

Deep dive into { git }

Michael Kaufmann

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

Getting started

Getting started

- › Local git configs



```
$ 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

```
git config --global --edit
```

```
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   mfixinit = "!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
~
```

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)

```

❯ git init
Initialized empty Git repository in /Users/m.kaufmann/source/repos/UnterDerHaube/.git/
❯ git hash-object txt/Doc.txt
f20acb769ad757b2058c8c3c96e7eac2e1258503
❯ git hash-object txt/Doc.txt
f85de0e273f9ed77d2a37b1dd7397e667b0ad335
❯ git add txt/Doc.txt
❯ git commit -m "Initial commit"
[master (root-commit) 5de0e27] Initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 txt/Doc.txt
❯ cat .git/HEAD
ref: refs/heads/main
DIRC_0*00
0_*0g000000>0]00s00WH,{09~f{
05
txt/Doc.txt0({0700030>0pC0k0J0%
❯ git ls-files --stage
100644 f85de0e273f9ed77d2a37b1dd7397e667b0ad335 0      txt/Doc.txt
❯ cat .git/objects/f8/5de0e273f9ed77d2a37b1dd7397e667b0ad335
xK00R03b0K0M0Rp)0Sp00M-QPr00VRp*0/000/0L00K0Q(0)2000000L0â000000%
❯ git cat-file -t f85de
blob
❯ git cat-file -p f85de
Name: Dr. Emmet "Doc" Brown
Original time: 1985
Nickname: Doc
❯ git commit -m "Add Doc's bio"
[master 5de0e27] Add Doc's bio
 1 file changed, 1 insertion(+)
 create mode 100644 txt/Doc.txt

```

10:56:10 ⓘ 10:56:36 ⓘ 10:57:58 ⓘ 14s ⏺ 10:58:21 ⓘ 10:58:24 ⓘ 10:59:03 ⓘ 11:00:27 ⓘ 11:00:43 ⓘ 11:01:05 ⓘ 11:03:05 ⓘ 11:05:52 ⓘ 11:05:57 ⓘ

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

```
x ..UnterDerHaube (-zsh)
apple ~ % source/repos/UnterDerHaube git ⚡ main
> git cat-file -t fa79
commit
apple ~ % source/repos/UnterDerHaube git ⚡ main
> 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
> git cat-file -t 70f8
tree
apple ~ % source/repos/UnterDerHaube git ⚡ main
> git cat-file -p 70f8
040000 tree ba105a49862bcf36b8ffff1dae703fc5678f7358a    txt
apple ~ % source/repos/UnterDerHaube git ⚡ main
> git ls-tree ba105
100644 blob f85de0e273f9ed77d2a37b1dd7397e667b0ad335   Doc.txt
apple ~ % source/repos/UnterDerHaube git ⚡ main
> cat .git/refs/heads/main
fa794f016773b66bed29596a0cec389295a5d758
apple ~ % source/repos/UnterDerHaube git ⚡ main
>

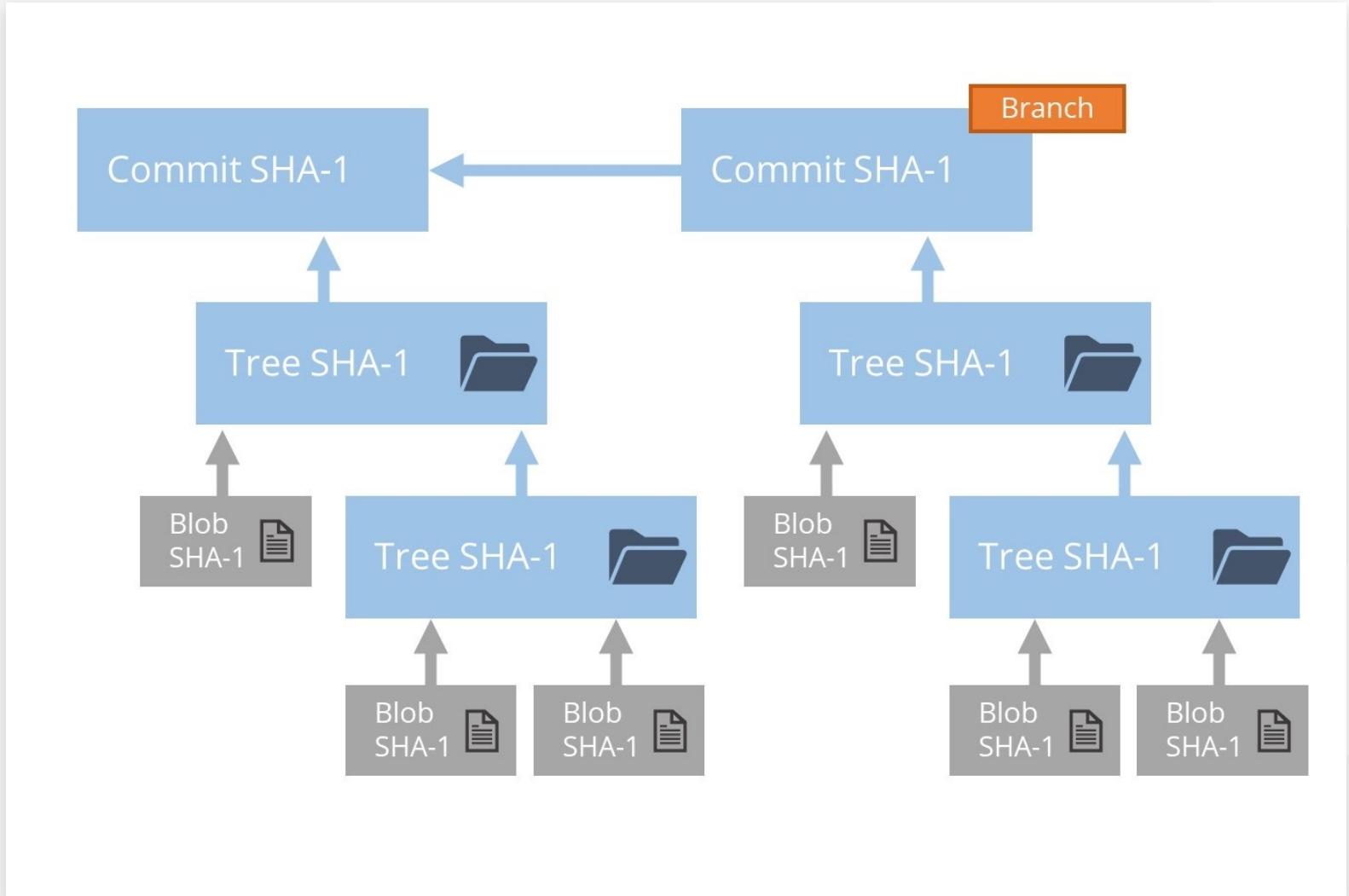
11:09:37 ⓘ
11:09:42 ⓘ
11:09:46 ⓘ
11:10:57 ⓘ
11:11:01 ⓘ
11:11:19 ⓘ
11:12:48 ⓘ
```

Every 2.0s... Andoria.local: Wed Nov 25 11:13:11 2020

```
.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

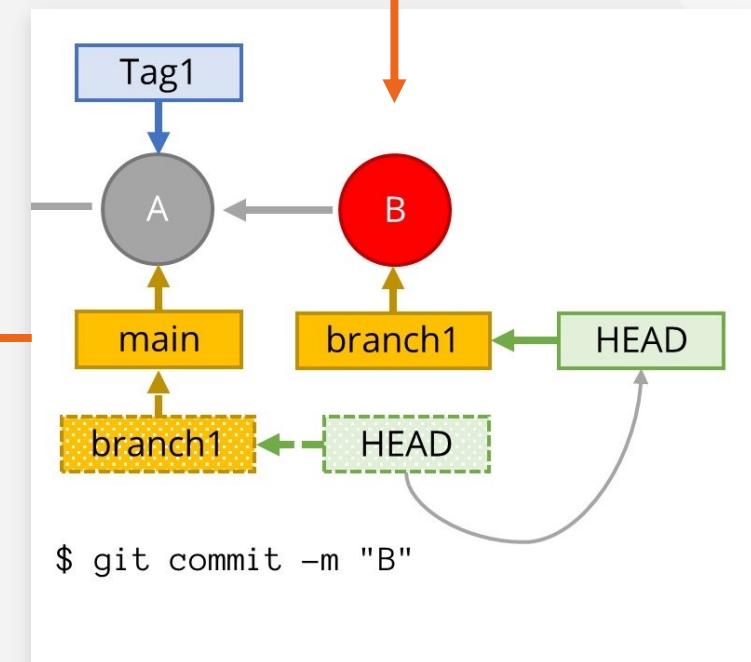
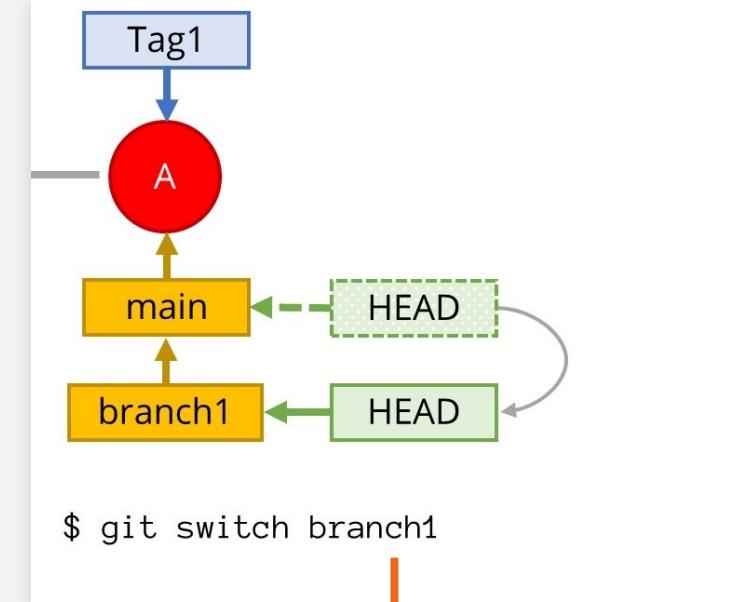
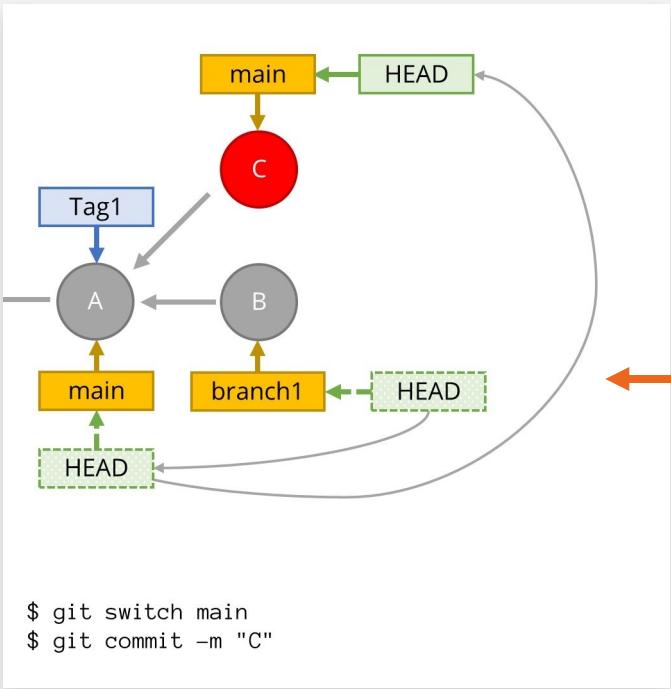
Everything is a rooted tree



Branches and tags

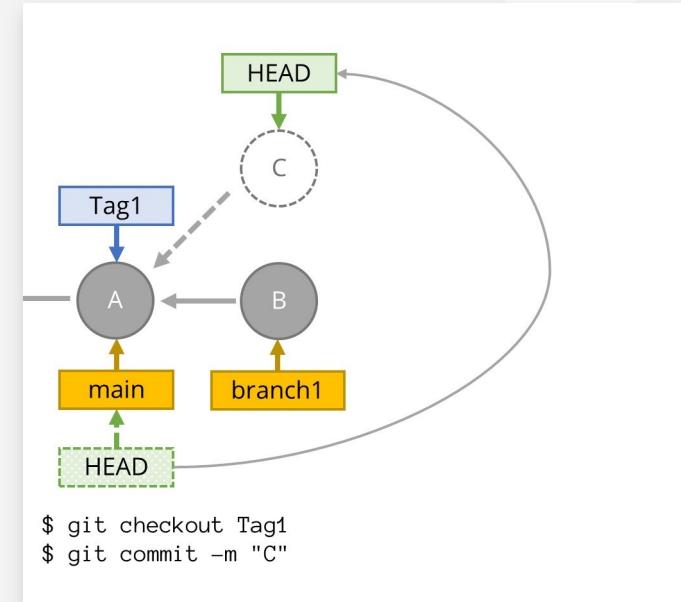
Branches

- › 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



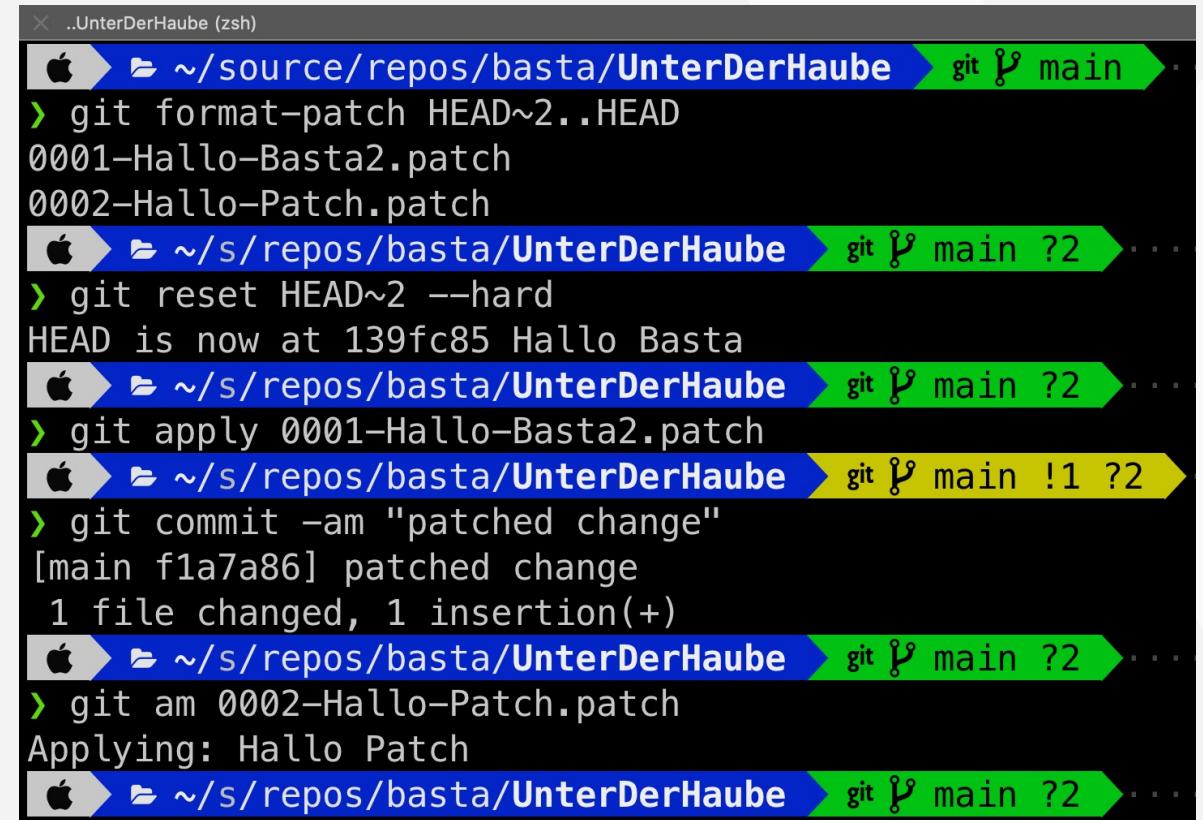
Referencing commits

- > **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**)

Diffs 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 patches for commits `HEAD~2..HEAD`, resulting in two patches: `0001-Hallo-Basta2.patch` and `0002-Hallo-Patch.patch`. It then resets the branch to `HEAD~2` using `--hard`. After applying the first patch, it commits the change with the message "patched change". 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

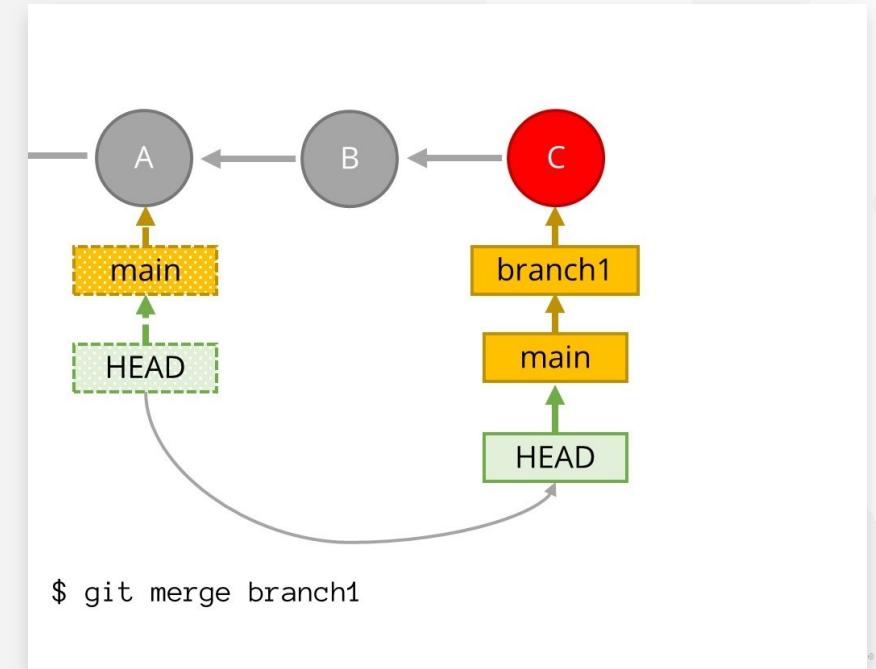
```
$ git add -p | --patch
```

```
git add -p index.html
git (perl)          %1           .bootcamp/test (-zsh)
git add -p index.html
git switch slow-down
Switched to branch 'slow-down'
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; }
     #tetriss { margin: 1em auto; padding: 1em; border: 4px solid black; border-radius: 10px; background-color: #F8F8F8; }
     #stats { display: inline-block; vertical-align: top; }
-    #canvases { display: inline-block; vertical-align: top; background: url(images/texture.jpg); box-shadow: 10px 10px 10px #999; border: 2px solid #333; }
+    #canvases { 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; }
```

Merging

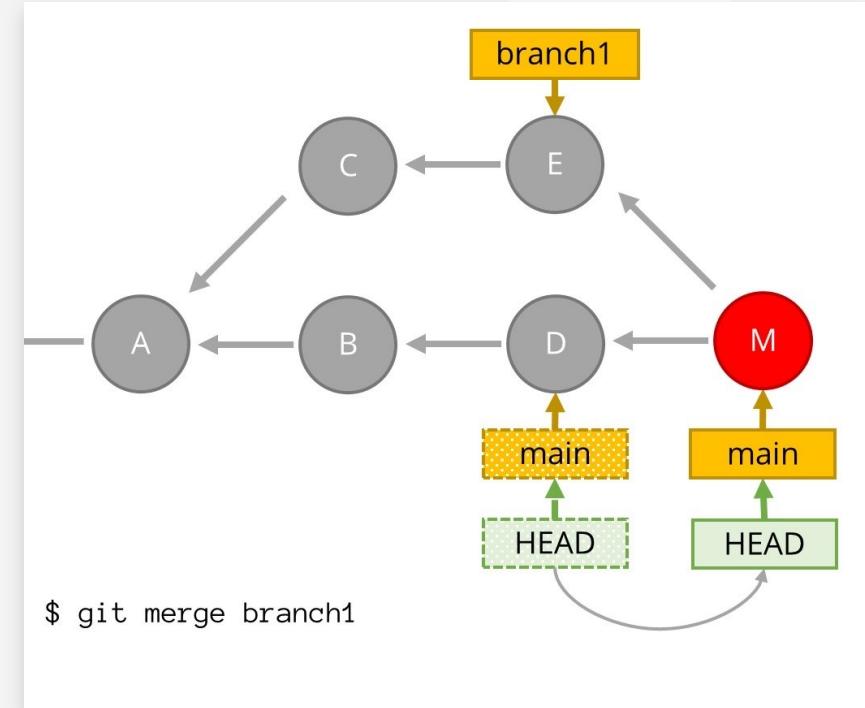
Fast-forward merge

```
$ 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
```



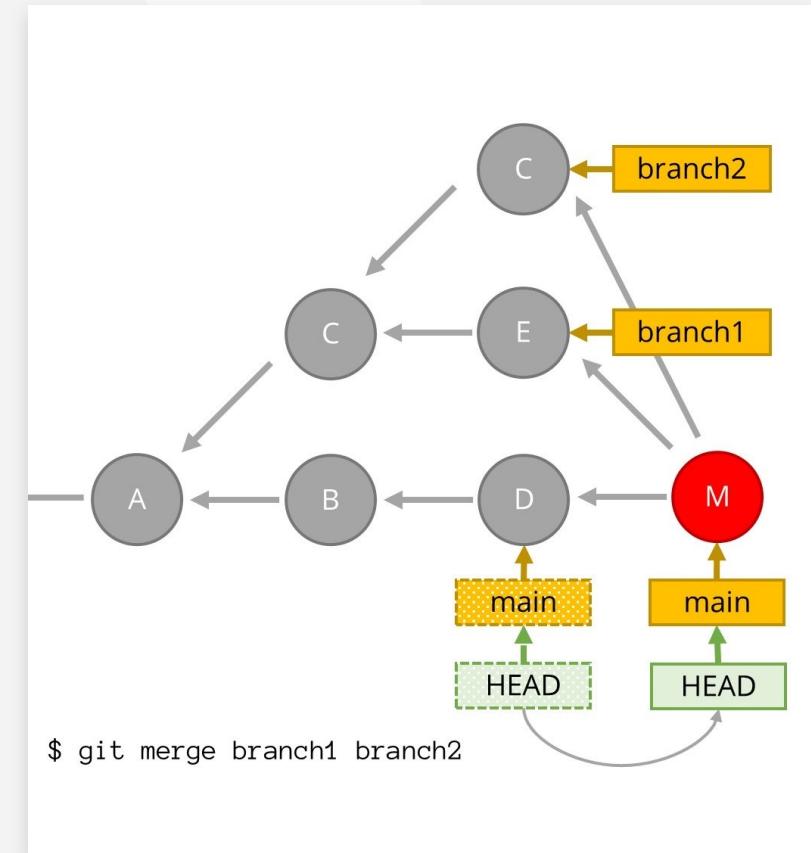
Merge commit

```
$ 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
```



A terminal window showing a git log --oneline --graph output. The log shows a merge commit at the top:

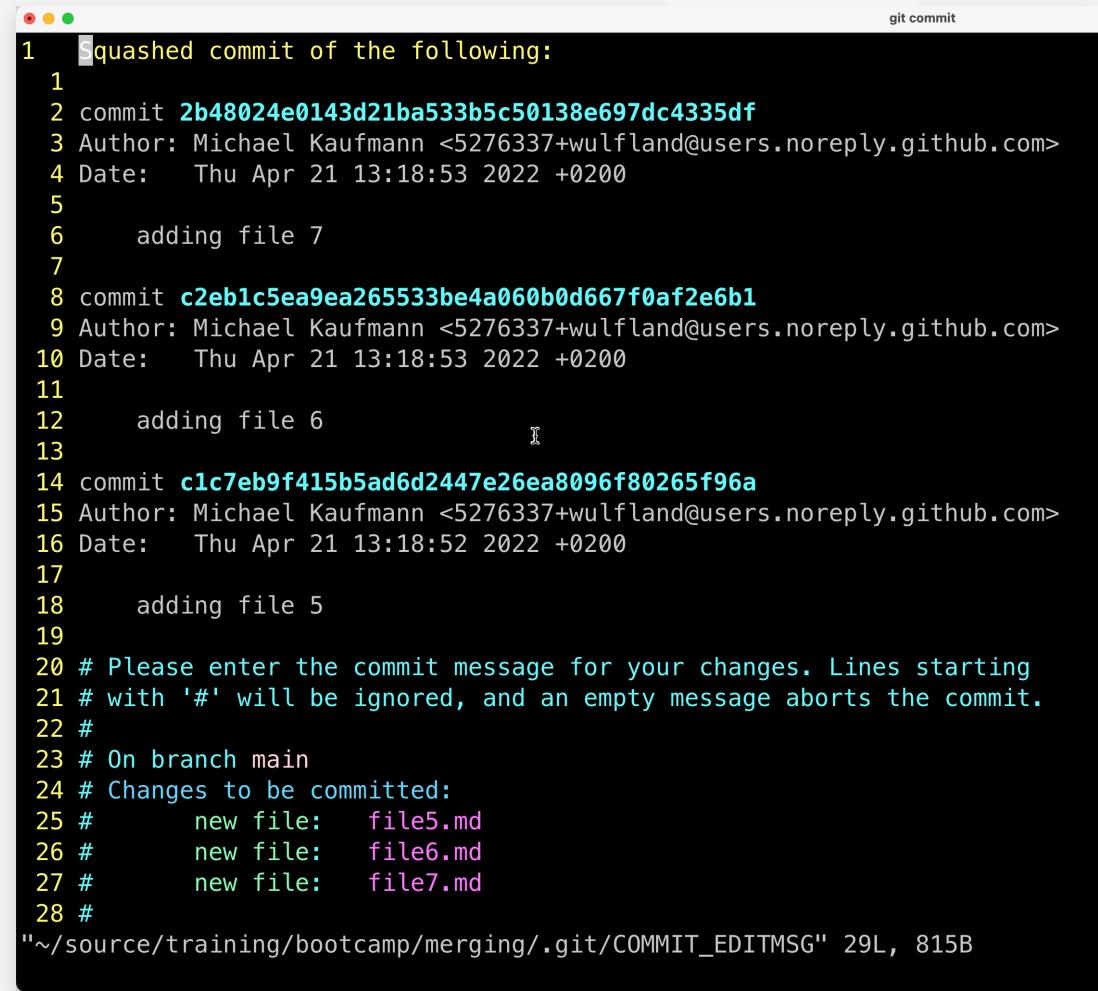
```
m.kaufmann@Michaels-MacBook-Pro: ~/source/repos/Merging
git log --oneline --graph
*-- 6fc5c3a (HEAD -> main) Merge branches 'branch1' and 'branch2' into main
|\ \
| * 6ecc1b9 (branch2) F
| * 1f7deeb (branch1) E
| /
| * eb3f7cf C
* | 48cffc6 D
* | 2bb6ada B
|/
* d781579 (tag: A) A
```

Below this, another git command is run:

```
git log --oneline --graph
```

Squash merge

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

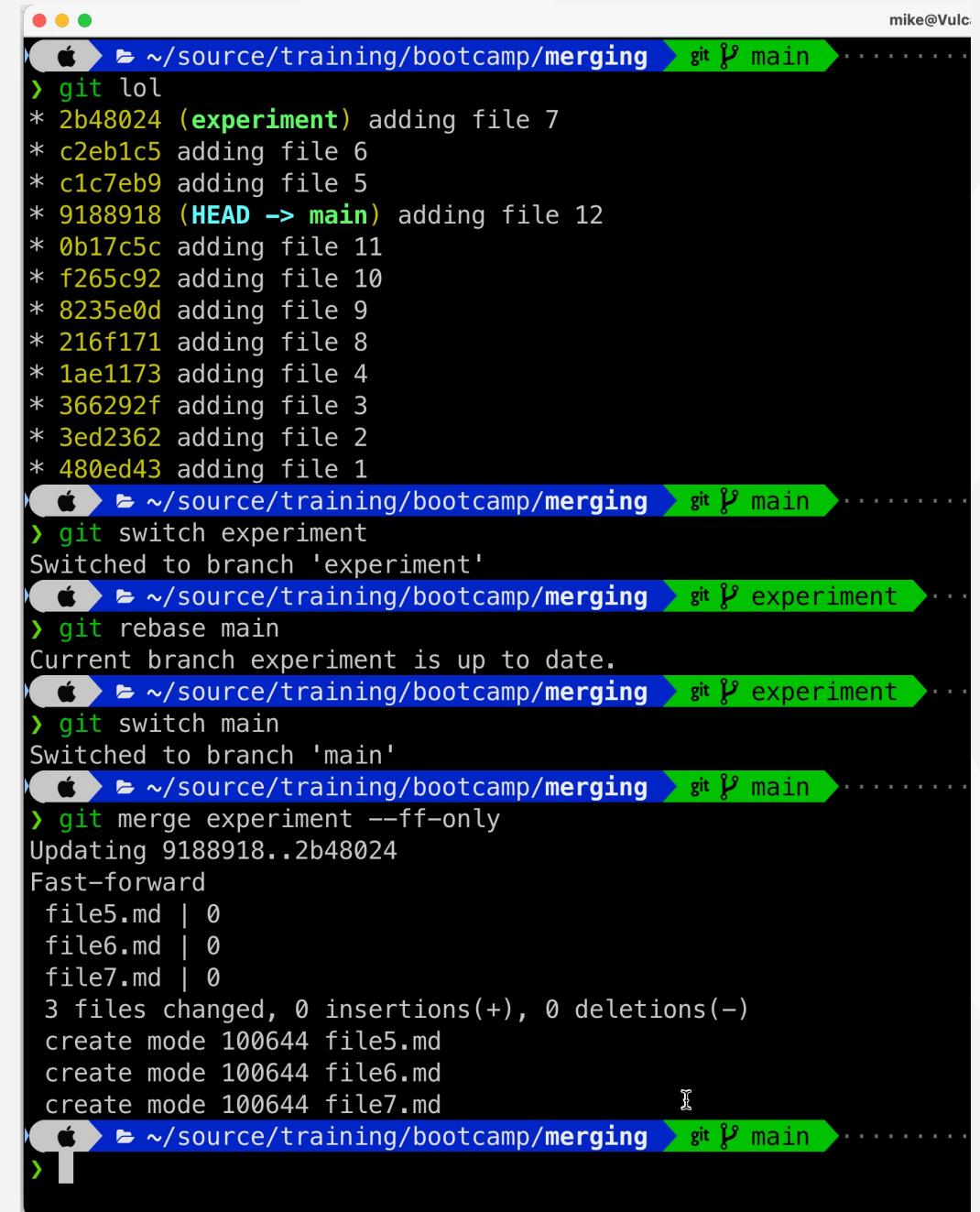


The screenshot shows a terminal window with the title "git commit". The content of the terminal is as follows:

```
1 Squashed commit of the following:  
1  
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
```



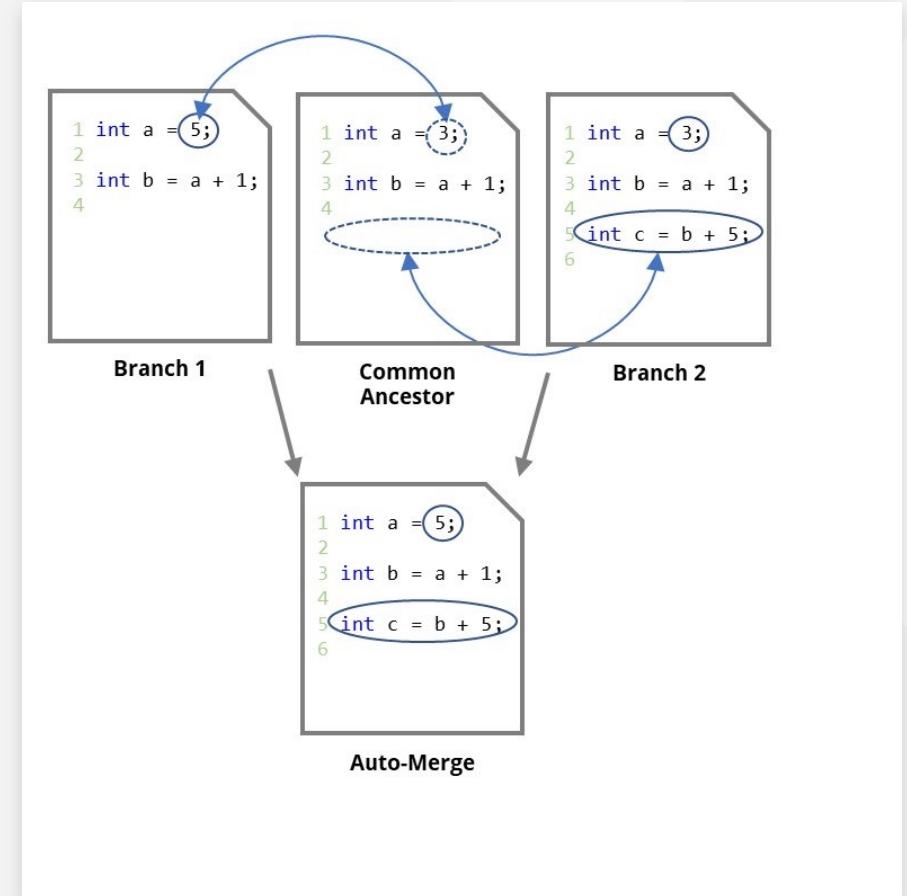
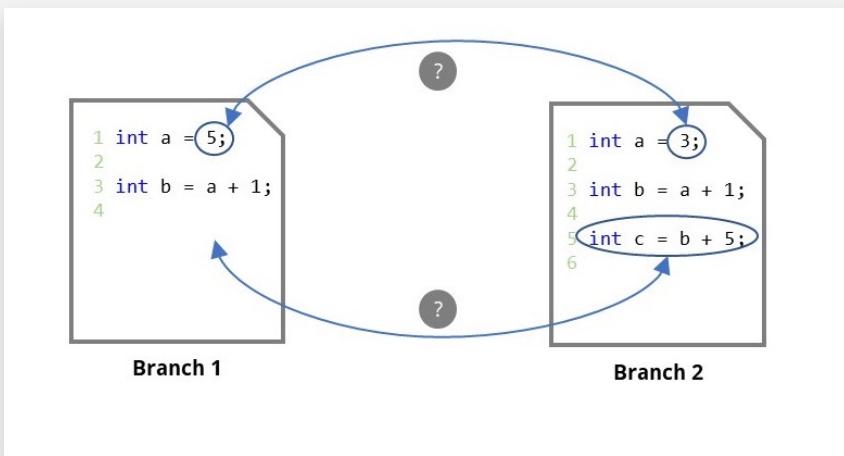
The screenshot shows a terminal window with the following session:

```
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  
git switch experiment  
Switched to branch 'experiment'  
git rebase main  
Current branch experiment is up to date.  
git switch main  
Switched to branch 'main'  
git merge experiment --ff-only  
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.

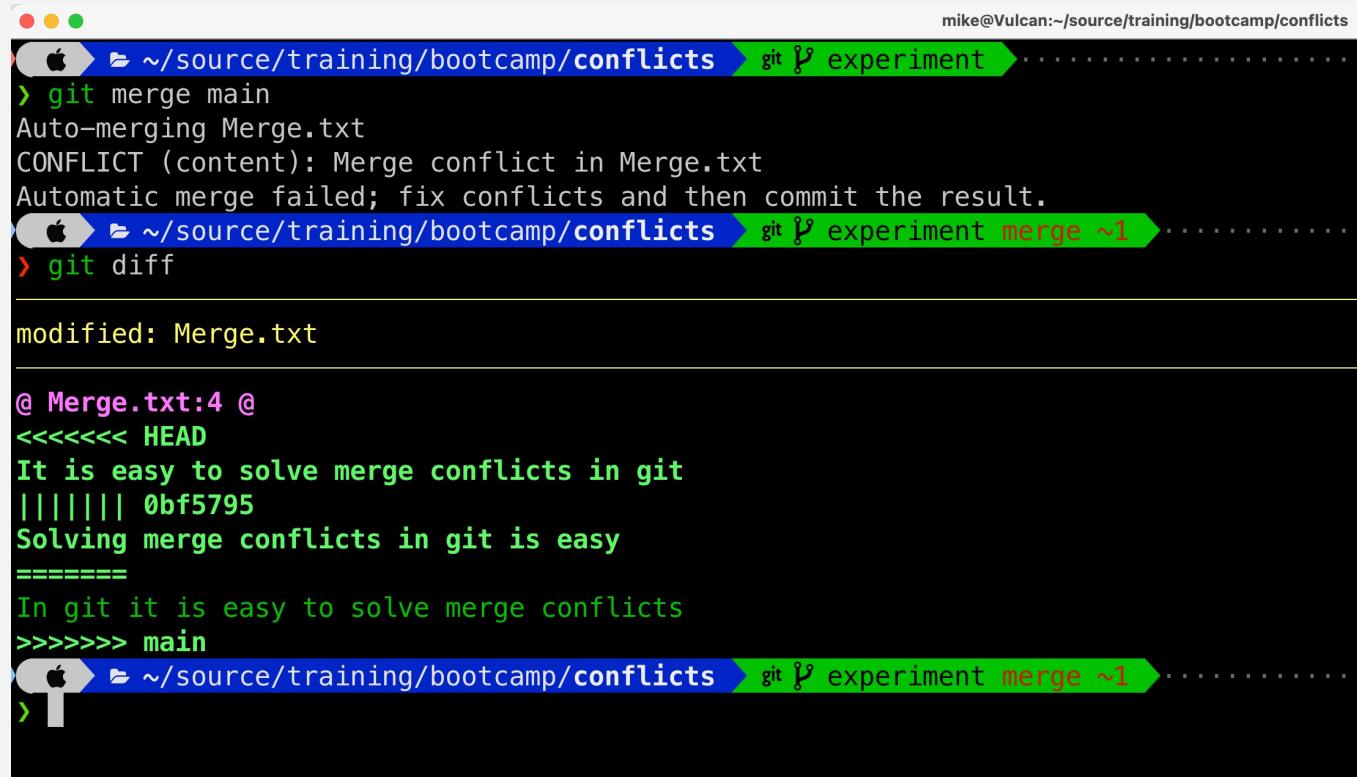
> <<<<<, =====, and >>>>>
<<<<<
HEAD (ours)
=====

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.  
❯ 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  
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  
❯
```

Resolving merge conflicts

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



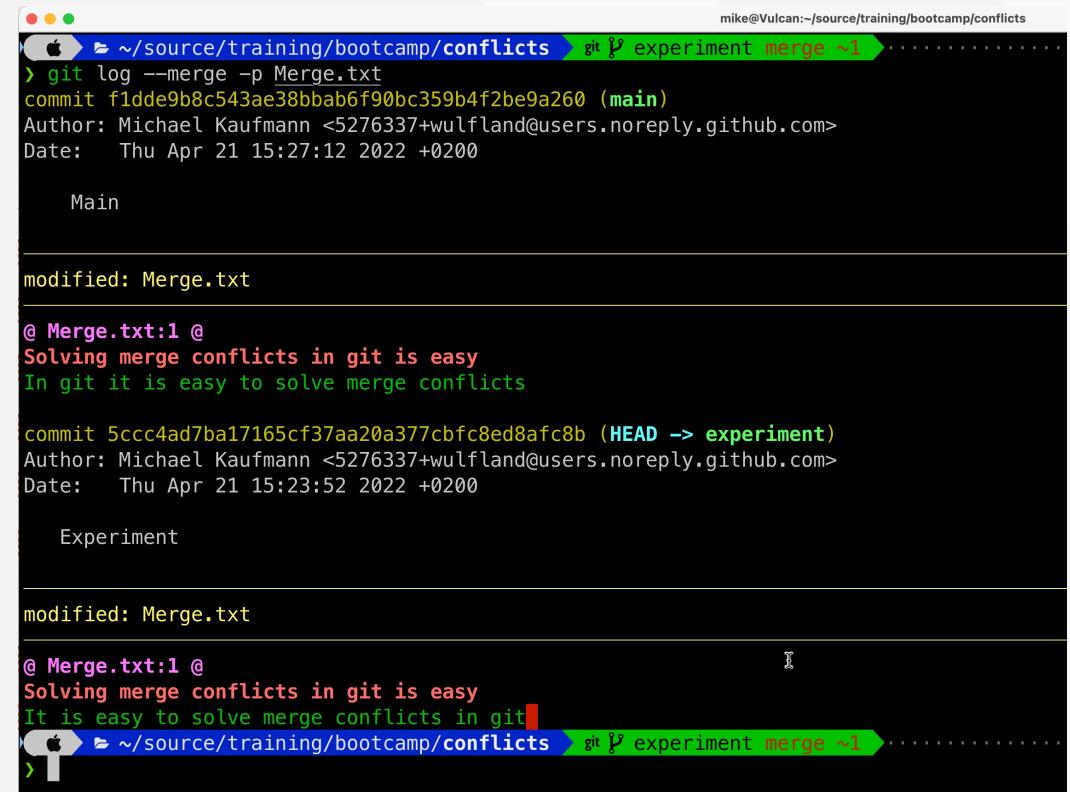
A terminal window titled "mike@Vulcan:~/source/training/bootcamp/conflicts" showing a git merge process. The user runs "git merge main", which auto-merges "Merge.txt". A conflict occurs, and the user runs "git diff" to view the conflict. The conflict shows two versions of the file "Merge.txt": one from HEAD (containing "It is easy to solve merge conflicts in git") and one from the main branch (containing "Solving merge conflicts in git is easy"). The user then runs "git merge ~1" to resolve the conflict.

```
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
```



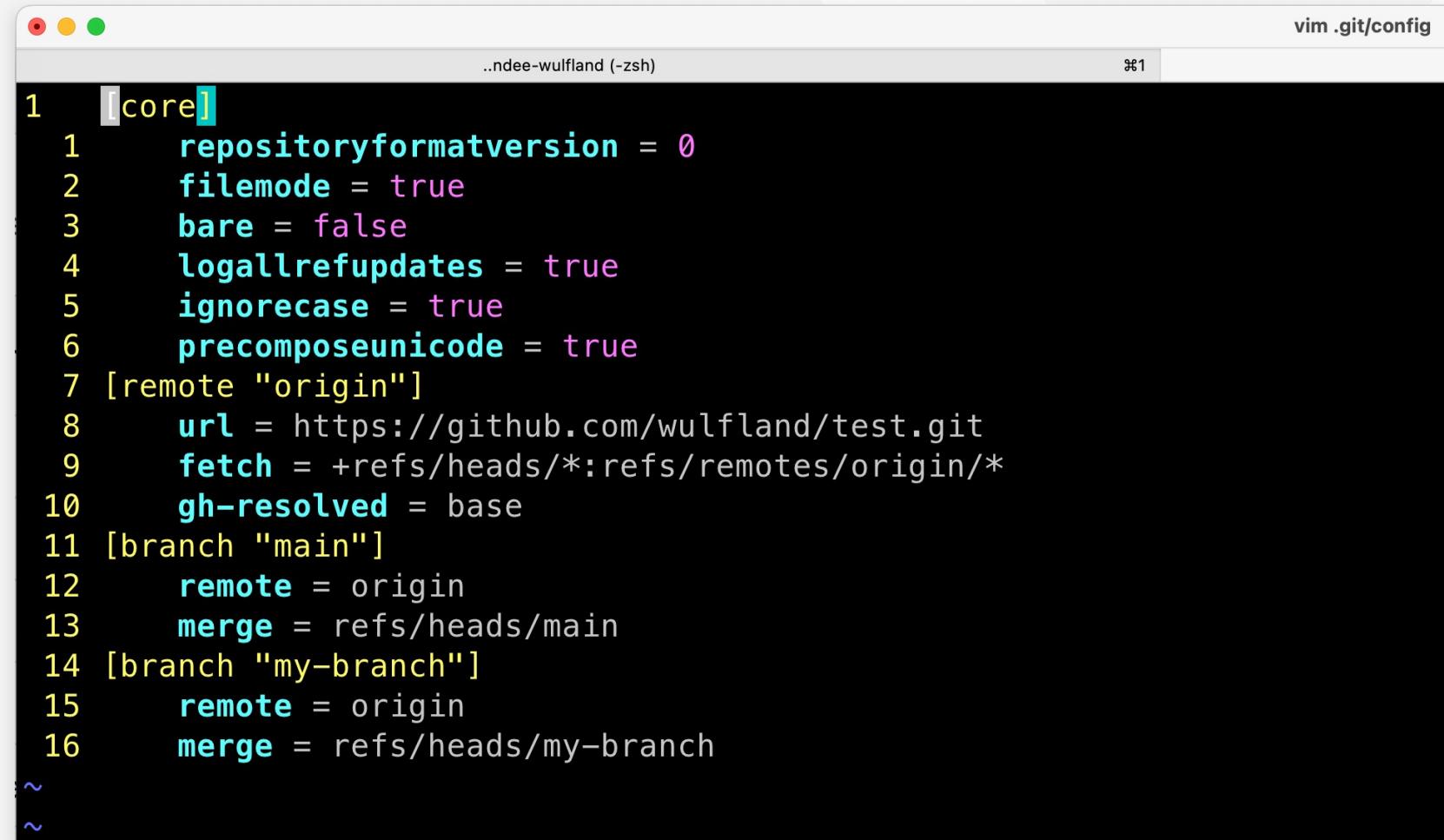
The screenshot shows a terminal window with the following command and output:

```
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
```

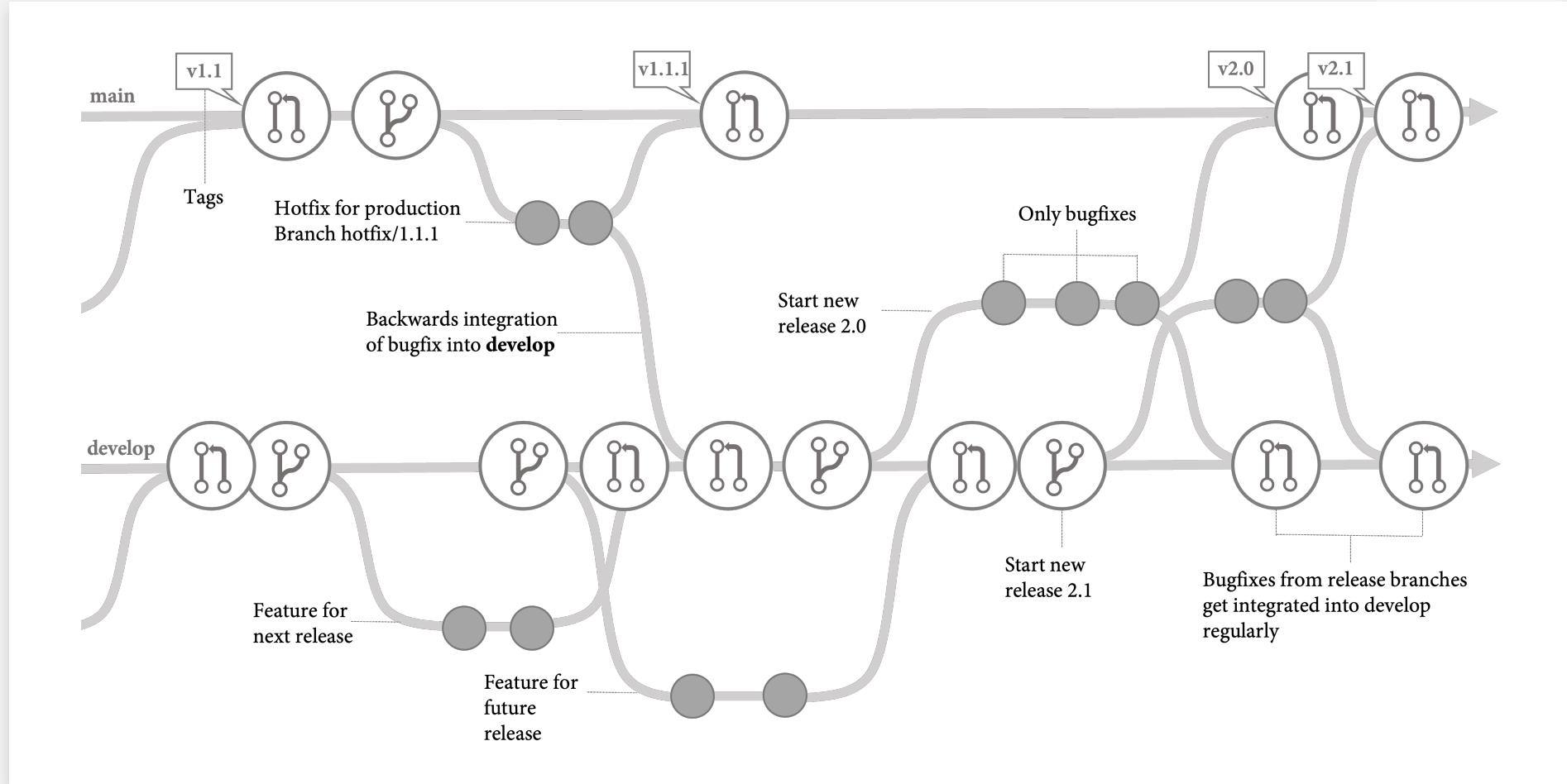


A screenshot of a terminal window titled "vim .git/config". The window shows the contents of the .git/config file. The configuration includes settings for the core section and two branches, "main" and "my-branch", both pointing to the "origin" remote.

```
..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  
~  
~
```

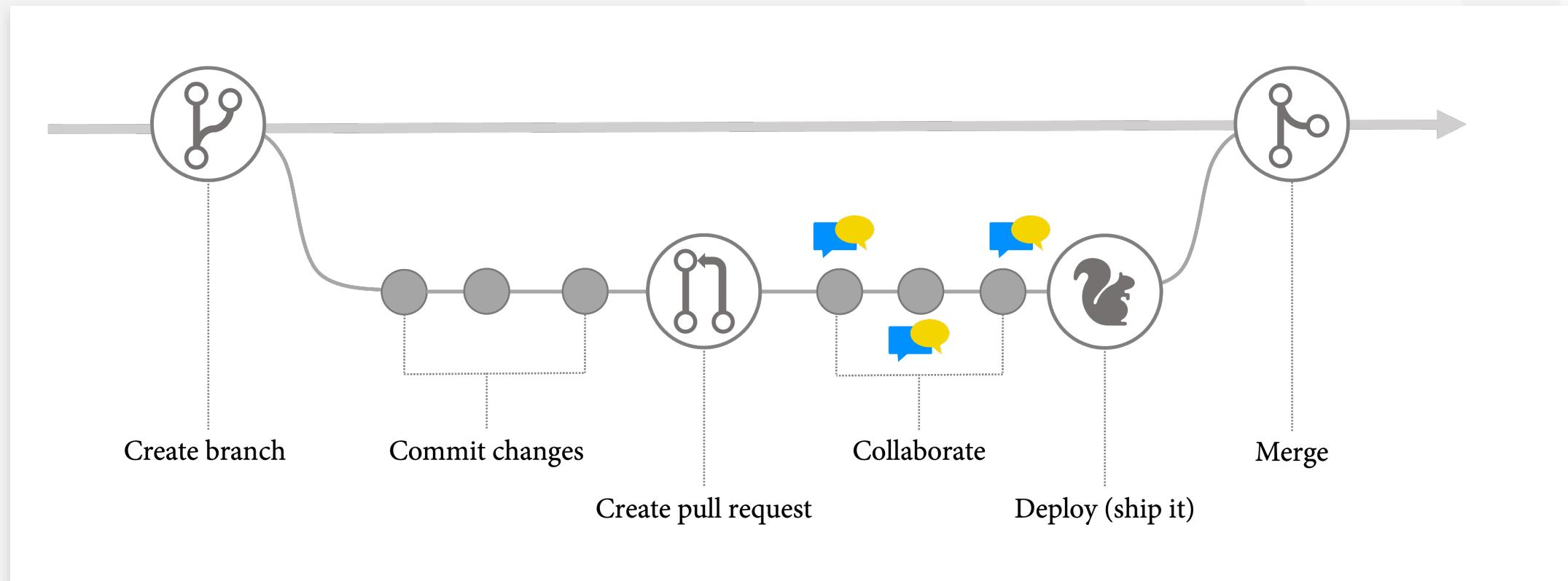
Branching workflows

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



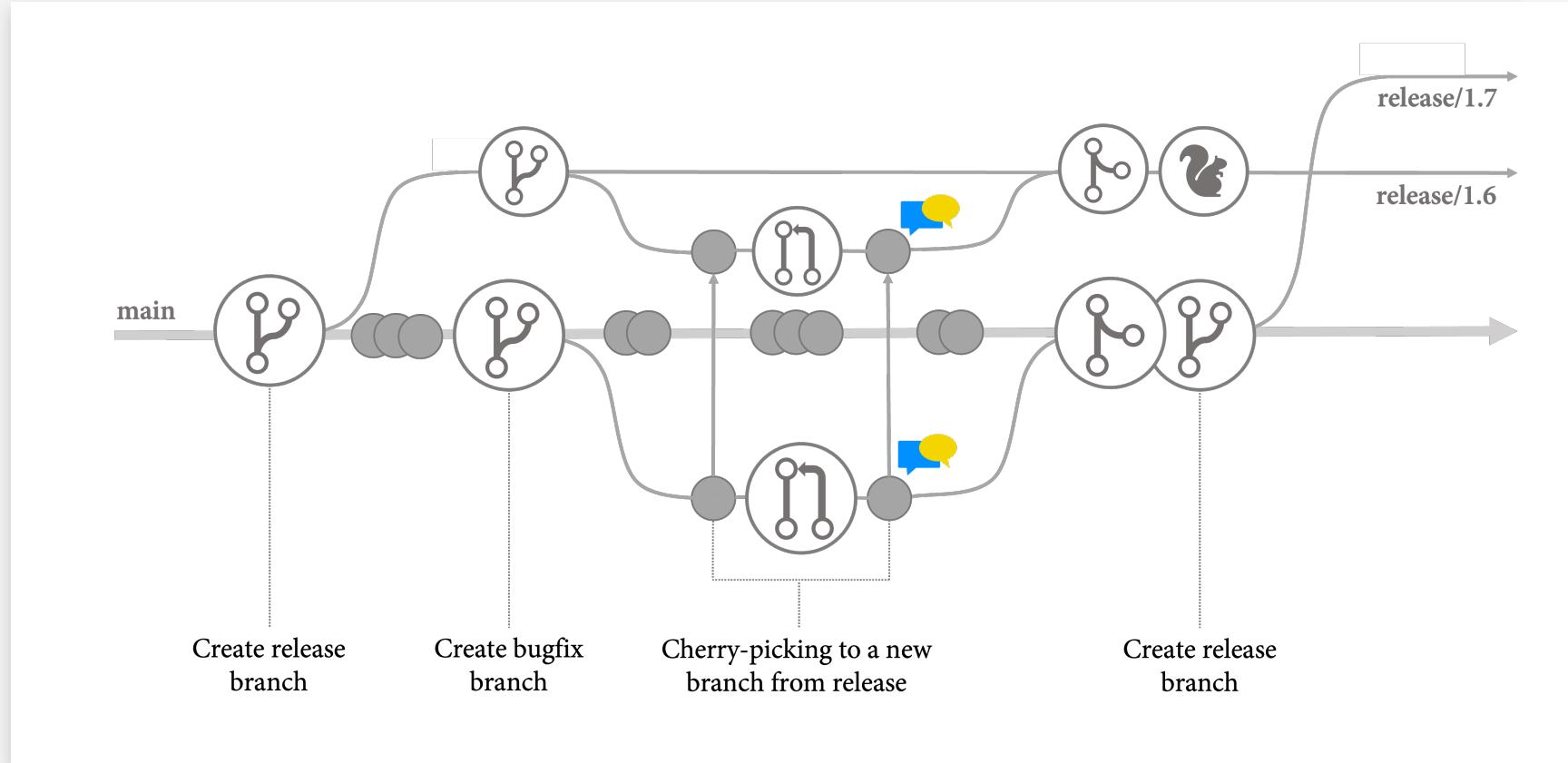
Branching workflows

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



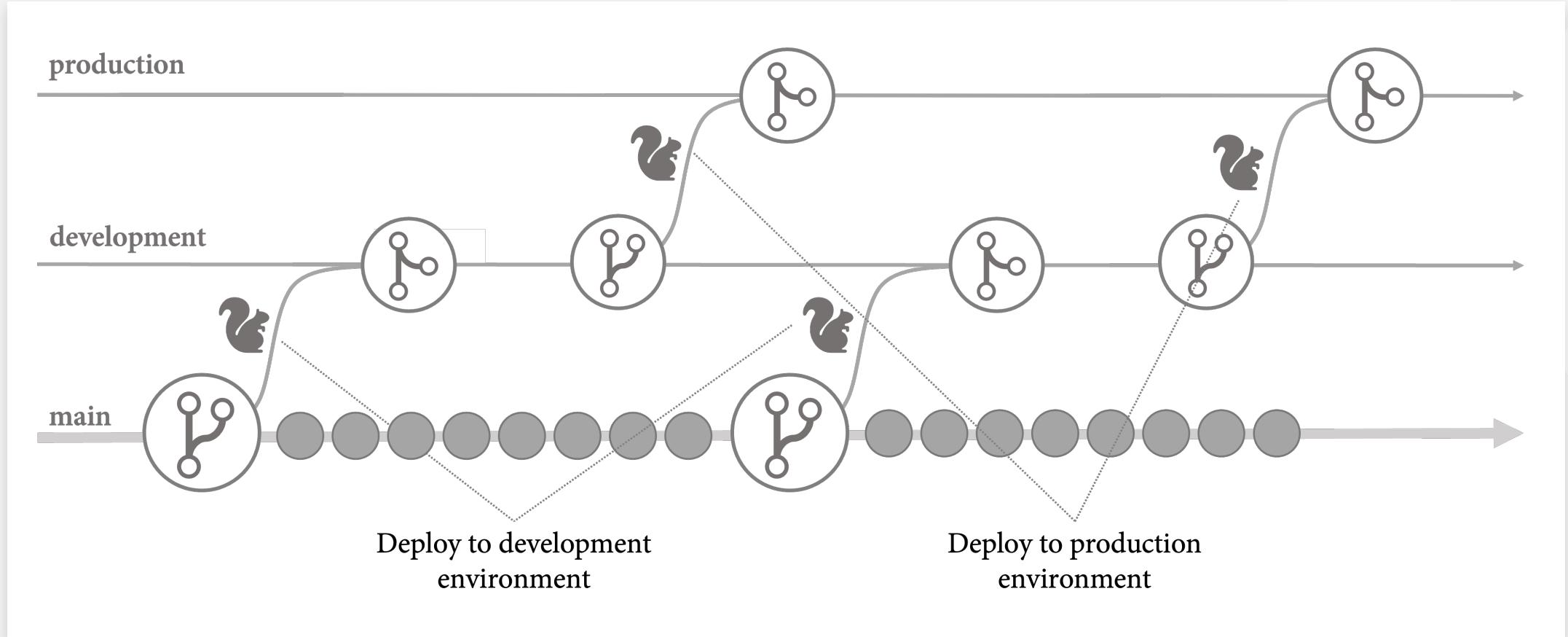
Branching workflows

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



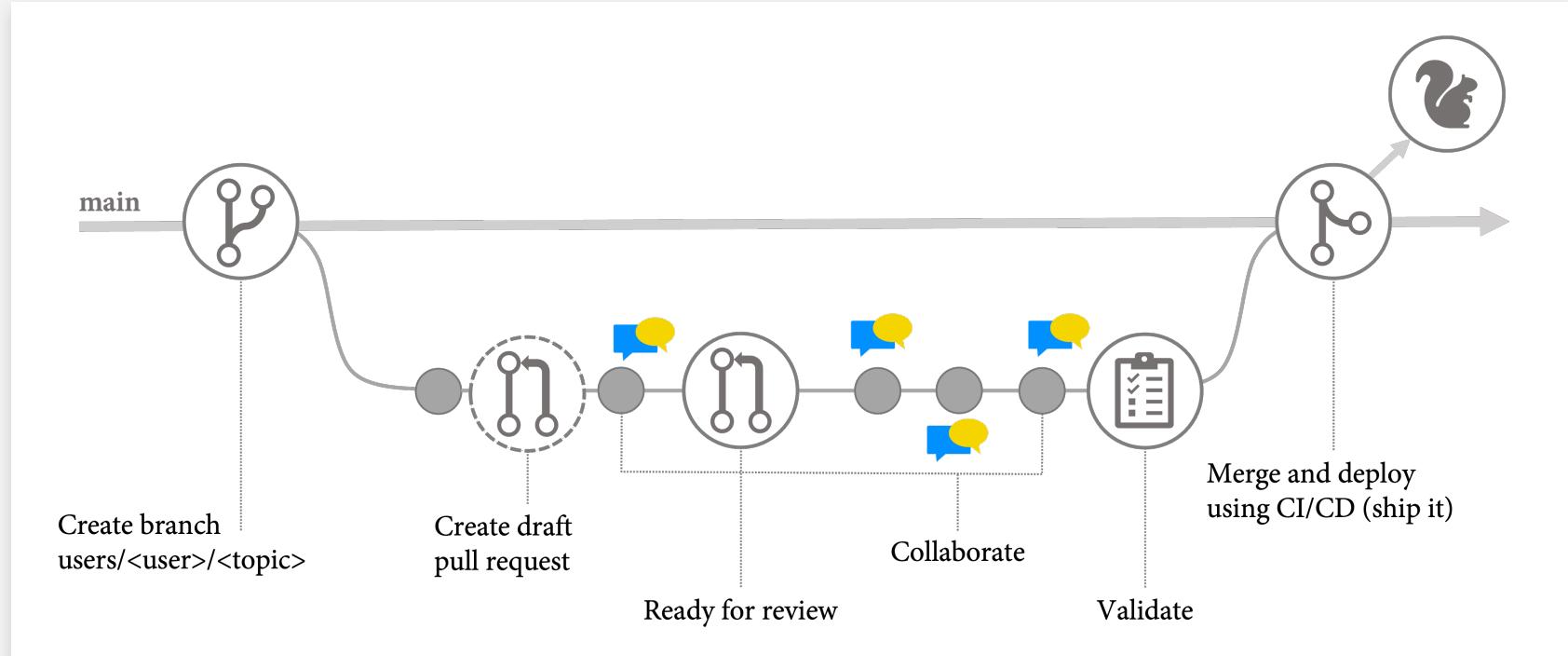
Branching workflows

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



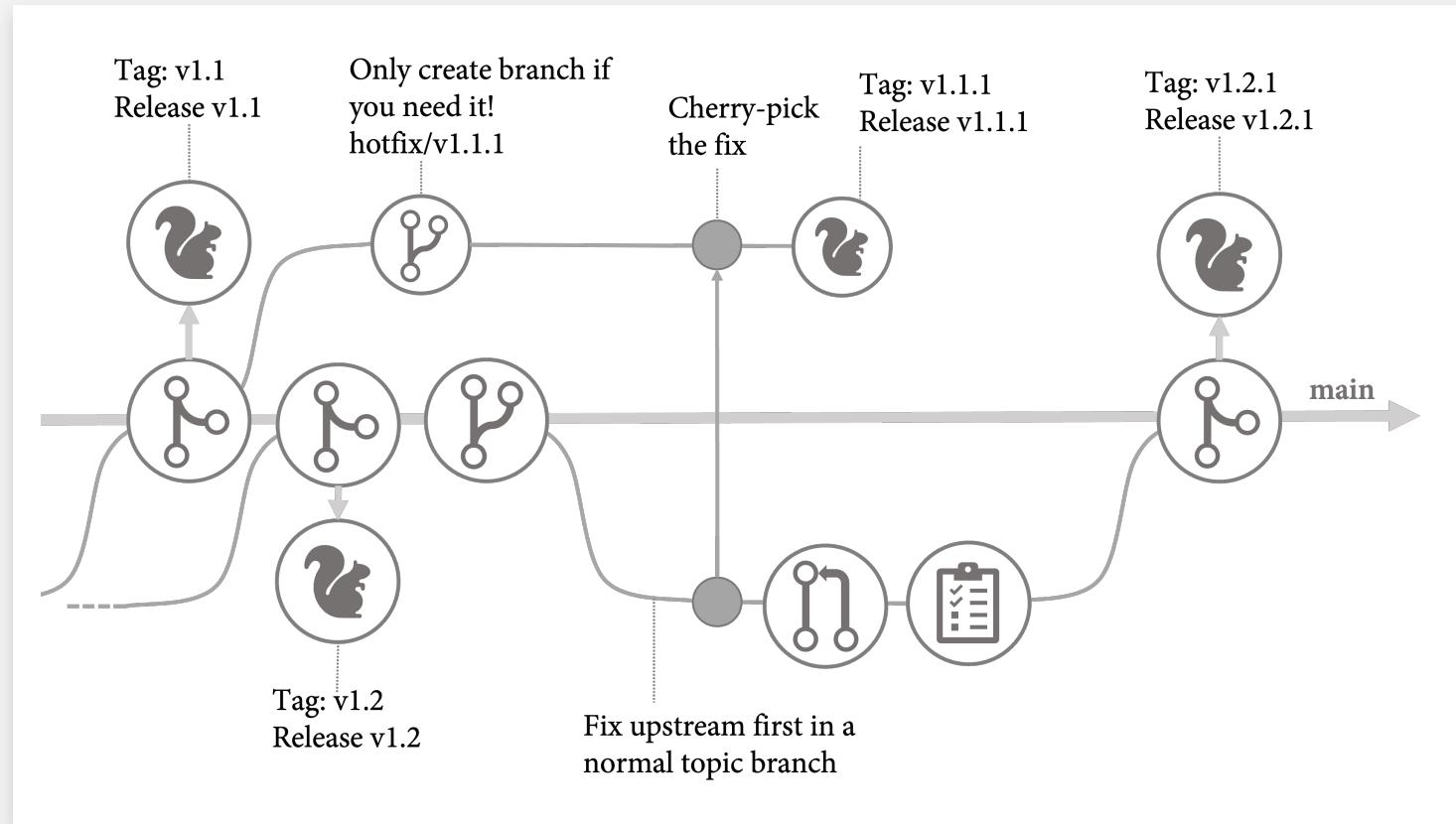
Branching workflows

- › **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

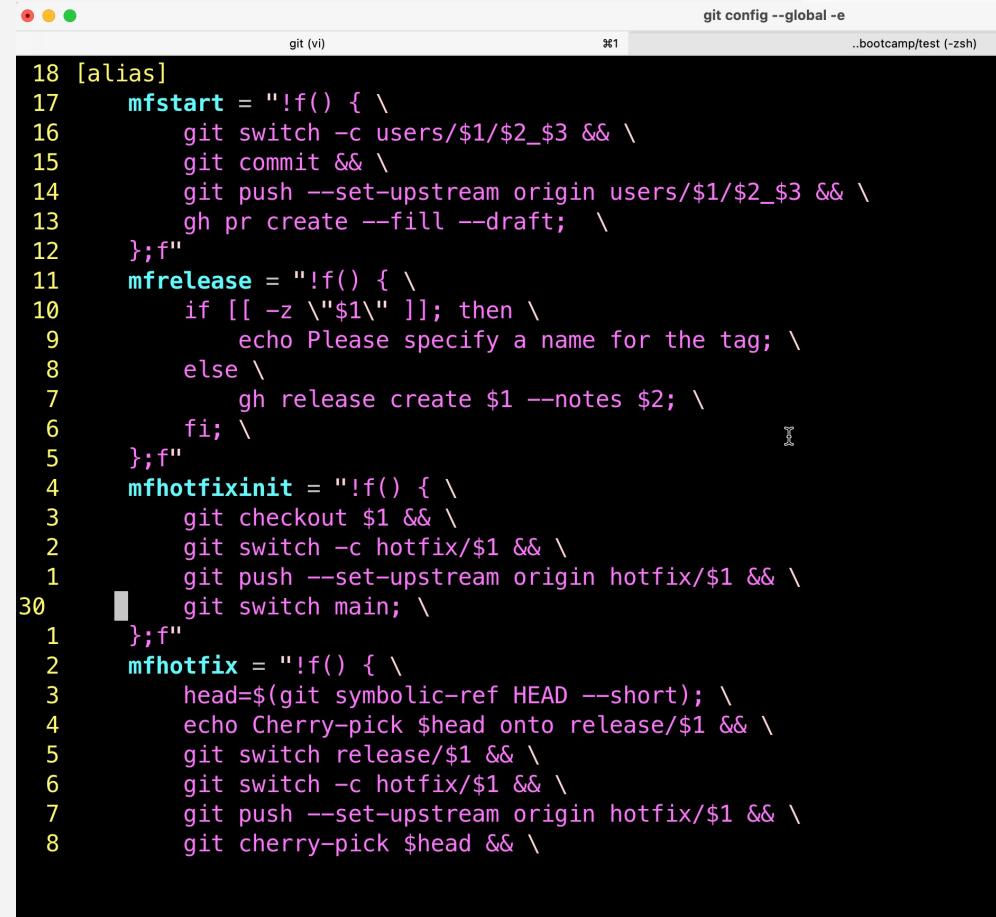


Git aliases

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 showing a .gitconfig file in a vi editor. The window title is "git config --global -e ..bootcamp/test (-zsh)". The file contains several aliases:

```
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>

Add Readme file #3

Merged wulfland merged 1 commit into main from users/kaufm/123_add-readme-file 18 hours ago

Conversation 0 Commits 1 Checks 0 Files changed 1

wulfland commented 18 hours ago

Fixes: #2

-o Add Readme file ... e8f3f60

wulfland marked this pull request as ready for review 18 hours ago

wulfland merged commit 6c71f44 into main 18 hours ago

Revert Create a new pull request to revert these changes

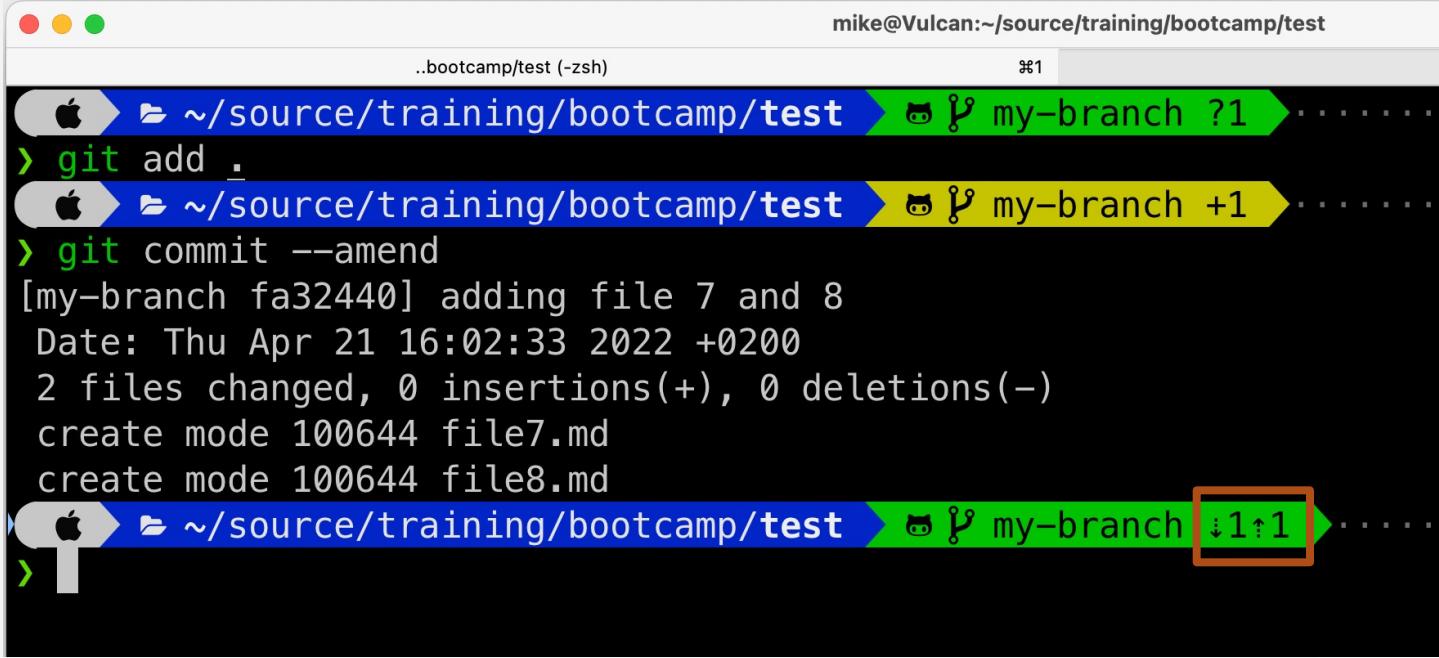
wulfland deleted the users/kaufm/123_add-readme-file branch 18 hours ago

Restore branch

Ammending changes

Ammending changes

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



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

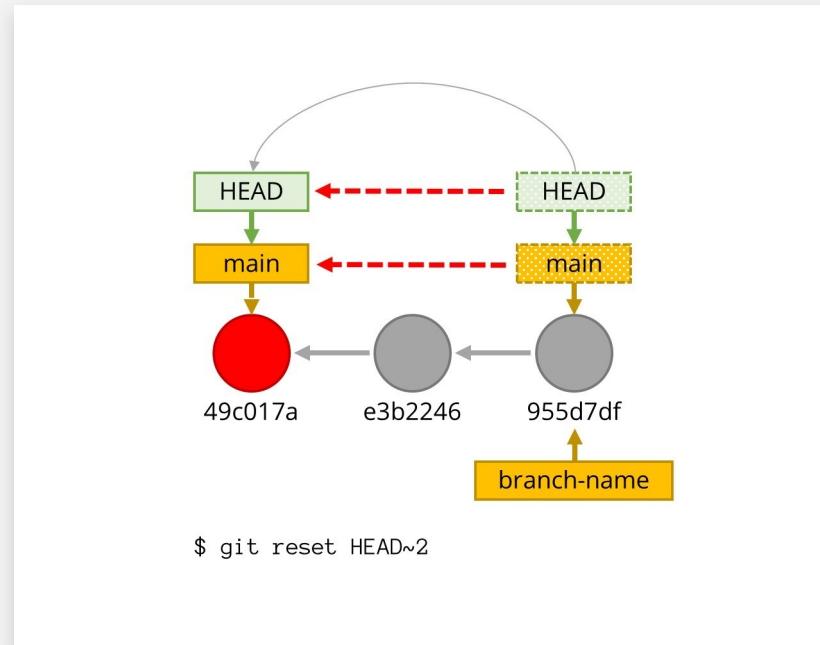
```
mike@Vulcan:~/source/training/bootcamp/test  
..bootcamp/test (-zsh) 361  
apple ~ source/training/bootcamp/test my-branch ?1  
> git add .  
apple ~ source/training/bootcamp/test my-branch +1  
> git commit --amend  
[my-branch fa32440] adding file 7 and 8  
Date: Thu Apr 21 16:02:33 2022 +0200  
2 files changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 file7.md  
create mode 100644 file8.md  
apple ~ source/training/bootcamp/test my-branch ↓1↑1  
>
```

The terminal title bar indicates the user is on the 'my-branch' branch. The command `git commit --amend` has been run, and the commit message shows it's amending the previous commit (fa32440). The commit message includes the addition of two files: `file7.md` and `file8.md`. The status bar at the bottom of the terminal shows the current branch as `my-branch` with a revision indicator `↓1↑1`.

Resetting your repository

Resetting your repository

- Moving the current branch to another position



Resetting your repository

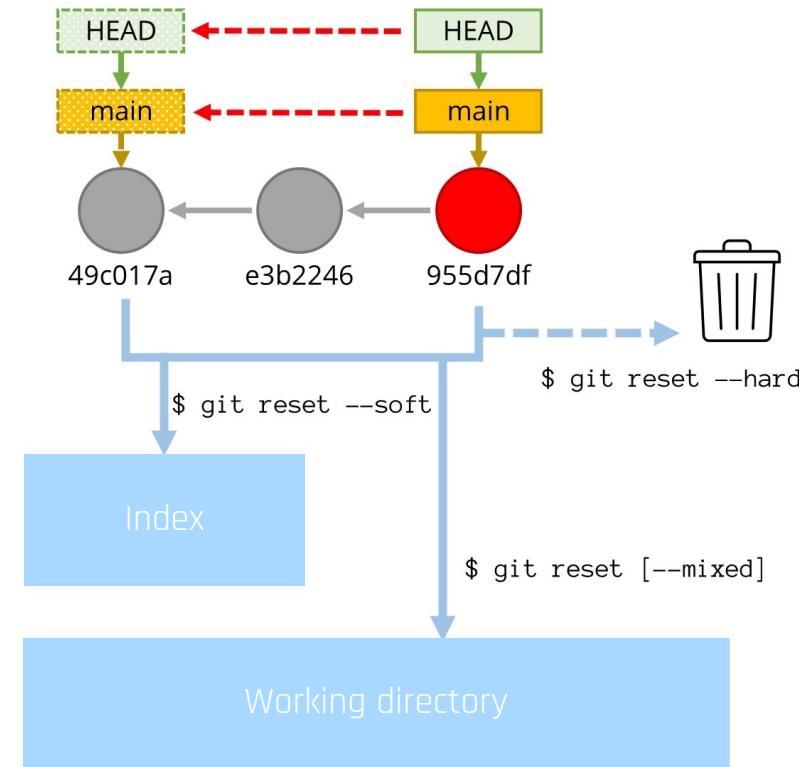
- › What happens with the changes?

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

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

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

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

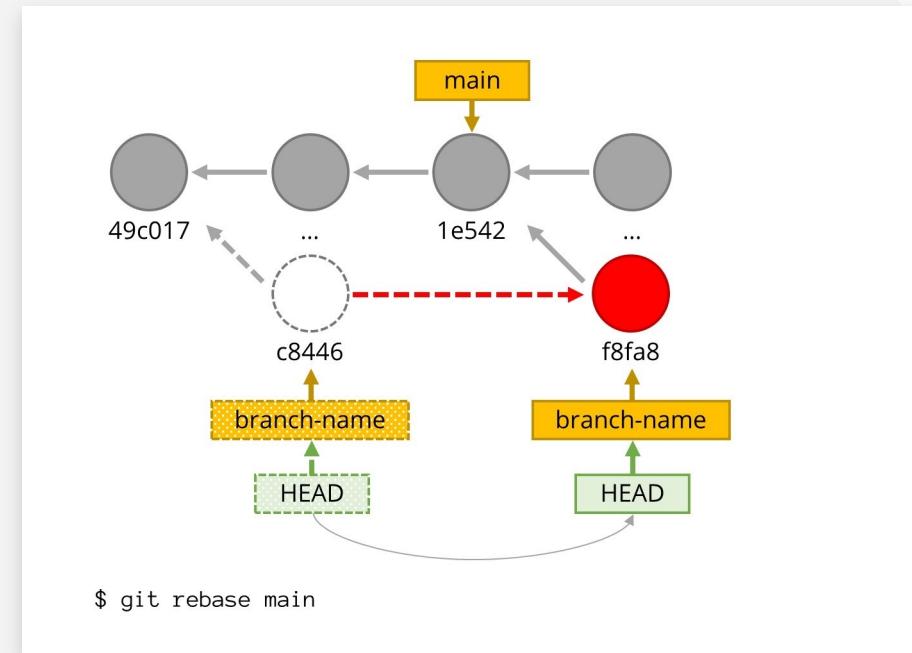


Rebase

Rebase

› Rebase

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



Cherry-picking and the reflog

Git RefLog

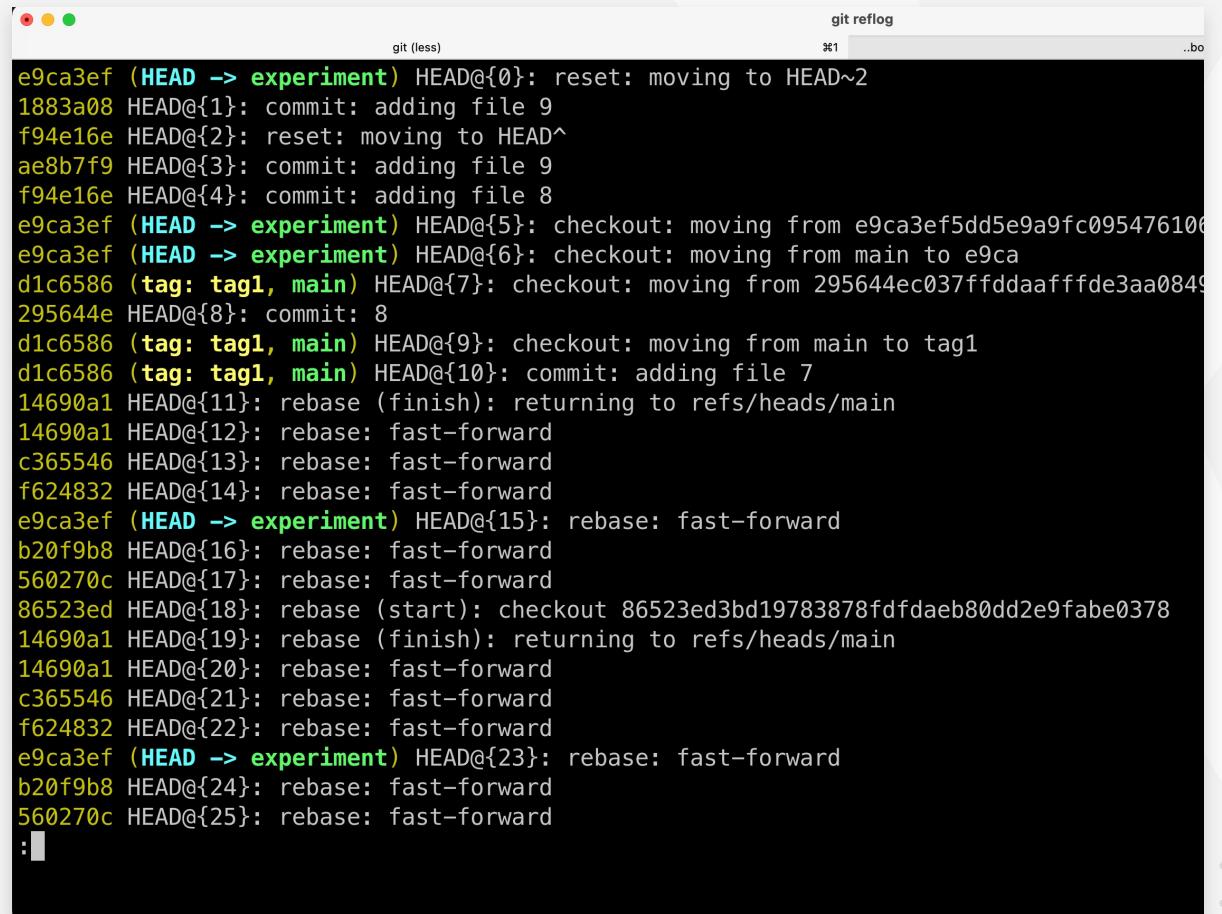
- › The RefLog is **local!**
- › Expires:
 - › `gc.reflogExpireUnreachable` (default **30** days)
 - › `gc.reflogExpire` (default **90** days)

```
$ 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

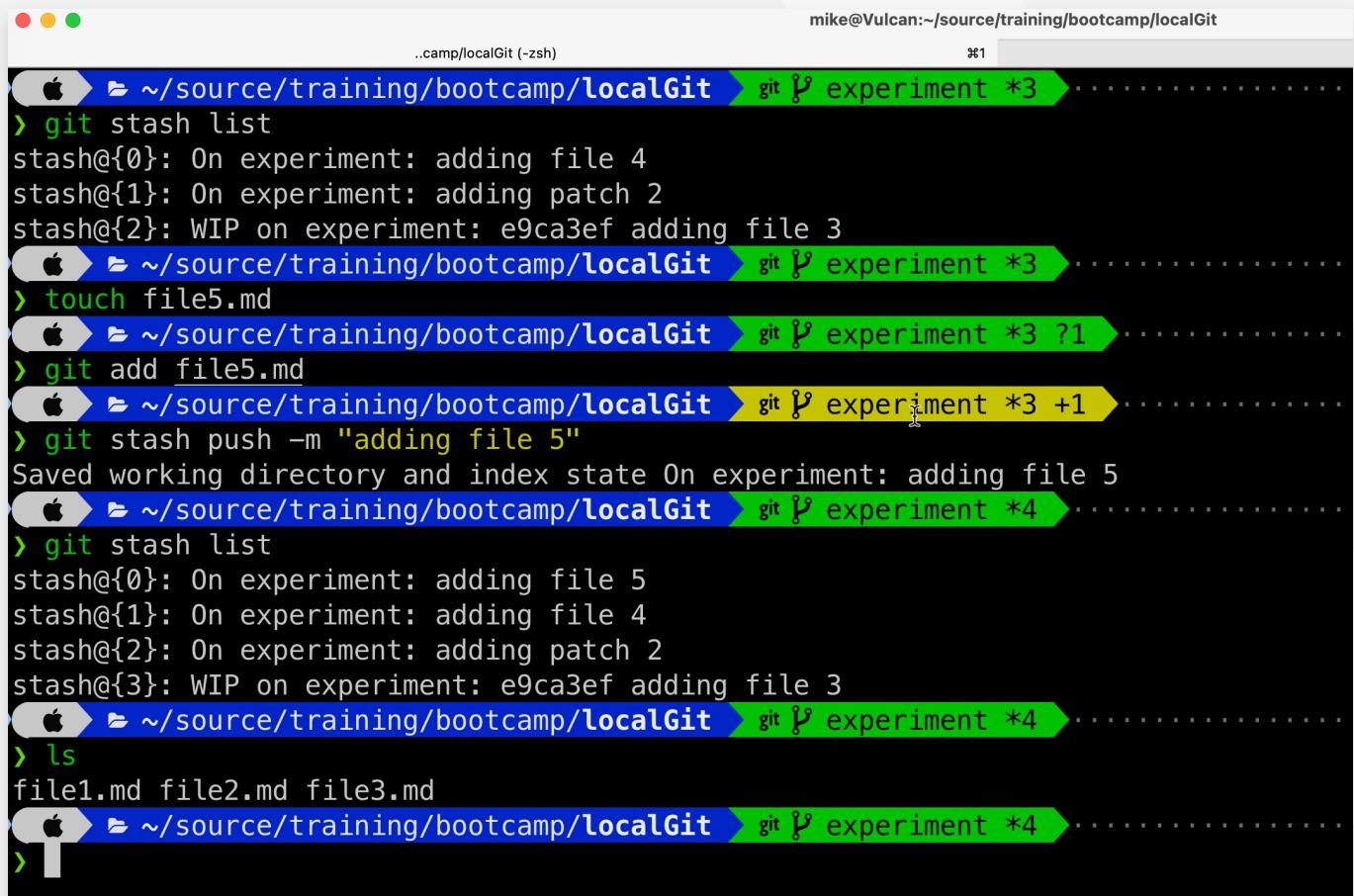
```
$ git stash [push]
```

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

```
$ git stash list
```

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

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



A terminal window titled 'mike@Vulcan:~/source/training/bootcamp/localGit' showing a sequence of git commands related to stashing changes.

```
mike@Vulcan:~/source/training/bootcamp/localGit git [experiment *3]
> git stash list
stash@{0}: On experiment: adding file 4
stash@{1}: On experiment: adding patch 2
stash@{2}: WIP on experiment: e9ca3ef adding file 3
> touch file5.md
> git add file5.md
> git stash push -m "adding file 5"
Saved working directory and index state
On experiment: adding file 5
> git stash list
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
> ls
file1.md file2.md file3.md
>
```

Let's connect



@mike_kaufmann



@wulfland



<https://writeabout.net>

Let's connect!



**Michael
Kaufmann**

Founder/MD Xpirit Germany
Microsoft Regional Director, MVP
[Linkedin.com/in/mikaufmann](https://www.linkedin.com/in/mikaufmann)
mkaufmann@xpirit.com