Open Source SW & Lab - Summer 2023
9. Git-Tools (2)

Walid Abdullah Al
Computer and Electronic Systems Engineering
Hankuk University of Foreign Studies

Based on:
Pro Git (2022) by Scott Chacon, Ben Straub

Computer Vision Lab
Hankuk University of Foreign Studies

Previously,

- **We covered some useful Git tools such as:**
  - Revision selection
    - Using commit hash
    - Branch reference
  - Interactive staging
    - Selective staging of files
    - Selective staging of patches

Stashing

- **Suppose**
  - You're working on branch-A
  - You need to switch to branch-B
  - But don't want to commit the half-done works on branch-A
- **Stashing (`git stash`)**
  - Saves the messy state of your work
  - On a stack of unfinished changes
  - That you can reapply anytime

Contd.

- **Suppose, you have the following unfinished works**

```
$ git status
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   index.html

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   lib/simplegit.rb
```

- **Now, want to switch branches without committing**
- **Therefore, stash the changes**

```
$ git stash
Saved working directory and index state \
  "WIP on master: 049d078 Create index file"
HEAD is now at 049d078 Create index file
(To restore them type "git stash apply")
```

```
$ git status
# On branch master
nothing to commit, working directory clean
```

Contd.

- **To check the stack of stashes:**

```
$ git stash list
stash@{0}: WIP on master: 049d078 Create index file
stash@{1}: WIP on master: c264051 Revert "Add file_size"
stash@{2}: WIP on master: 21d80a5 Add number to log
```

- **To apply a stash:**
  - git stash apply
  - git stash apply stash@{2}

```
$ git stash apply
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.html
        modified:   lib/simplegit.rb

no changes added to commit (use "git add" and/or "git commit -a")
```

- **To remove a stash:** git stash drop

Branch from stash

- **Suppose:**
  - You stashed some work on branch-A
  - Continued committing on the same branch
  - Now, reapplying stash on branch-A can be problematic
  - (merge conflict)
- **git stash branch <branchname> <stash-ref>**
  - Creates a new branch on the stash-commit
  - And reapplies the stashed changes on the new branch
- **Note:**
  - Untracked files are not stashed
  - Use –u to stash the untracked ones also
- **Finally, if you do not want to stash but just clean your working directory**
  - $ git clean

Rewriting history

- **Changing**
  - Commit meta/info
  - Commit file-contents
- **Changing the last commit**
  - $ git commit --amend
- **Changing previous 3 commits**

```
$ git rebase -i HEAD~3
```

Reset

- **Think of Git being a content manager of three different trees**

Tree	Role
HEAD	Last commit snapshot, next parent
Index	Proposed next commit snapshot
Working Directory	Sandbox

Contd.

- **Think of Git being a content manager of three different trees**

Tree	Role
HEAD	Last commit snapshot, next parent
Index	Proposed next commit snapshot
Working Directory	Sandbox

Contd.

- **HEAD**
  - Pointer to the current branch reference
  - Pointer to the last commit made on that branch
  - Parent of the next commit
  - **Snapshot of the last commit on the current branch**

```
$ git cat-file -p HEAD
tree cfda3bf379e4f8dba8717dee55aab78aef7f4daf
author Scott Chacon  1301511835 -0700
committer Scott Chacon  1301511835 -0700

initial commit

$ git ls-tree -r HEAD
100644 blob a906cb2a4a904a152...   README
100644 blob 8f94139338f9404f2...   Rakefile
040000 tree 99f1a6d12cb4b6f19...   lib
```

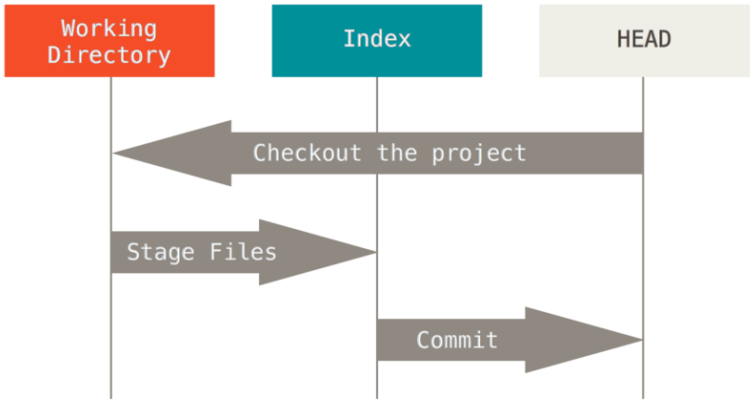
Contd.

- **Index**
  - Proposed next commit
  - Can be referred to as the "staging area"
- **Working directory**
  - Other two trees store contents in .git folder
  - Working tree unpacks its contents as files so that we can access, edit, delete, etc.
  - This is the directory that we see in our file system browser

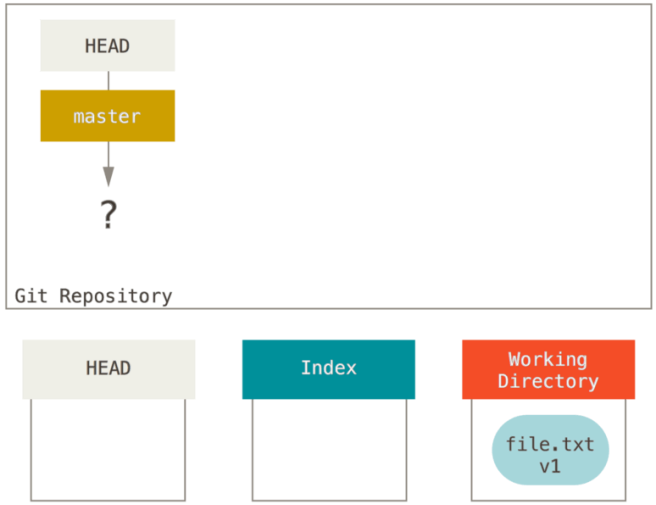
```
$ tree
.
├── README
├── Rakefile
└── lib
    └── simplegit.rb

1 directory, 3 files
```

Contd.

- **Git workflow**



- **Suppose**
  - A new directory with a new file (v1 of the file)
  - Now, run `git init`



Contd.

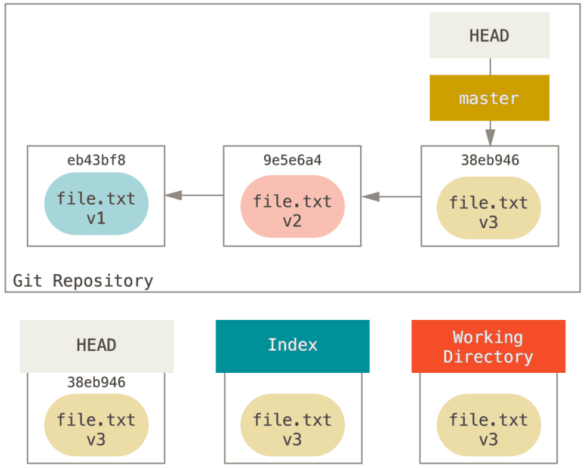

HEAD
master
?
Git Repository
HEAD
Index
Working Directory
file.txt v1
git add
git commit
edit file
eb43bf8
9e5e6a4
file.txt v2

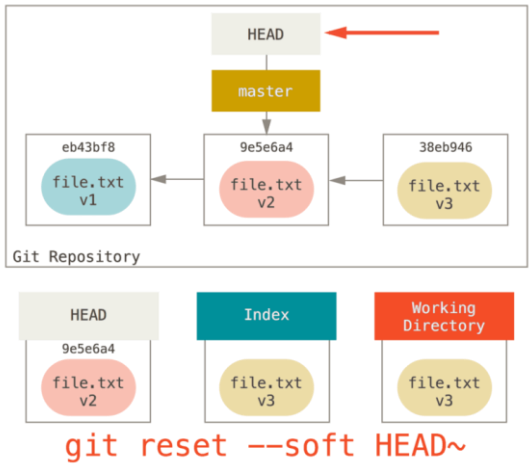
Contd.

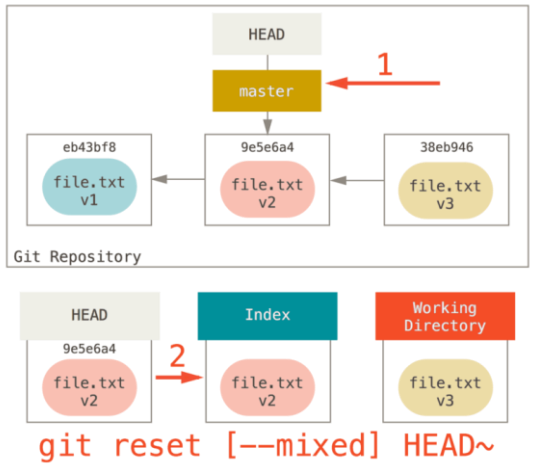
- **If all the files are same across HEAD, INDEX, WORKING trees,**
  - Git status outputs nothing (working tree clean)
- **When checkout a branch**
  - HEAD → last commit of that branch
  - Index → snapshot of that commit
  - Copies the Index contents to working directory

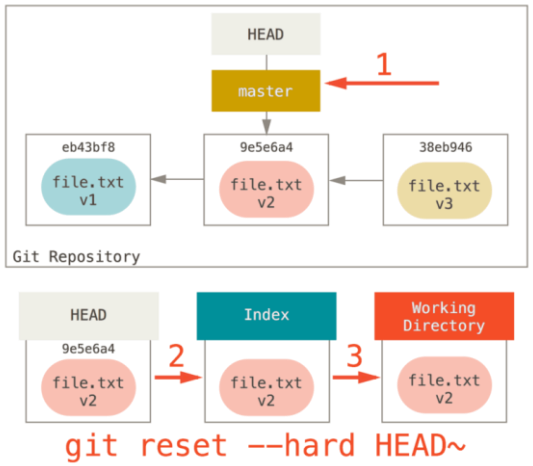
Contd.

- **The role of reset (3 steps)**
  - Step-1: move HEAD (--soft)
  - Step-2: Update Index (--mixed; default)
  - Step-3: Update the working directory (--hard)



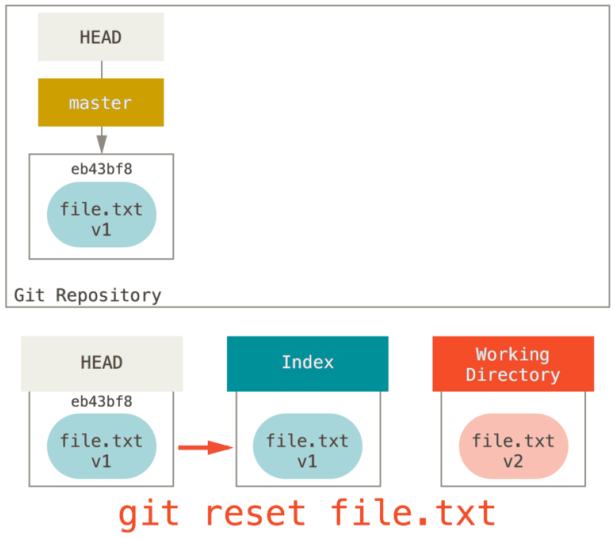


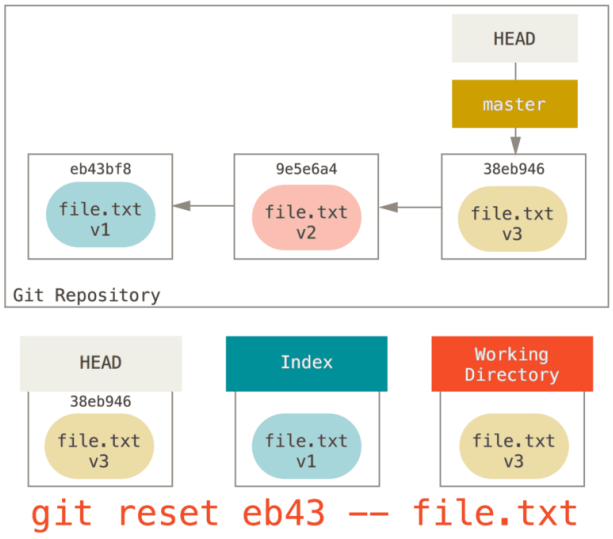




Contd.

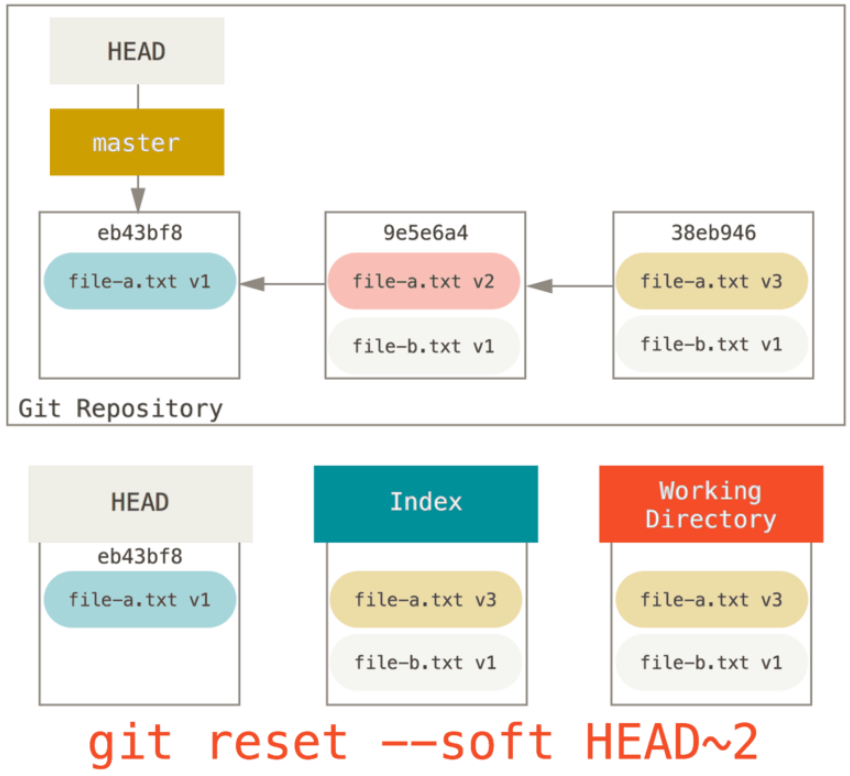
- **Reset with a path**
  - Step-1 will be skipped

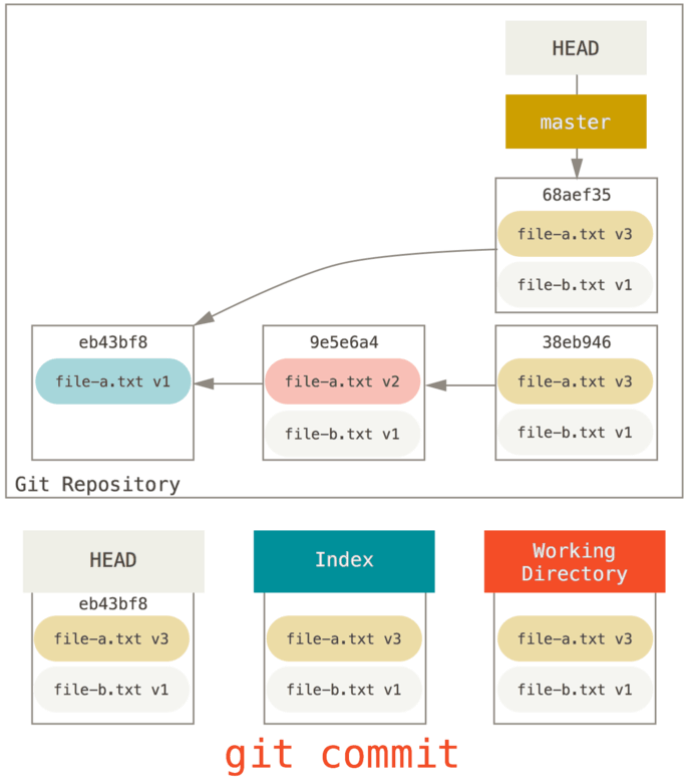




Contd.

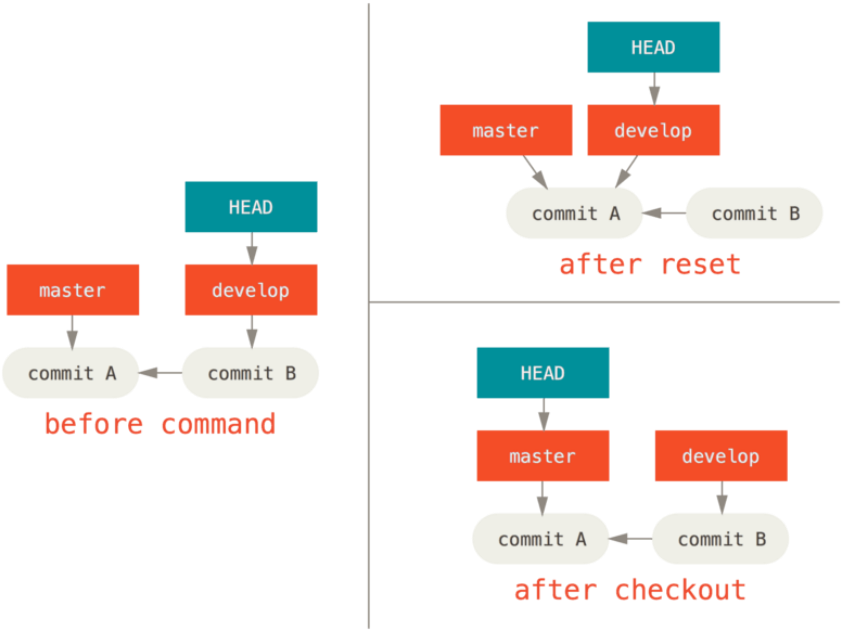
- **Squashing**
  - git reset --soft **and** git commit





Contd.

- **Checkout**
  - git reset --hard; (updates all the three trees)
  - However, performs trivial merge in the working directory
    - Any changes will not be updated
    - Unmodified files will be updated
  - And, no branch pointer update



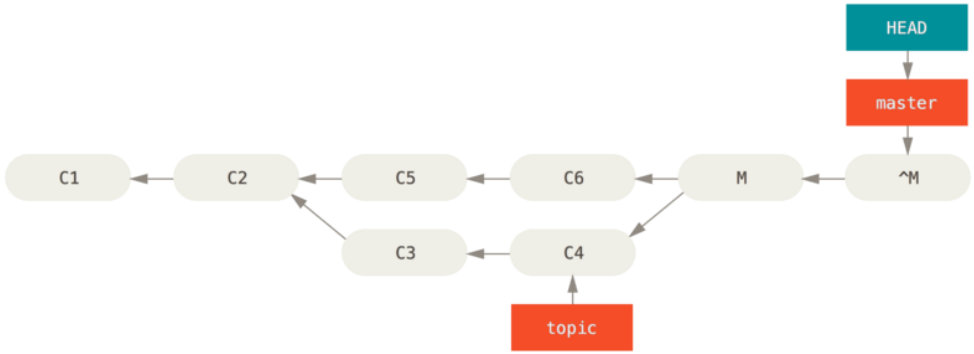
Contd.

- **Summary**

	HEAD	Index	Workdir	WD Safe?
Commit Level				
`reset --soft [commit]`	REF	NO	NO	YES
`reset [commit]`	REF	YES	NO	YES
`reset --hard [commit]`	REF	YES	YES	**NO**
`checkout <commit>`	HEAD	YES	YES	YES
File Level				
`reset [commit] <paths>`	NO	YES	NO	YES
`checkout [commit] <paths>`	NO	YES	YES	**NO**

Advanced merging

- **Abort a merge (**while in a merge conflict**)**
  - git merge --abort
  - git reset --hard HEAD
- **Ignoring Whitespace**
  - $ git merge -Xignore-space-change <branch>
  - $ git merge -Xignore-all-space <branch>
- **Undoing a merge**
  - Fix the reference: $ git reset --hard HEAD~
  - Revert commit: $ git revert -m 1 HEAD



Today's labwork

- Complete the UI part/prototype of your project
  - (if you haven't done it already)
- Let's take the example of our complex number operation

```
while True:
    op, oprnd1, oprnd2 = input_expression()
    if oprnd1=='q':
        print("thank you")
        break
    res = apply_operation(op, oprnd1, oprnd2)
    print_result(res)
```

```
def input_expression():
    print("input expression:")
    expr = input()

    op, oprnd1, oprnd2 = '+', 0, 0
    return op, oprnd1, oprnd2

def apply_operation(op, oprnd1, oprnd2):
    res=0
    return res

def print_result(res):
    print("result:", res)
```

Submit

- Screenshot of your prototype/UI running window.
- + your repository link