Chapter 05
GIT Advanced

Open Source SW Development
CSE22300

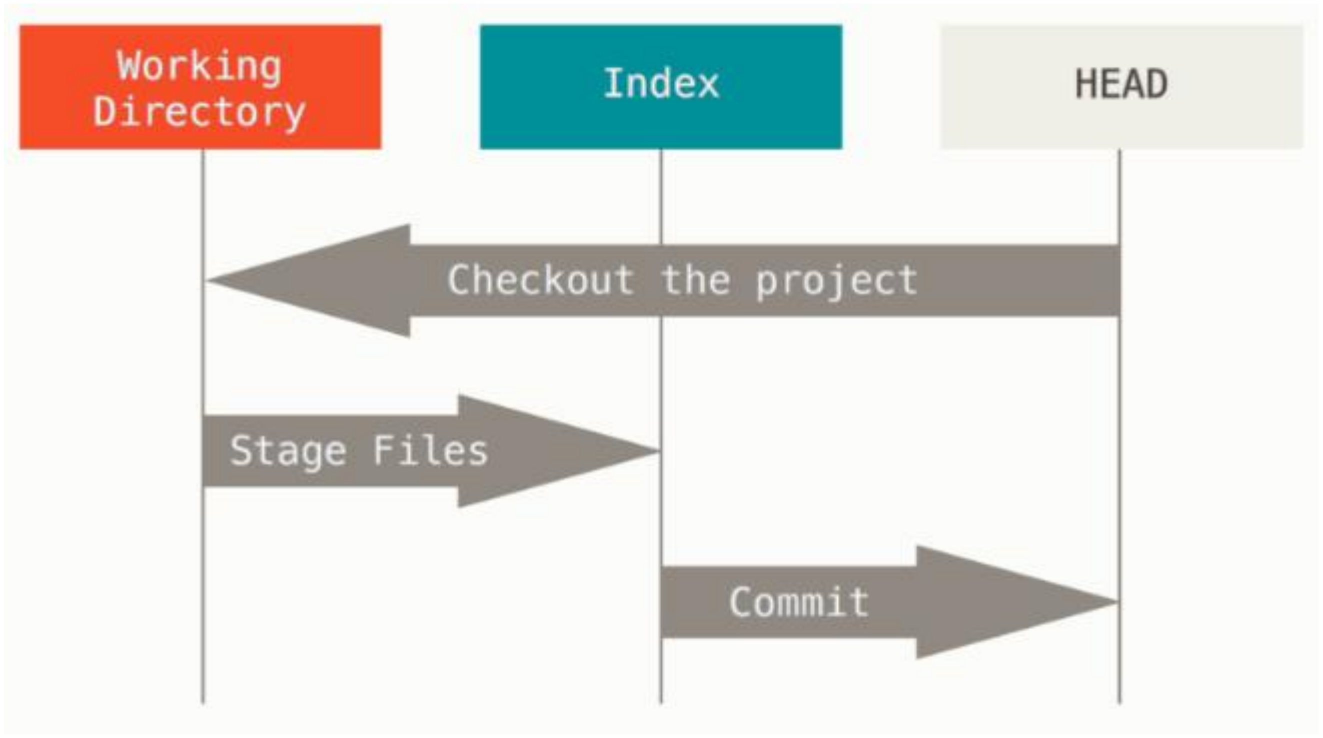
Reset

The Three Tress

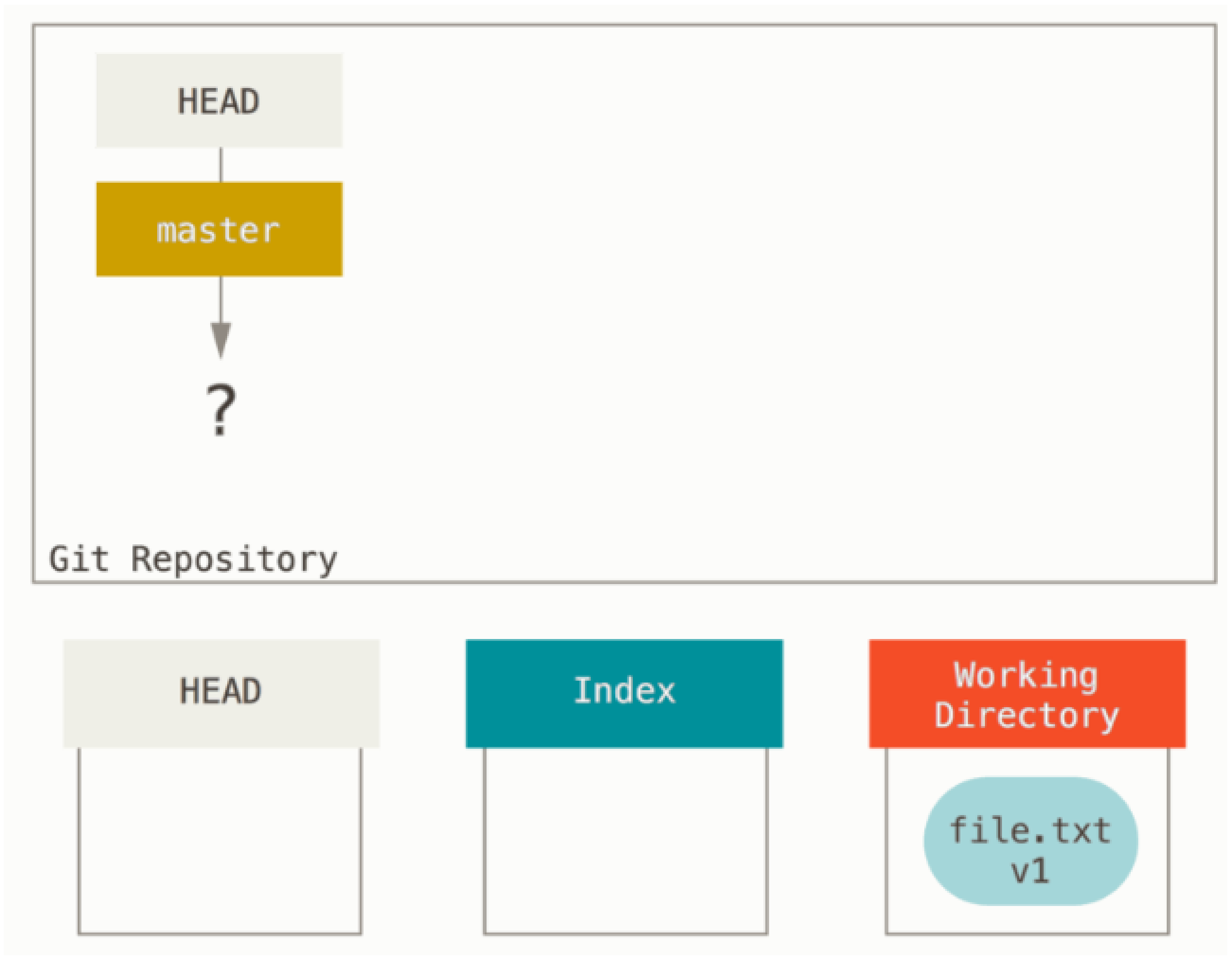
- **Tree**
  - **Collection of files, not specifically the data structure**

Tree	Role
HEAD	Last commit snapshot, next parent
Index	Proposed next commit snapshot, Staging area
Working Directory	Working Tree, Sandbox, Actual files and directories

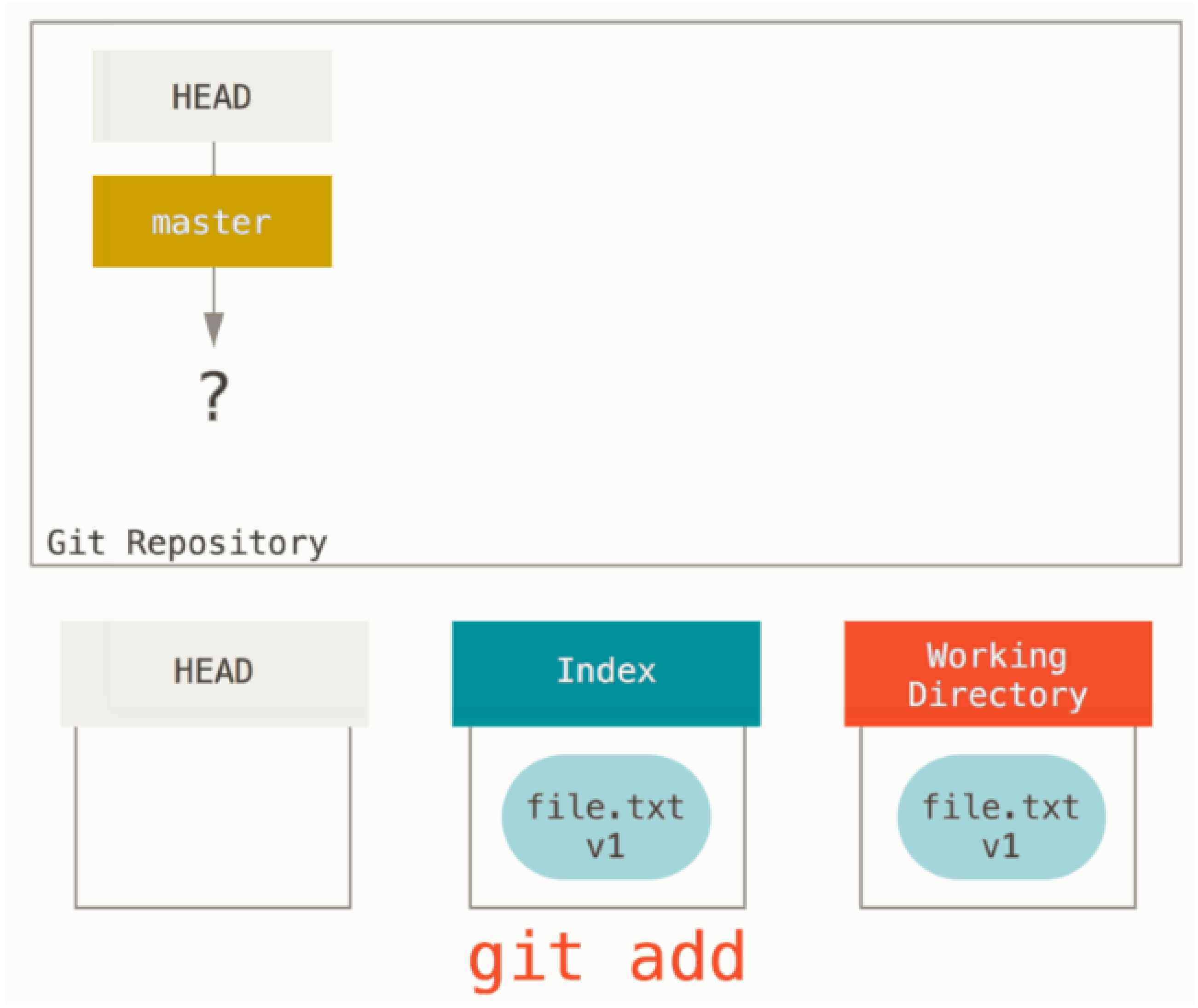
The Workflows



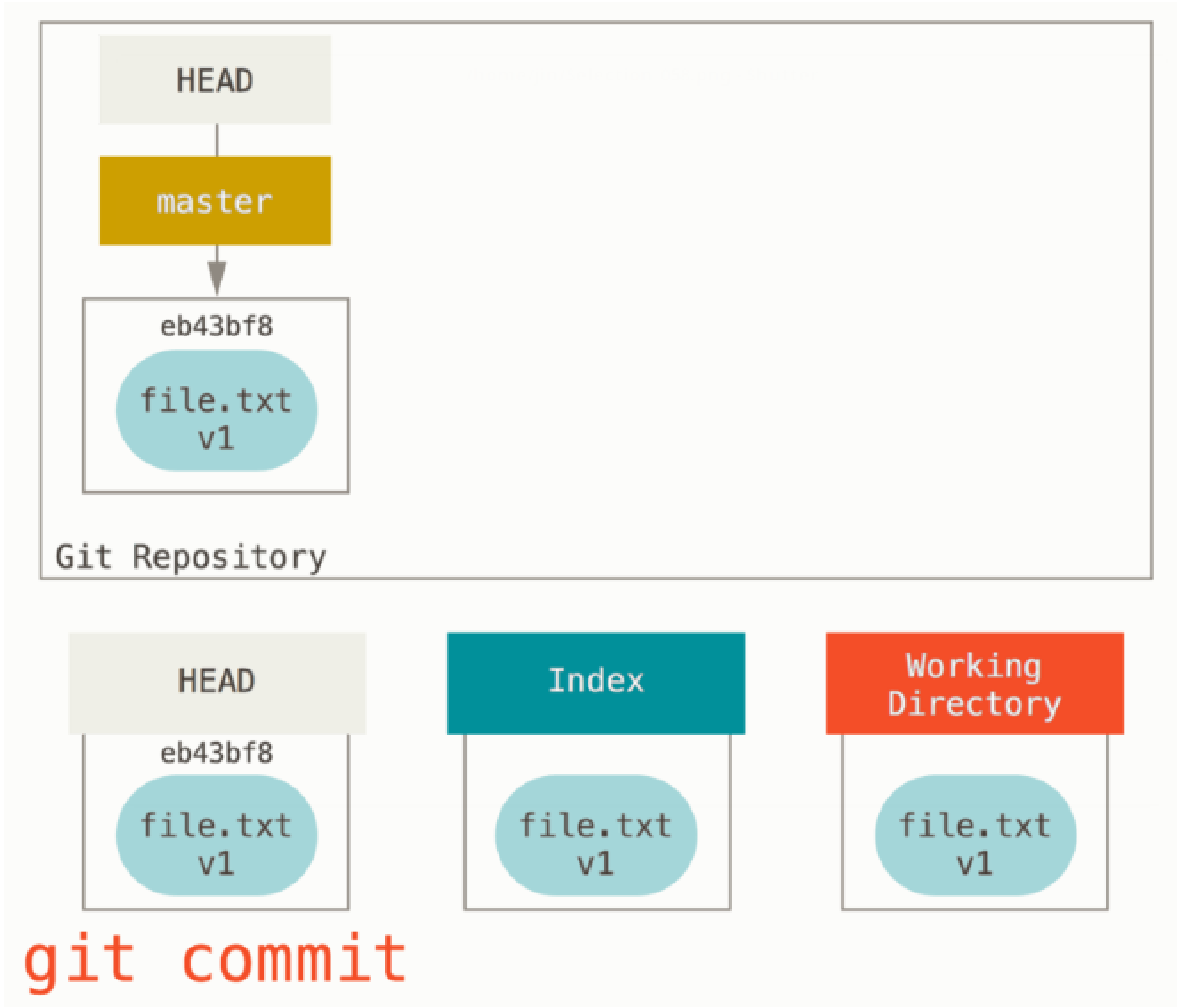
git init


HEAD
master
?
Git Repository
HEAD
Index
Working Directory
file.txt v1

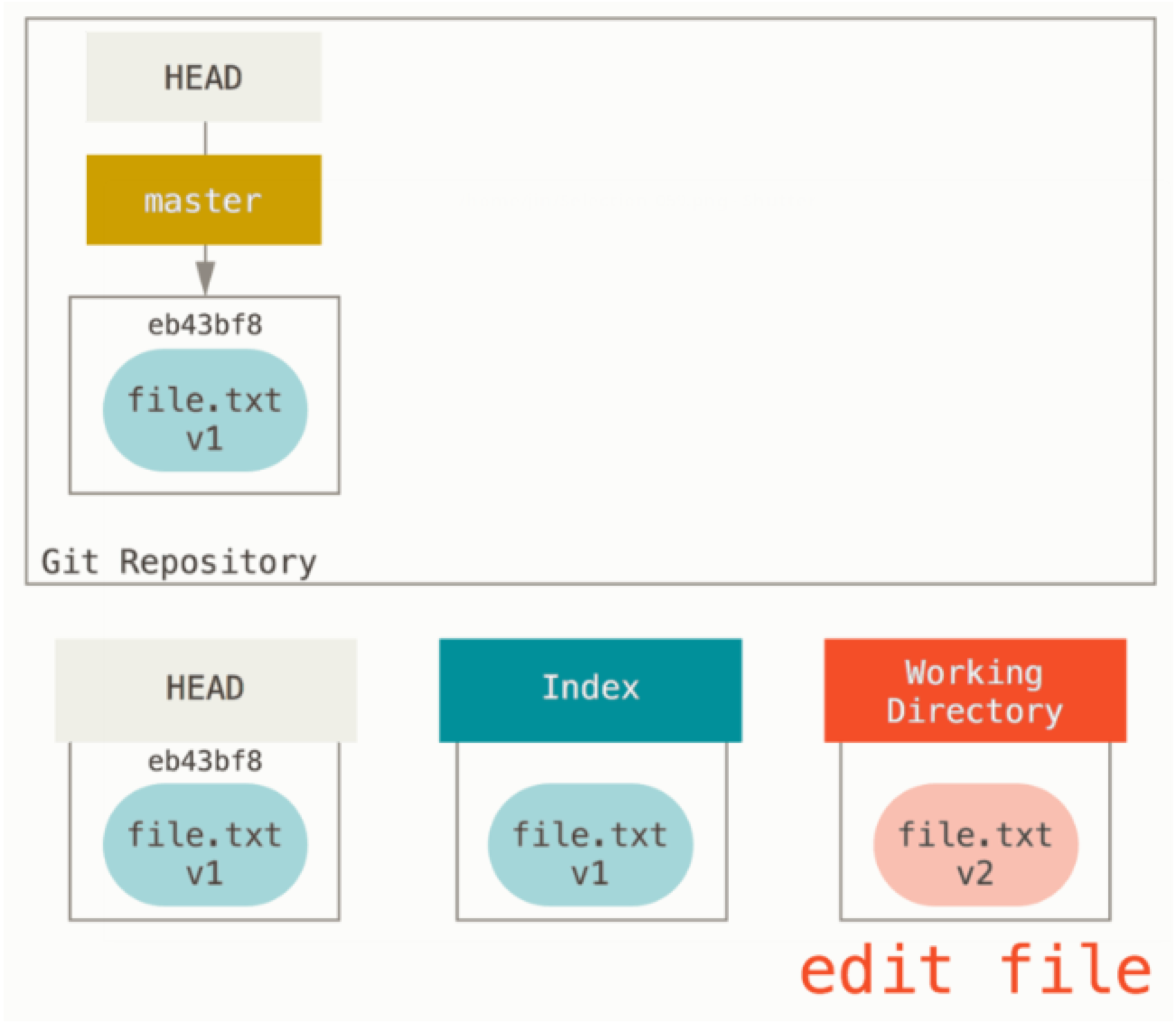
git add


HEAD
master
?
Git Repository
HEAD
Index
Working Directory
file.txt v1
file.txt v1
git add

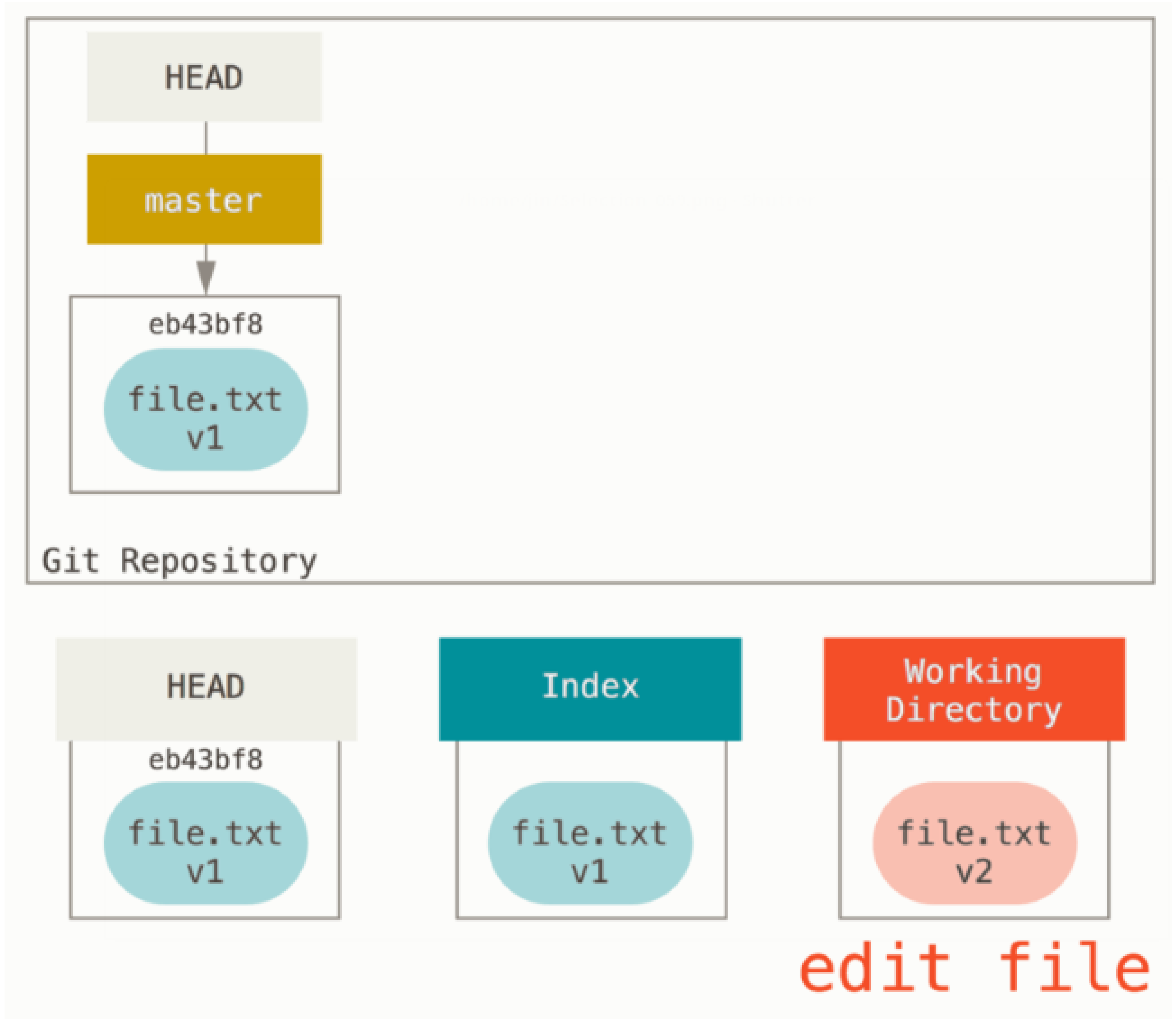
git commit


HEAD
master
eb43bf8
file.txt v1
Git Repository
HEAD
eb43bf8
file.txt v1
Index
file.txt v1
Working Directory
file.txt v1
git commit

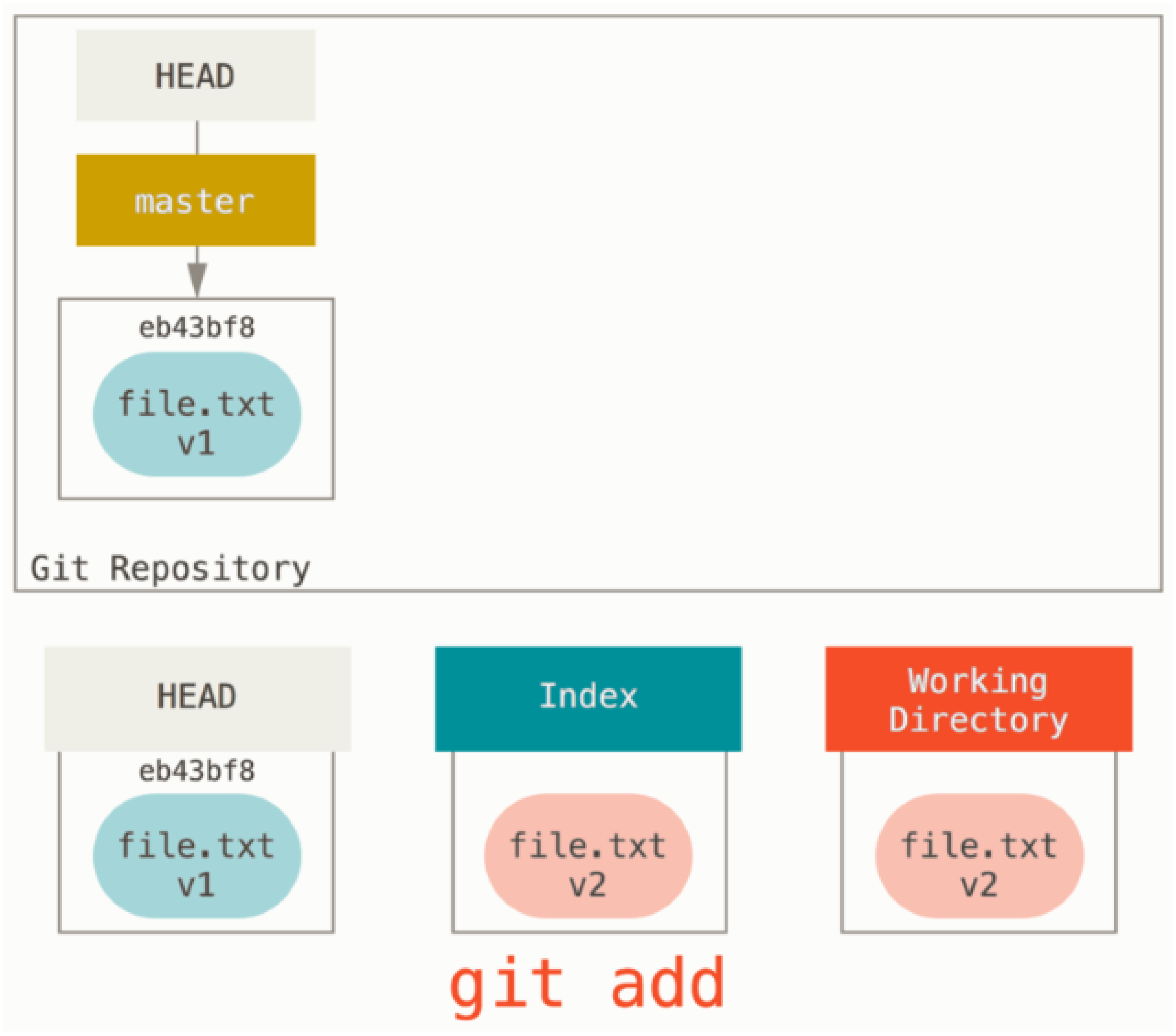
edit file


HEAD
master
eb43bf8
file.txt v1
Git Repository
HEAD
eb43bf8
file.txt v1
Index
file.txt v1
Working Directory
file.txt v2
edit file

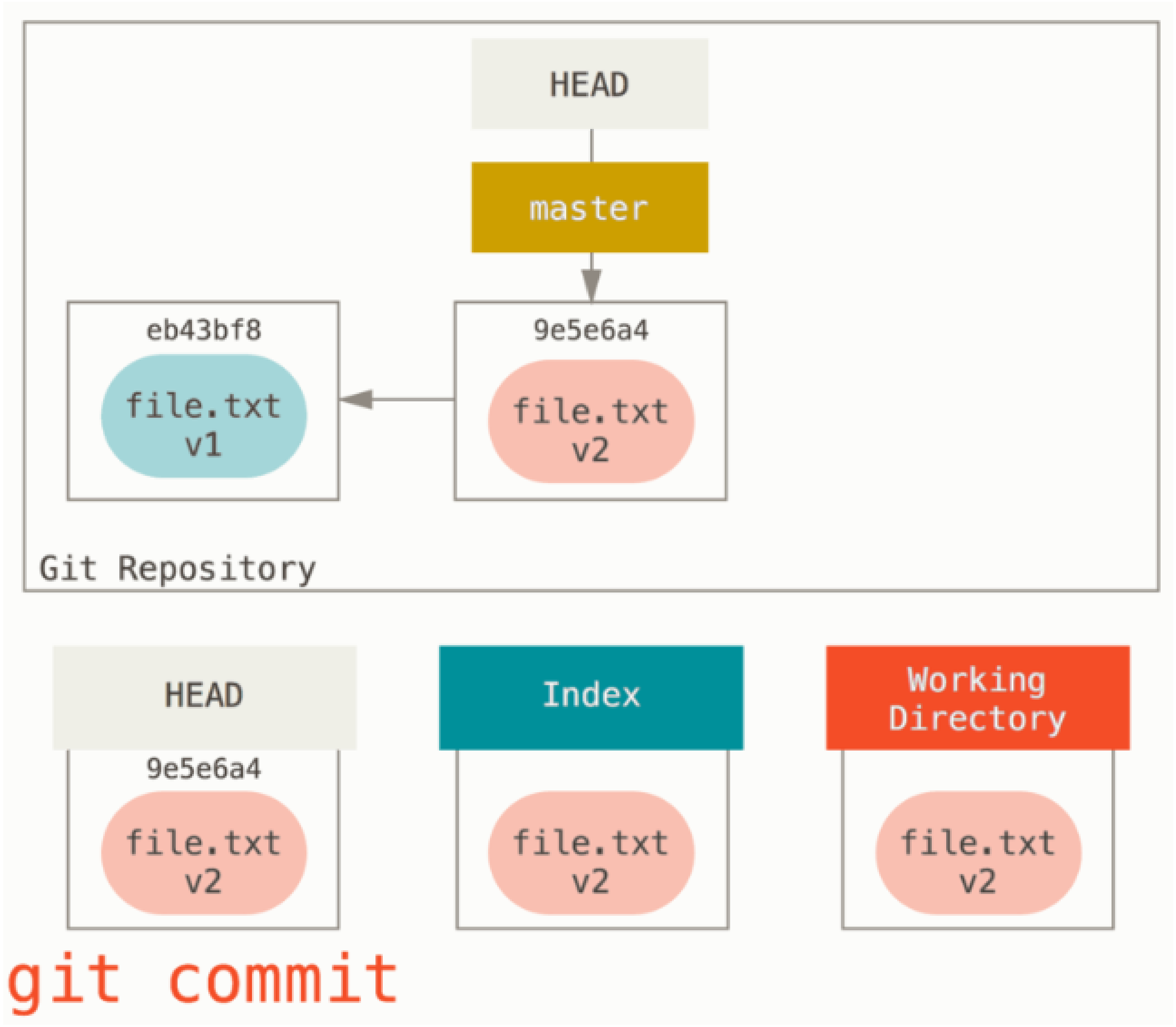
edit file


HEAD
master
eb43bf8
file.txt v1
Git Repository
HEAD
eb43bf8
file.txt v1
Index
file.txt v1
Working Directory
file.txt v2
edit file

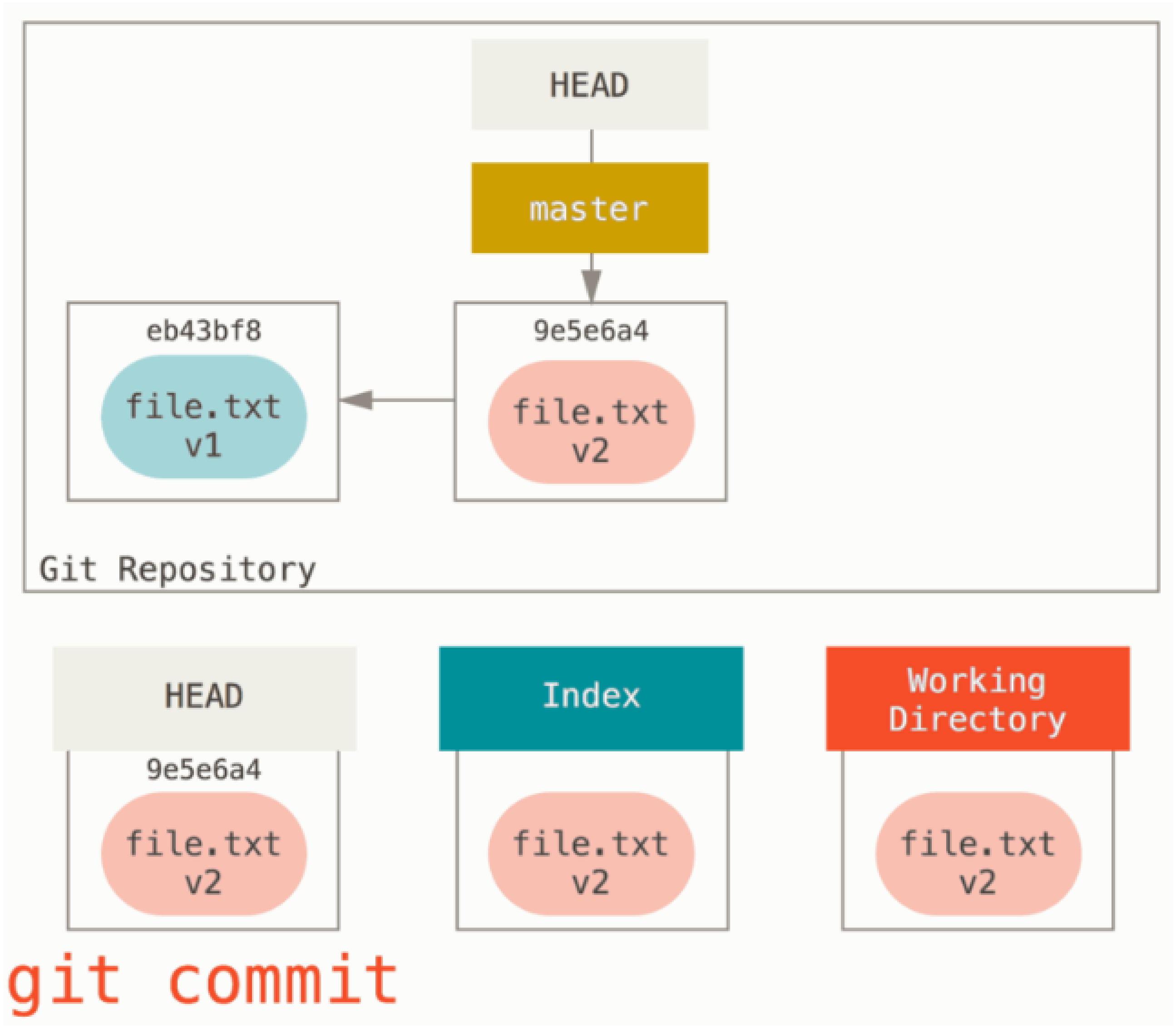
git add edited file



git add edited file



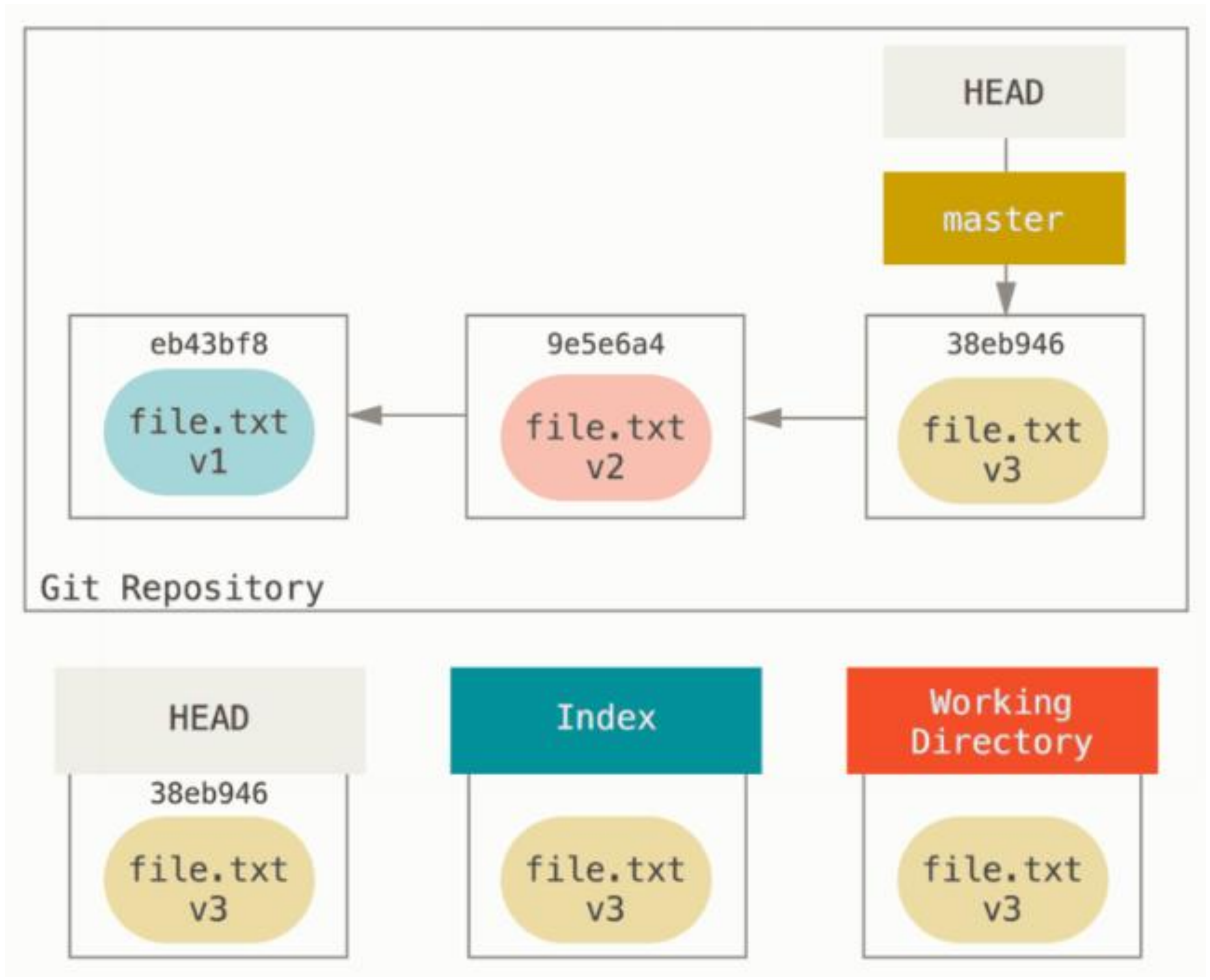
git add edited file



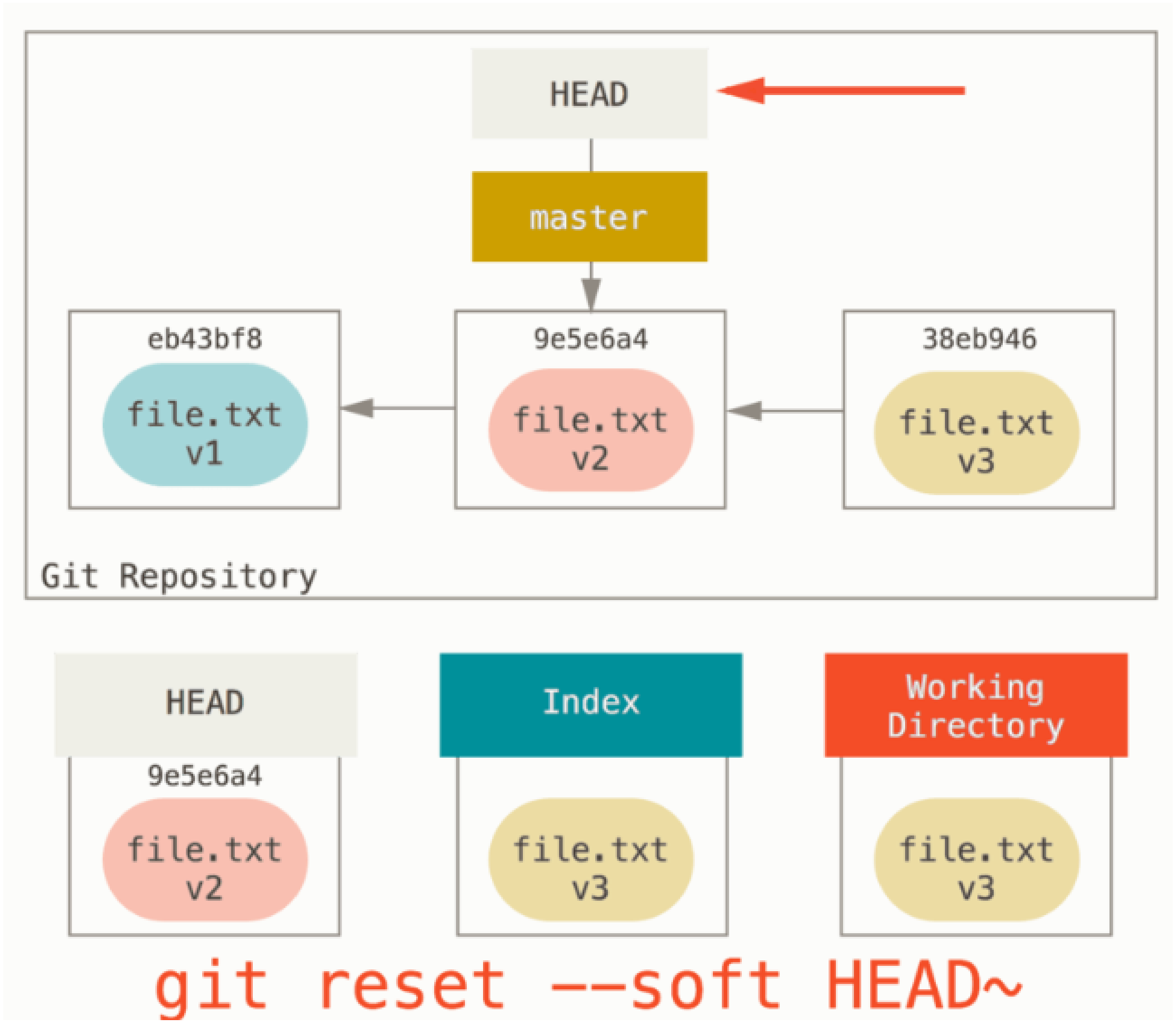
Reset

- Three steps
  - Move the branch HEAD points to (stop here if –soft)
  - Move the index look like HEAD (stop here unless –hard)
  - Move the Working Directory look like the index

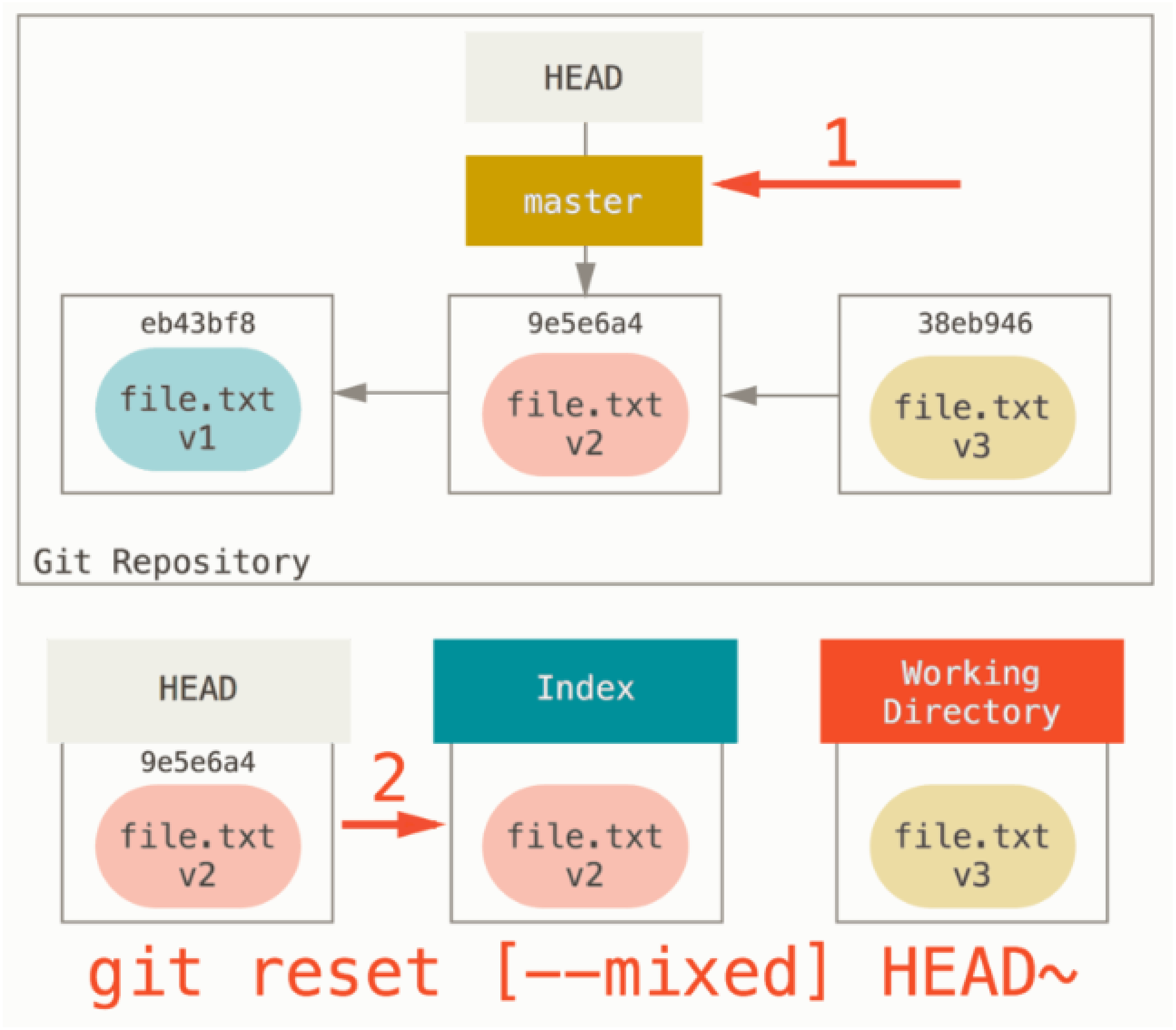
Reset


HEAD
master
eb43bf8
file.txt v1
9e5e6a4
file.txt v2
38eb946
file.txt v3
Git Repository
HEAD
38eb946
file.txt v3
Index
file.txt v3
Working Directory
file.txt v3

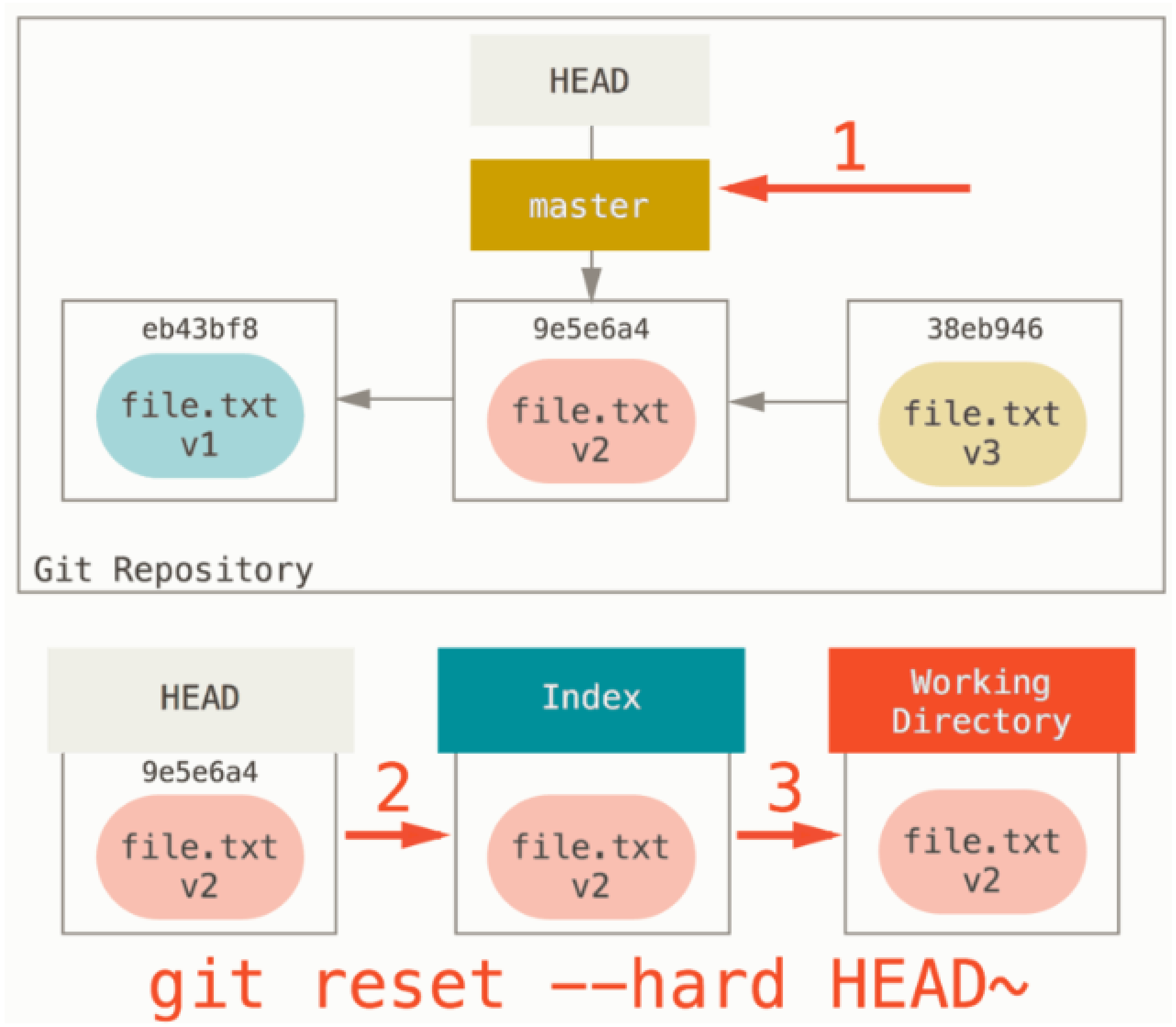
Reset


HEAD
master
eb43bf8
file.txt v1
9e5e6a4
file.txt v2
38eb946
file.txt v3
Git Repository
HEAD
9e5e6a4
file.txt v2
Index
file.txt v3
Working Directory
file.txt v3
git reset --soft HEAD~

Reset


HEAD
master
1
eb43bf8
file.txt v1
9e5e6a4
file.txt v2
38eb946
file.txt v3
Git Repository
HEAD
Index
Working Directory
9e5e6a4
file.txt v2
2
file.txt v2
file.txt v3
git reset [--mixed] HEAD~

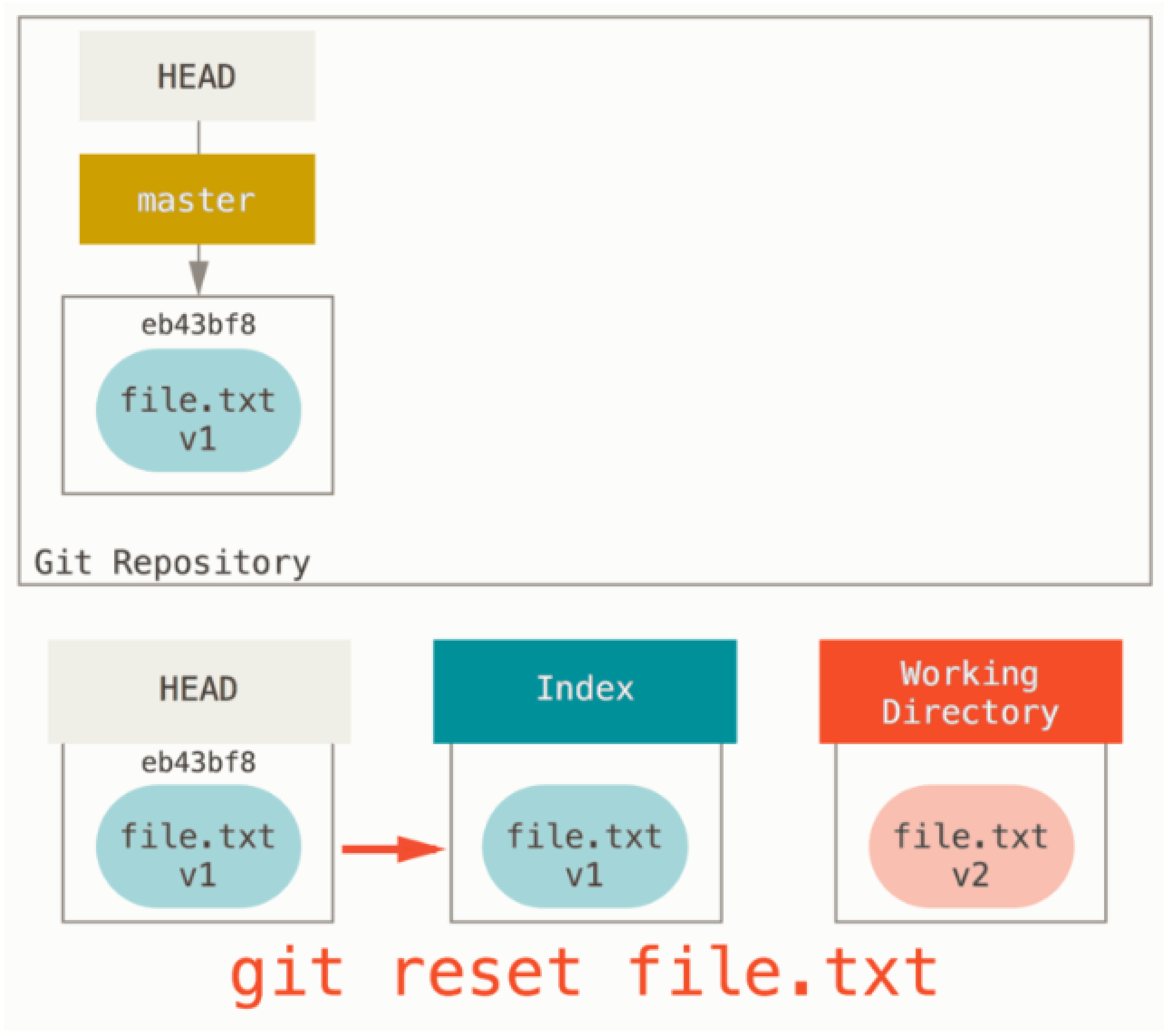
Reset


HEAD
master
1
eb43bf8
file.txt v1
9e5e6a4
file.txt v2
38eb946
file.txt v3
Git Repository
HEAD
Index
Working Directory
9e5e6a4
file.txt v2
2
file.txt v2
3
file.txt v2
git reset --hard HEAD~

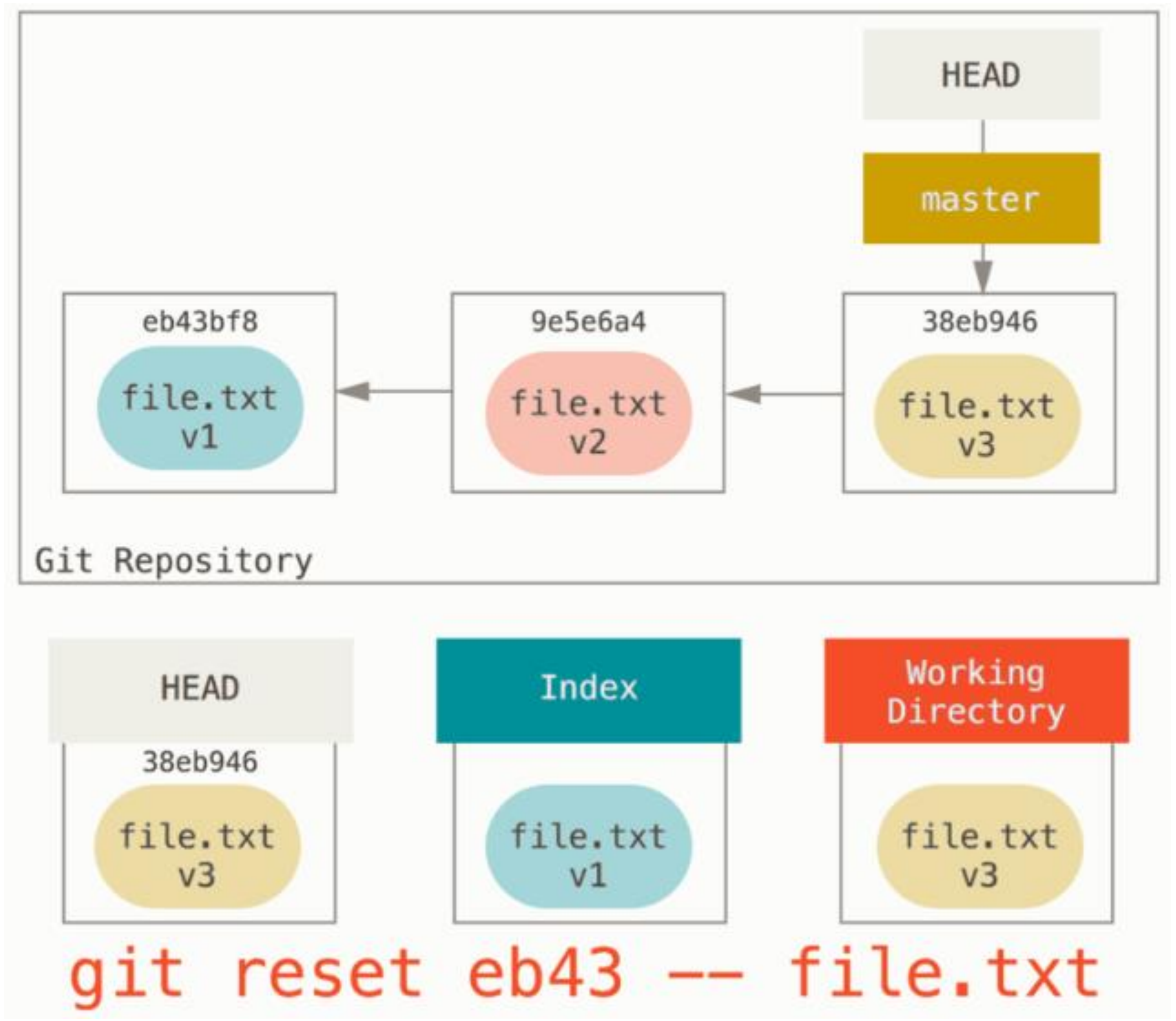
Reset with Path

- git reset file.txt
  - opposites of git add
  - Move the branch HEAD points to (skipped)
  - Make the index look like HEAD (stop here)

Reset with Path

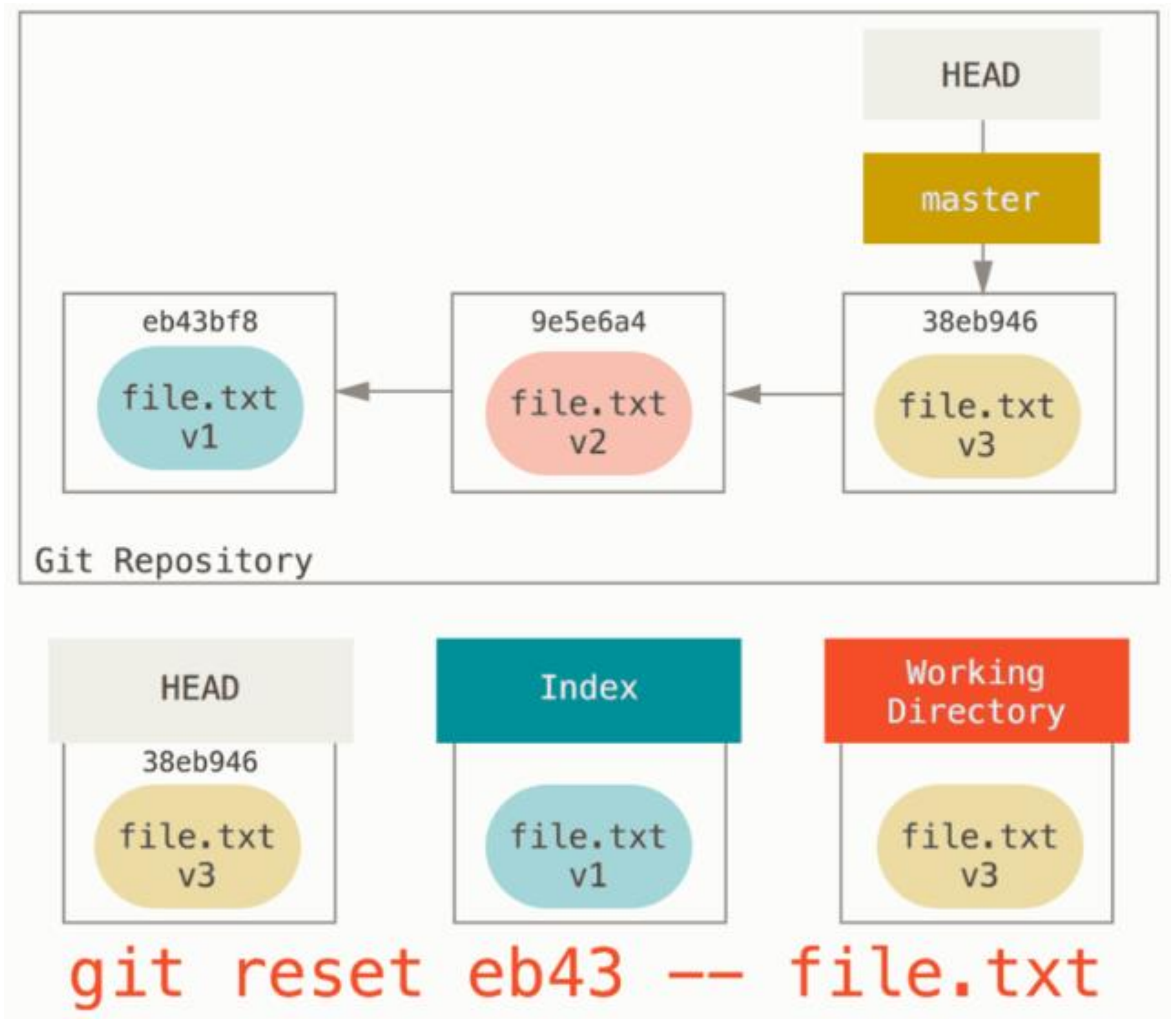


Reset with Path



```
git reset eb43 -- file.txt
```

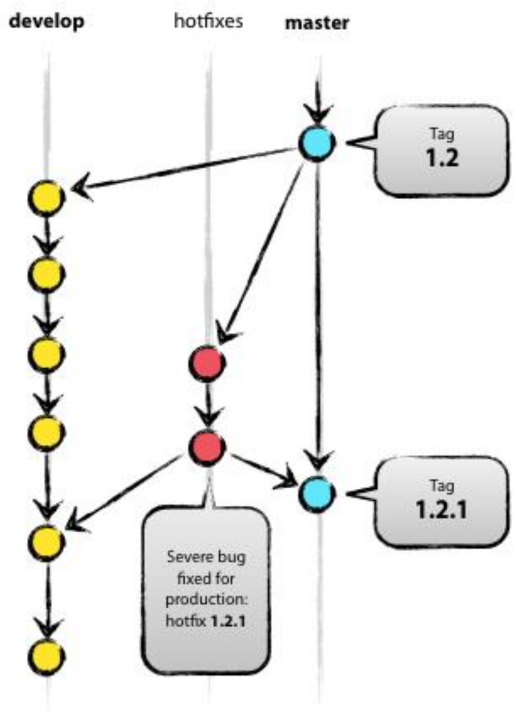
Reset with Path



Advance Features

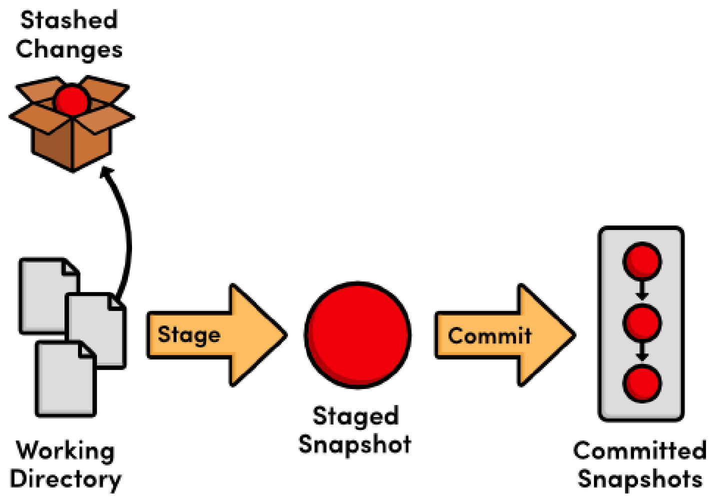
Tags

- **Adds a symbolic name**
  - **git tag <tag name> <commit name>**
- **A tag cannot be changed. However, it can be moved (See the advanced session for instructions)**



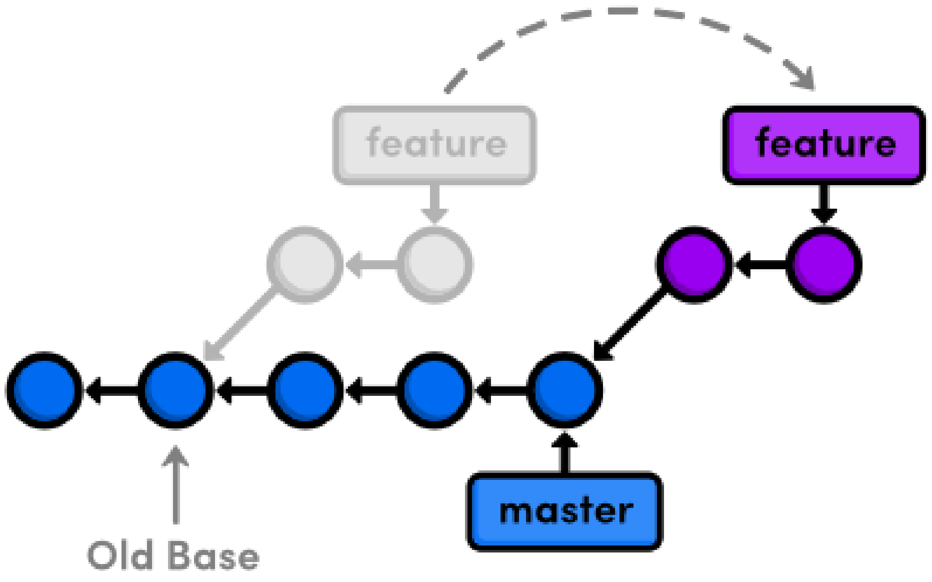
Stash

- There can be multiple stashes, each with a name.
  - git stash list
- You can apply a stash other than the last one.
  - git stash apply <stash name>
- Applying a stash does not delete it.
  - git stash drop <stash name>



Git Rebase

- The main topic of this part
- git rebase allows you to rewrite your history.
  - It alters the repository so that the commits you can see before and after its usage are different.
  - It can change, merge, split, add, remove, modify commits



Git Rebase

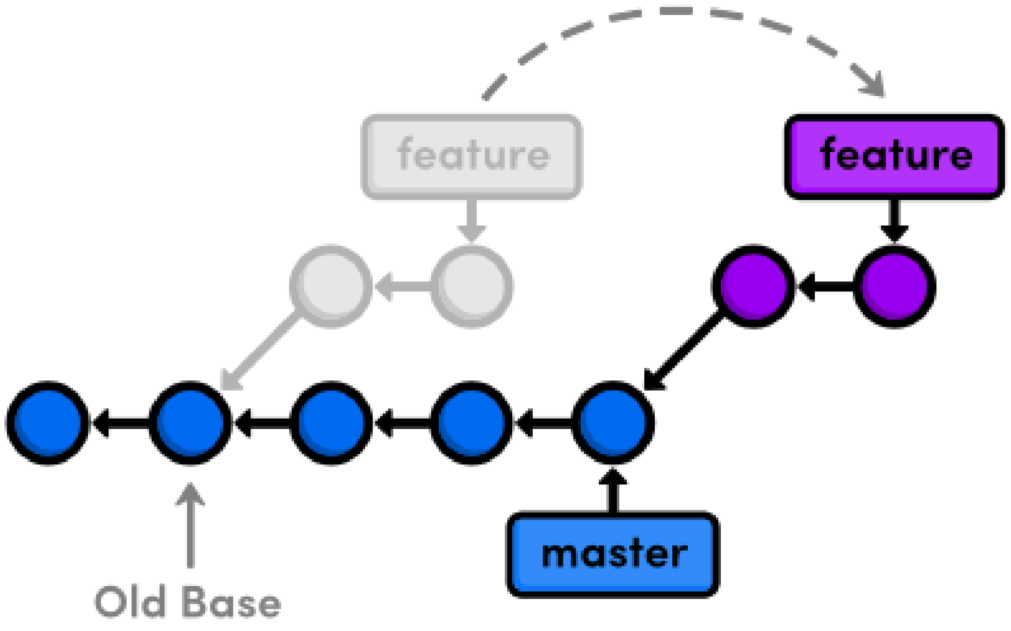
- **IT IS A DANGEROUS COMMAND!**
  - **If you change history, you will break merges for EVERYONE that has already ‘pulled’ your branch.**
  - **git tries to protect you from it**
  - **If it detects that pushing will probably cause it, it tries to stop you.**
  - **You will have to use a different syntax to go ahead anyway.**
  - **However, you should not rely on this**

Git Rebase

- No commit should EVER be rebased if it is already public.
  - If you have already “pushed” it, or
  - If you have “pulled” or “fetched” it, or
  - These include commits inherited from branches created from “pushed” or “pulled” ones.
- No commit should be rebase if:
  - You have merged it on a different branch
- A suggestion:
  - Only use it on private branches you have not merged anywhere.

Git Rebase

- **Two main usages:**
  - **Batch**
  - **Interactive**



Git Rebase – Batch Usage

- Takes a branch, and modify it to make it look like the branch never existed
- Example:

  -
    ```
    A-B-C-D            master
         \-E-F-G       topic
    ```
- Becomes:
  - A-B-C-D-E’-F’-G’   master
- From ‘topic’ branch: ‘git rebase master’

Git Rebase – Batch Usage

- **Git rebase does an actual merge**
- **Merges may have conflict. You have three choices**

- **Solve the conflicts:**
  - **git add the resolutions**
  - **git rebase –continue**
- **Skip this commit**
  - **git rebase –skip**
- **Abort the rebase**
  - **git rebase --abort**

Git Rebase – Interactive

- **git rebase –i <commit>**
  - **<commit> should be the commit BEFORE the first one you wish to alter.**
  - **Will open your $EDITOR with the following buffer:**

Git Rebase – Interactive

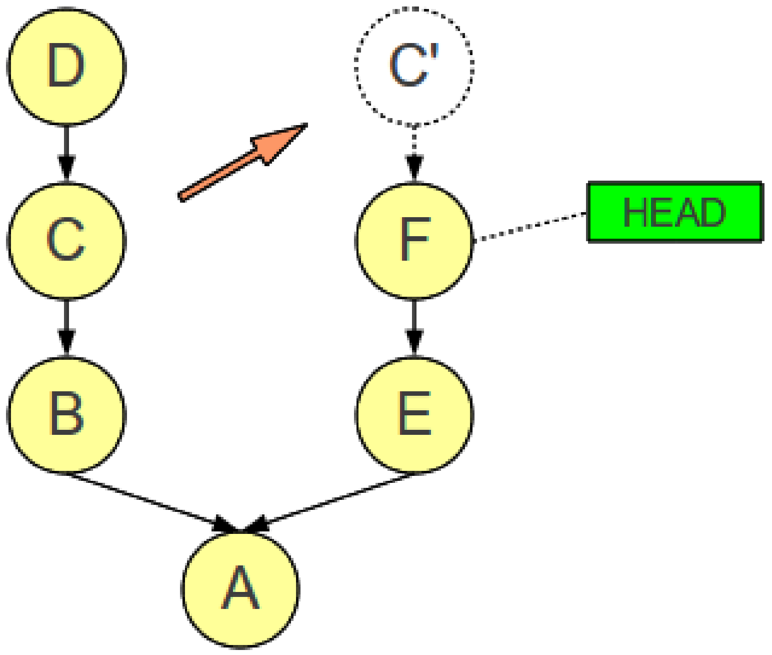
```
pick d6a7c25 Added README file.
pick dce0696 changed README file.
pick 8e5ba04 Makefile
pick b42cffd rgheguie
pick 2306a37 new line.

# Rebase 68bcfea..2306a37 onto 68bcfea
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log message
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
```

Git Rebase – Interactive

- **If you do nothing, or remove all the lines, nothing happens**
- **squash – This commit gets deleted, but its contents are added to the previous commit. Commit messages are merged.**
- **fixup – Like squash, but the commit message gets lost**
- **reword – change the commit message**
- **edit – stop there to allow modifying the commit**

Git Cherry Pick

- The ability to get a specific commit from a branch and merge *only that commit* on a different branch.
  - Also known as: “backport fix for bug ##### to an older branch.”
  - This is a merge operation. Conflicts may occur and will have to be resolved normally



Git Cherry Pick

- **git cherry-pick <commit>**
  - **Will merge commit <commit> on the current branch.**

- **If you are merging from a public branch, add "-x", i.e.:**
  - **git cherry-pick –x <commit>**
  - **Reason: Will add a note to the commit message specifying the source of the cherry-pick**