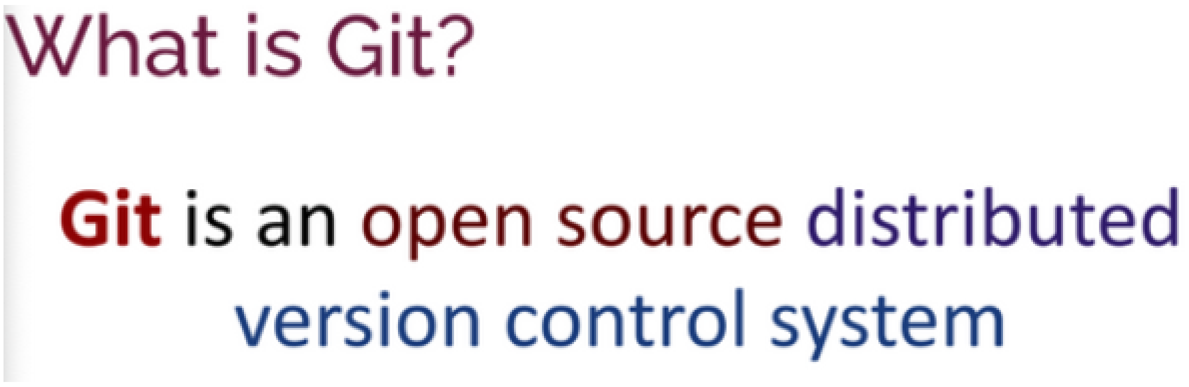
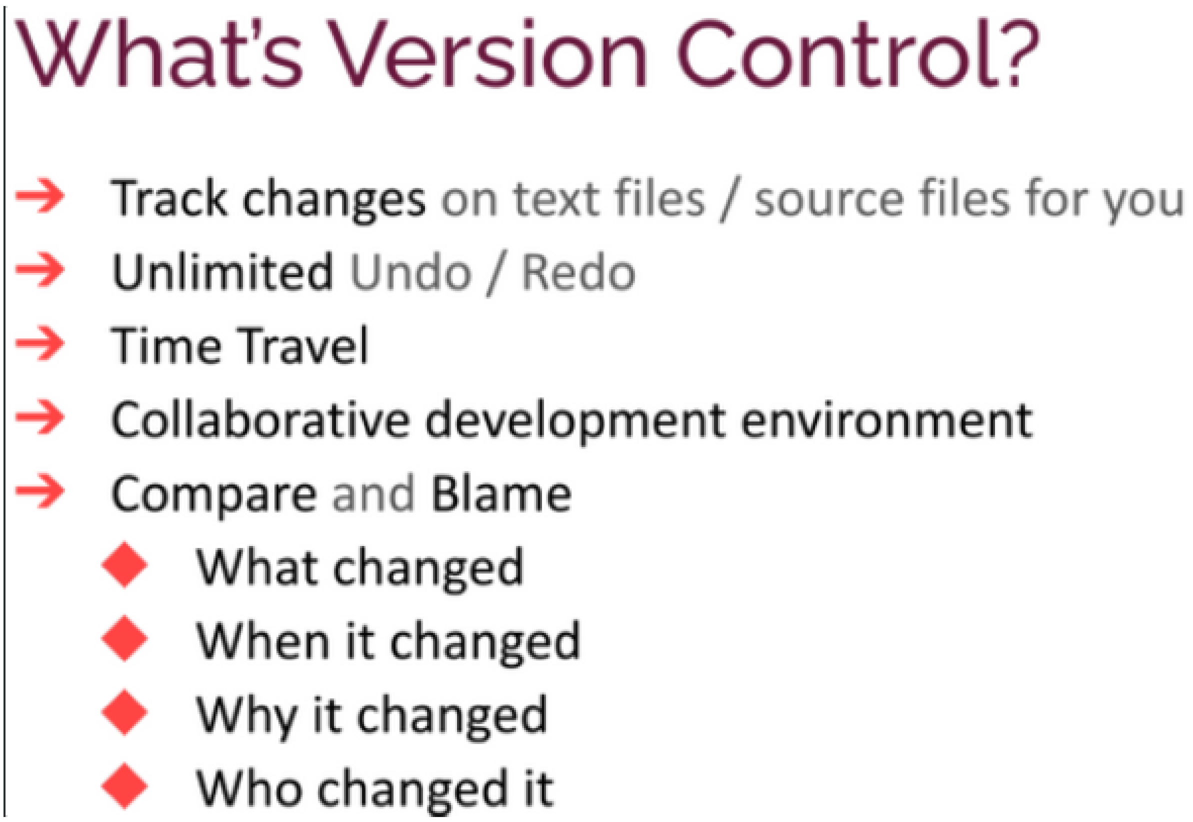
Git ≤≥ GitHub *Summary_bytepe

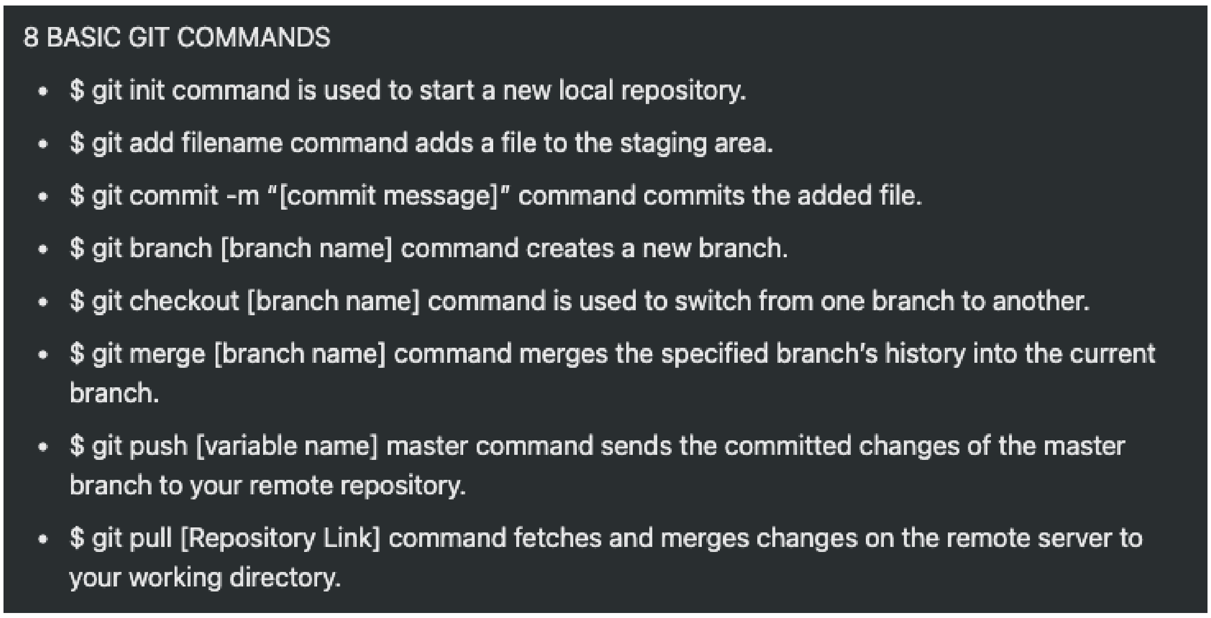

What is Git?

Git is an open source distributed version control system



What's Version Control?

- Track changes on text files / source files for you
- Unlimited Undo / Redo
- Time Travel
- Collaborative development environment
- Compare and Blame
  - What changed
  - When it changed
  - Why it changed
  - Who changed it



8 BASIC GIT COMMANDS

- $ git init command is used to start a new local repository.
- $ git add filename command adds a file to the staging area.
- $ git commit -m "[commit message]" command commits the added file.
- $ git branch [branch name] command creates a new branch.
- $ git checkout [branch name] command is used to switch from one branch to another.
- $ git merge [branch name] command merges the specified branch's history into the current branch.
- $ git push [variable name] master command sends the committed changes of the master branch to your remote repository.
- $ git pull [Repository Link] command fetches and merges changes on the remote server to your working directory.

Git Repository

→ Let's check if you have git in your computer

```
git --version
```

→ git needs your identity to mark/label changes / editor

```
git config --global user.name "Your Name"
```

```
git config --global user.email "Your Email"
```

```
git config --global core.editor "vim"
```

```
git config --list
```

→ to create a new remote repo and connect it with your local repo (after you create a remote repo on Github/Bitbucket etc.)

```
git clone address
```

Track a new file

→ let's create a new file in our project folder

```
touch file1.txt
```

→ let's edit this file

```
vim file1.txt
```

→ let's check the status of our project

```
git status
```

Stage files options

→ stage one file

```
git add filename
```

→ stage all files (new, modified)

```
git add .
```

→ stage all changes

```
git add -A
```

→ stage modified and deleted files only

```
git add -u
```

Commit

→ Commit the files on the stage

```
git commit -m ”message”
```

→ Add and commit all tracked files

```
git commit -am ”message”
```

→ amend commit message

```
git commit --ammend
```

Changing the Last Commit: `git commit --amend`

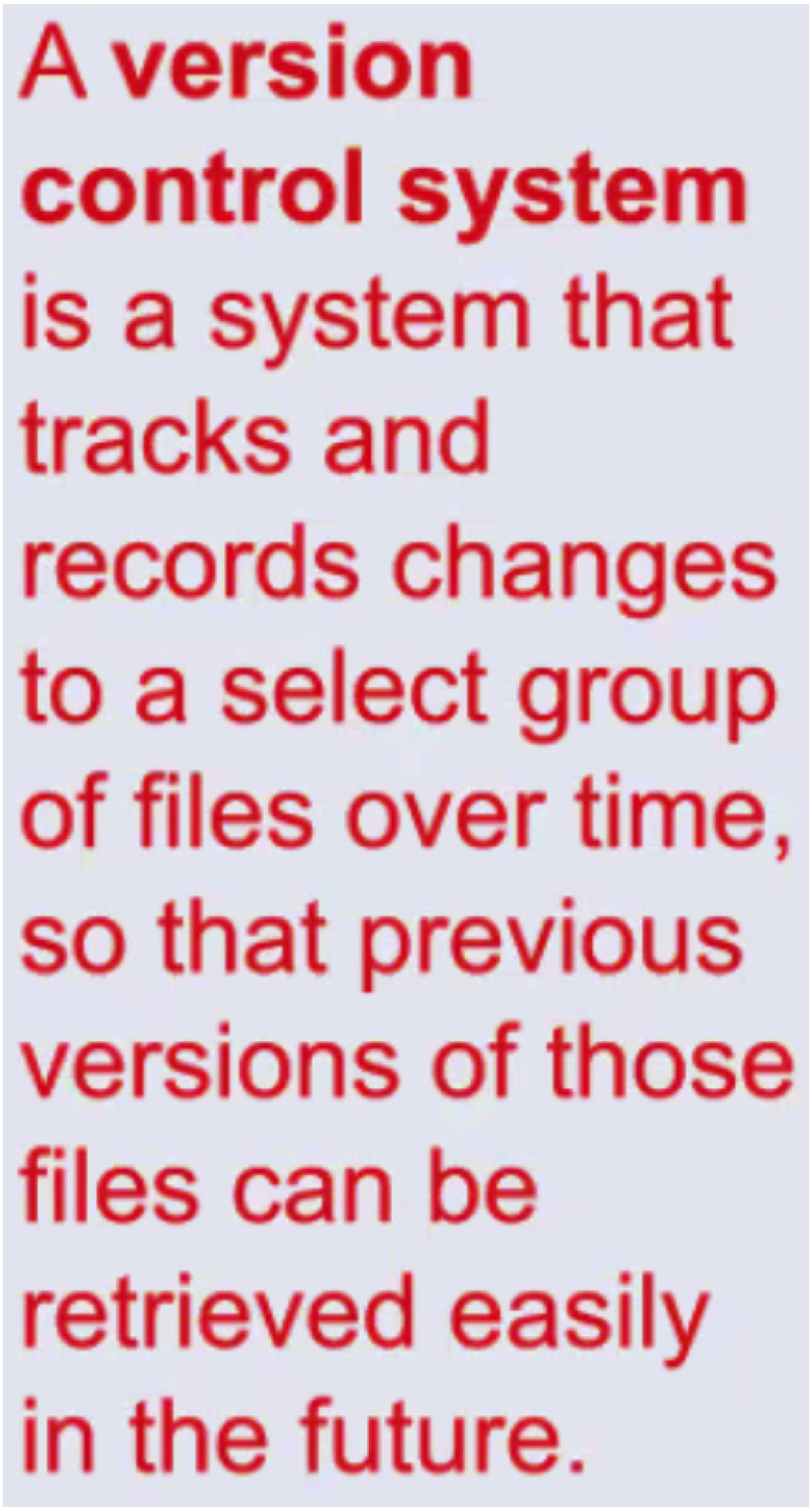
The `git commit --amend` command is a convenient way to modify the most recent commit. It lets you combine staged changes with the previous commit instead of creating an entirely new commit. It can also be used to simply edit the previous commit message without changing its snapshot. But, amending does not just alter the most recent commit, it replaces it entirely, meaning the amended commit will be a new entity with its own ref. To Git, it will look like a brand new commit, which is visualized with an asterisk (*) in the diagram below. There are a few common scenarios for using `git commit --amend`. We'll cover usage examples in the following sections.

Change most recent Git commit message

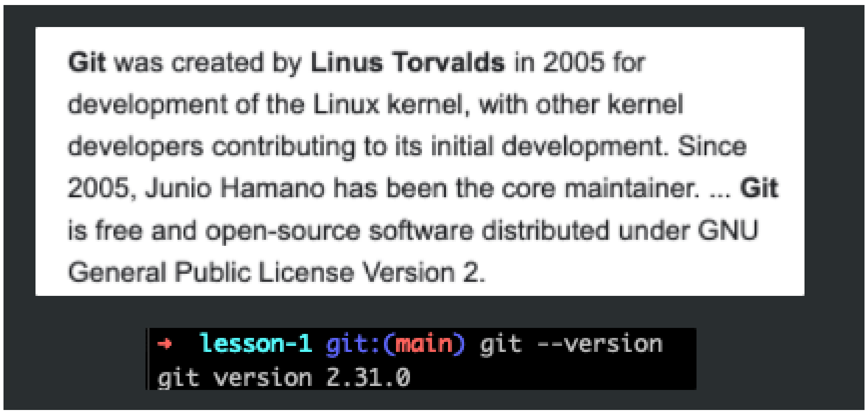
```
git commit --amend
```

```
git commit --amend -m "an updated commit message"
```

Adding the `-m` option allows you to pass in a new message from the command line without being prompted to open an editor.

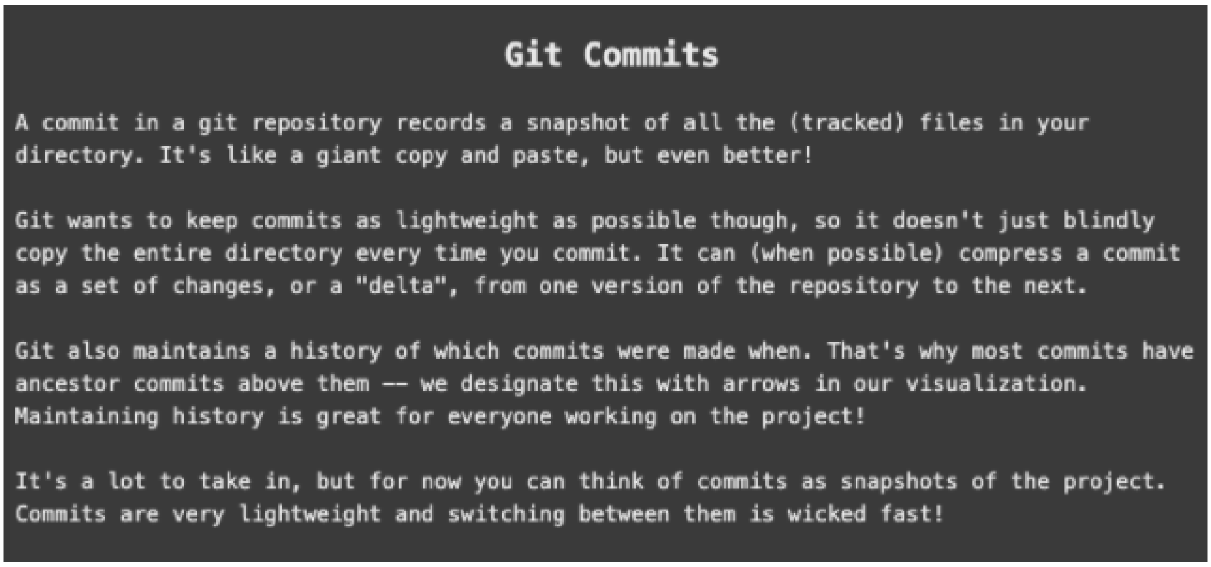


A **version control system** is a system that tracks and records changes to a select group of files over time, so that previous versions of those files can be retrieved easily in the future.



Git was created by **Linus Torvalds** in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development. Since 2005, Junio Hamano has been the core maintainer. ... **Git** is free and open-source software distributed under GNU General Public License Version 2.

```
➜  lesson-1 git:(main) git --version
git version 2.31.0
```



Git Commits

A commit in a git repository records a snapshot of all the (tracked) files in your directory. It's like a giant copy and paste, but even better!

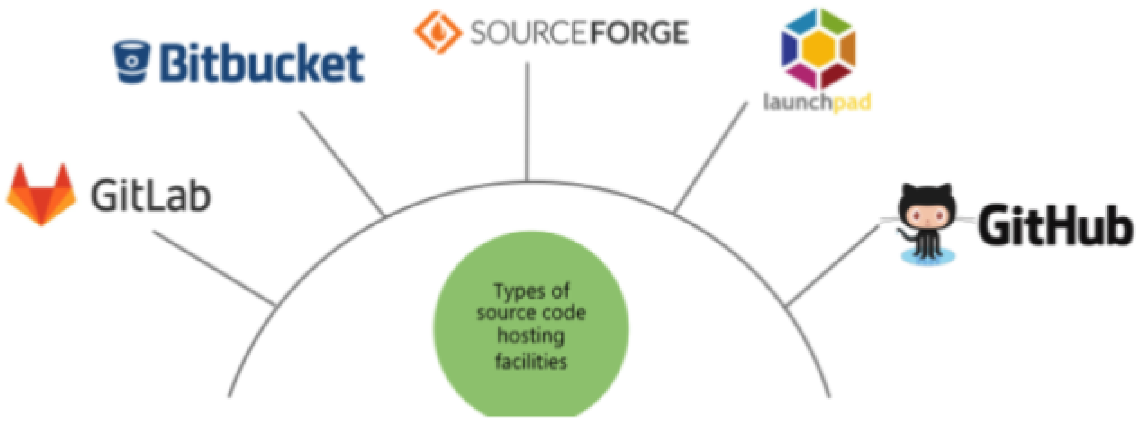
Git wants to keep commits as lightweight as possible though, so it doesn't just blindly copy the entire directory every time you commit. It can (when possible) compress a commit as a set of changes, or a "delta", from one version of the repository to the next.

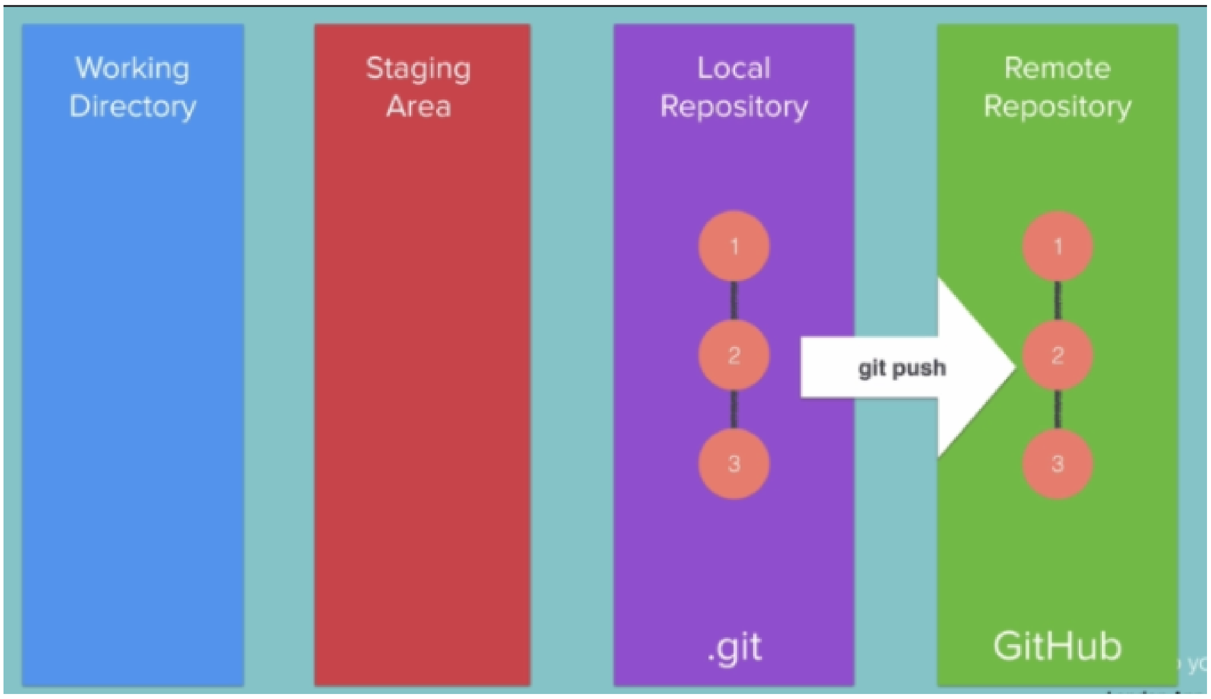
Git also maintains a history of which commits were made when. That's why most commits have ancestor commits above them -- we designate this with arrows in our visualization. Maintaining history is great for everyone working on the project!

It's a lot to take in, but for now you can think of commits as snapshots of the project. Commits are very lightweight and switching between them is wicked fast!

GitHub is a Git repository hosting (Source Code Hosting) service , but it adds many of its own features. It is a web-based platform used for version control and it provides a Web-based graphical interface. It also provides access control and several collaboration features, such as a wikis and basic task management tools for every project.

Git is a version control system that lets you manage and keep track of your source code history locally. GitHub is a cloud-based hosting service that lets you manage Git repositories.



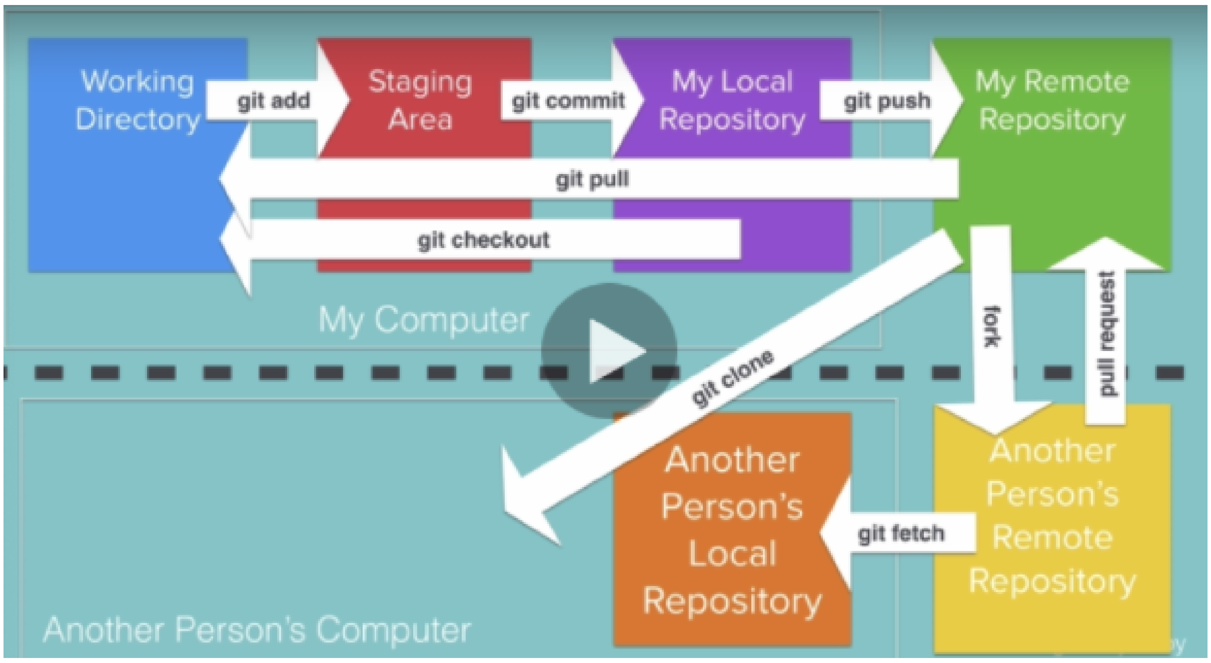


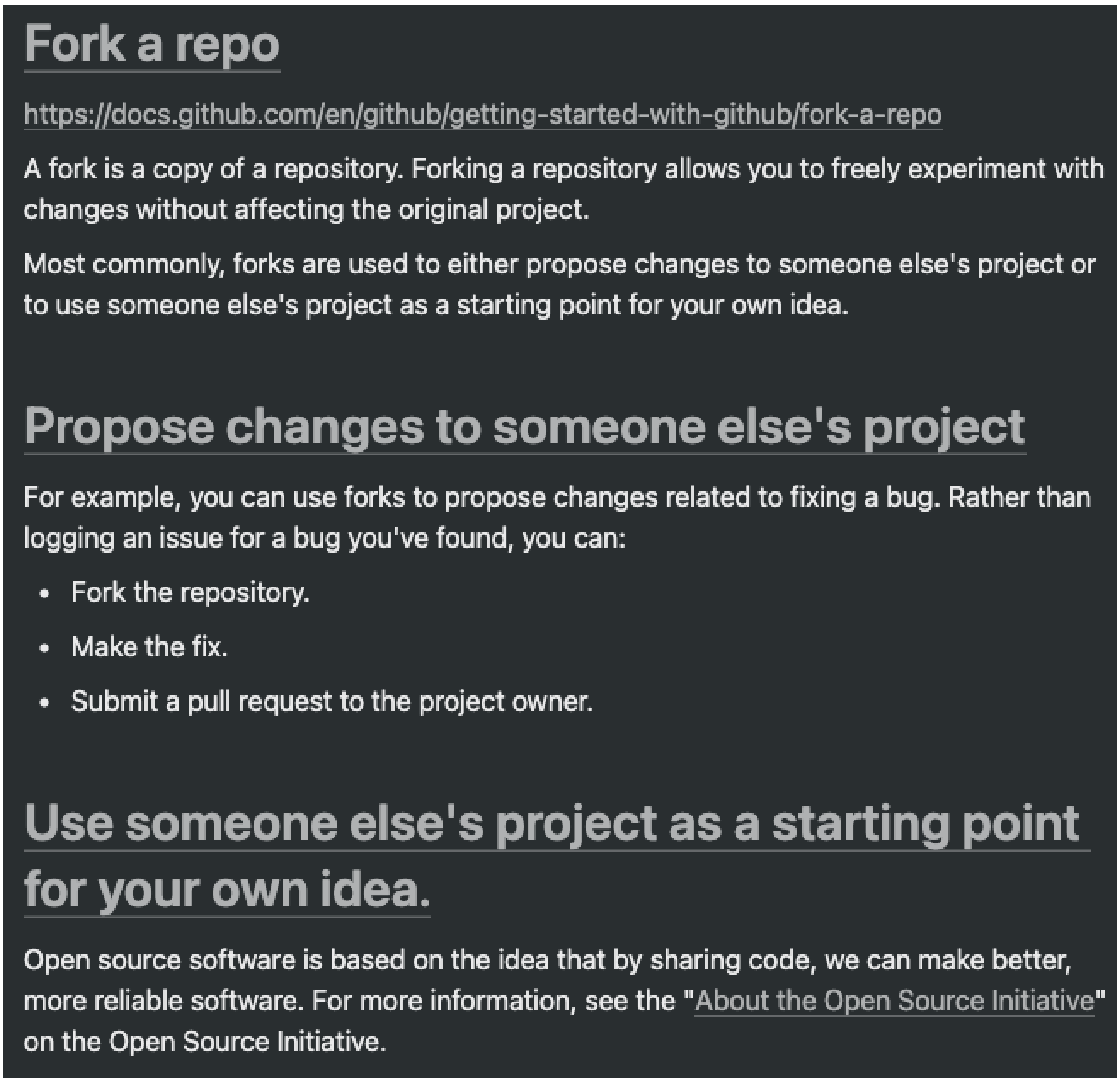
Great news everyone!

It was recently announced that free users will now also get access to private repositories on GitHub. So now, there's absolutely no reason to not keep your code version controlled and backed up on GitHub!

Github - Remote Repository

- Act of copying a repository from remote server to your local machine is called **cloning**
- Cloning allows team to work together
- Downloading commits from others : **fetch, merge**
- Downloading commits from others : **pull (fetch + merge)**
- Uploading your commits (local changes) to remote : **push**





Fork a repo

https://docs.github.com/en/github/getting-started-with-github/fork-a-repo

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Most commonly, forks are used to either propose changes to someone else's project or to use someone else's project as a starting point for your own idea.

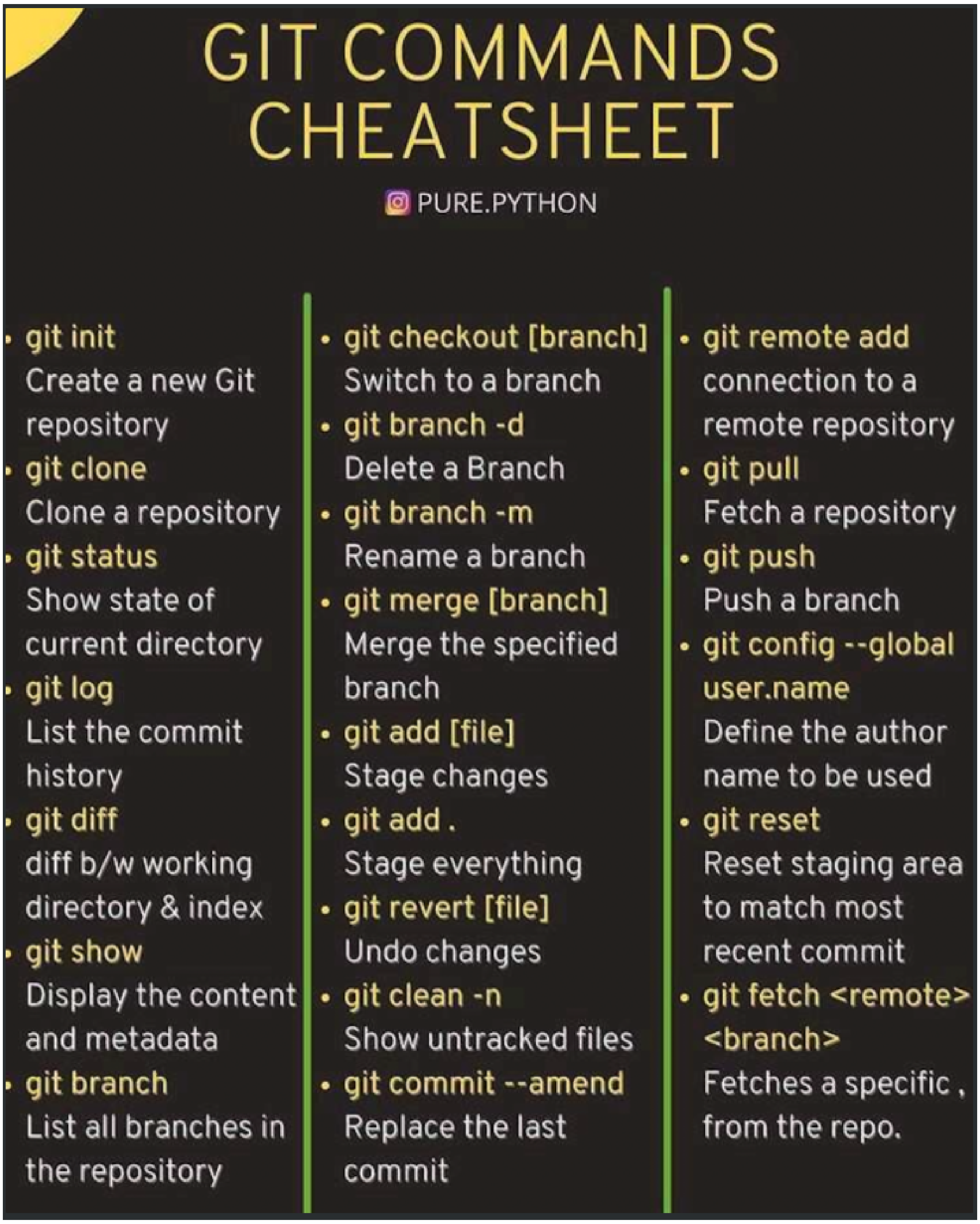
Propose changes to someone else's project

For example, you can use forks to propose changes related to fixing a bug. Rather than logging an issue for a bug you've found, you can:

- Fork the repository.
- Make the fix.
- Submit a pull request to the project owner.

Use someone else's project as a starting point for your own idea.

Open source software is based on the idea that by sharing code, we can make better, more reliable software. For more information, see the "About the Open Source Initiative" on the Open Source Initiative.



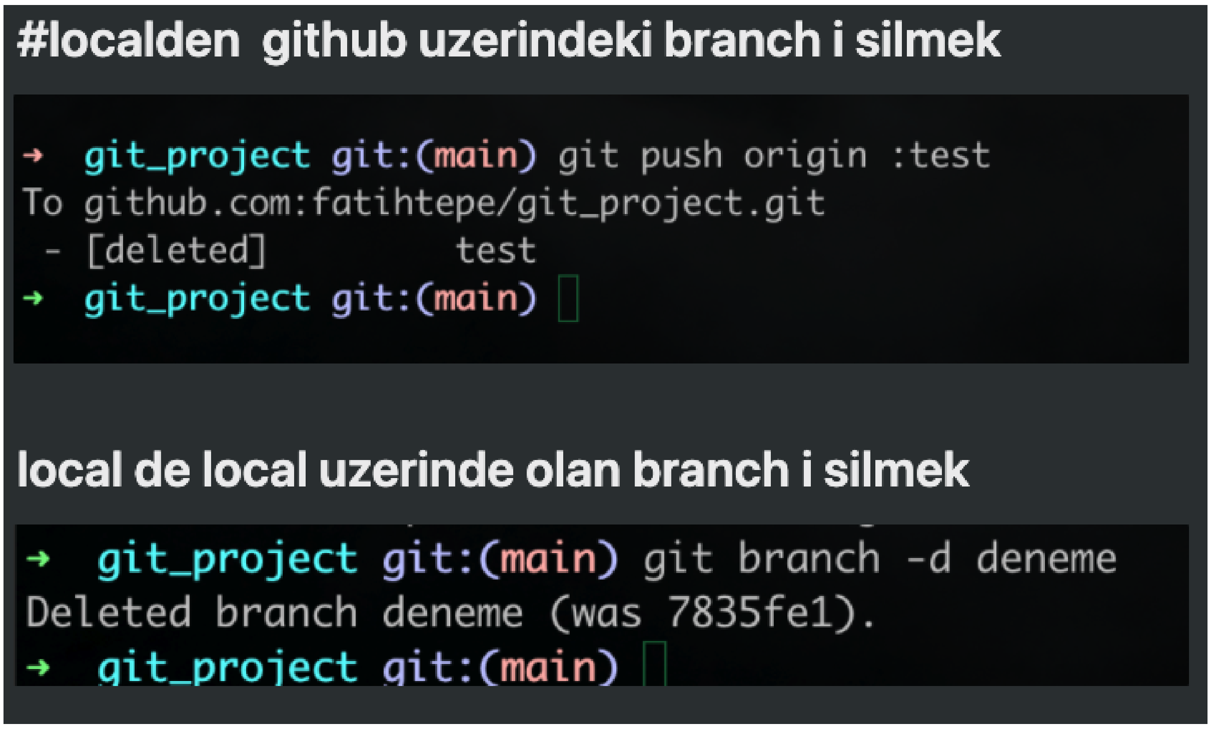
GIT COMMANDS CHEATSHEET

PURE.PYTHON

- git init
  Create a new Git repository
- git clone
  Clone a repository
- git status
  Show state of current directory
- git log
  List the commit history
- git diff
  diff b/w working directory & index
- git show
  Display the content and metadata
- git branch
  List all branches in the repository

- git checkout [branch]
  Switch to a branch
- git branch -d
  Delete a Branch
- git branch -m
  Rename a branch
- git merge [branch]
  Merge the specified branch
- git add [file]
  Stage changes
- git add .
  Stage everything
- git revert [file]
  Undo changes
- git clean -n
  Show untracked files
- git commit --amend
  Replace the last commit

- git remote add
  connection to a remote repository
- git pull
  Fetch a repository
- git push
  Push a branch
- git config --global user.name
  Define the author name to be used
- git reset
  Reset staging area to match most recent commit
- git fetch <remote> <branch>
  Fetches a specific , from the repo.

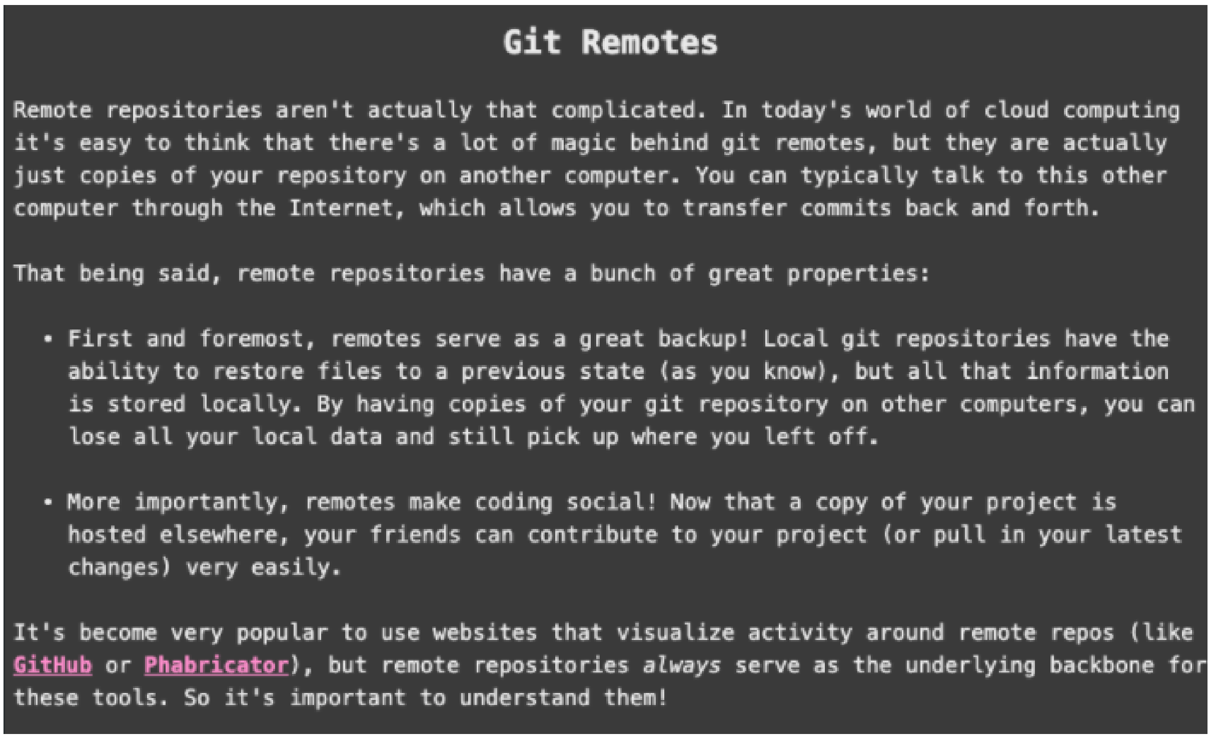


#localden github uzerindeki branch i silmek

```
➜  git_project git:(main) git push origin :test
To github.com:fatihtepe/git_project.git
 - [deleted]         test
➜  git_project git:(main) 
```

local de local uzerinde olan branch i silmek

```
➜  git_project git:(main) git branch -d deneme
Deleted branch deneme (was 7835fe1).
➜  git_project git:(main) 
```



Git Remotes

Remote repositories aren't actually that complicated. In today's world of cloud computing it's easy to think that there's a lot of magic behind git remotes, but they are actually just copies of your repository on another computer. You can typically talk to this other computer through the Internet, which allows you to transfer commits back and forth.

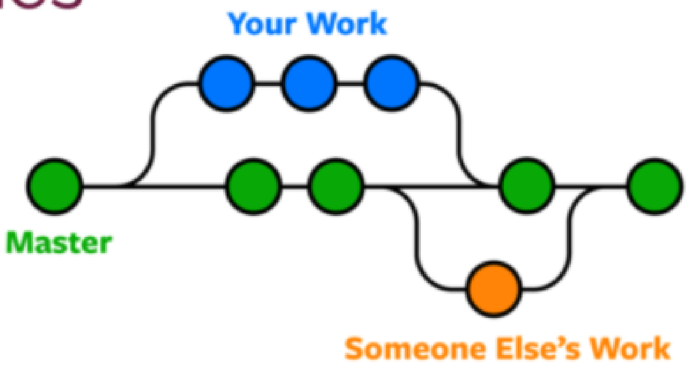
That being said, remote repositories have a bunch of great properties:

- First and foremost, remotes serve as a great backup! Local git repositories have the ability to restore files to a previous state (as you know), but all that information is stored locally. By having copies of your git repository on other computers, you can lose all your local data and still pick up where you left off.

- More importantly, remotes make coding social! Now that a copy of your project is hosted elsewhere, your friends can contribute to your project (or pull in your latest changes) very easily.

It's become very popular to use websites that visualize activity around remote repos (like GitHub or Phabricator), but remote repositories *always* serve as the underlying backbone for these tools. So it's important to understand them!

Branches



- Production of the project lives on master/main branch
- Branches are reference to a commit

Creating/switching branches

- create a new branch

```
git branch Branch name
```

- switch to a branch

```
git checkout Branch name
```

- create a new branch and switch to that branch

```
git checkout -b Branch name
```

Deleting branches

→ delete a local branch

```
git branch -d Branch name
```

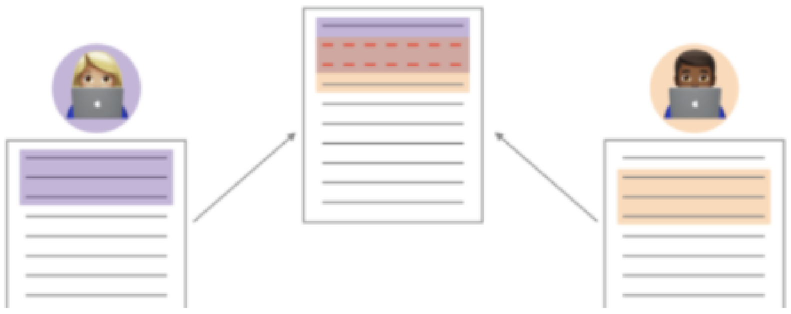
```
git branch -D Branch name
```

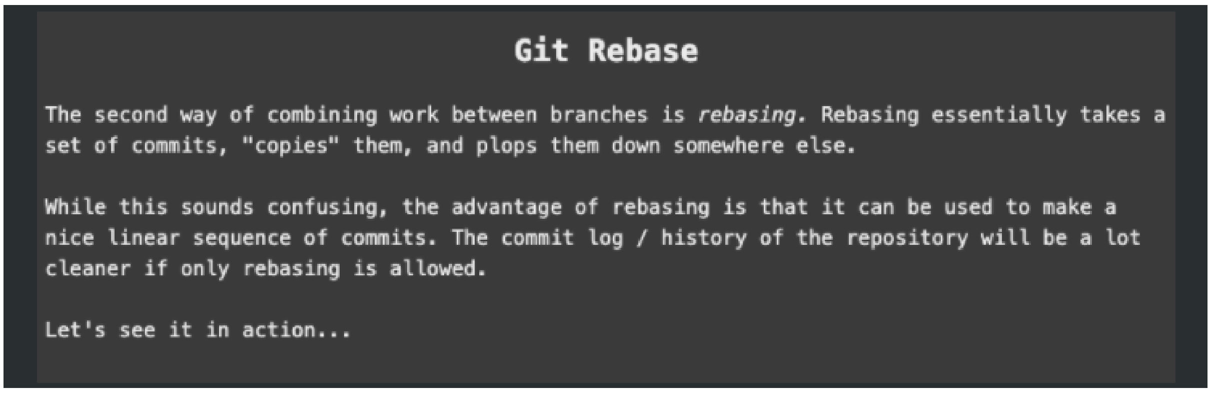
```
-d, --delete
        Delete a branch. The branch must be fully merged in its upstream branch,
or in HEAD if no upstream was set with --track or --set-upstream.

   -D
        Shortcut for --delete --force.
```

Github - Merge Conflict

→ **Merge conflicts** happen when you merge branches that have competing commits, and Git needs your help to decide which changes to incorporate in the final merge.



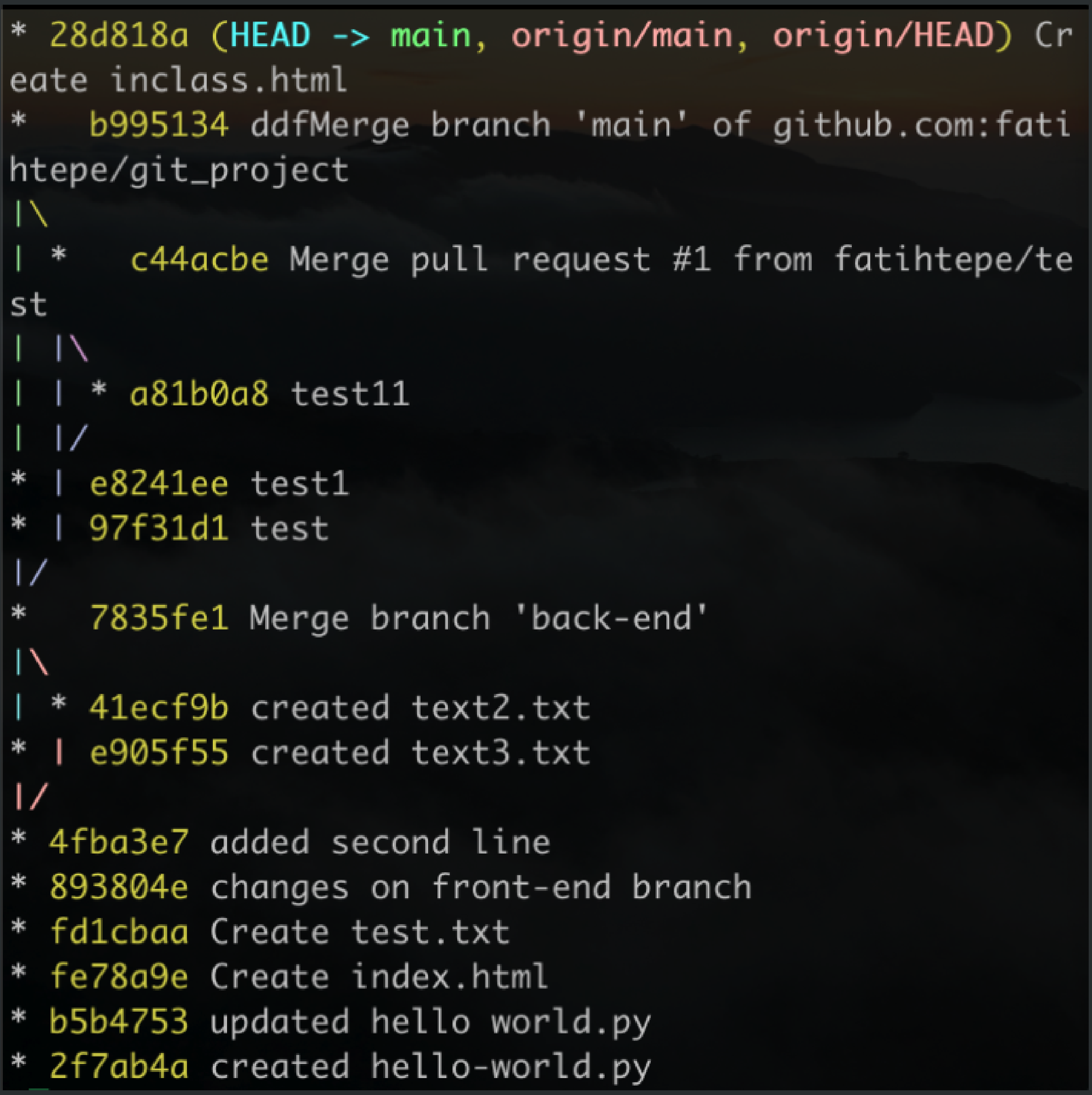

Git Rebase

The second way of combining work between branches is *rebasing.* Rebasing essentially takes a set of commits, "copies" them, and plops them down somewhere else.

While this sounds confusing, the advantage of rebasing is that it can be used to make a nice linear sequence of commits. The commit log / history of the repository will be a lot cleaner if only rebasing is allowed.

Let's see it in action...

alias— git log etc

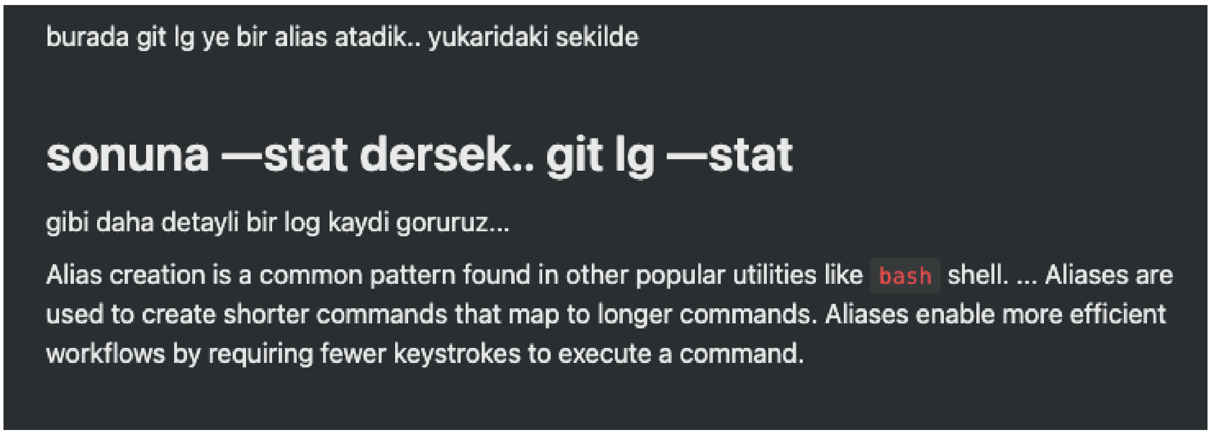


```
* 28d818a (HEAD -> main, origin/main, origin/HEAD) Cr
eate inclass.html
*   b995134 ddfMerge branch 'main' of github.com:fati
htepe/git_project
|\
| *   c44acbe Merge pull request #1 from fatihtepe/te
st
| |\
| | * a81b0a8 test11
| |/
* | e8241ee test1
* | 97f31d1 test
|/
*   7835fe1 Merge branch 'back-end'
|\
| * 41ecf9b created text2.txt
* | e905f55 created text3.txt
|/
* 4fba3e7 added second line
* 893804e changes on front-end branch
* fd1cbaa Create test.txt
* fe78a9e Create index.html
* b5b4753 updated hello world.py
* 2f7ab4a created hello-world.py
```

```
git config --global alias.lg "log --graph --decorate --oneline --all"

git help lg

`git lg' is aliased to `log --graph --decorate --oneline --all'
```

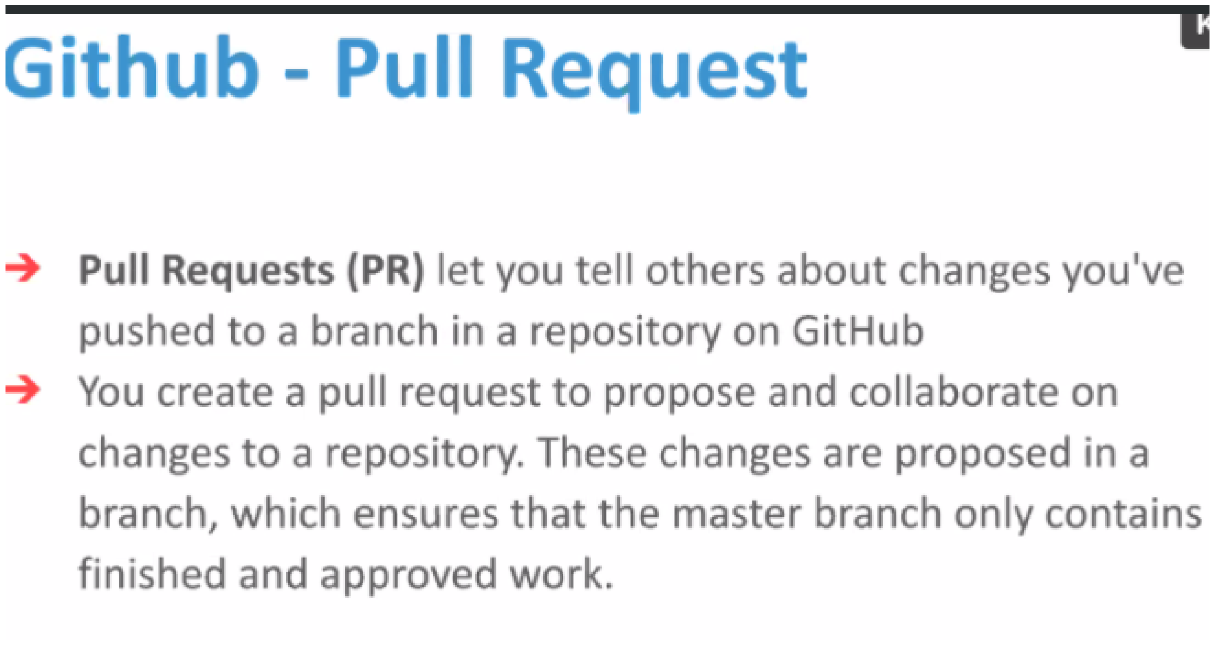


burada git lg ye bir alias atadik.. yukaridaki sekilde

sonuna —stat dersek.. git lg —stat

gibi daha detayli bir log kaydi goruruz...

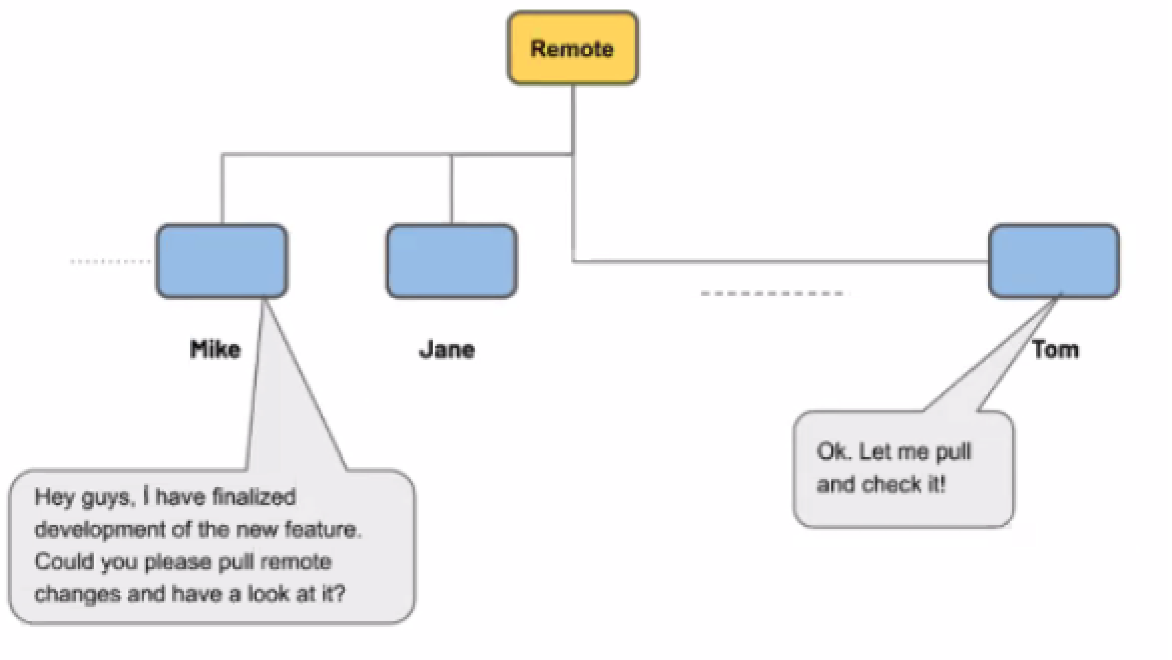
Alias creation is a common pattern found in other popular utilities like `bash` shell. ... Aliases are used to create shorter commands that map to longer commands. Aliases enable more efficient workflows by requiring fewer keystrokes to execute a command.



Github - Pull Request

- **Pull Requests (PR)** let you tell others about changes you've pushed to a branch in a repository on GitHub
- You create a pull request to propose and collaborate on changes to a repository. These changes are proposed in a branch, which ensures that the master branch only contains finished and approved work.

Why “pull” request?


Remote
Mike
Jane
Tom
Hey guys, I have finalized development of the new feature. Could you please pull remote changes and have a look at it?
Ok. Let me pull and check it!