Efficient tools and techniques for modern software development

Git – Part 2

Vineel Kovvuri

Senior SDE @ Microsoft

Agenda

- Recap from part 1
- Branches
- Merging Branches
- Rebasing Branches
- Resolving Conflicts

Not in Agenda

- Push/Pull/Fetch
- Github

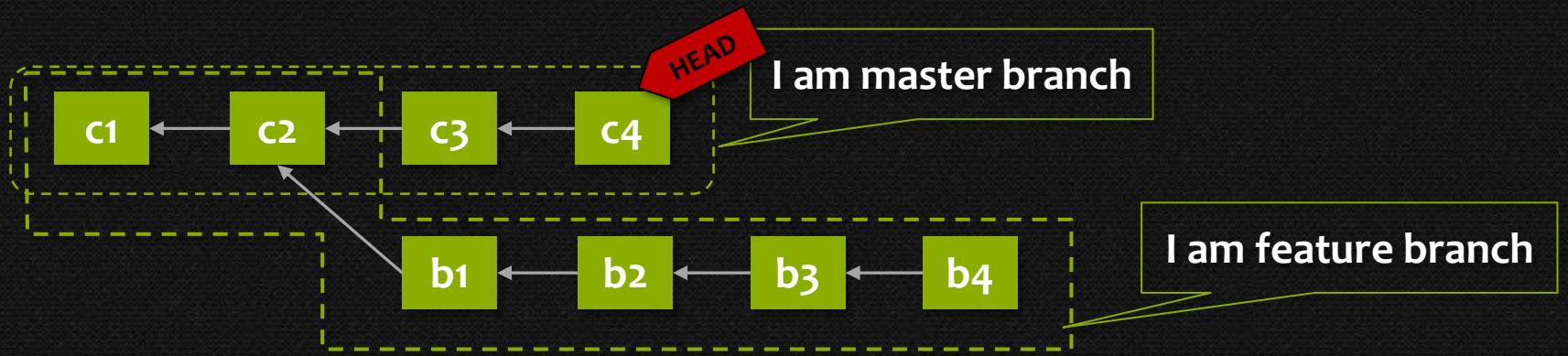
Recap

Create Repo	git init	Initialize a repository
Inspect Repo	git status	Know the status of the repository
Create Commits	git add	Add files for staging
	git commit	Create commit of the staged files
Inspect Commits	git log	View the commit log
	git diff/difftool	See changes between the commits
Undo Commits	git reset	Undo commit(unpack the commit)
	git checkout	Discard the changes

What are branches and why should I care?

- Branch is just a sequence of commits with a parent child relationship
- The default branch is always referred as *master* or *main*

- Branching helps in working with multiple features independently
- At any given point in time, There can be only one *active* branch in a repository

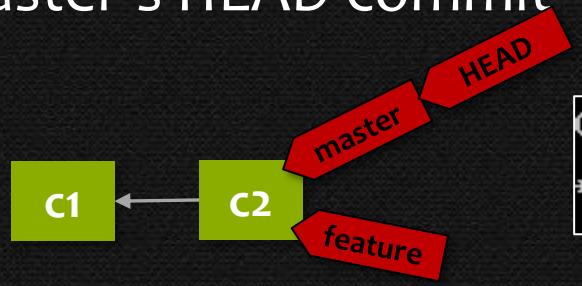


- The content of the file and folder structure of the repo is determined by the commits on current *active* branch

- *git branch* will show *all branches and highlights the current active branch

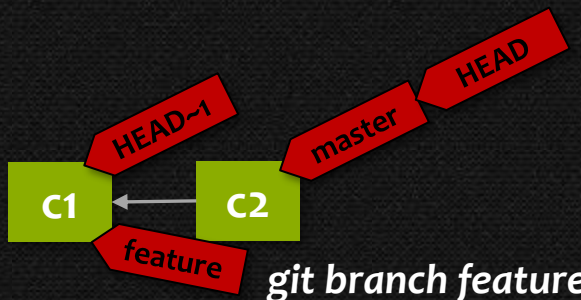
Branching

- *git branch feature master* will create a new branch named '*feature*' from master's HEAD commit



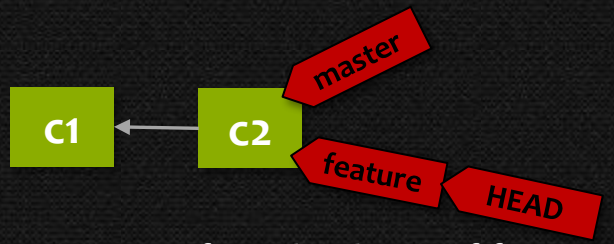
git branch feature HEAD

```
C:\MyProject>git branch
  feature
* master
```



git branch feature HEAD~1

- *git checkout feature* is used to switch to the branch named '*feature*'

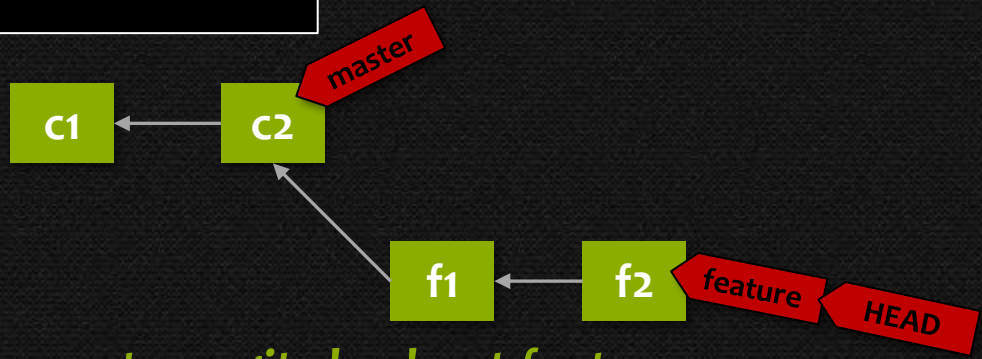


HEAD after checkout of feature

```
C:\MyProject>git checkout feature
Switched to branch 'feature'

C:\MyProject>git branch
* feature
  master
```

- With each commit on the feature branch, The HEAD moves forward on the feature branch

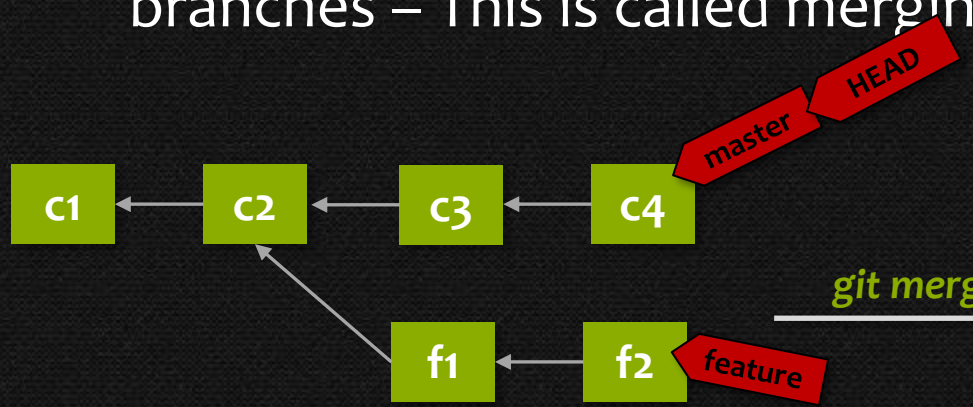


git checkout –b feature master = git branch feature master + git checkout feature

Vineel Kovvuri

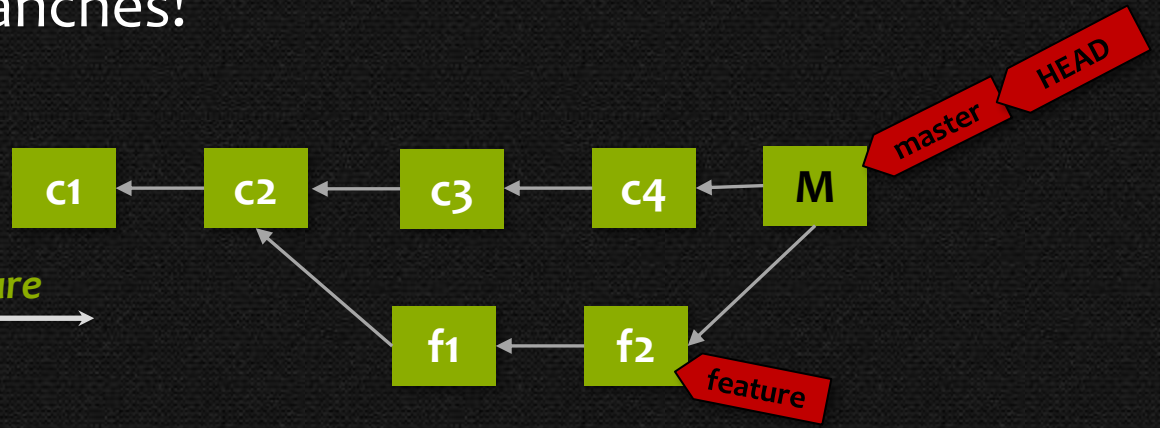
Merging

- *git merge* is used to create a merge commit between two or more branches – This is called merging branches!



HEAD on master branch after checkout of master

git merge feature



Merge commit created on master after git merge command

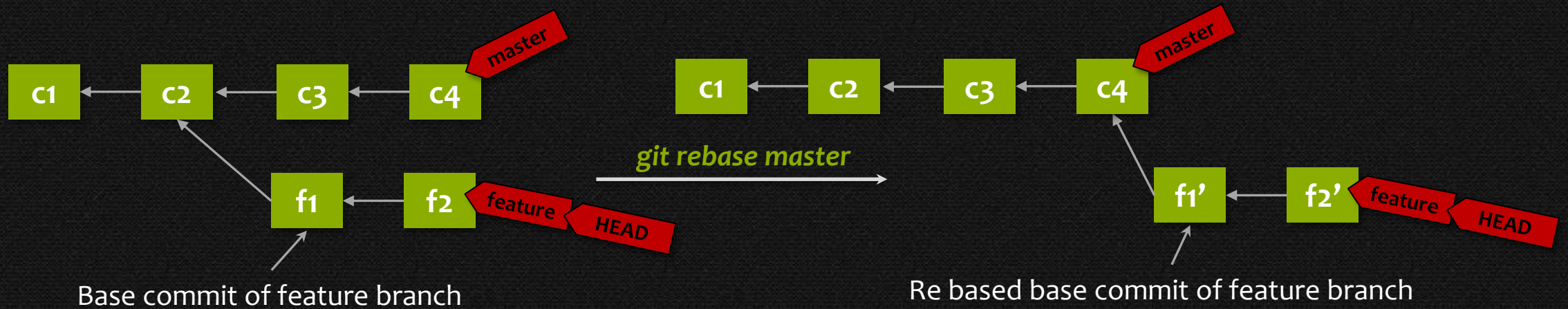
```
C:\MyProject>git log
commit a8a5250f3ee66af7e4a4afdfb2a5a0a32bbb97d3
Merge: d751102 f3f8a35
Author: Vineel Kumar Reddy Kovvuri <vineel.kovvuri@gmail.com>
Date:   Tue Jul 14 19:05:02 2015 +0530

    Merge branch 'feature'
```

```
C:\MyProject>git log --graph --oneline --decorate --all
*   a8a5250 (HEAD, master) Merge branch 'feature'
|\
| * f3f8a35 (feature) Optimised Hello World
* | d751102 Comment added
|/
* ee8a73a Adding .gitignore
* 940a3a6 My First helloWorld commit
```

Rebasing

- *git rebase* realigns the base commit of the current branch with other branch



Feature branch before git rebase

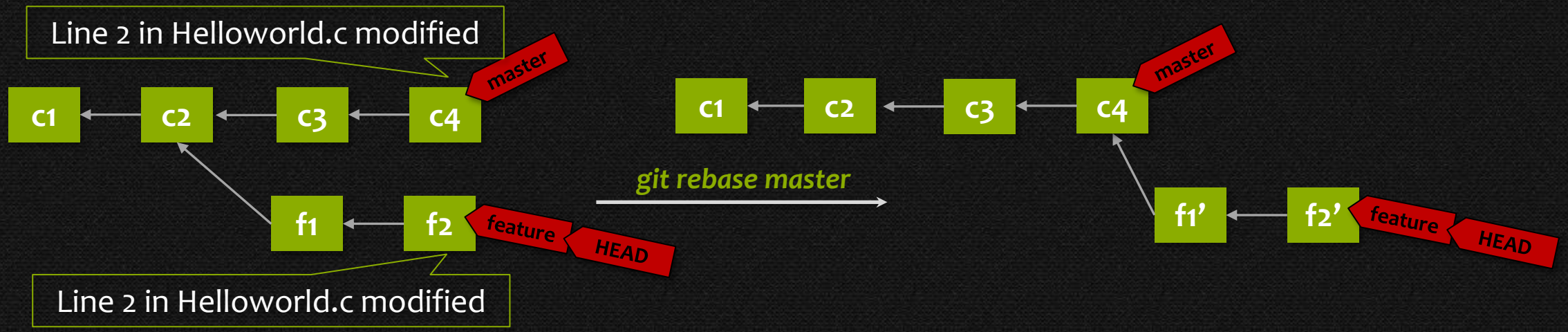
Feature branch after git rebase

f1 Contains changes made before rebase

f1' May not contain the same changes as f1 because of merge conflicts

Resolving conflicts manually in Git

- *git merge* and *git rebase* can sometime lead to merge conflicts




```
C:\HelloWorldProject>git rebase master
First, rewinding head to replay your work on top of it...
Applying: Comment updated in feature
Using index info to reconstruct a base tree...
M       HelloWorld.c
Falling back to patching base and 3-way merge...
Auto-merging HelloWorld.c
CONFLICT (content): Merge conflict in HelloWorld.c
Failed to merge in the changes.
Patch failed at 0001 Comment updated in feature
The copy of the patch that failed is found in:
   c:/HelloWorldProject/.git/rebase-apply/patch

When you have resolved this problem, run "git rebase --continue".
If you prefer to skip this patch, run "git rebase --skip" instead.
To check out the original branch and stop rebasing, run "git rebase --abort".
```

HelloWorld.c

```
1   #include<stdio.h>
2   <<<<<<< HEAD
3   //Comments add in master
4   =======
5   //Comments add feature branch
6   >>>>>>> Comment updated in feature
7   int main()
8   {
9       printf("Hello World!\n");
10      return 0;
11  }
12
```

Recap

Branching Commands	git branch	List all branches
	git branch <new> <existing>	Create <new> branch from <existing> branch
	git checkout <branch>	Switch to <branch>
	git checkout –b <new> <existing>	Create a new branch and switch to that branch
Merge Command	git merge <feature>	Merge current branch with <feature> branch
Rebase Command	git rebase <feature>	Rebase current branch with <feature> branch

References

- https://github.com/vineelkovvuri/gvpcoe-sessions-2024/blob/master/Git-Part2
- https://stackoverflow.com/

Thank You

Vineel.Kovvuri@gmail.com

https://edgroom.com/vineelkovvuri

https://github.com/vineelkovvuri

https://www.linkedin.com/in/vineelkovvuri