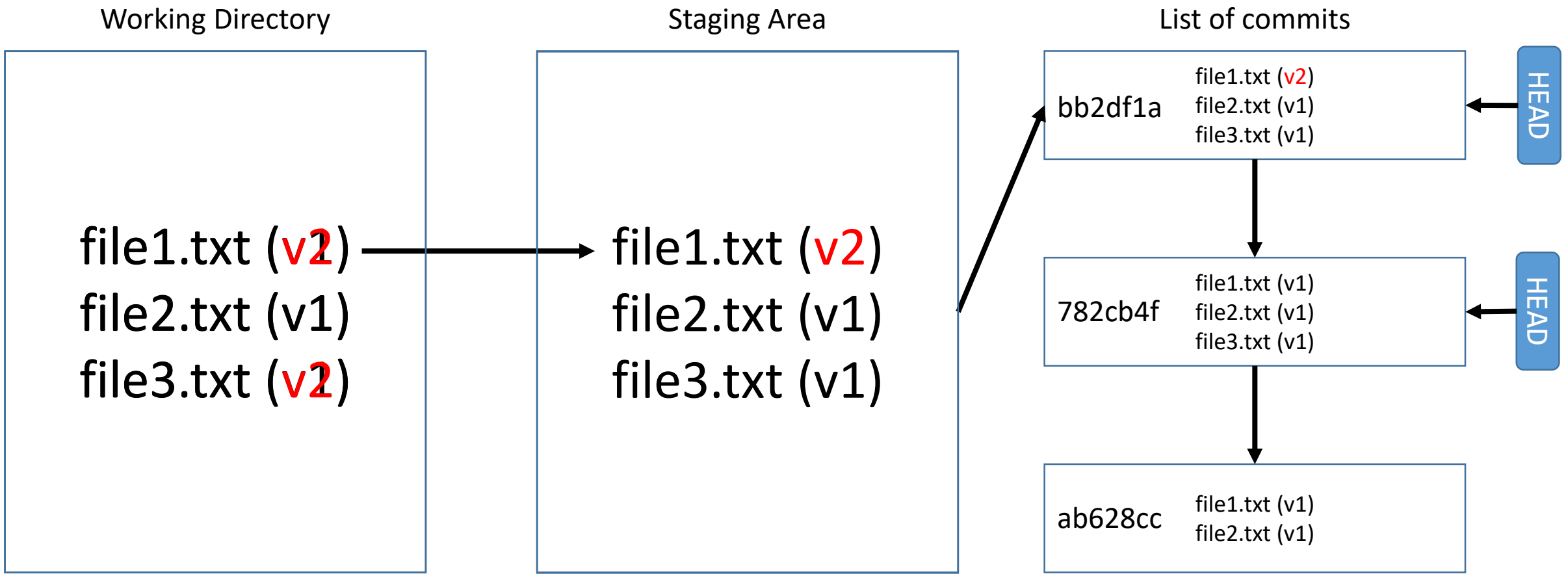
Lecture 4
More on Commits and Branches

Homework 3 Review

Review: The Git Commit Workflow (Edit, Add, Commit)

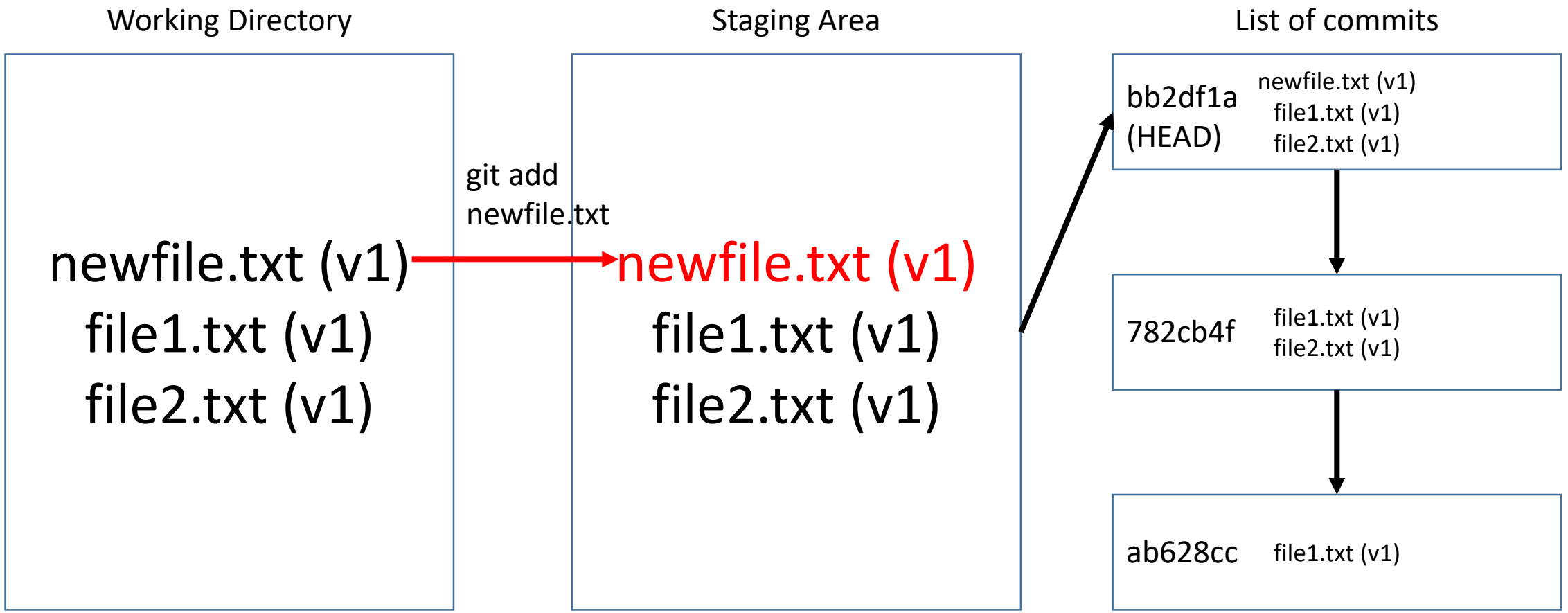


1. Make changes to files
`vim file1.txt file3.txt`

2. Add changes to the staging area
`git add file1.txt`

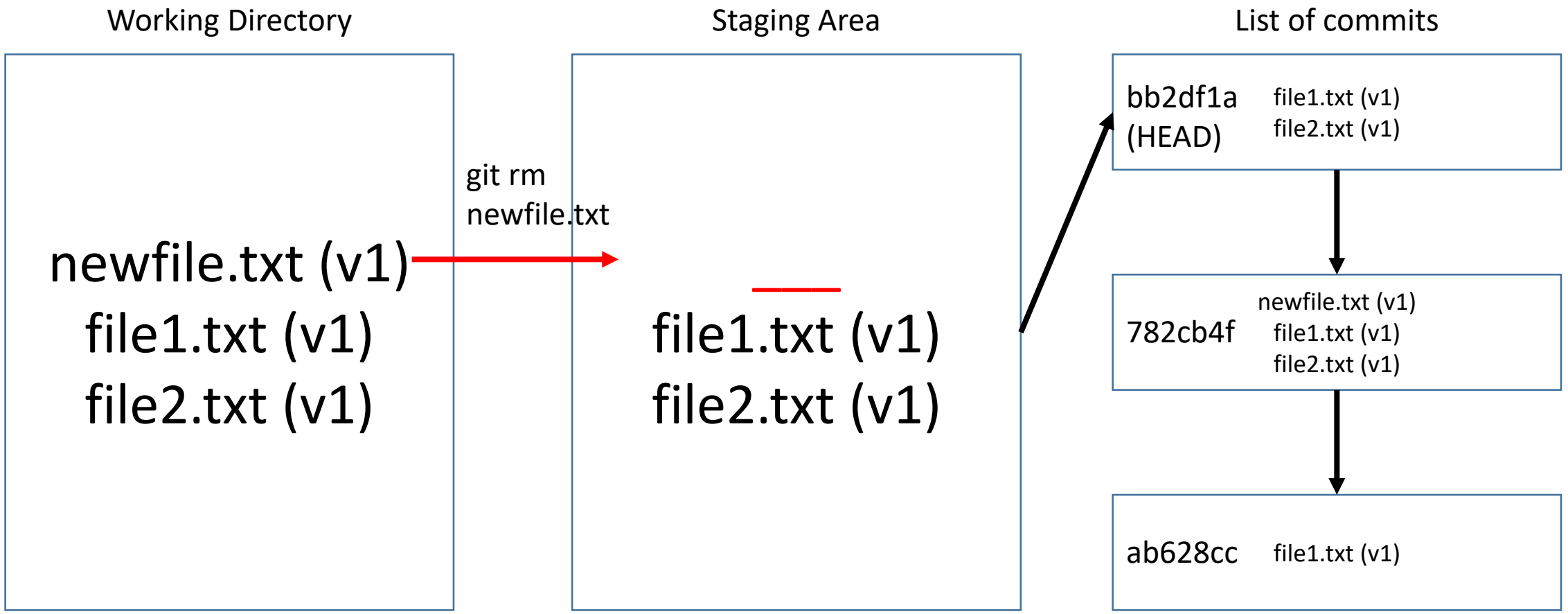
3. Commit changes in staging area
`git commit -m “fixed bug in file1.txt”`

What about new files?



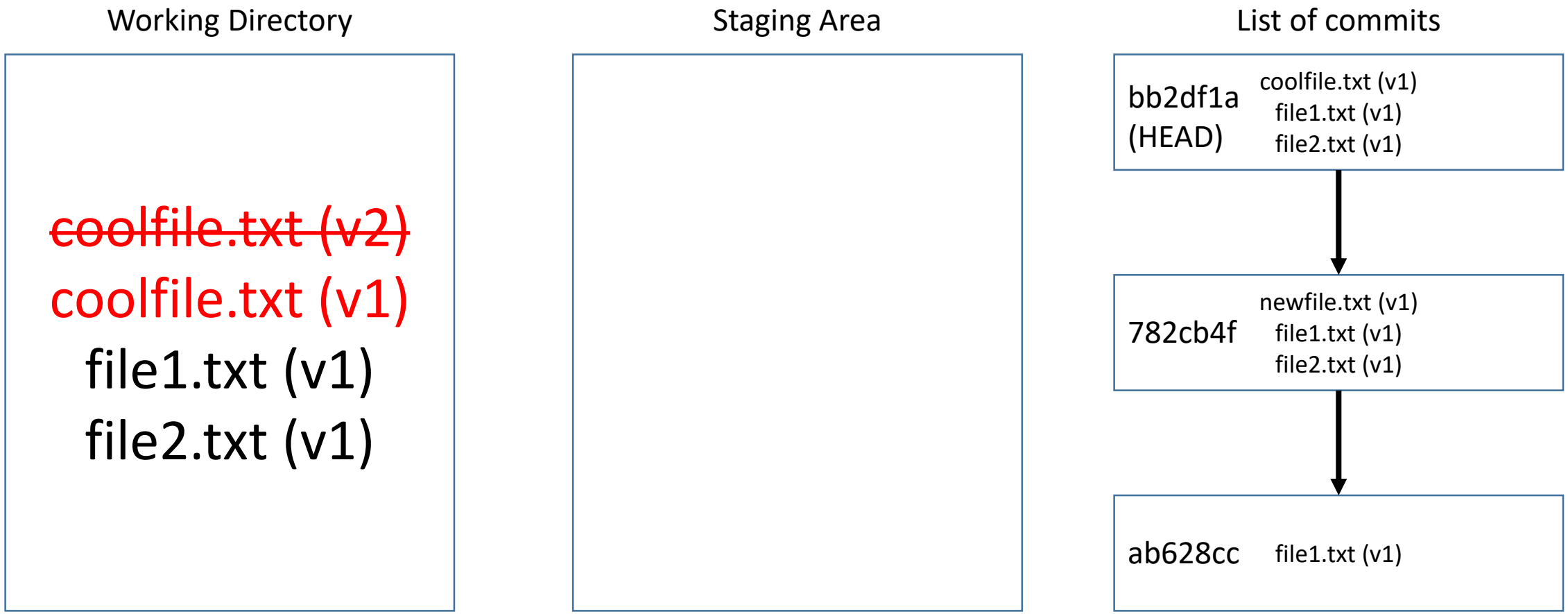
No difference from an edit, use git add newfile.txt.

What about removing files?



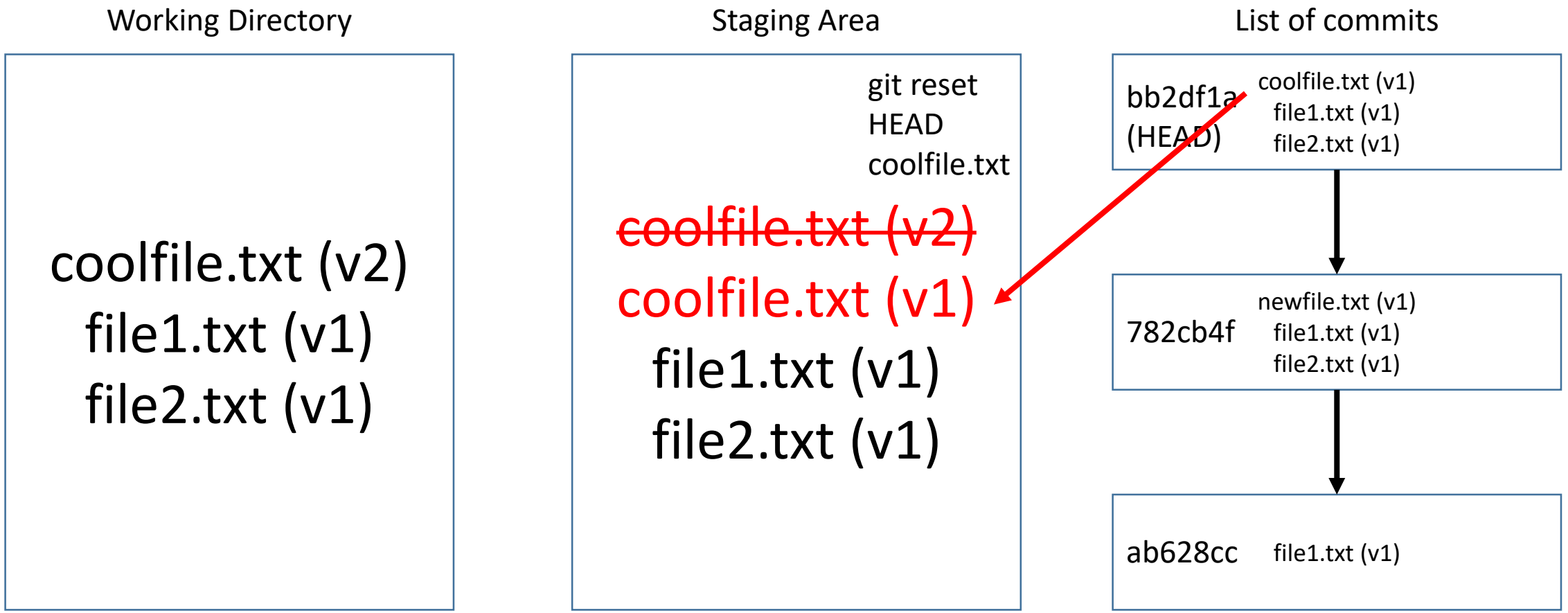
git rm newfile.txt (also deletes newfile.txt from working directory!)

What if I want to undo changes in the Working Dir?



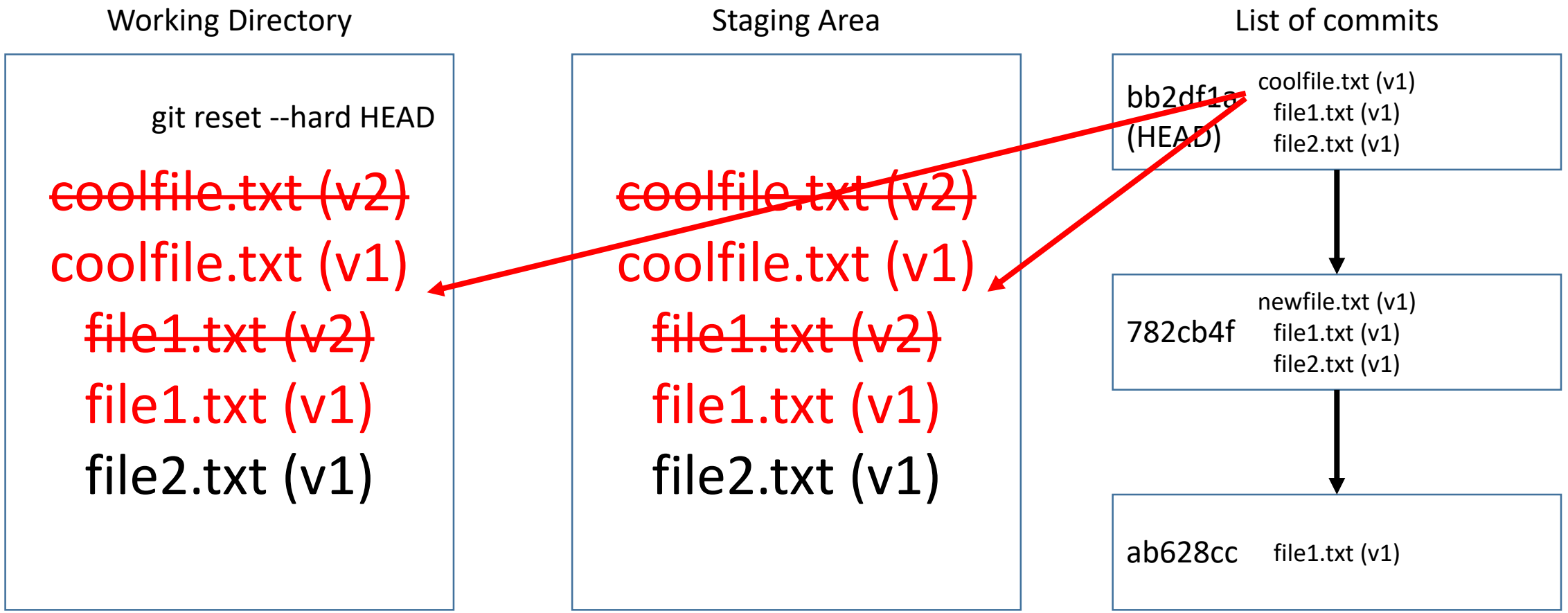
git checkout -- coolfile.txt (Note staging area is unaffected)

What if I want to ‘unstage’ a file?



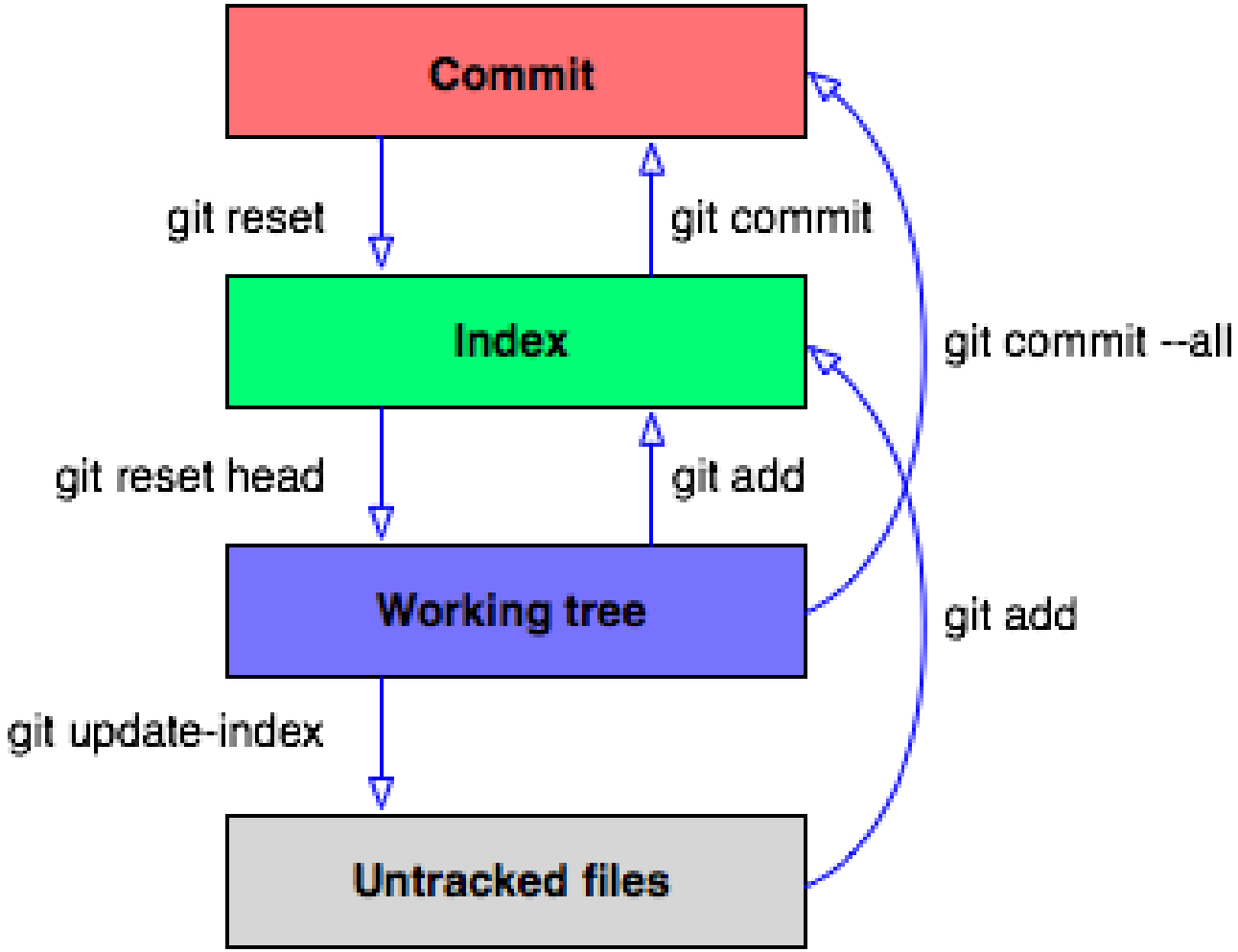
git reset HEAD coolfile.txt (Note WD is unaffected)

What if I want to start over and go back to exactly what the HEAD looks like (in both WD and SA)?



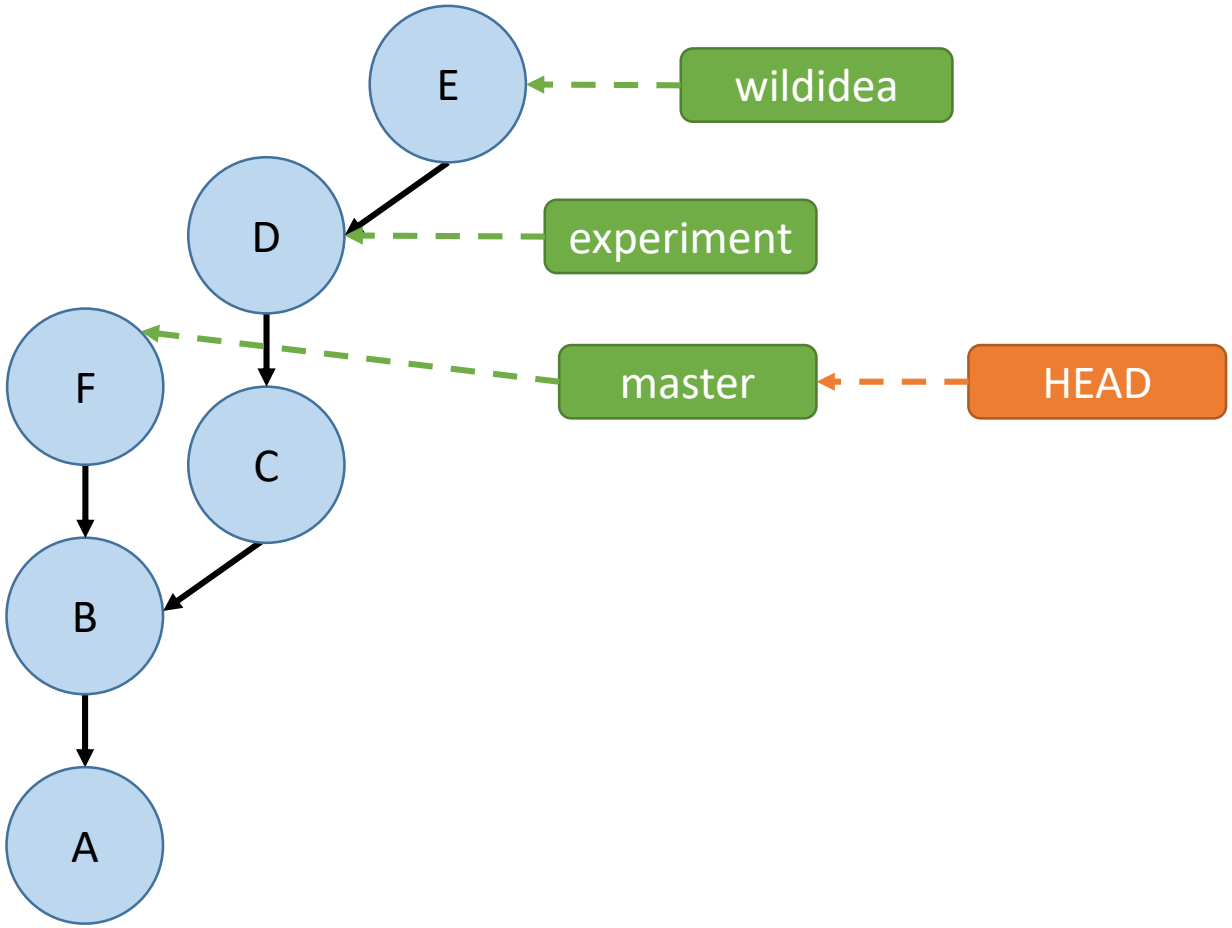
git reset --hard HEAD (overwrites entire WD!)

Summary


Commit
git reset
git commit
Index
git commit --all
git reset head
git add
Working tree
git add
git update-index
Untracked files

Last Time

- Branches are pointers to specific commits
- Branches allow us to create commit histories that diverge
- We can merge diverged histories back together

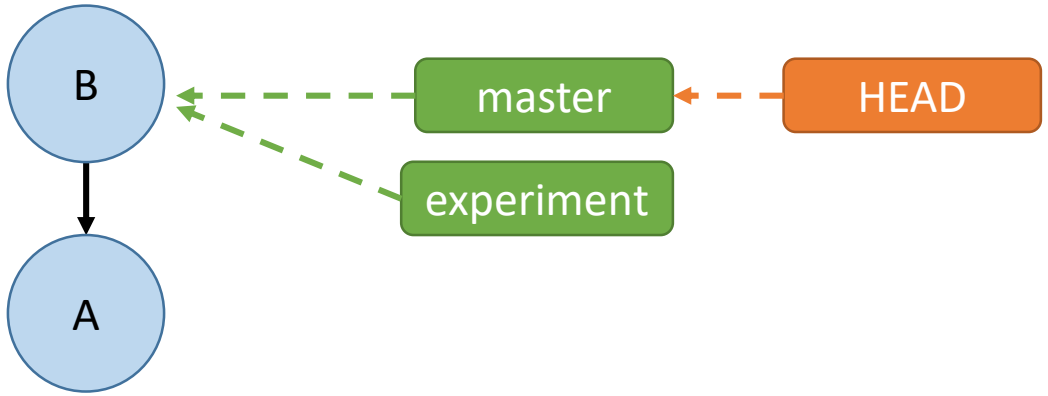


`git branch <newbranchname>`

Example use:

```
git branch experiment
```

- Creates a new branch called “experiment” that **points** to wherever you are right now (i.e. wherever HEAD is right now)

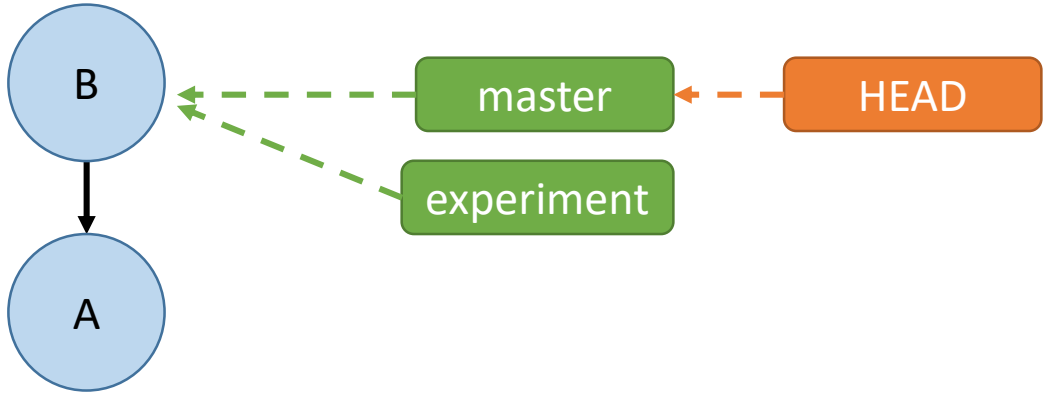


`git checkout <branchname>`

Example use:

```
git checkout experiment
```

Switches the HEAD to the branch named “develop”

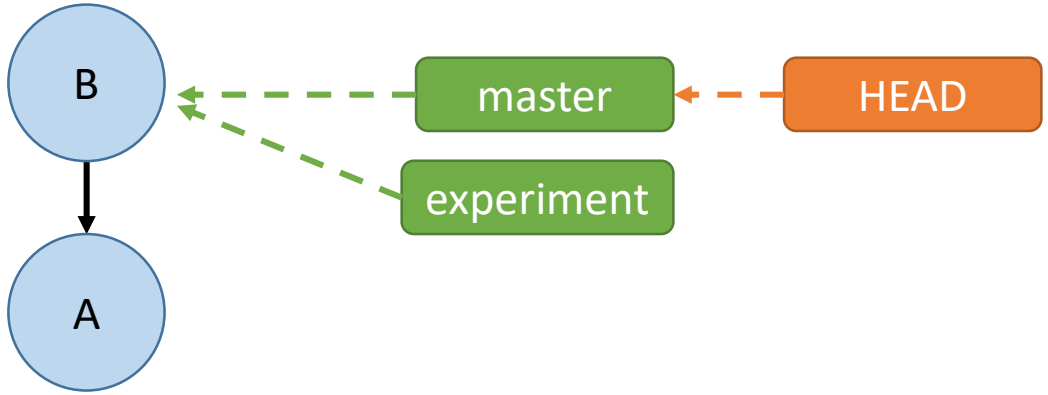


`git reset --hard <commit_hash>`

Example use:

```
git reset --hard HEAD~
```

Moves the current branch to point to a different commit



Naming Commits Relative to the HEAD

`<commit-ish>~`: The parent of the commit

`<commit-ish>~n`: The nth parent of the commit

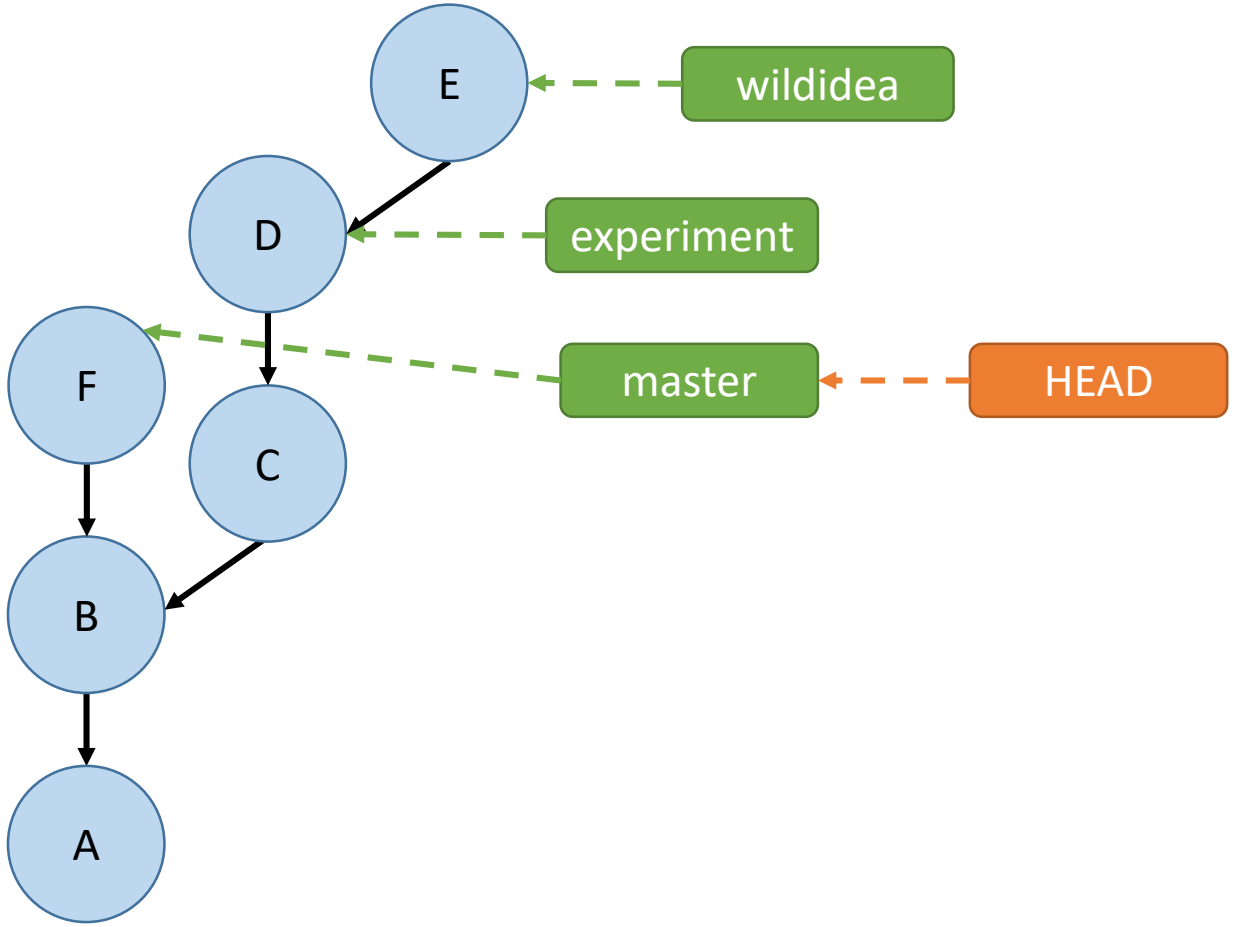
`<commit-ish>` is anything that is or points to a commit:

- short hash `a39dcf5`
- branch name
- `HEAD`

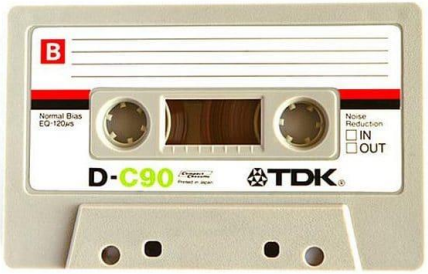
Merging

`git merge experiment`

“Will replay the changes made on the experiment branch since it diverged from master (i.e. B) until its current commit (D) on top of master, and record the result in a new commit along with the names of the two parent commits.” (from git help merge)

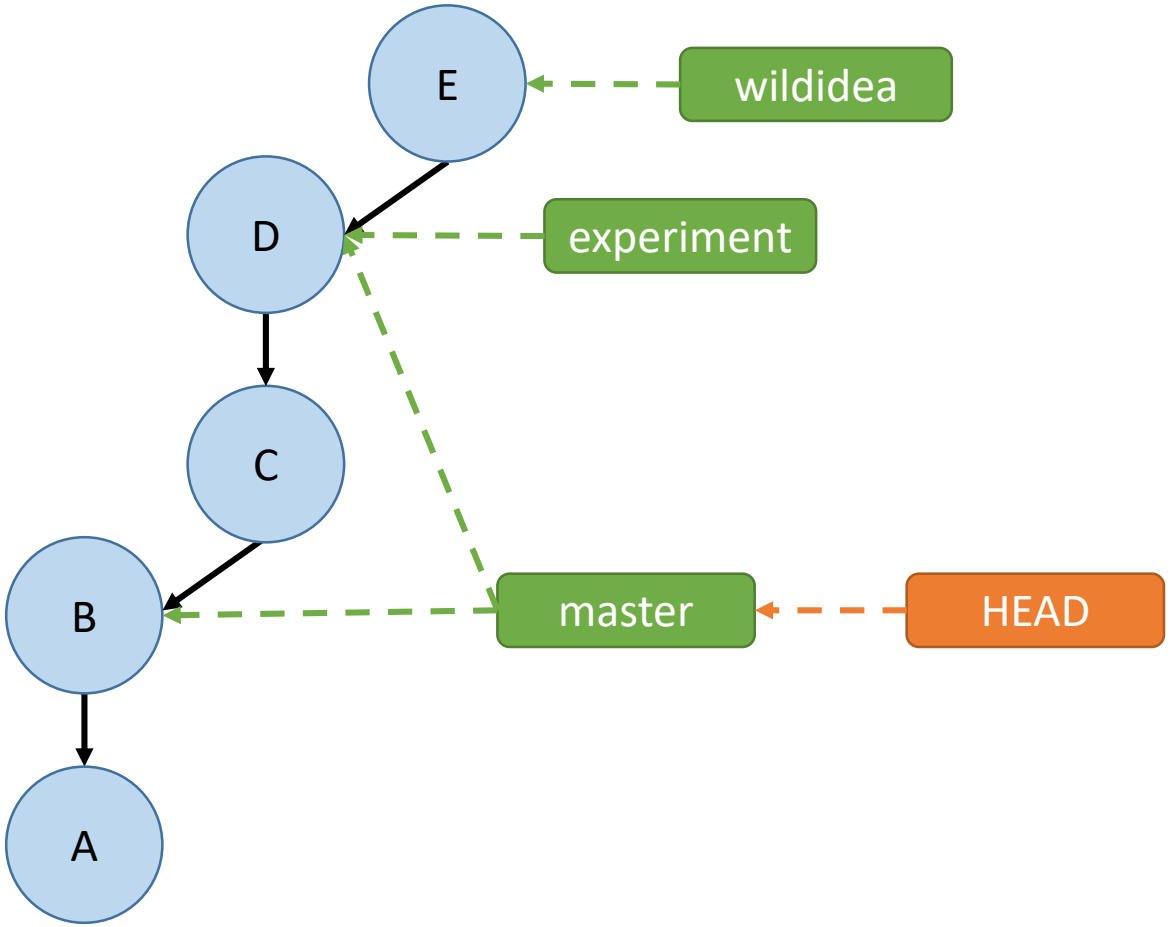


Fast Forward Merges



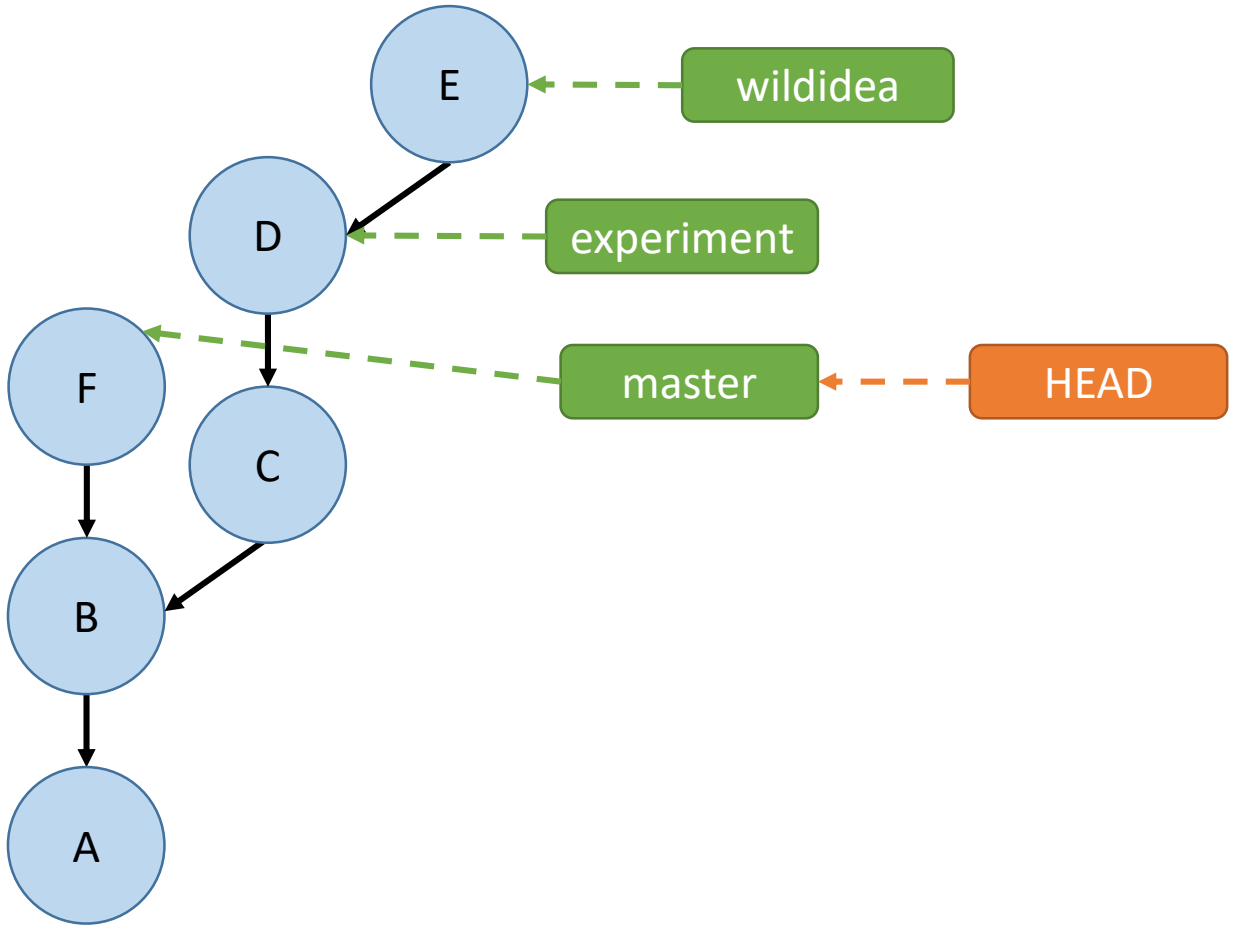
- Occur when the branch being merged **onto** is an **ancestor** of the branch being **in**.
- No merge commit is made unless --no-ff flag is used
- Will never cause conflicts!

```
git merge experiment
```



Three-Way Merges

- Occur when the branch being merged **onto** is **not a descendent** of the branch being merged **in**.
- The branch being merged onto has “moved on” since the split.
- Creates a merge commit, can cause conflicts!



`git merge experiment`

Performs a “three way merge” between B, F, and D

MERGE CONFLICT

```
:( andrew@hydreigon ~/temp3
03:57 PM (master)$ git merge goodidea
Auto-merging D
CONFLICT (add/add): Merge conflict in D
Automatic merge failed; fix conflicts and then commit the result.
```

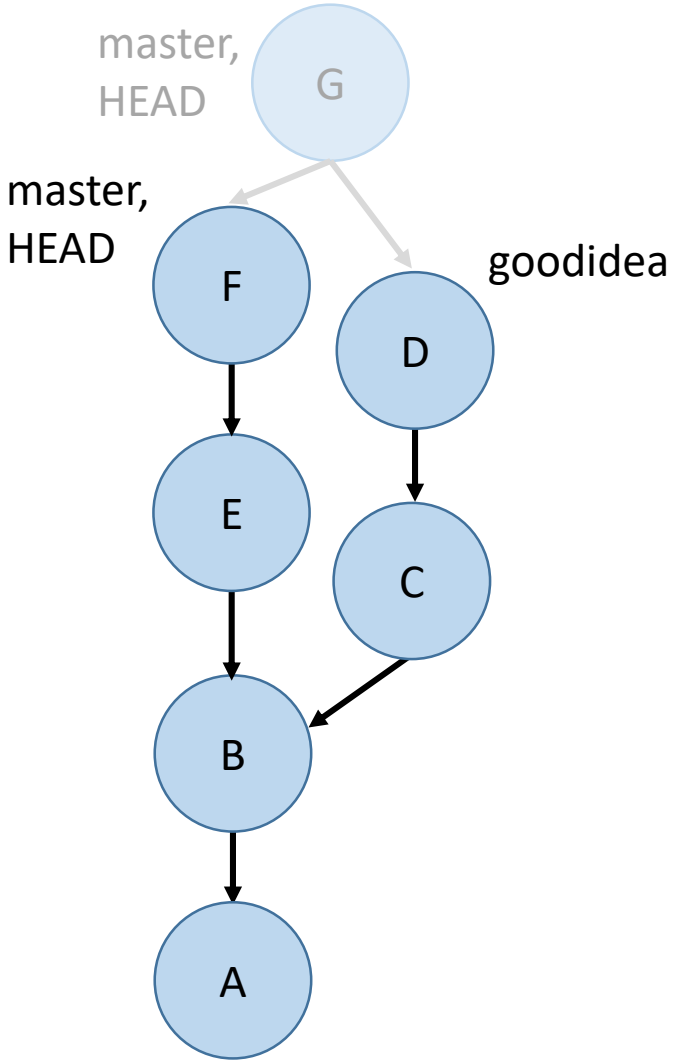
```
:( andrew@hydreigon ~/temp3
03:57 PM (master)$ git s
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

Changes to be committed:

        new file:   C

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

        both added:      D
```



MERGE CONFLICT

```
This file is demo.txt

<<<<<<< HEAD
Here is another line. modified in master
=======
Here is another line. modified in goodidea
>>>>>>> goodidea
```

“How to fix a merge conflict”

- Run `git status` to find the files that are in conflict.
- For each of these files, look for lines like “<<<<<< HEAD” or “>>>>>> 3de67ca” that indicate a conflict.
- Edit the lines to match what you want them to be.
- After you finish doing this for each conflict in each file, `git add` these conflicted files and run `git commit` to complete the merge.

```
:( andrew@hydreigon ~/temp3
03:57 PM (master)$ git s
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

Changes to be committed:

        new file:   C

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

        both added:      D
```

Activity!

Start the homework!