1. `git init` is used to initialize a local repository
2. `git add <file_name>` is to stage changes for that file
3. `git add .` is used to stage changes for all files in the current directory
4. `git commit -m "a description about the commit done"` commit the changes
5. `git status` is used to check the status of our staging area
6. `git log` tells us what we have committed (like our entire committed history)
7. `git reset .` will unstage all the changes
8. `git checkout -- <file_name>` is used to undo any changes
9. `git checkout <hash>` to return to a previous version of the commit
10. `git log --all` to show the commits ahead and also before
11. `q` to exit out of git log in case it becomes too long
12. `git log --all --graph` shows possible branches and previous version history
13. `rm -rf .git` removes the entire git history

---

14. `git remote add origin <remote_repo_url>` adding our local repository to an online repository like github (origin is like a convention for the naming)
    very important concept, is that regardless of what branch we are currently on. It does not affect the `git remote add origin command`
15. `git remote` tells us how many remote repositories we have
16. `git remote -v` verbose just means like give more details, like showing us the full link
17. `git remote remove <repo_name>` to remove an online repository

---

when we upload code to github, it's called push
when we download code from github, it's called pull

- To sync computer --> Github (like local repo to online repo)

18. `git push <online_repo_name> <branch_name>` to push local repository to online repository

- the above command ensures that our local repository is in sync with our online repository
- origin/master is like a remote tracking branch

19. `git push <online_repo_name> <branch_name> --setupstream` is like saying next time we just run this code instead --> `git push` will do

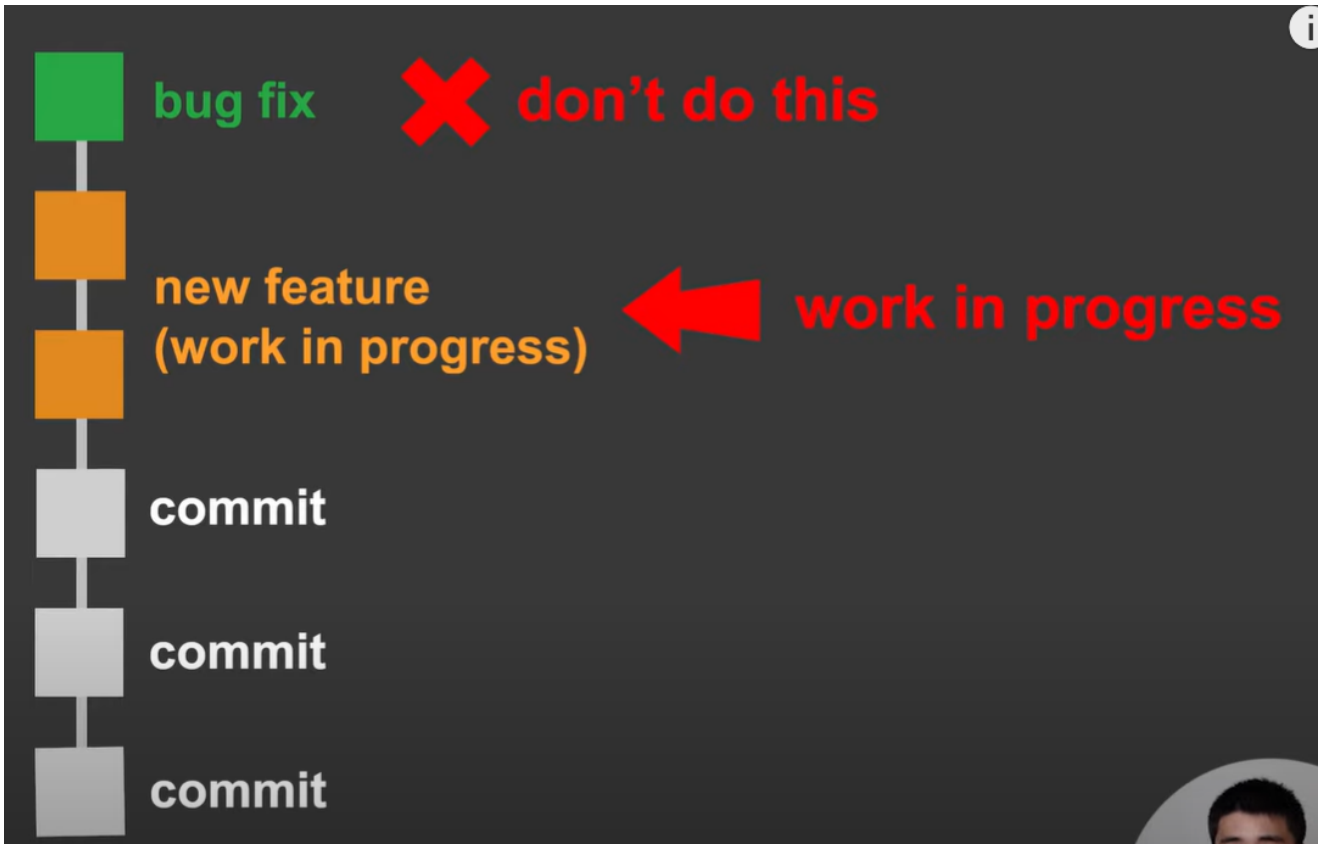
- note that, `git push` will only pushes on commits

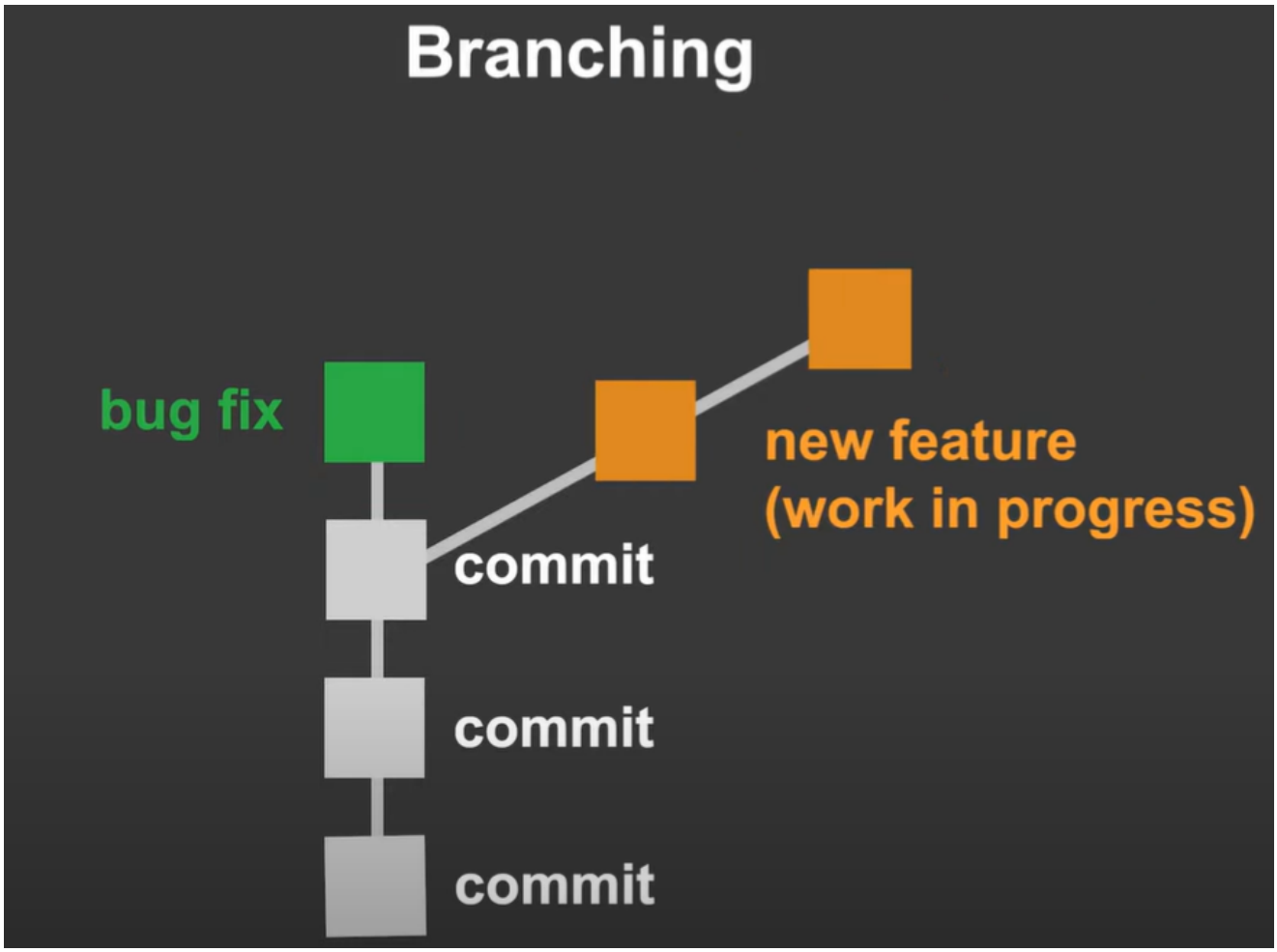
20. `git push origin master -f` this means that we force push the repository
21. `git fetch` update all the remote tracking branches with to the current state in github

---

- Used to sync Github --> Computer (like online repo, back to local repo)

22. `git pull <online_repo_name> <branch_name>` this is to sync changes from remote branch back to local branch....when remote branch is ahead.
23. `git pull --setupstream` , then we can just run `git pull`
24. `git commit --amend -m "your_message"` we are amending a commit





24. `git branch <branch_name>` to create a branch (the purpose of creating a branch is for when we have a large feature to work but at the same time we want to separate it from a simple bug fix)

- depending on what branch we are working on, it merges to that branch

25. `git merge <branch_name>` <branch_name> is the branch we would like to merge to current working branch
26. `git branch -D <branch_name>` to delete an existing branch, but we first have to switch to another branch first
27. `git merge origin/master` merges online repo changes to local changes

```
- this is super important concept, you might wonder why must be merge our online branch to local branch?
- why cannot do the other way around??
- this is because we are unable to checkout to a remote branch
```

28. `git branch -a` to show all current branches available
29. `git merge <branch_name> --allow-unrelated-histories` to merge 2 unrelated repositories
30. `git branch -d <branch_name>` to delete a local branch

31. `git push origin --delete <branch_name>` to delete online repository branch
32. `git reset --merge HEAD~1` revert to previous merge commit
33. `git push -f <remote_branch_name> <branch_name>` force push a remote branch

34) this happens when, we initialized a new local repository but did not `git add .` and also `git commit -m "your_message"`
35) `git checkout <branch_name>` note that any checkout is only made locally
36) `git revert <hash>` to revert to a previous commit
37) `git restore .` to discard local changes done
38) `git reset --hard origin/<branch_name>` reset local branch to match current state of the remote branch

- merge a specific folder
  1. First, ensure you are on the branch where you want the changes to be merged into (in this case, `v1.1.5`):

     ```
     git checkout v1.1.5
     ```

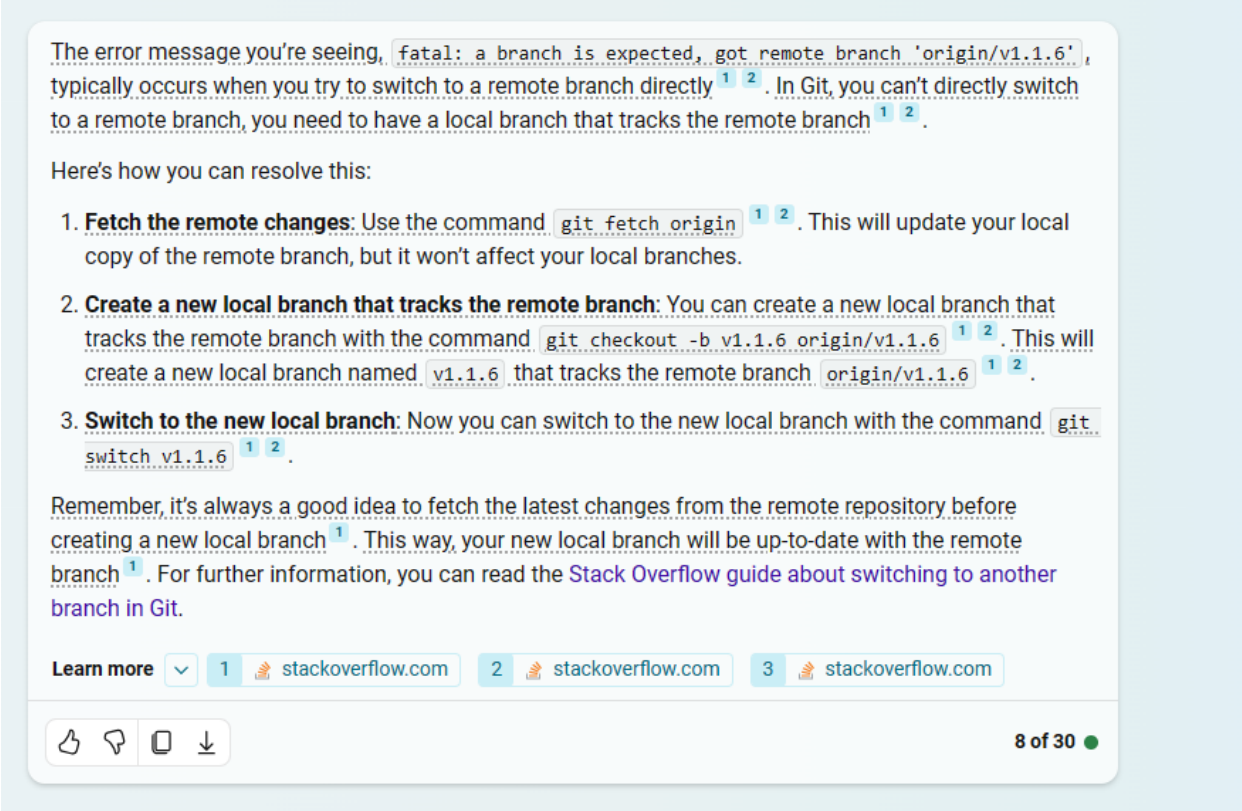
  2. Then, use the `checkout` command to apply changes from the specific folder at `v1.1.4`:

     ```
     git checkout v1.1.4 -- path_to_folder
     ```

     Replace `path_to_folder` with the actual path to the folder you want to merge.

  3. This will apply changes from `v1.1.4` to `v1.1.5` for that folder only. Now, you can commit these changes:

     ```
     git commit -m "Merged specific folder from v1.1.4"
     ```
- Question of the day, can we directly checkout to a remote branch??
- No!!



The error message you're seeing, `fatal: a branch is expected, got remote branch 'origin/v1.1.6'`, typically occurs when you try to switch to a remote branch directly [1] [2]. In Git, you can't directly switch to a remote branch, you need to have a local branch that tracks the remote branch [1] [2].

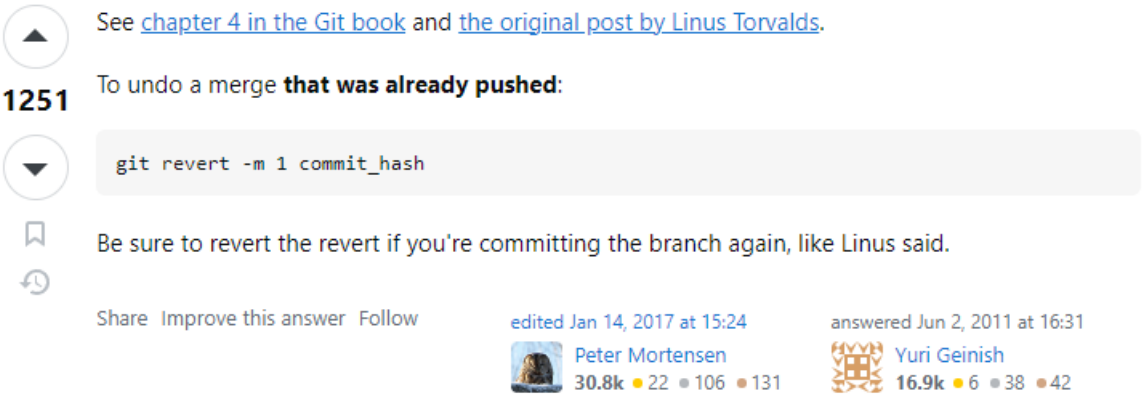
Here's how you can resolve this:

1. **Fetch the remote changes**: Use the command `git fetch origin` [1] [2]. This will update your local copy of the remote branch, but it won't affect your local branches.
2. **Create a new local branch that tracks the remote branch**: You can create a new local branch that tracks the remote branch with the command `git checkout -b v1.1.6 origin/v1.1.6` [1] [2]. This will create a new local branch named `v1.1.6` that tracks the remote branch `origin/v1.1.6` [1] [2].
3. **Switch to the new local branch**: Now you can switch to the new local branch with the command `git switch v1.1.6` [1] [2].

Remember, it's always a good idea to fetch the latest changes from the remote repository before creating a new local branch [1]. This way, your new local branch will be up-to-date with the remote branch [1]. For further information, you can read the Stack Overflow guide about switching to another branch in Git.

Learn more 1 stackoverflow.com 2 stackoverflow.com 3 stackoverflow.com

8 of 30



1251

See chapter 4 in the Git book and the original post by Linus Torvalds.

To undo a merge **that was already pushed**:

```
git revert -m 1 commit_hash
```

Be sure to revert the revert if you're committing the branch again, like Linus said.

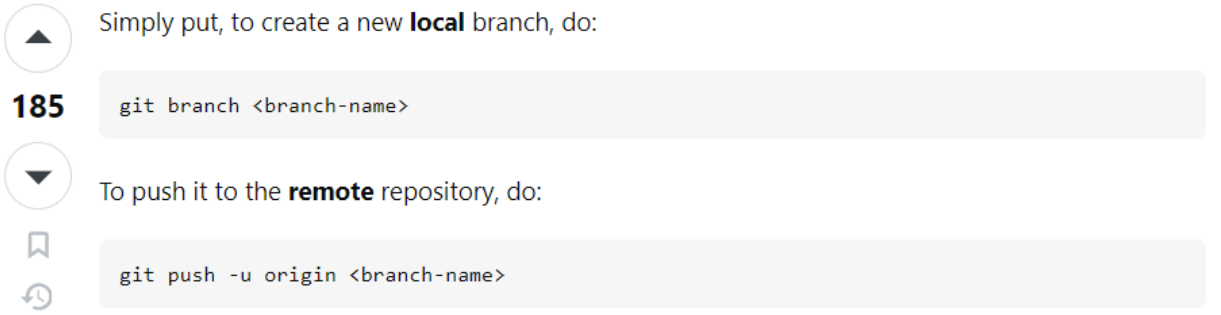
Share Improve this answer Follow

edited Jan 14, 2017 at 15:24 Peter Mortensen 30.8k ● 22 ● 106 ● 131

answered Jun 2, 2011 at 16:31 Yuri Geinish 16.9k ● 6 ● 38 ● 42

- to undo a merge that has already been pushed

40. `git diff <local_branch> <origin/remote_branch>` to show the difference between local and remote branch
41. `git push -u origin <branch_name>` pushes local branch to remote branch



185

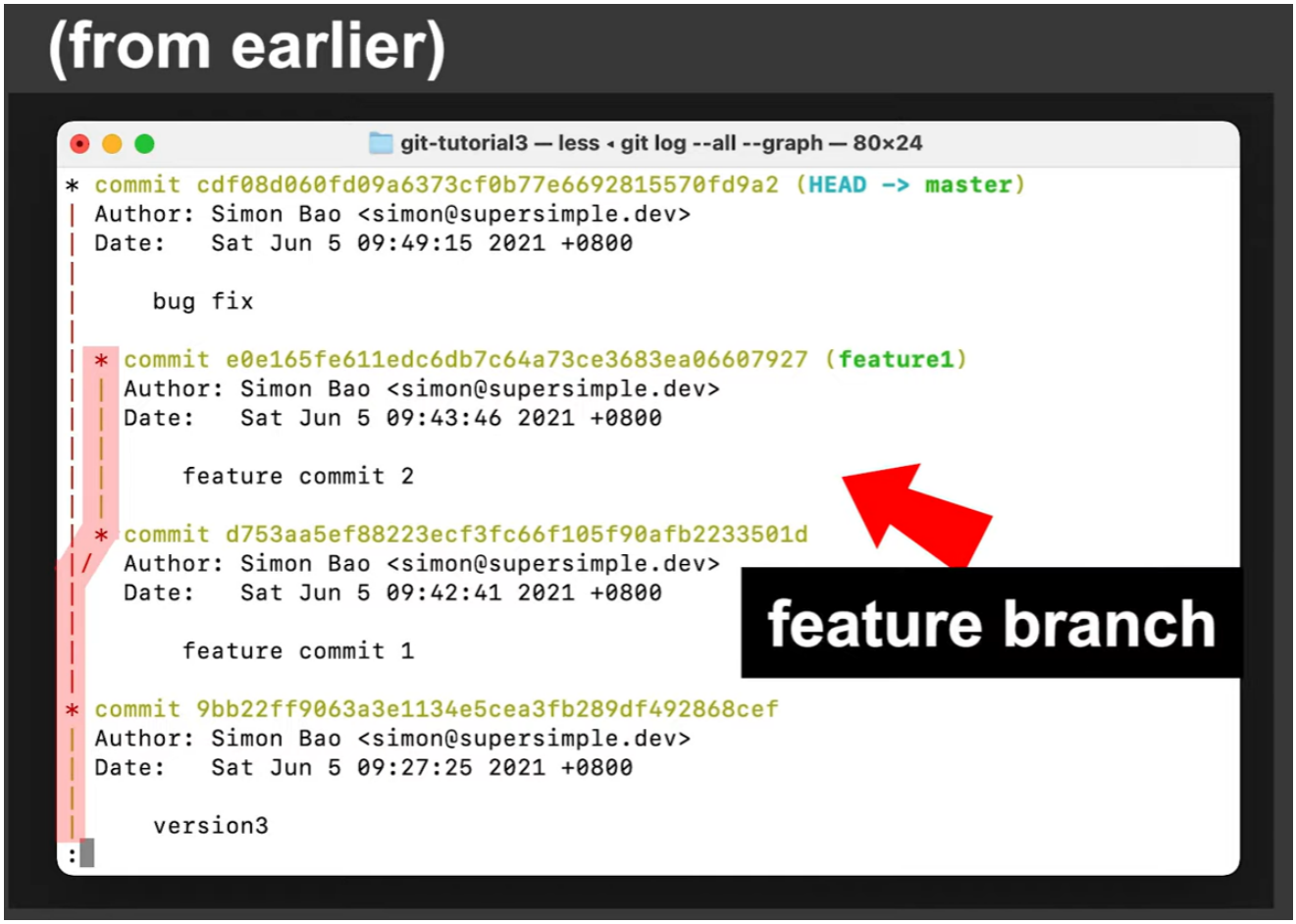
Simply put, to create a new **local** branch, do:

```
git branch <branch-name>
```

To push it to the **remote** repository, do:

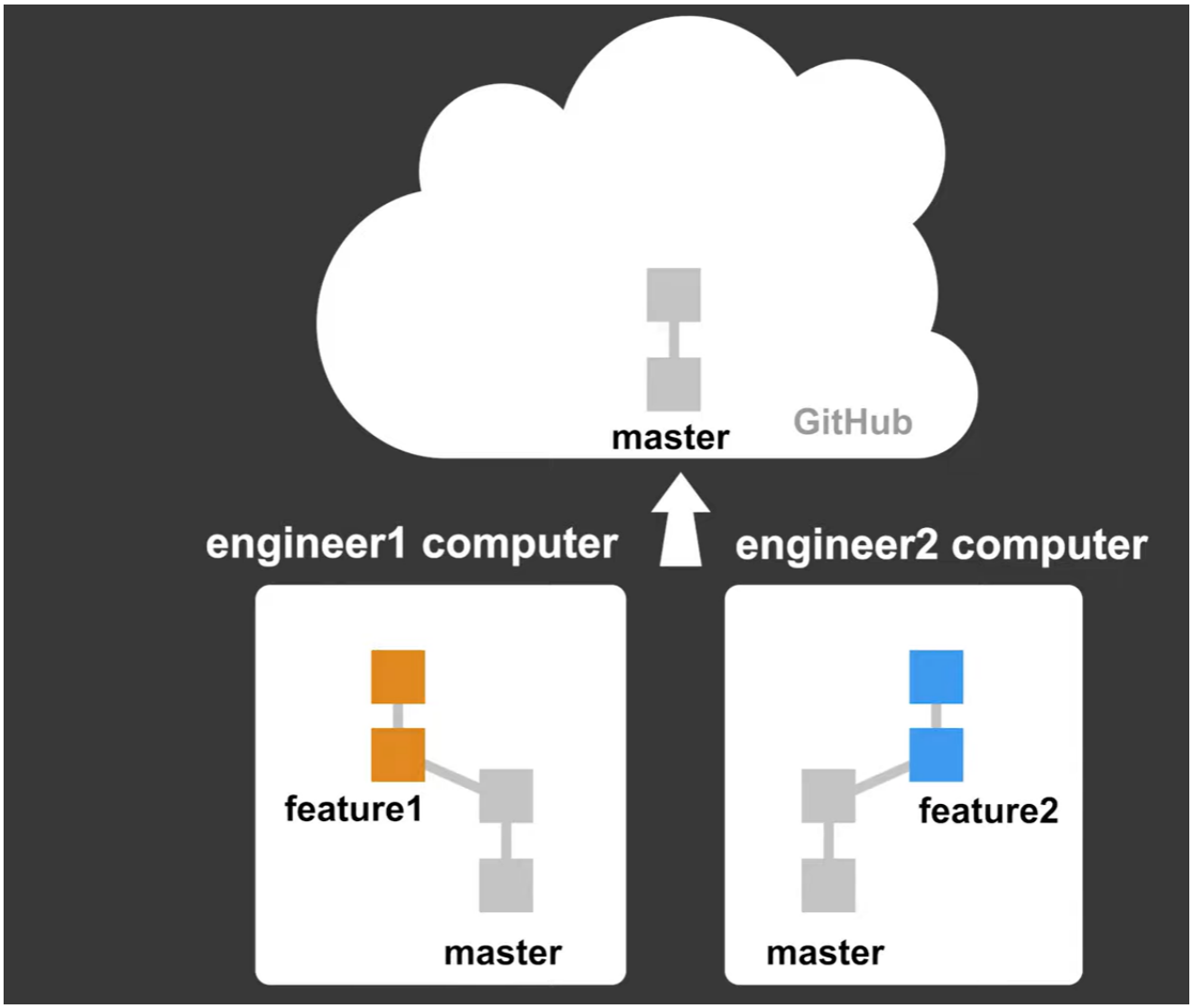
```
git push -u origin <branch-name>
```

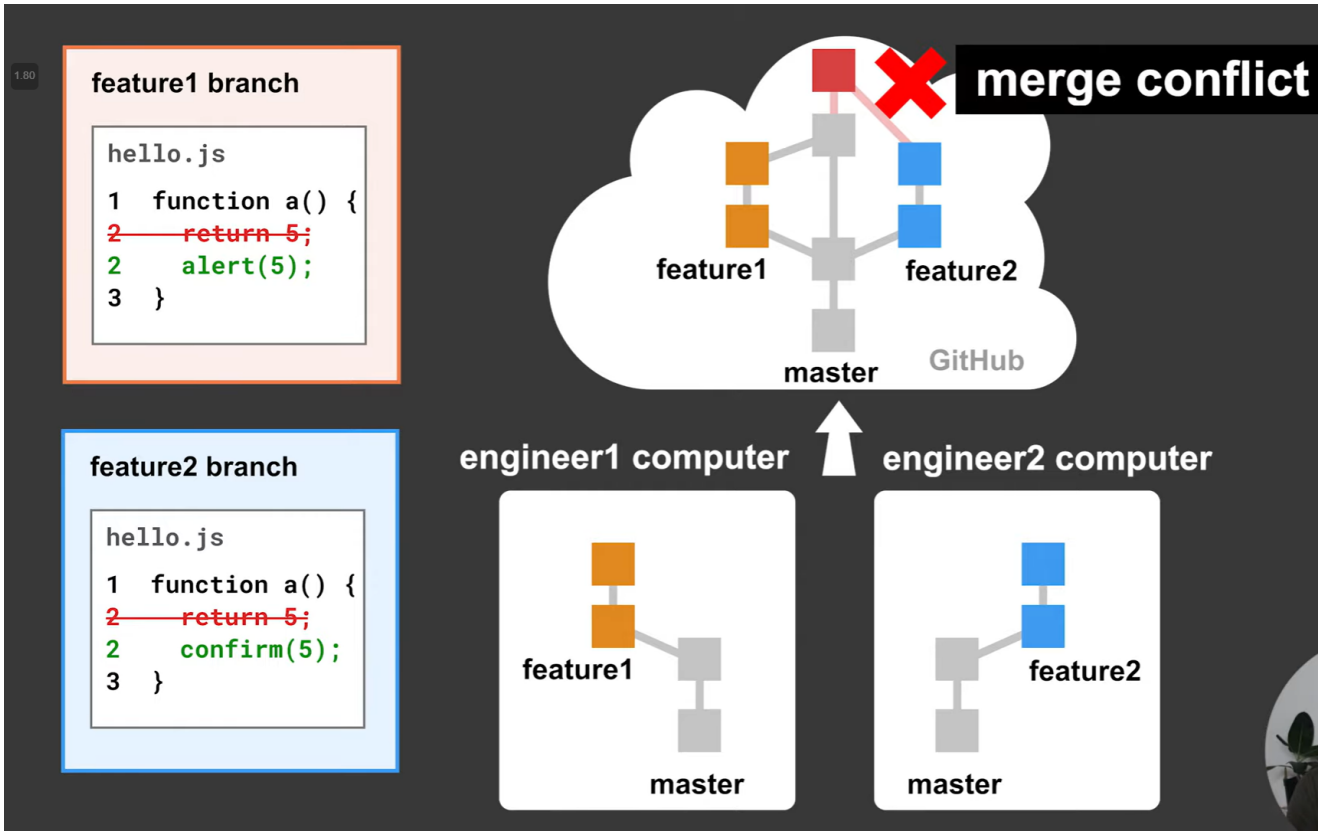
- Feature branches
- is like a workflow, a step-by-step process for using Git and Github



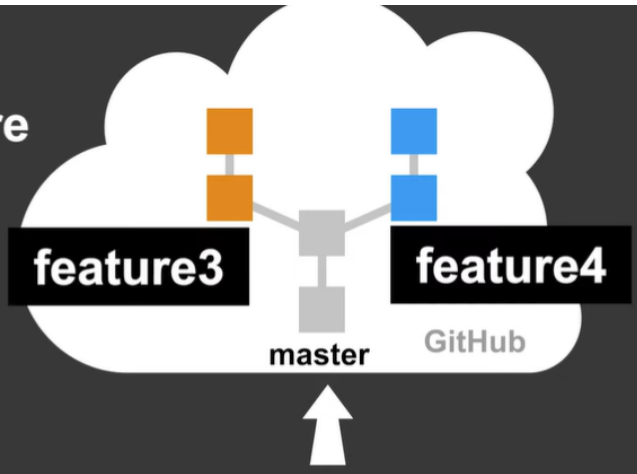
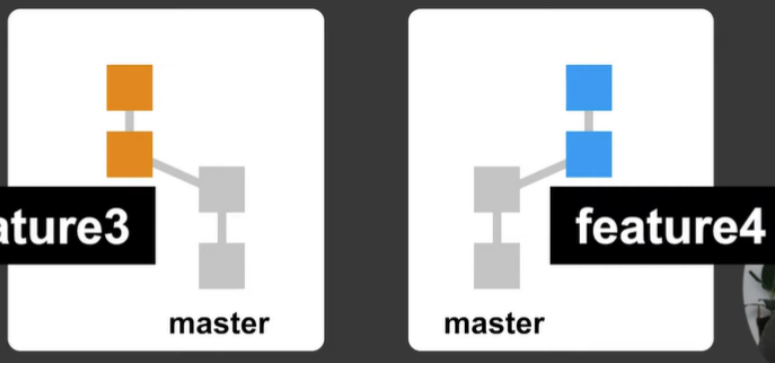
- Feature branch workflow

1. Create a feature branch
2. Upload feature branch to Github
3. Create a Pull Request (for code reviews)
4. Merge feature branch into master/main branch


master
GitHub
engineer1 computer
engineer2 computer
feature1
master
feature2
master


feature1 branch
hello.js
1 function a() {
2 return 5;
2 alert(5);
3 }
feature2 branch
hello.js
1 function a() {
2 return 5;
2 confirm(5);
3 }
merge conflict
feature1
feature2
master
GitHub
engineer1 computer
engineer2 computer
feature1
master
feature2
master

Set up required:

- same scenario as before
- git checkout master
  git pull origin master
- create 2 branches
  that modify same
  file, same line
- push to GitHub &
  create pull requests

Pause and try setting this up. You got this!