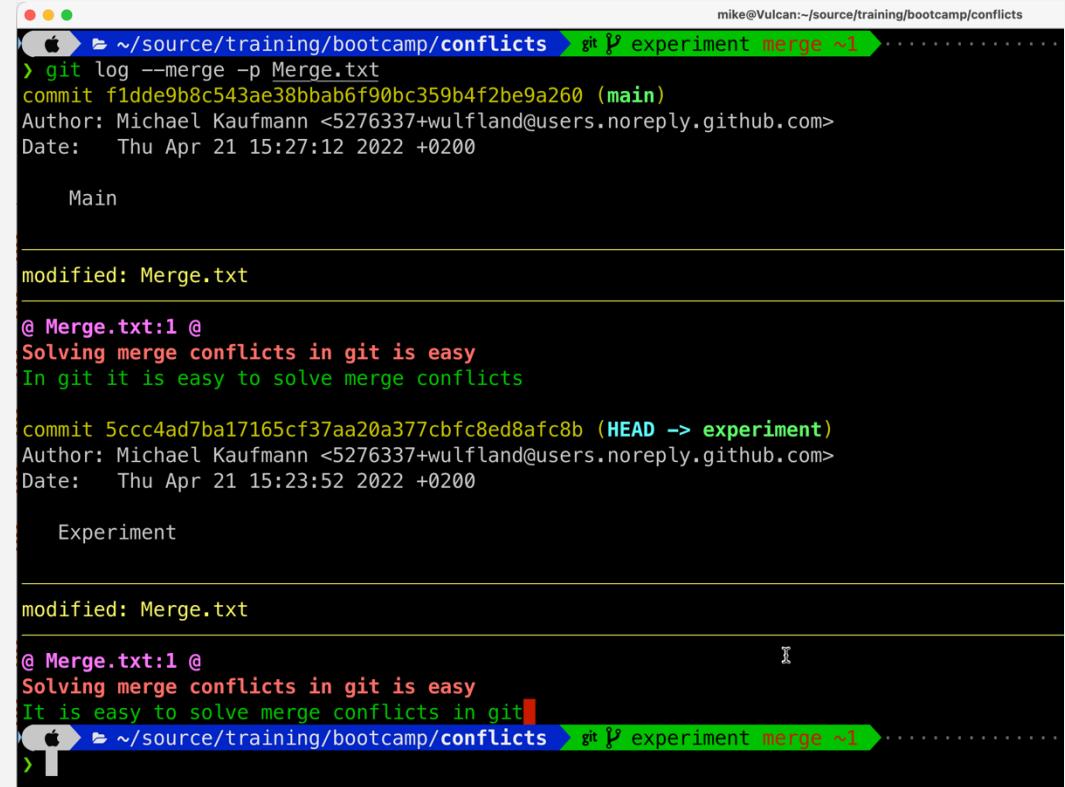
mike@Vulcan:~/source/training/bootcamp/conflicts

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

@ Merge.txt:4 @
<<<<< HEAD
It is easy to solve merge conflicts in git
||||||| 0bf5795
Solving merge conflicts in git is easy
=====
In git it is easy to solve merge conflicts
>>>>> main
```

# Resolving merge conflicts

```
$ git merge --abort  
$ git diff  
$ git log --merge -p <filename>  
$ git show :1:<filename> (common ancestor)  
$ git show :2:<filename> (HEAD)  
$ git show :3:<filename> (MERGE_HEAD)  
  
$ git add <filename>  
$ git merge --continue
```



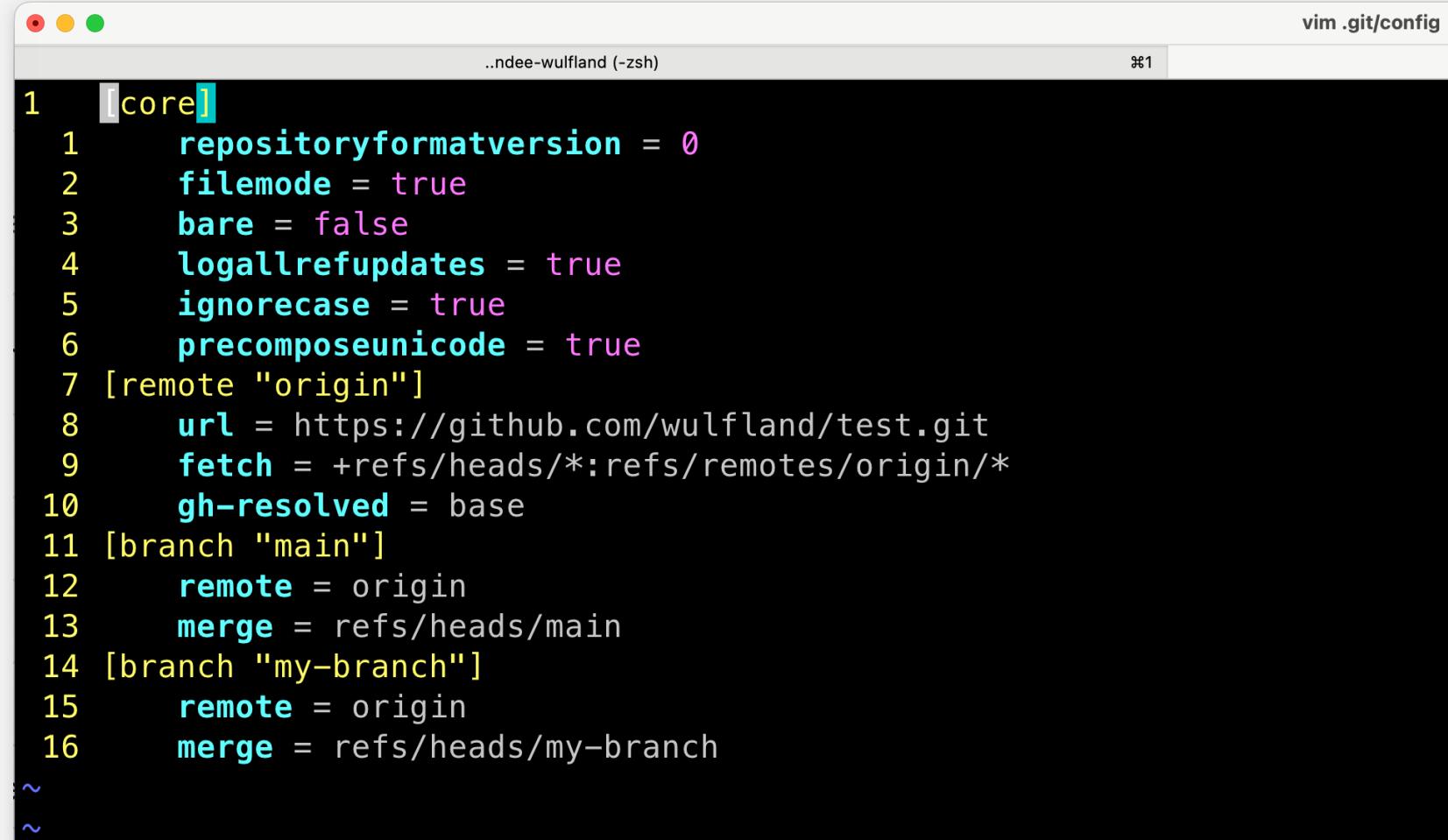
mike@Vulcan:~/source/training/bootcamp/conflicts

```
git log --merge -p Merge.txt  
commit f1dde9b8c543ae38bbab6f90bc359b4f2be9a260 (main)  
Author: Michael Kaufmann <5276337+wulfland@users.noreply.github.com>  
Date: Thu Apr 21 15:27:12 2022 +0200  
  
Main  
  
modified: Merge.txt  
  
@ Merge.txt:1 @  
Solving merge conflicts in git is easy  
In git it is easy to solve merge conflicts  
  
commit 5ccc4ad7ba17165cf37aa20a377cbfc8ed8afc8b (HEAD -> experiment)  
Author: Michael Kaufmann <5276337+wulfland@users.noreply.github.com>  
Date: Thu Apr 21 15:23:52 2022 +0200  
  
Experiment  
  
modified: Merge.txt  
  
@ Merge.txt:1 @  
Solving merge conflicts in git is easy  
It is easy to solve merge conflicts in git  
|
```

# Remotes

# Remotes

```
$ git remote add  
$ git push -u  
$ git branch -vv
```

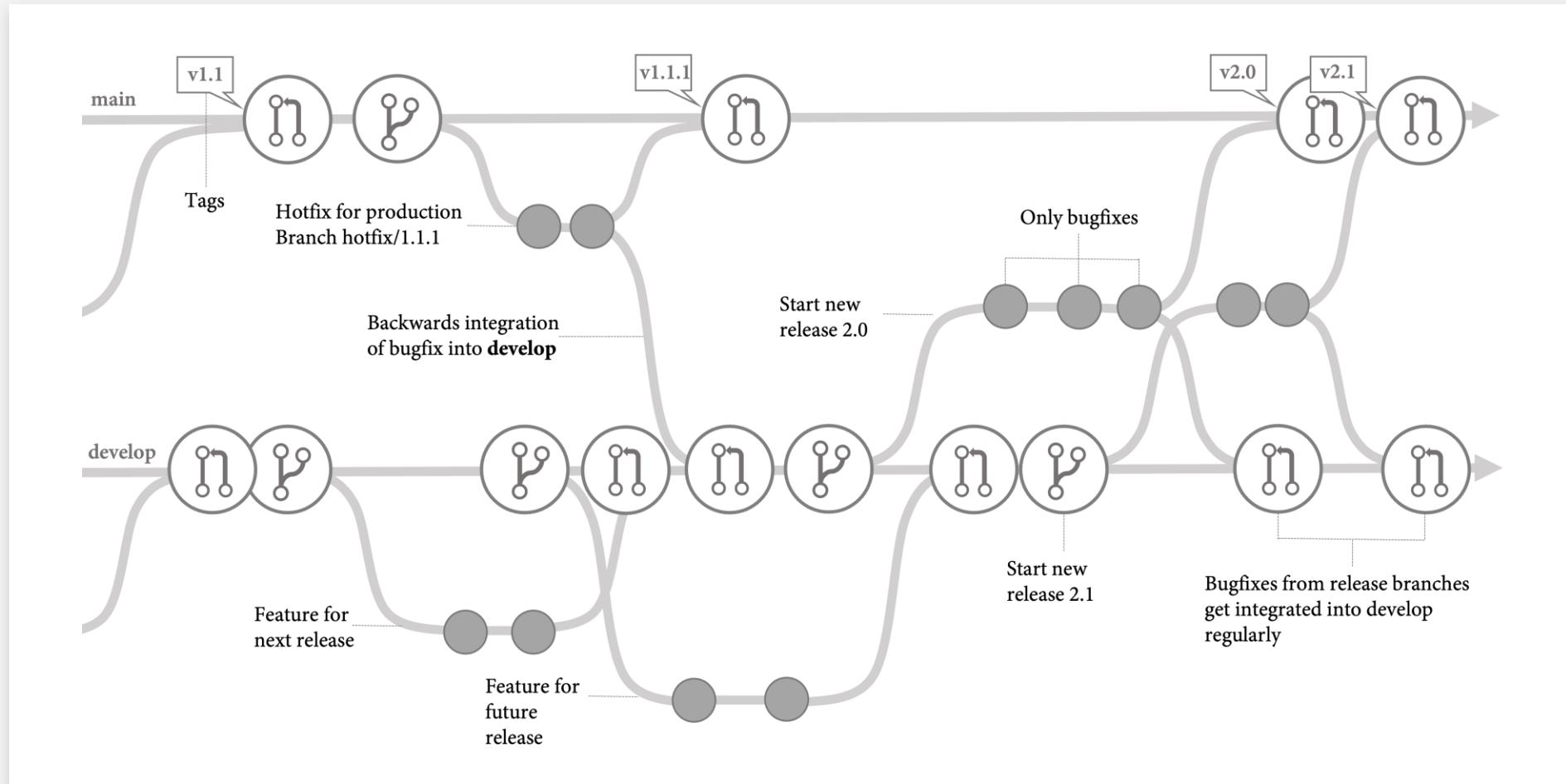


A screenshot of a terminal window titled "vim .git/config". The window shows the contents of the .git/config file. The file is a configuration file for a Git repository, containing sections for [core], [remote "origin"], [branch "main"], and [branch "my-branch"]. The [core] section contains various repository settings. The [remote "origin"] section specifies the URL and fetch strategy. The [branch] sections define the merge strategy for the main and my-branch branches.

```
..ndee-wulfland (-zsh)  
vim .git/config  
1 [core]  
1   repositoryformatversion = 0  
2   filemode = true  
3   bare = false  
4   logallrefupdates = true  
5   ignorecase = true  
6   precomposeunicode = true  
7 [remote "origin"]  
8   url = https://github.com/wulfland/test.git  
9   fetch = +refs/heads/*:refs/remotes/origin/*  
10  gh-resolved = base  
11 [branch "main"]  
12   remote = origin  
13   merge = refs/heads/main  
14 [branch "my-branch"]  
15   remote = origin  
16   merge = refs/heads/my-branch  
~  
~
```

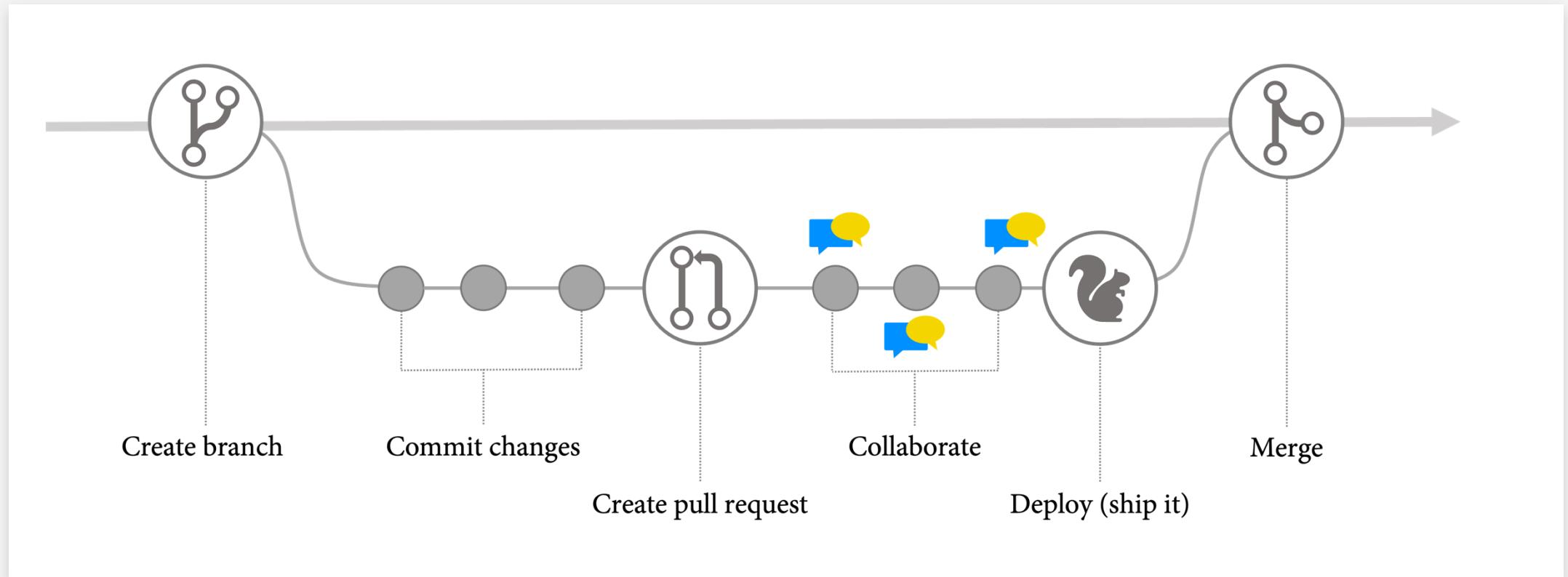
# Branching workflows

- › **Git flow (<https://nvie.com/posts/a-successful-git-branching-model>)**



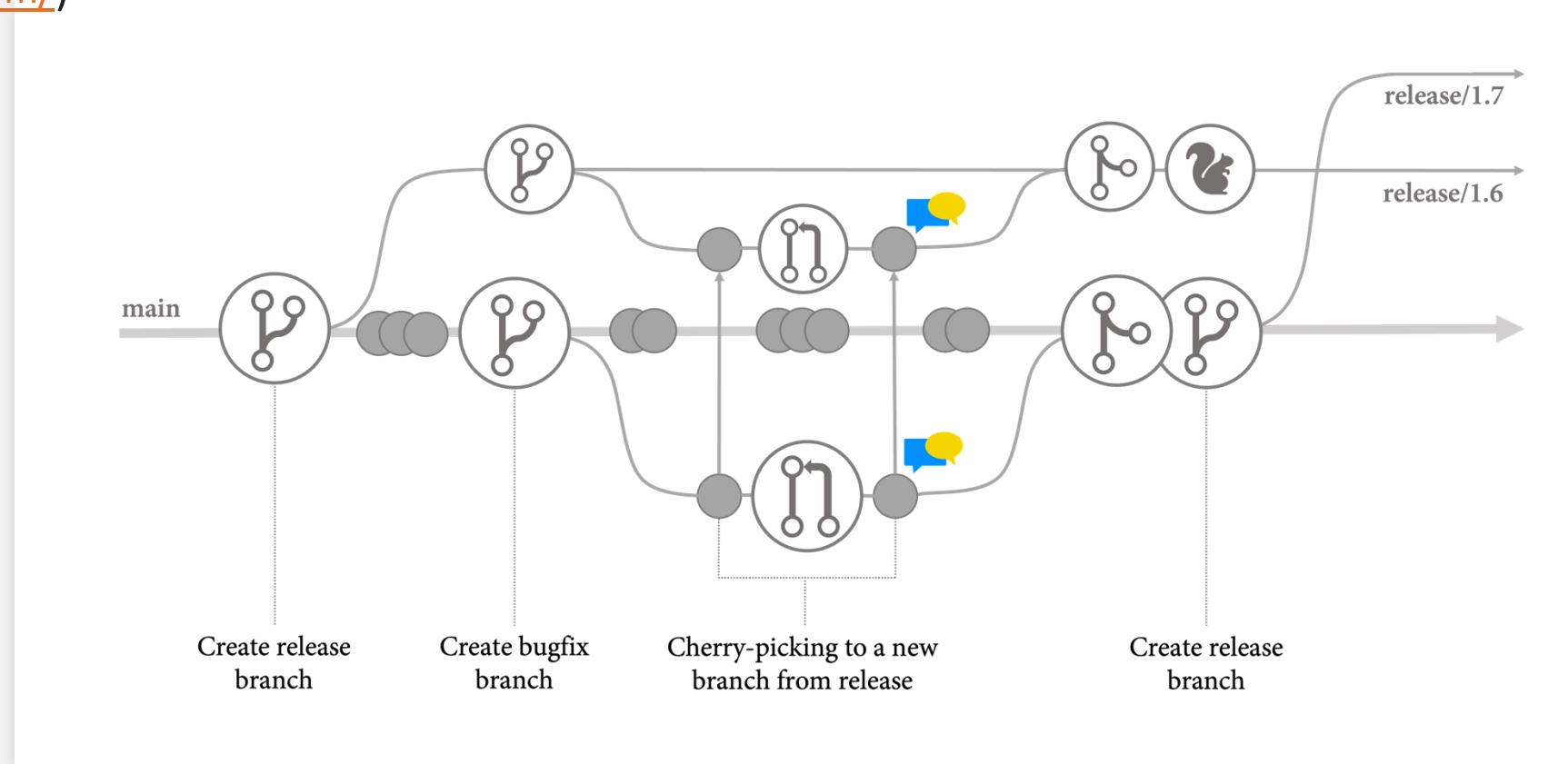
# Branching workflows

- › GitHub flow (trunk-based workflow, <https://docs.github.com/en/get-started/quickstart/github-flow>)



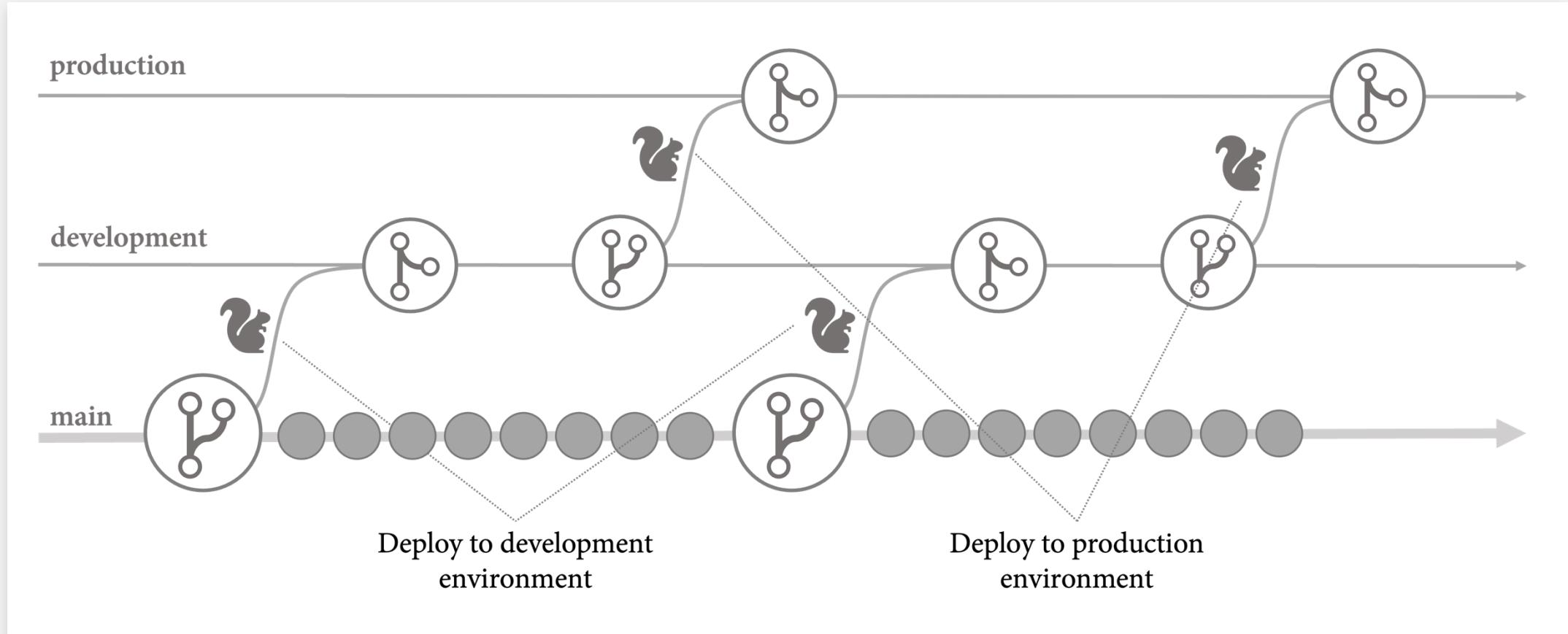
# Branching workflows

- Release flow (<https://devblogs.microsoft.com/devops/release-flow-how-we-do-branching-on-the-vsts-team/>)



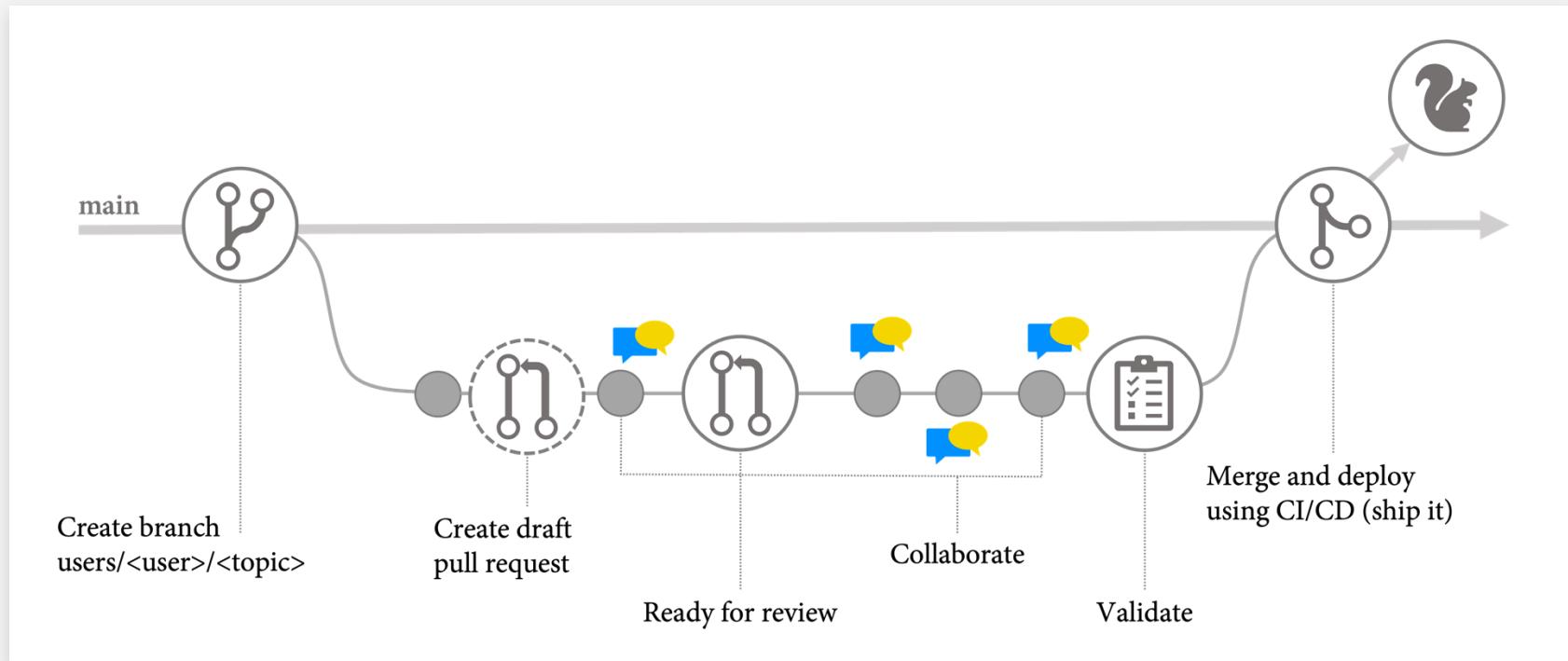
# Branching workflows

- › **GitLab flow** ([https://docs.gitlab.com/ee/topics/gitlab\\_flow.html](https://docs.gitlab.com/ee/topics/gitlab_flow.html))



# Branching workflows

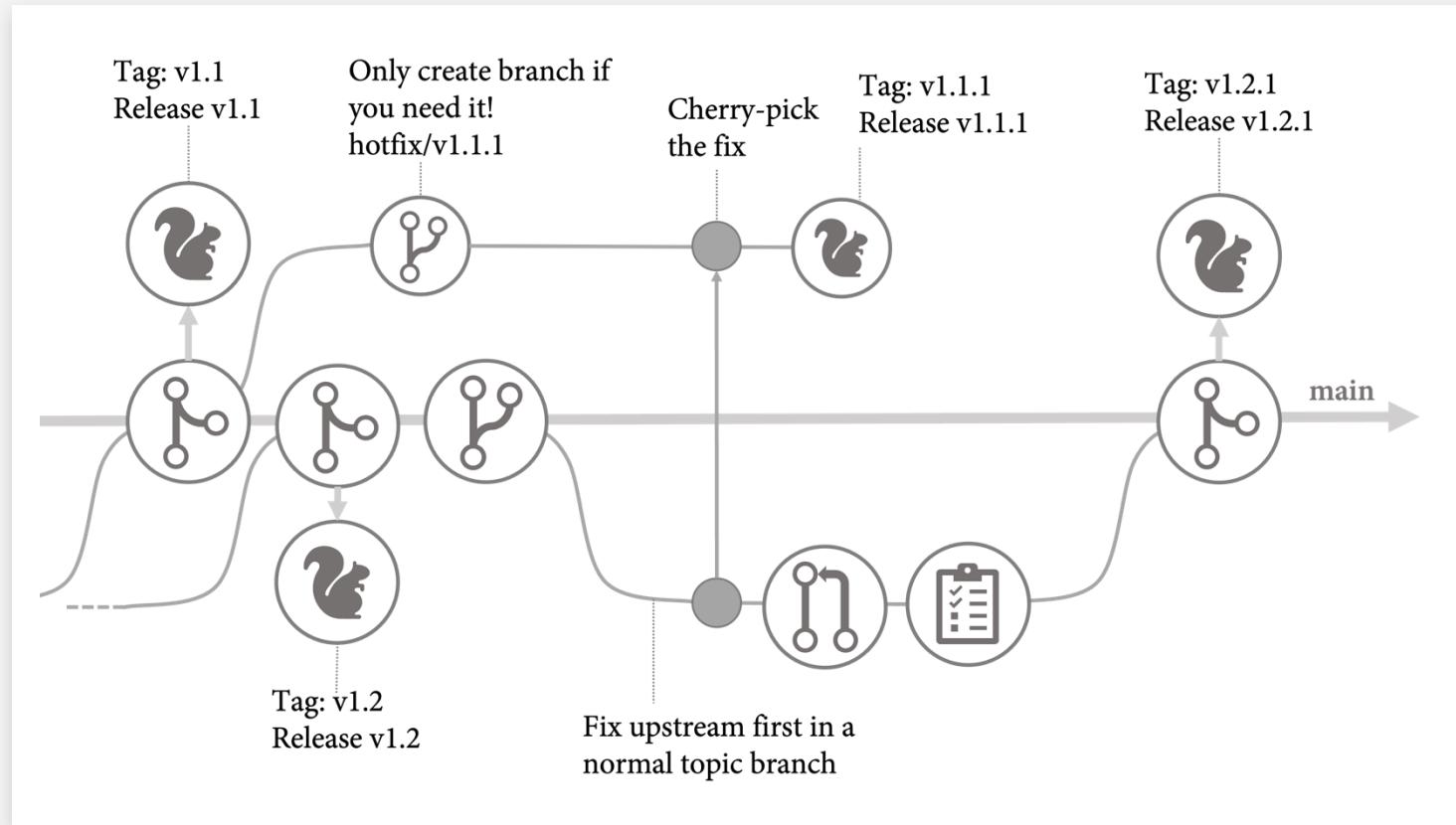
- › **MyFlow** (<https://wulfland.github.io/MyFlow/>)
  - › Trunk-based (main, branch protection, CODEOWNERS)
  - › Private topic branches (`users/<username>/<id>_<topic>`)
  - › (Draft) pull requests, auto merge, `git push -f` / `git push origin +<branch-name>`



# Branching workflows

- › **MyFlow** (<https://wulfland.github.io/MyFlow/>)

- › Releases / Tags / cherry-pick

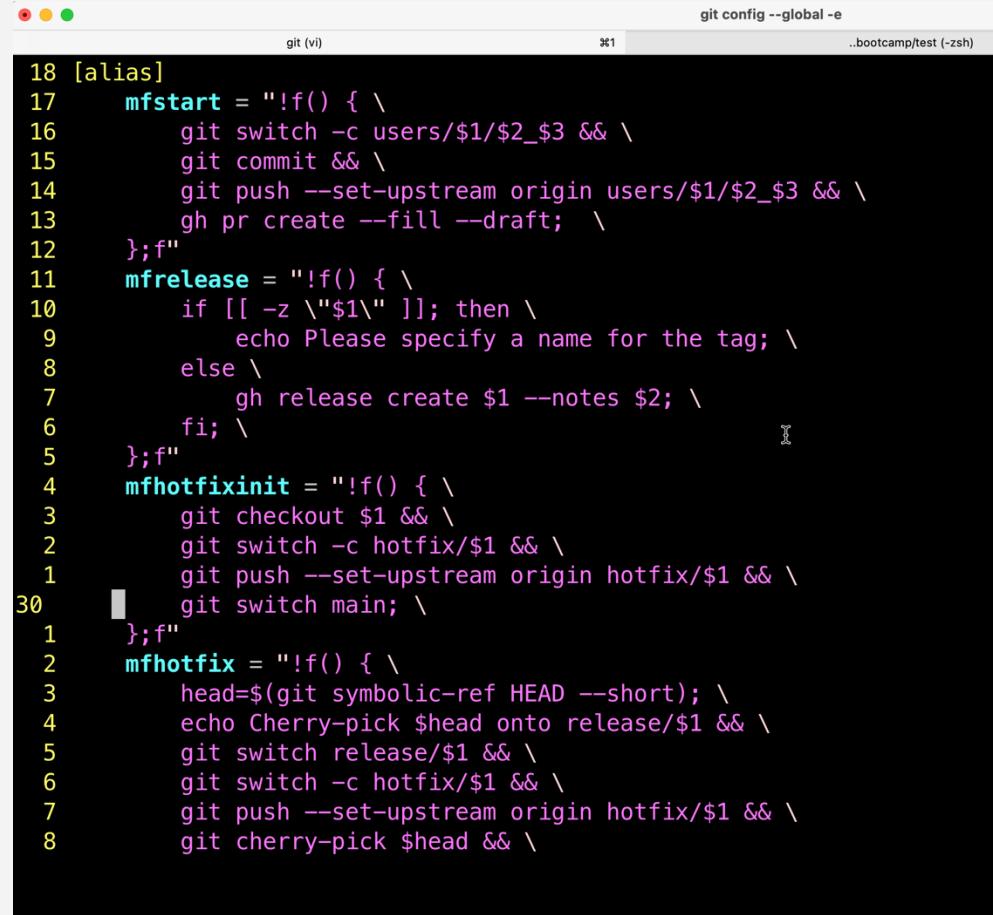


# Git aliases

# Git aliases

```
$ git config --global alias.last "log -1"  
$ git config --global alias.lol "log --oneline --graph --decorate --all"  
$ git config --global -e
```

- › Functions "!f(){ ... };f"
- › \$1, \$2, \$3, ... \$@
- › Line endings: \
- › && or ; to separate commands



A screenshot of a terminal window titled "git config --global -e ..bootcamp/test (-zsh)". The window displays a portion of a .gitconfig file. The code uses shell functions to define aliases for git operations. It includes functions for switching branches, committing, pushing, creating pull requests, specifying release tags, initializing hotfix branches, and performing cherry-picks.

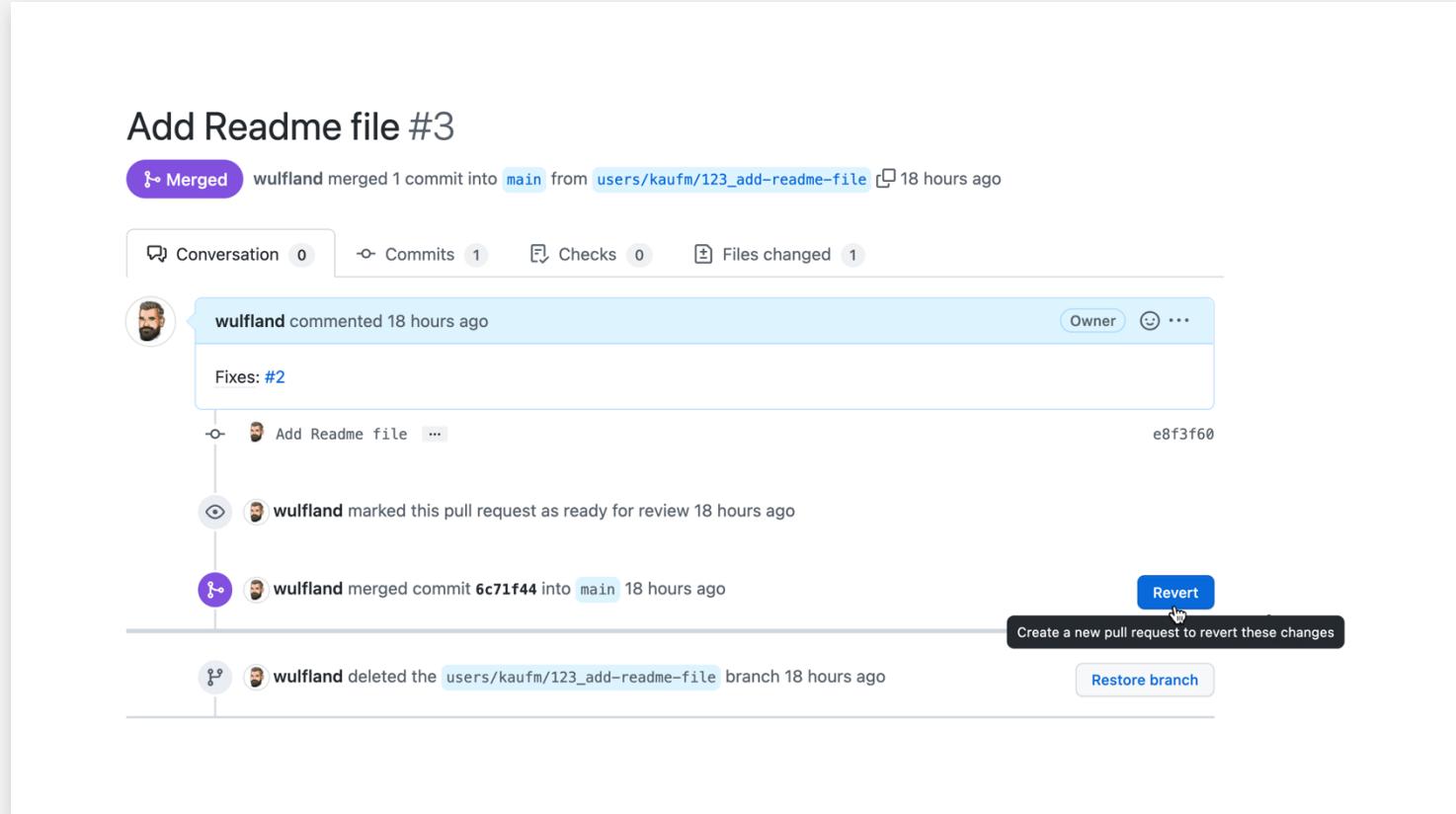
```
18 [alias]  
17     mfstart = "!f() { \  
16         git switch -c users/$1/$2_$3 && \  
15         git commit && \  
14         git push --set-upstream origin users/$1/$2_$3 && \  
13         gh pr create --fill --draft; \  
12     };f"  
11     mfrelease = "!f() { \  
10         if [[ -z \"$1\" ]]; then \  
9             echo Please specify a name for the tag; \  
8             else \  
7                 gh release create $1 --notes $2; \  
6                 fi; \  
5     };f"  
4     mfhfixinit = "!f() { \  
3         git checkout $1 && \  
2         git switch -c hotfix/$1 && \  
1         git push --set-upstream origin hotfix/$1 && \  
30         git switch main; \  
1     };f"  
2     mfhfix = "!f() { \  
3         head=$(git symbolic-ref HEAD --short); \  
4         echo Cherry-pick $head onto release/$1 && \  
5         git switch release/$1 && \  
6         git switch -c hotfix/$1 && \  
7         git push --set-upstream origin hotfix/$1 && \  
8         git cherry-pick $head && \  
9     };f"
```

# Reverting changes

# Reverting changes

Undo changes in a safe way

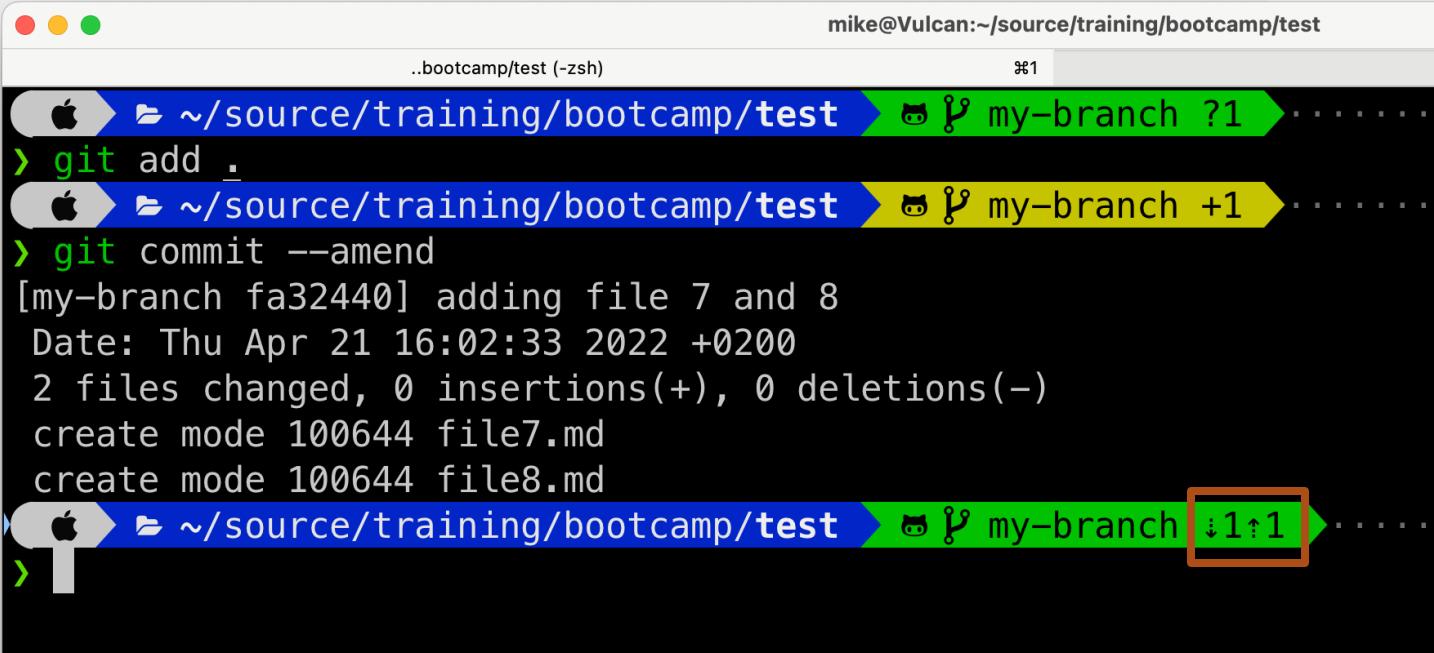
\$ git revert <REF>



# Ammending changes

# Ammending changes

```
$ git commit --amend  
$ git push --force
```



The screenshot shows a macOS terminal window with the following session:

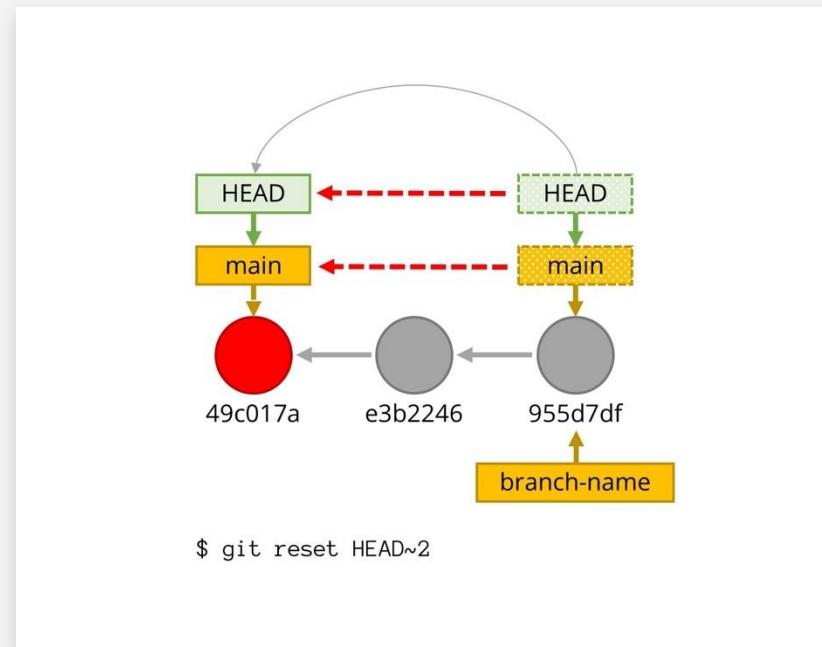
```
mike@Vulcan:~/source/training/bootcamp/test  
..bootcamp/test (-zsh) 361  
❯ git add .  
❯ git commit --amend  
[my-branch fa32440] adding file 7 and 8  
Date: Thu Apr 21 16:02:33 2022 +0200  
2 files changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 file7.md  
create mode 100644 file8.md  
❯
```

A yellow box highlights the command `git commit --amend`. A red box highlights the status bar at the bottom right of the terminal window, which shows `my-branch ↓1↑1`.

# Reseting your repository

# Resetting your repository

- › Moving the current branch to another position



# Resetting your repository

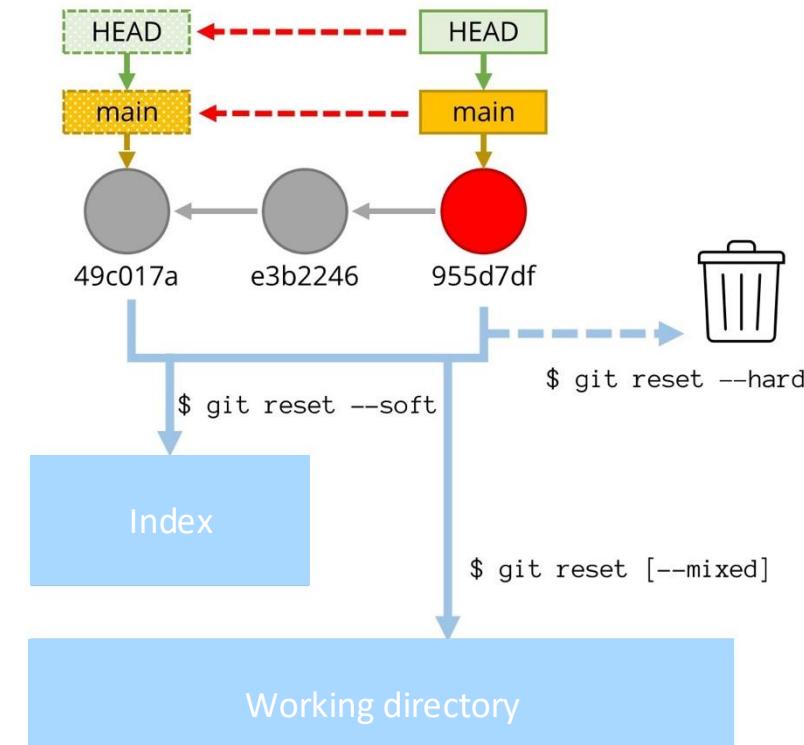
- › What happens with the changes?

```
$ git reset [--mixed]
```

```
$ git reset --hard
```

```
$ git reset --soft
```

```
$ git reset --merge | --keep
```

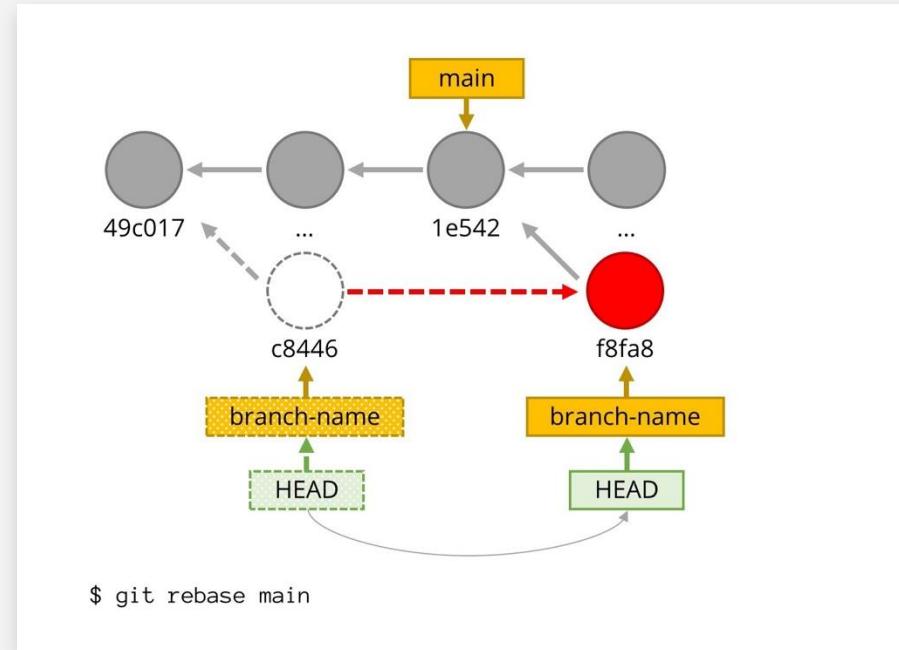


# Rebase

# Rebase

› Rebase

```
$ git rebase <REF>  
$ git rebase -I  
$ git commit --amend  
$ git rebase --continue  
$ git rebase --abort
```



# Cherry-picking and the reflog

# Git RefLog

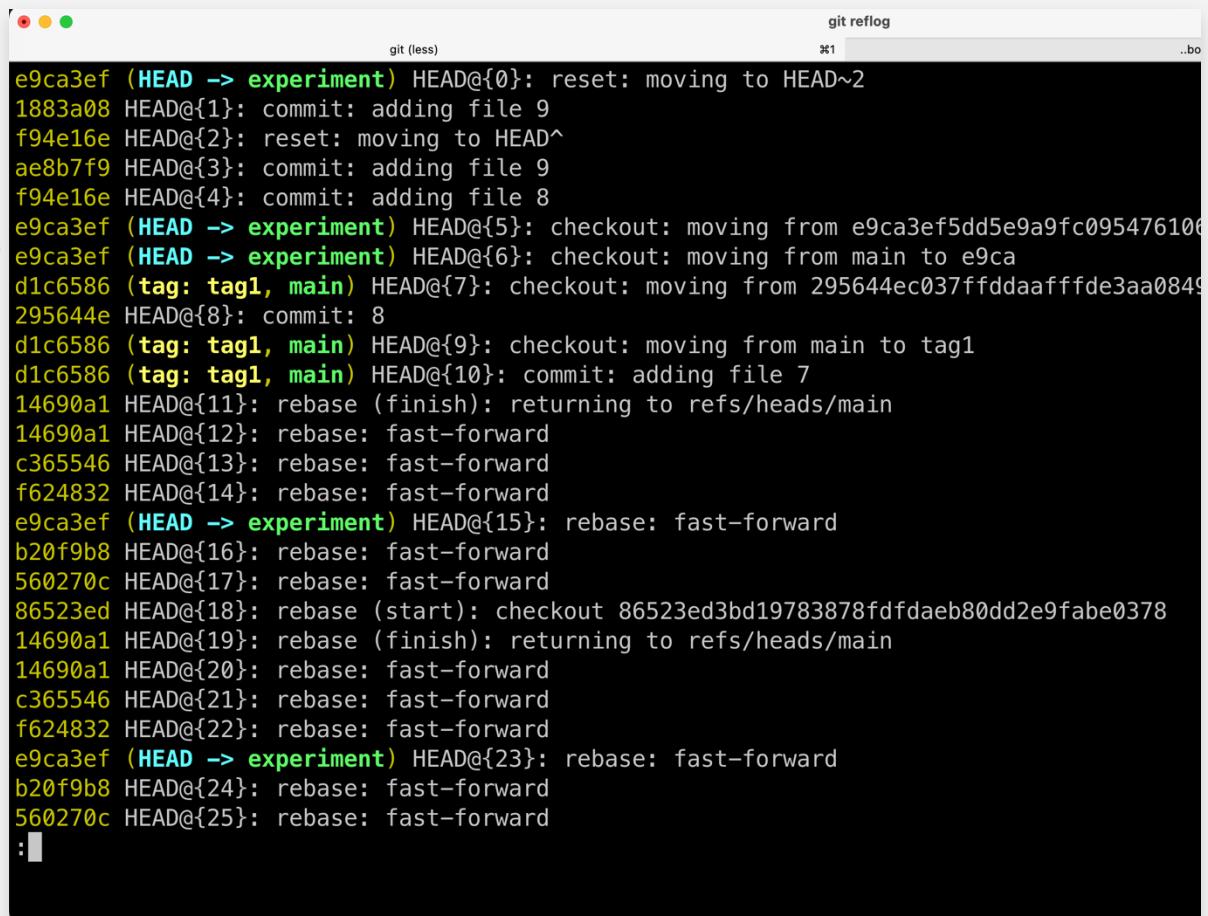
- › The RefLog is **local!**
- › Expires:
  - › `gc.reflogExpireUnreachable` (default **30 days**)
  - › `gc.reflogExpire` (default **90 days**)

```
$ git reflog [show]
```

```
$ git cherry-pick <REF>
```

```
$ git cherry-pick --edit <REF>
```

```
$ git cherry-pick -no-commit <REF>
```



The screenshot shows a terminal window titled "git reflog" with the command "git (less)" running. The output displays a log of Git operations:

```
e9ca3ef (HEAD -> experiment) HEAD@{0}: reset: moving to HEAD~2
1883a08 HEAD@{1}: commit: adding file 9
f94e16e HEAD@{2}: reset: moving to HEAD^
ae8b7f9 HEAD@{3}: commit: adding file 9
f94e16e HEAD@{4}: commit: adding file 8
e9ca3ef (HEAD -> experiment) HEAD@{5}: checkout: moving from e9ca3ef5dd5e9a9fc095476106
e9ca3ef (HEAD -> experiment) HEAD@{6}: checkout: moving from main to e9ca
d1c6586 (tag: tag1, main) HEAD@{7}: checkout: moving from 295644ec037ffddaffde3aa0849
295644e HEAD@{8}: commit: 8
d1c6586 (tag: tag1, main) HEAD@{9}: checkout: moving from main to tag1
d1c6586 (tag: tag1, main) HEAD@{10}: commit: adding file 7
14690a1 HEAD@{11}: rebase (finish): returning to refs/heads/main
14690a1 HEAD@{12}: rebase: fast-forward
c365546 HEAD@{13}: rebase: fast-forward
f624832 HEAD@{14}: rebase: fast-forward
e9ca3ef (HEAD -> experiment) HEAD@{15}: rebase: fast-forward
b20f9b8 HEAD@{16}: rebase: fast-forward
560270c HEAD@{17}: rebase: fast-forward
86523ed HEAD@{18}: rebase (start): checkout 86523ed3bd19783878fdfdaeb80dd2e9fabe0378
14690a1 HEAD@{19}: rebase (finish): returning to refs/heads/main
14690a1 HEAD@{20}: rebase: fast-forward
c365546 HEAD@{21}: rebase: fast-forward
f624832 HEAD@{22}: rebase: fast-forward
e9ca3ef (HEAD -> experiment) HEAD@{23}: rebase: fast-forward
b20f9b8 HEAD@{24}: rebase: fast-forward
560270c HEAD@{25}: rebase: fast-forward
```

# The stash

# The stash

› LIFO: Last In – First Out

```
$ git stash [push]
```

```
$ git stash push -m "<message>"
```

```
$ git stash list
```

```
$ git stash pop [--index]
```

```
$ git stash branch <name>
```

The terminal window shows a session on a Mac OS X system (Vulcan) with the command-line interface zsh. The user is in a directory named 'localGit' under 'source/training/bootcamp'. The session demonstrates the use of the 'git stash' command:

- The user runs `git stash list`, which shows three stashes:
  - stash@{0}: On experiment: adding file 4
  - stash@{1}: On experiment: adding patch 2
  - stash@{2}: WIP on experiment: e9ca3ef adding file 3
- The user adds a new file, `file5.md`, to the working directory.
- The user runs `git add file5.md`.
- The user runs `git stash push -m "adding file 5"`. This saves the current state of the working directory and index under a new stash entry, stash@{3}.
- The user runs `git stash list` again, showing four stashes:
  - stash@{0}: On experiment: adding file 5
  - stash@{1}: On experiment: adding file 4
  - stash@{2}: On experiment: adding patch 2
  - stash@{3}: WIP on experiment: e9ca3ef adding file 3
- The user runs `git stash pop` (implied by the context), which applies the changes from stash@{3} back to the working directory.
- The user runs `ls` to see the files: `file1.md`, `file2.md`, and `file3.md`.
- The user runs `git stash list` again, showing three stashes:
  - stash@{0}: On experiment: adding file 5
  - stash@{1}: On experiment: adding file 4
  - stash@{2}: WIP on experiment: e9ca3ef adding file 3

# Let's connect

---



@mike\_kaufmann



@wulfland



<https://writeabout.net>



Let's connect!



Michael  
Kaufmann

Managing Director Xebia  
Germany  
Microsoft Regional Director,  
MVP

[Linkedin.com/in/mikaufmann](https://www.linkedin.com/in/mikaufmann)  
[mkaufmann@xpirit.com](mailto:mkaufmann@xpirit.com)