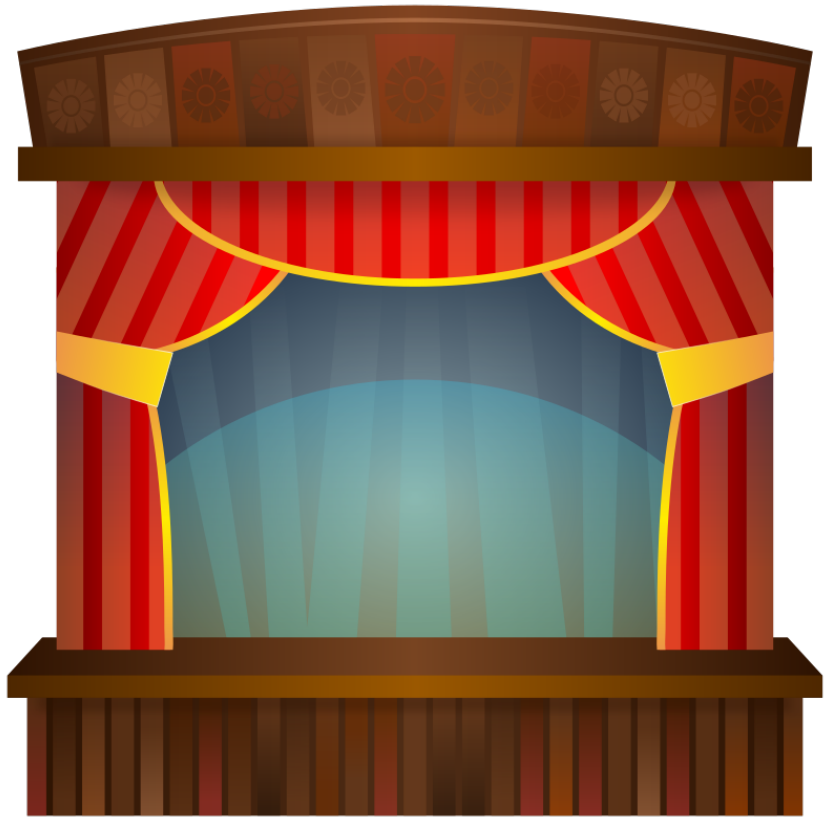
Lecture 3
More on Git Commits

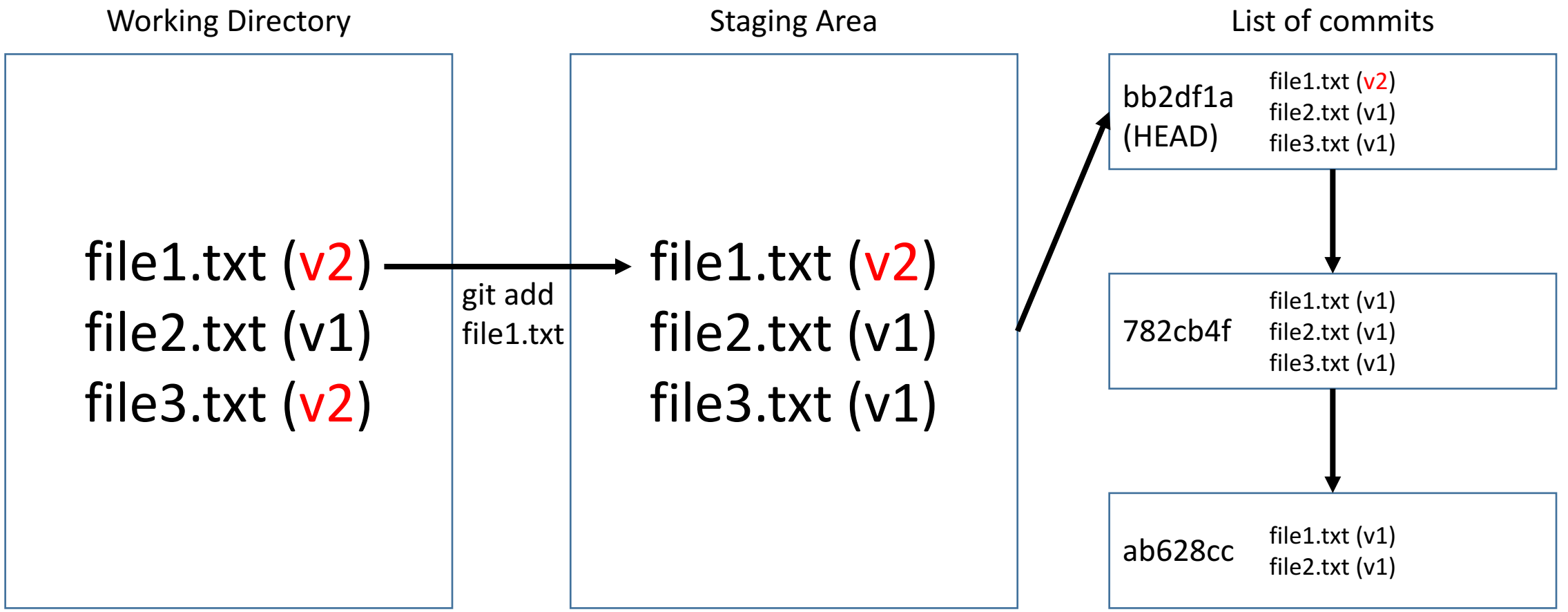
Sign in on the attendance sheet!



Homework policy (clarified)

- This course is meant to be fun, but assignments still have due dates
- Contact us before Tuesday if you have issues
- Come up to us before or after class if you had issues
- We'll be flexible - this isn't meant to be stressful
- But above all, please always hand in your homework on Tuesday to avoid complications

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

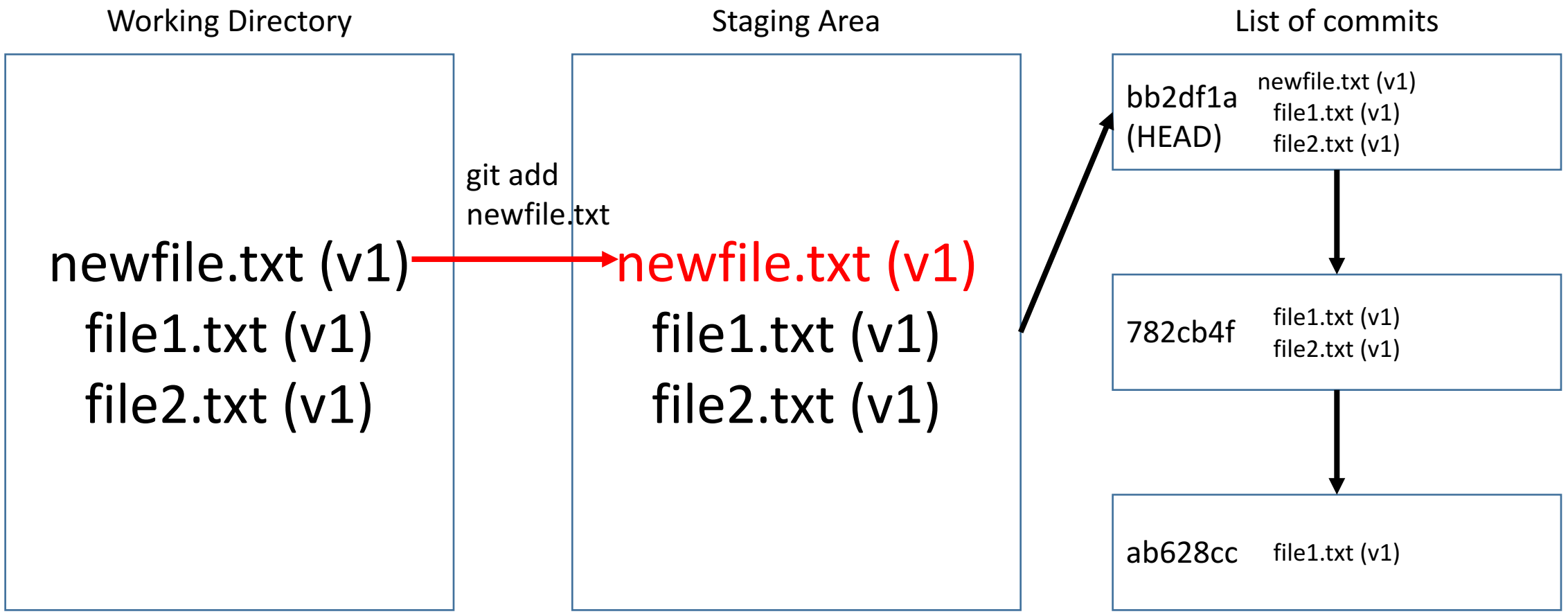


1. Make changes to files
vim file1.txt file3.txt

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

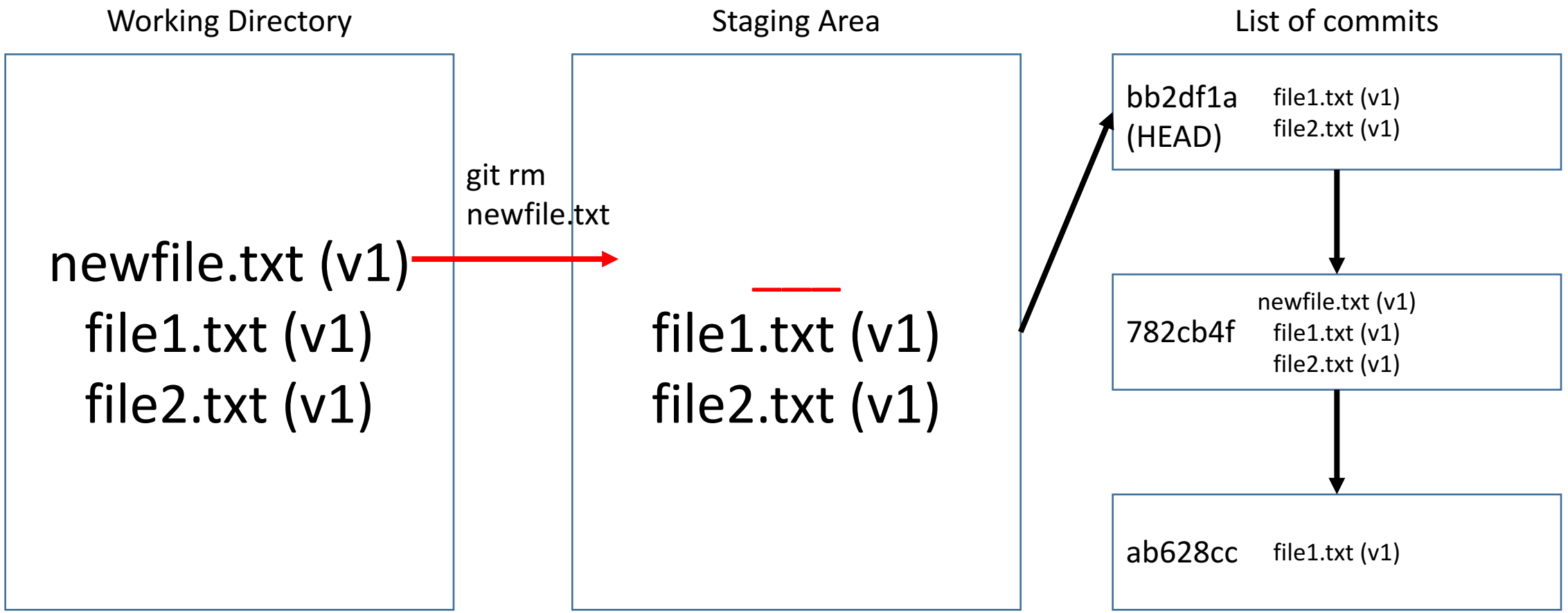
3. Commit changes in staging area
git commit -m “fixed bug in file1.txt”

What about new files?



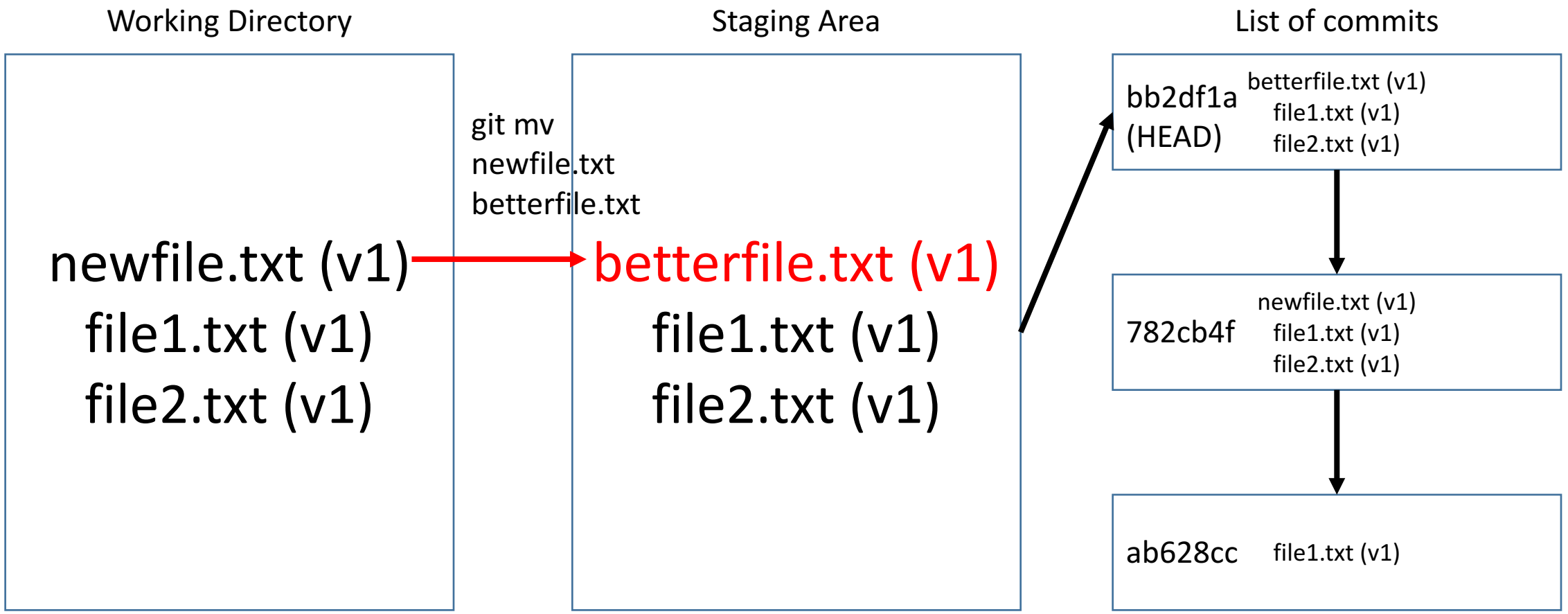
No difference from an edit, use git add newfile.txt.

What about removing files?



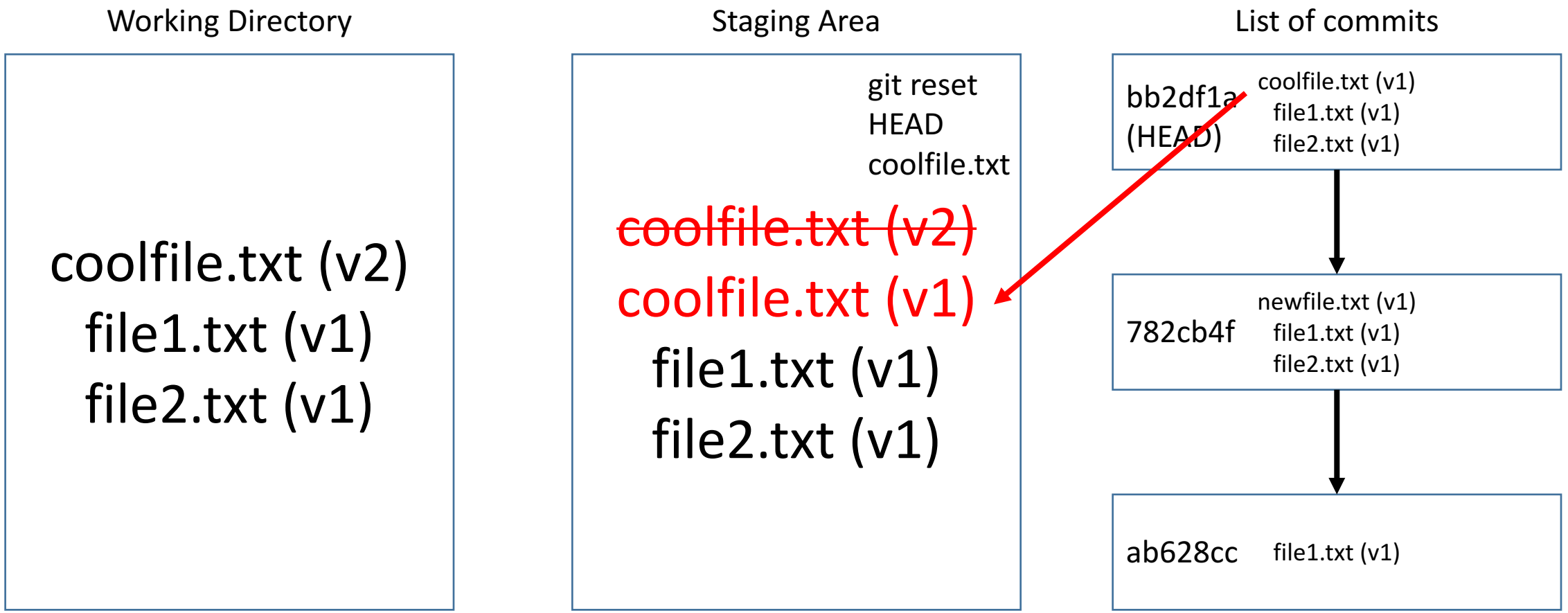
git rm newfile.txt (also deletes newfile.txt from working directory!)

What about renaming files?



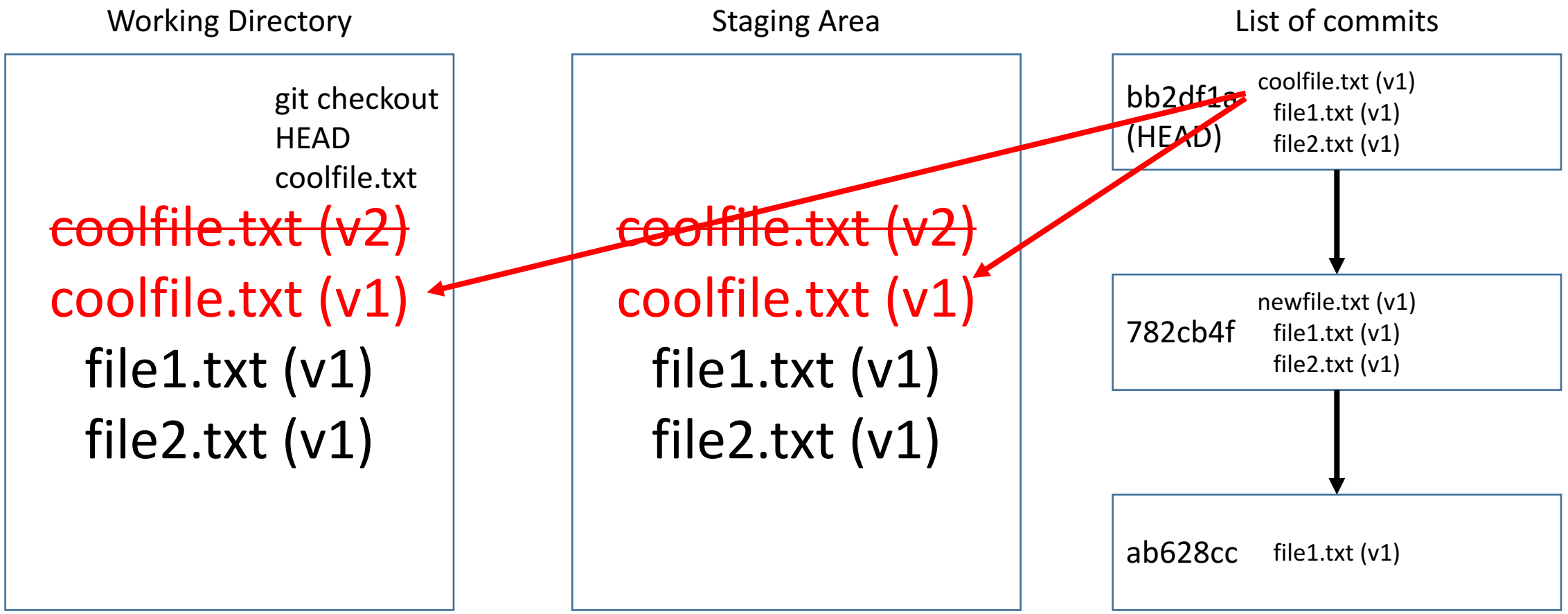
git mv newfile.txt betterfile.txt

What if I want to ‘unstage’ a file?



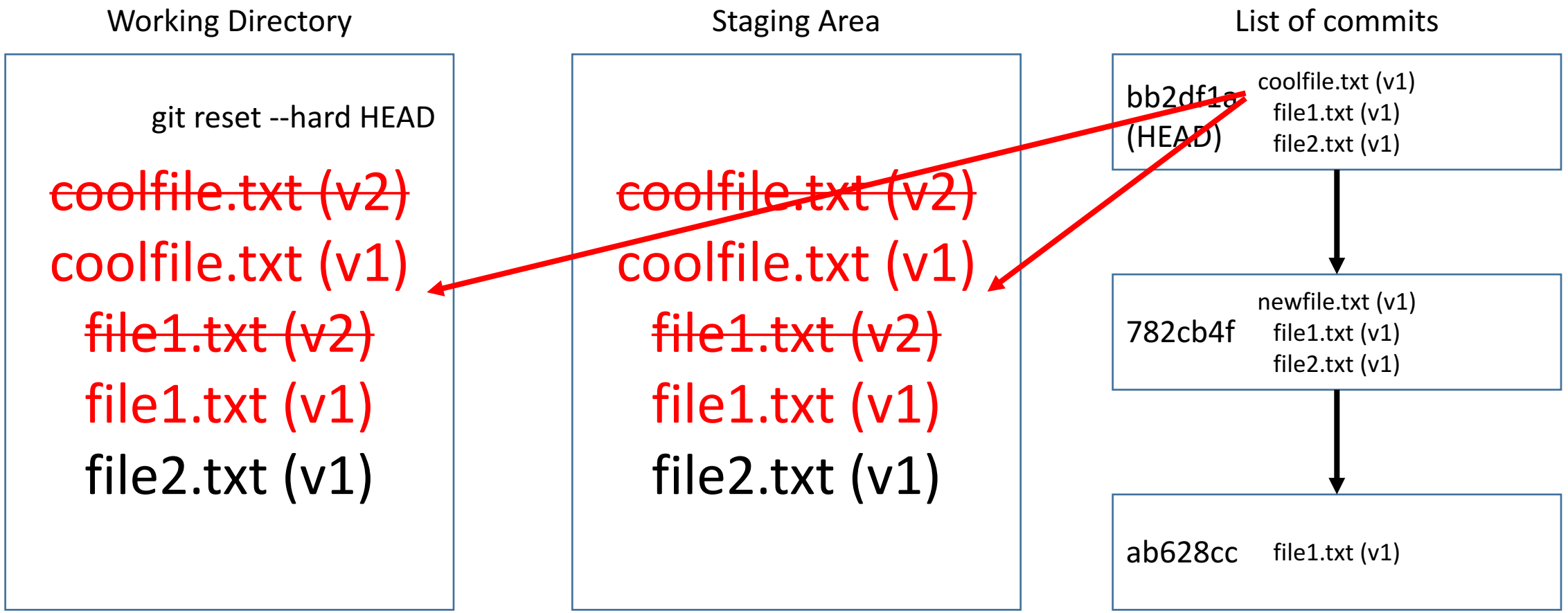
git reset HEAD coolfile.txt (Note WD is unaffected)

What if I want to start over on a file (in the WD)?



git checkout HEAD coolfile.txt

What if I want to start over (in both WD and SA)?



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

Summary: Manipulating the Staging Area

- To update the staging area with files from your working directory, use “git add”.
- To update the staging area with files from HEAD, use “git reset”.
- To delete files from the staging area, use “git rm”.

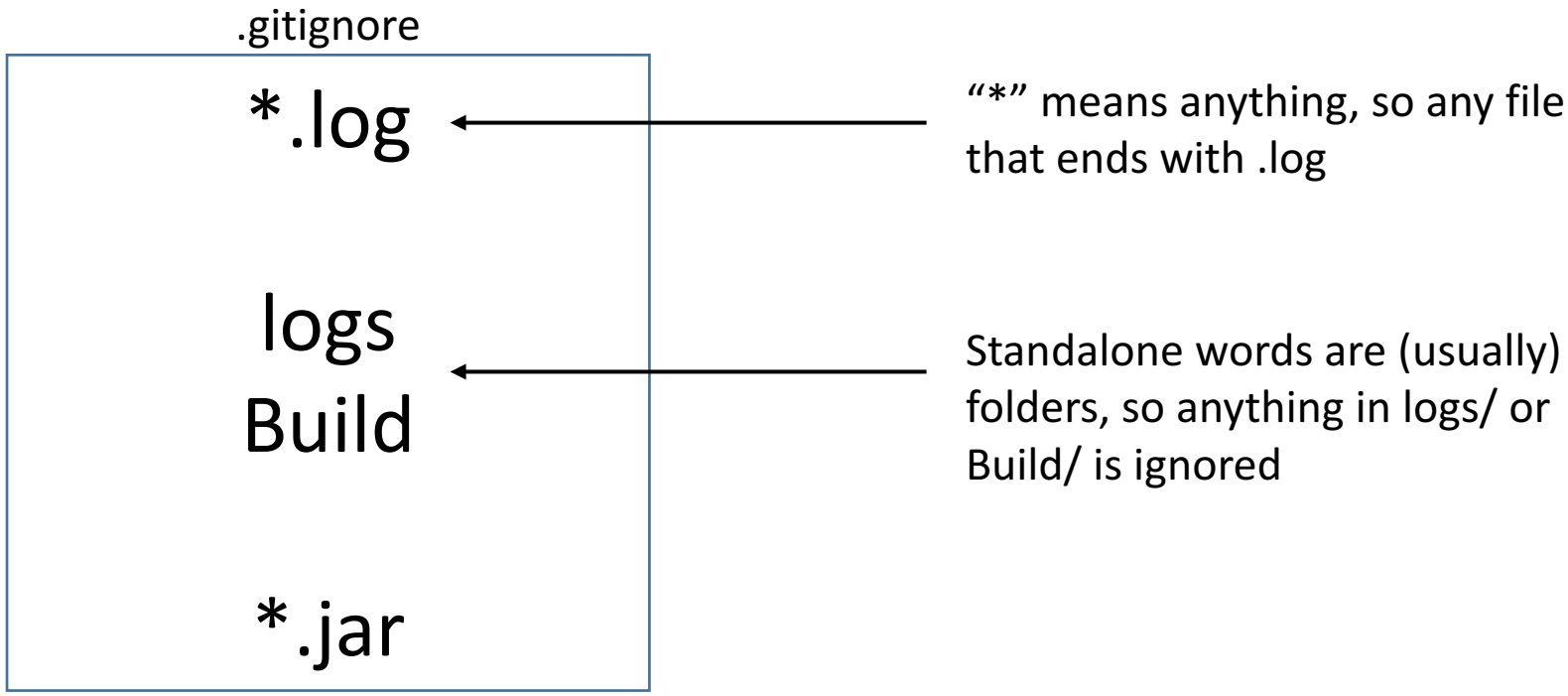
That’s how you manipulate the staging area. How about the working directory?

Summary: Manipulating the Working Directory

- To update files in the working directory, edit files with vim or your preferred text editor.
- To reset files in the working directory to how they were in a particular commit, use “git checkout”.
- If you want to reset the staging area at the same time (which is often the case), use “git reset --hard” (but with caution).

Ignoring files

- By default Git tracks everything in your repository
- Not always a good thing – log files, compiled files, cache files, etc.
- Tell git to ignore these files using a .gitignore file
- https://github.com/github/gitignore for examples



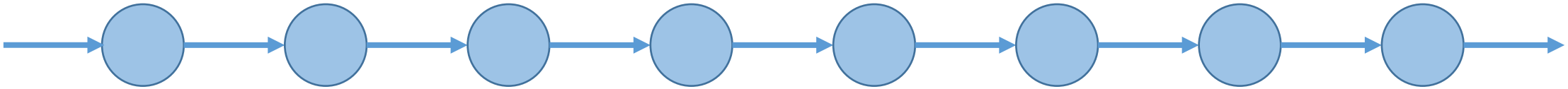
Configuring Git

- Git has certain settings by default
- Provide Git with your name, email
- Customize Git to take advantage of its features, integration with other tools, different settings with special powers, etc.
- git config --global user.name "John Doe"
- git config --global user.email johndoe@example.com

Activity

- Groups of two or three
- One person create a new Git repository using “git init” in a new folder
- Add some files and make some commits, write down your steps if you won’t remember
- Ask the other person to try to work backwards and figure out a possible set of steps that brought the repository to this state
- Switch places and do this one more time

Where we are



- This wraps up our discussion of “how to make commits”.
- So far, our commits were made in a very linear fashion – every commit had exactly one parent, and had a maximum of one child.
- In larger projects, this probably won’t happen – the commits will begin branching off each other.
- Next week: branches