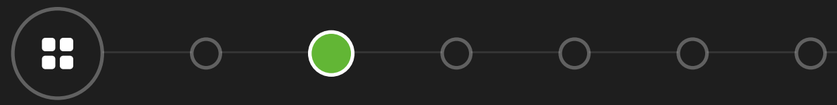
Introduction to Git and Github

Lesson: T6.1, T6.2, T6.3

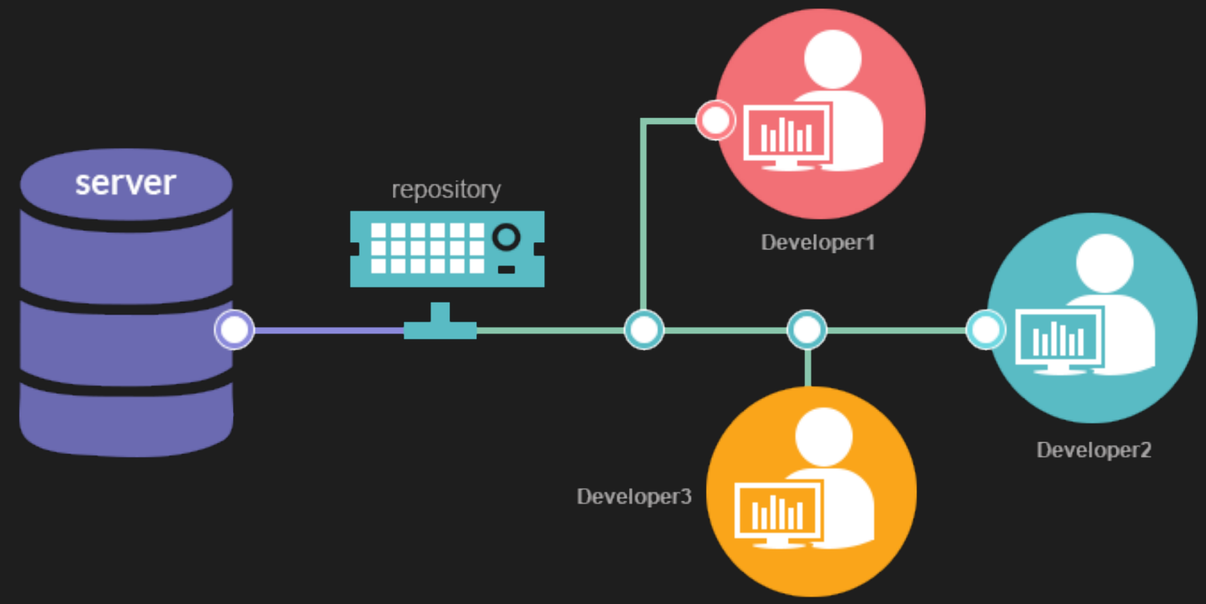


You will be able to learn (Таны энэ хичээлээс сурах зүйлс)

- What is Version Control System?
- How to work on a project in a team?
- What is GIT and how to use it?
- What is GitHub and how to use it?

What is Version Control System?

Version Control Systems are basically a category of software tools that helps in saving and identifying the changes made in the files of computer programs, documents, or other collections of information so that it becomes easier to track those changes even after a long period of time.



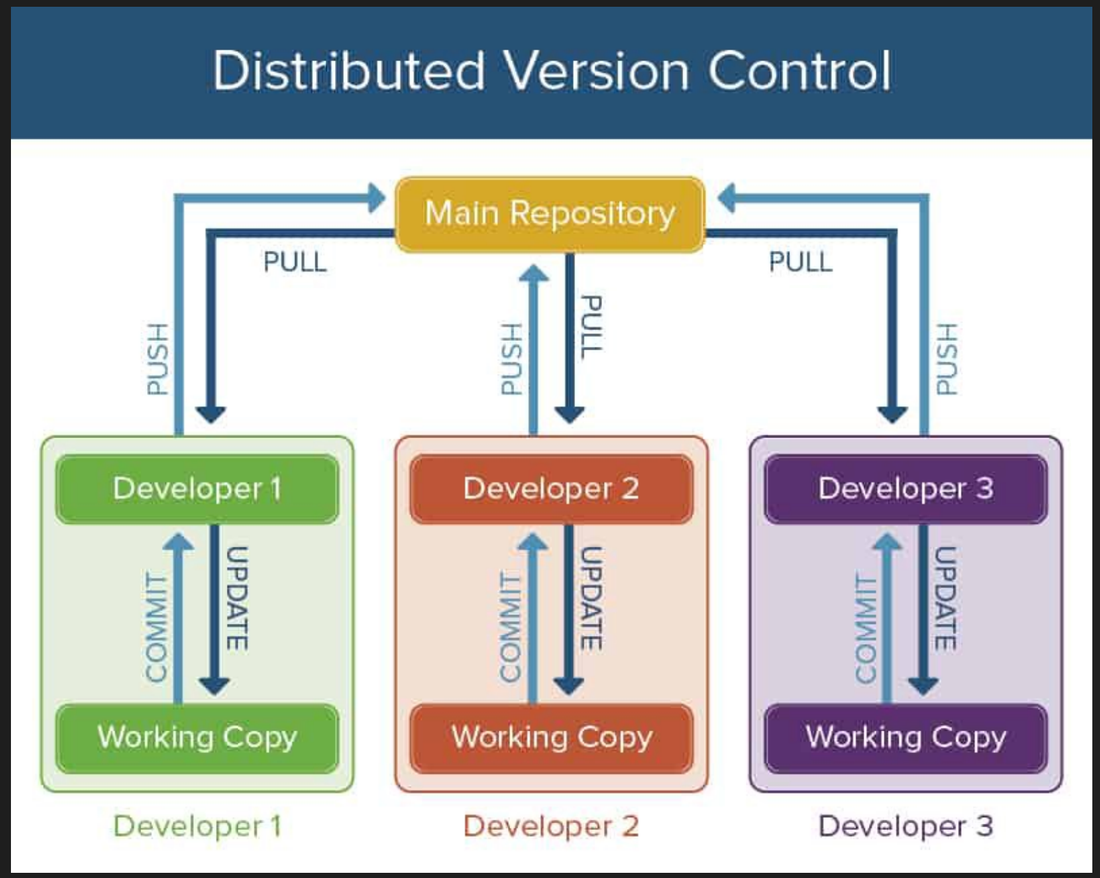
Benefits that you can't ignore

- Enhances the project development speed
- Provide a very efficient collaboration within the team
- Reduces the possibility of error and conflicts
- Even a very small change is traceable
- Devs can contribute to the project from anywhere
- For different contributors, a different copy can be maintained and can be verified before merging to the main project
- Recovery of the previous state is possible
- A lot of information about who, why, and when can be accessed
- Saves a lot of time for your coffee.

What is Version Control System?

Types of Version Control System

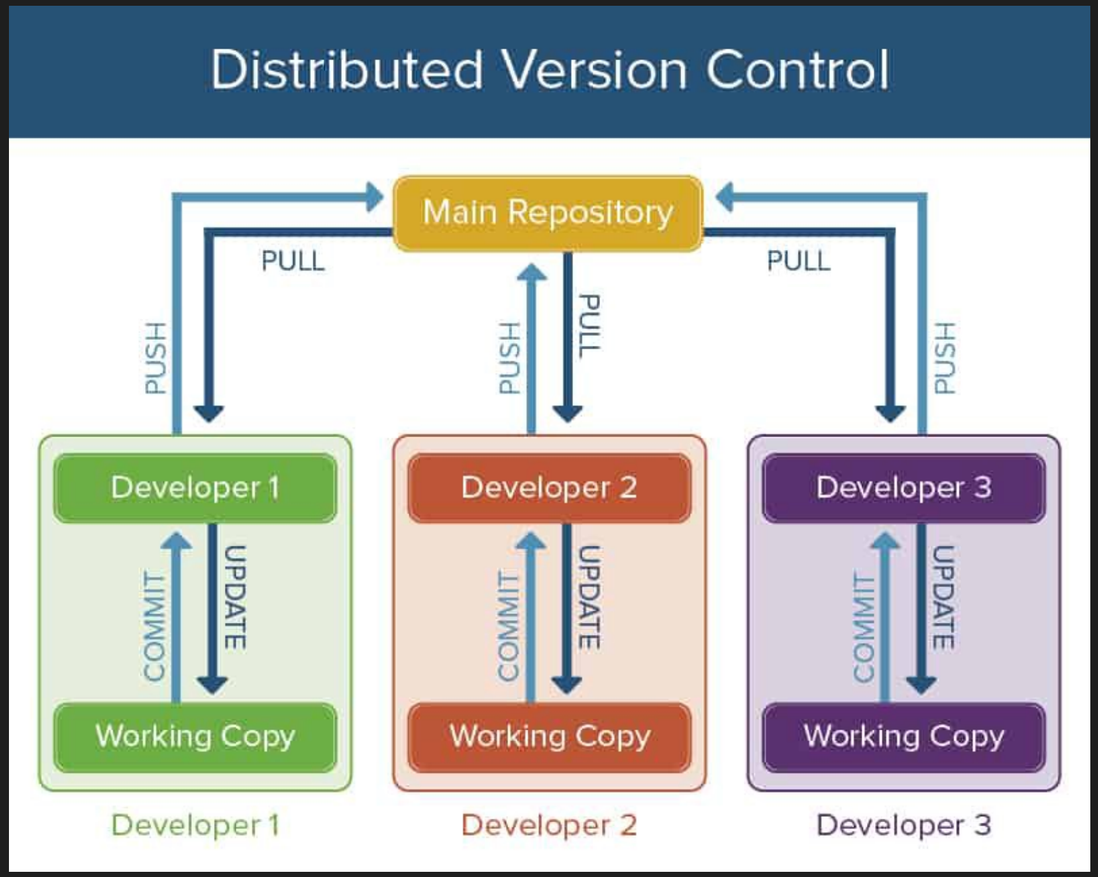
- Local Version Control Systems
- Centralized Version Control Systems
- Distributed Version Control Systems



What is Version Control System?

5 Free Open-source VCS Tools to check

- Git
- Concurrent Version controls (CVS)
- Apache Subversion (SVN)
- Mercurial
- Bazaar



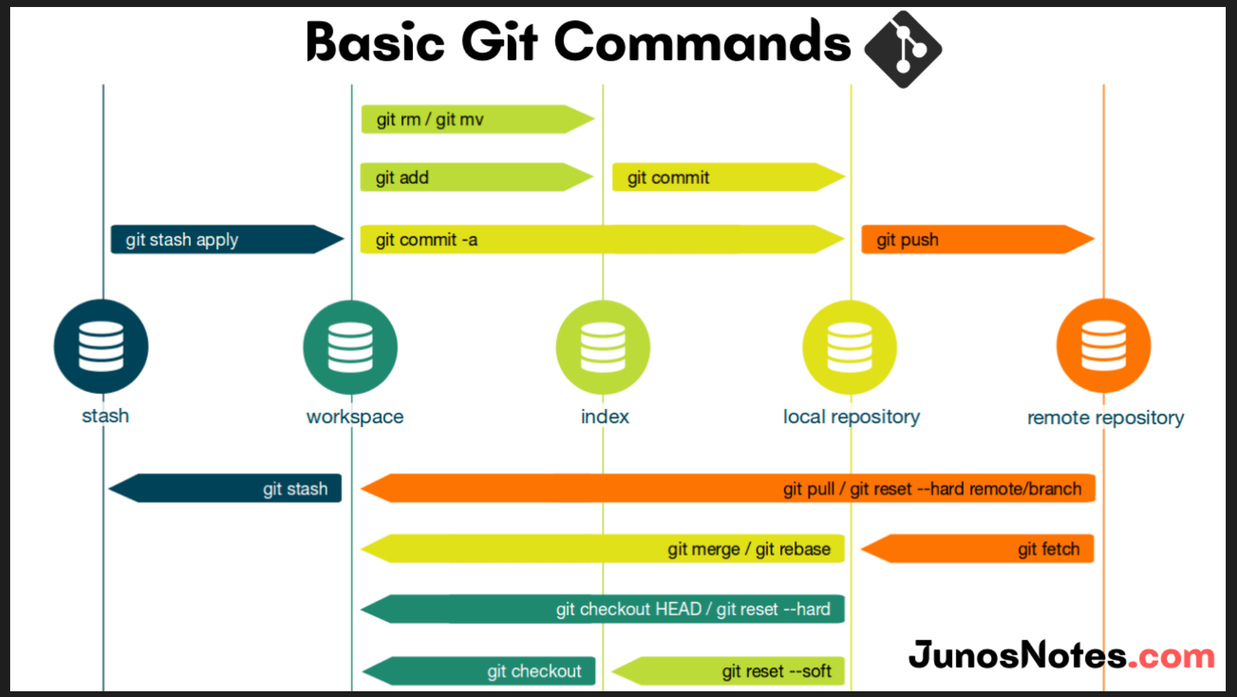
What is GIT?

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Background and History

- Initial Release - 7 April 2005
- Original Author - Linus Torvalds, who created Linux Kernel

Git is a distributed version control system, as opposed to a centralized system. In a distributed system, you can copy a complete repository with the full project history to every developer's machine.



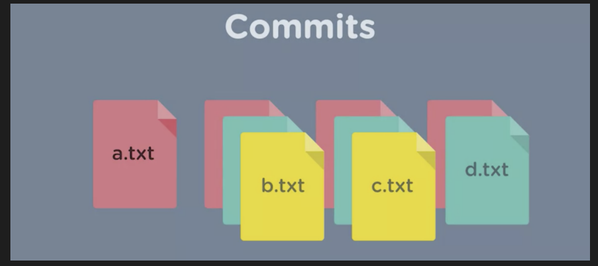
What is GIT?

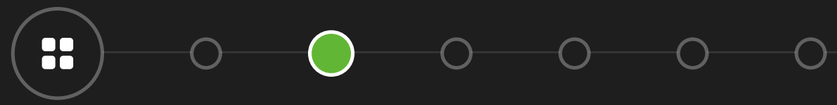
Git Repository

- It is basically a folder with your project files
- In this folder, we run Git commands to store your changes
- You can have multiple projects and repositories on your local computer

Git Commits

- Each time you complete a change to some or all of your project's files, you can take a snapshot of their current contents. These snapshots are known as **commits**.





What is GIT?

Let’s apply our knowledge in practice

- Download **medals.zip** file from Google Classroom Topic Git and Github

What is GIT?

Let’s do some preparation

1. Download MacOS git installer from https://git-scm.com
2. After successfully installation open your terminal by typing on SpotLight TERMINAL
3. Move the downloaded folder medals on your Desktop
4. Go to your Desktop by writing on TERMINAL
    a. **cd** ~/Desktop
5. Go into your medals folder on your Desktop on TERMINAL
    a. cd medals

Some important terminal commands

- **a. ls**
    - i. lists all files of the current directory
    - ii. **ls -a**
        1. lists all files and hidden files of the current directory
- **b. cd**
    - i. go to named folder
    - ii. for example:
        1. **cd** ~/ - go to home directory of current user
        2. **cd** ~/Desktop - go to Desktop folder of current user
        3. **cd** .. - go out from current directory
        4. etc...

Git commands

git --help
this command will show all the commands of git

git clone
cloning remote repository into your local computer

git init
set up new repositories

git add
add local changes to the repository

git status
seeing what is changed on your local machine on the repository

git commit
after adding the local changes, commiting as a new version of the changes to the repository

git log
view a list of old commits

git mv
move files tracked by Git

git rm
remove files tracked by Git

git push
pushing to the remote repository with the current commited changes

git pull
getting all remote repository changes to the local computer

git diff
it will show current changes of changed files in the repo

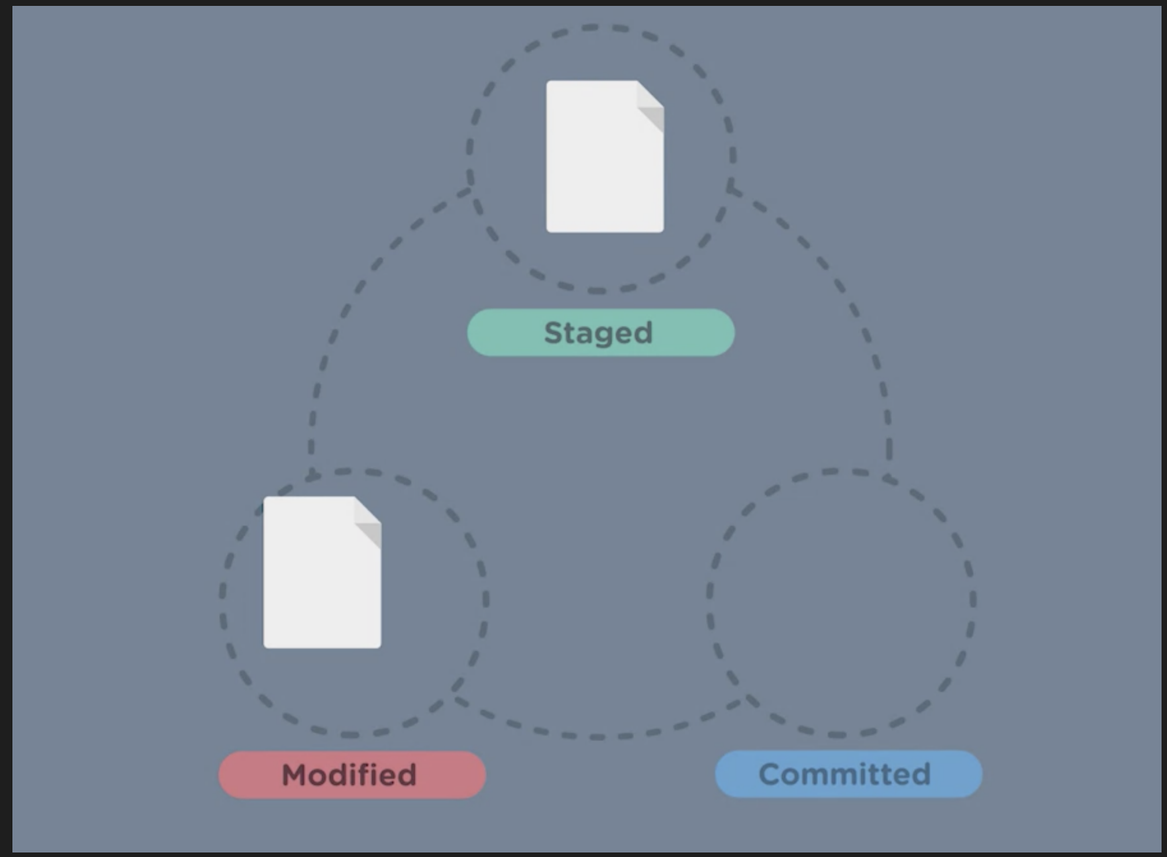
git revert
it will revert the specified SHA hash commit

Git changes

Modified
The repository has local change

Staged
The local change has been added as staged before committed

Committed
The change is committed and ready to be pushed to the remote repository



Git initialize, add changes and commit the changes

Let's do most important steps using Git for tracking changes on the project

1. In your medals folder, execute the following command
   a. **git init**
2. Then execute
   a. **git add medals.html**
3. After that
   a. **git commit -m "Add main site page"**

Sample commit messages

1. *Good commit messages*
   a. Add main site page
   b. Remove sale description
   c. Add new Products
2. *Bad commit messages*
   a. commit
   b. test
   c. fix
   d. etc.

Git challenge

Task 1

We're working on our first novel, and we want to track our changes in Git. But first, we need to change into the directory where our text files are. Using the terminal, change into the **novel** directory on the Desktop. If you have not the novel directory, create the folder.

Task 2

Initialize a new Git repository here in the current directory.

Task 3

Add the file **chapter1.txt** to the staging area for committing.

Task 4

Finally, commit the staged file, with a message of **"Add chapter 1"**. Rather than letting Git launch an editor, specify your commit message on the command line: use the **-m** command line option, and specify your message between **"quotation marks"**.

Git Diff – showing local change differences

Git diff

It will show the difference of the local changes

Git diff --staged

It will show the difference of the local changed files on staging

Git challenge – use change differences

Task 1

We've updated one file in the repo, and created another file. Run the git command that will show you the status of the repo's files.

Task 2

Now run the git command that will show you the changes made within the files. (Changes to the tracked file will be shown, but not the untracked file.)

Task 3

Stage chapter1.txt

Task 4

Now stage chapter2.txt

Task 5

Run the "git diff" command again, but this time add the option that will show you the staged changes. (When you have the command right, changes to both the file that was previously tracked and the file that was previously untracked will be shown.)

Task 6

Finally add all the staged files with commit option -m. To the message, use what ever message you want to add, but please use meaninful message.

Git – managing committed files

Managing committed files

1. **Deleting files**
2. Moving/renaming files
3. Unstaging files
4. Discarding modifications
5. Retrieving deleted files
6. Undoing commits

1. Deleting files

- **rm tin.html**
  - we can delete tin.html file from the system, but it is not deleted from the repository. For that reason, we use the following command to remove files from repository.
- **git rm tin.html**
  - will delete this file from the git repository

Git challenge – removing files from repository

Task 1

We've committed an appendix.txt file to the repo, but we decided we don't want it any more. Run the git command to remove the file.

Task 2

Run the command to display the status of the repo’s files

Task 3

Commit the file removal. Use the -m flag, with any commit message you like

Git – managing committed files

Managing committed files

1. Deleting files
2. **Moving/renaming files**
3. Unstaging files
4. Discarding modifications
5. Retrieving deleted files
6. Undoing commits

2. Moving/renaming files

- **git mv silver.txt silver.html**
  - will change the wrong name of the file into the correct file

Git challenge – renaming files from repository

Task 1

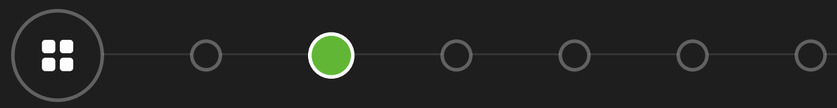
We accidentally committed a file with the wrong file name extension. Run the git command to move the "chapter3.text" file to the name "chapter3.txt".

Task 2

Run the command to display the status of the repo's files.

Task 3

Commit the file removal. Use the -m flag, with any commit message you like



Git – managing committed files

Managing committed files

1. Deleting files
2. Moving/renaming files
3. **Unstaging files**
4. Discarding modifications
5. Retrieving deleted files
6. Undoing commits

3. Unstaging files

- **git reset HEAD medals.html**
  - will reset the staged change to a file of the current repository

Git challenge – unstaging files from repository

Task 1

We staged a change to a file, but we've decided it wasn't a good idea. Run "git status" to display the staged file names.

Task 2

Running git status gave the following output:

```
On branch master Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   chapter3.txt
```

Let's run the command it suggests to unstage the file.

Git – managing committed files

Managing committed files

1. Deleting files
2. Moving/renaming files
3. Unstaging files
4. **Discarding modifications**
5. Retrieving deleted files
6. Undoing commits

4. Discarding modifications

- **git checkout -- medals.html**
  - will discard the local changes made to the file medals.html

Git challenge – discarding changes from repository

Task 1

We've unstaged chapter3.txt, but the file is still modified in the working directory. Run "git status" to display its status.

Task 2

Running git status gave the following output:

```
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   chapter3.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Let's run the command it suggests to discard the changes to chapter3.txt.

Git – managing committed files

Managing committed files

1. Deleting files
2. Moving/renaming files
3. Unstaging files
4. Discarding modifications
5. **Retrieving deleted files**
6. Undoing commits

5. Retrieving deleted files

- **git checkout -- medals.html**
  - will discard the local changes made to the file medals.html

Git challenge – doing uncommits from repository

Task 1

Oops! We accidentally deleted some files from the working directory! Run "git status" to see which ones.

Task 3

Now git status gives this output:

```
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        deleted:    chapter3.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

We want to bring chapter3.txt back as well. Run the necessary command.

Task 2

Running git status gave the following output:

```
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        deleted:    chapter2.txt
        deleted:    chapter3.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

We want to bring chapter2.txt back, and we can do so by "discarding the change" to the file. ("Discarding" the deletion will bring the file back.) Run the command suggested in the output to discard changes to chapter2.txt.

Git – managing committed files

Managing committed files

1. Deleting files
2. Moving/renaming files
3. Unstaging files
4. Discarding modifications
5. Retrieving deleted files
6. **Undoing commits**

6. Undoing commits

- **git revert 094ba15da4f01515a382272b9d38dd82504ecfa9**
  - will revert back the specific commit with SHA
    - value = 094ba15da4f01515a38
- **git revert HEAD**
  - will revert back the most recent commit

Git challenge – retrieving deleted files from repository

Task 1

We've decided we need to get back the **appendix.txt** file that we deleted before. First, we need to find the commit where we removed it. Run the command that will show the log of all prior commits.

Task 2

Running git log gave the following output:

```
commit 70692324faa22f3eaeeb4d80e0958f349f4155bc
Author: Treehouse Student <me@example.com>
Date:   Fri Jan 19 13:13:30 2018 -0700

    Fix file name

commit 962f0e9138bab1ebb5b3e21d18a747fe18236714
Author: Treehouse Student <me@example.com>
Date:   Fri Jan 19 12:57:04 2018 -0700

    Remove appendix
...
```

We want to bring appendix.txt back, which we can do by reverting the commit with the message "Remove appendix". Find the SHA checksum for that commit, and use it in the appropriate command. (This command triggers a new commit, so normally, an editor would pop up to allow you to edit that new commit's message. For this challenge, though, we just want you to run the command.)

Git Practice

Task 1

Download the file

- https://s3.amazonaws.com/treehouse-code-samples/IntroductionToGit/git_practice_managing_committed_files.zip

and

- Save it in your Desktop and unzip it

Task 2

We want to keep the **f.txt file**, but we accidentally ran **git rm f.txt**. Unstage the file deletion, then **restore** the copy in the working directory.

Task 3

The **a.txt file** has been committed to the repo. Make a commit that deletes it from the repo.

Task 4

The **c.txt file** has accidentally been deleted. Restore the copy in the working directory.

Task 5

The **z.txt file** should actually be named **e.txt**. Make a commit that moves it to the correct name.

Task 6

In the Git log, find the commit with the message **"Remove g.txt and h.txt"**, and revert it.

Task 7

You'll know you're done when running **ls** **produces** this output:

```
a.txt  b.txt  c.txt  d.txt  e.txt  f.txt  g.txt  h.txt
```

Task 8

And when running **git status** produces this output:

```
On branch master
nothing to commit, working tree clean
```

Github and Other Remote Repositories

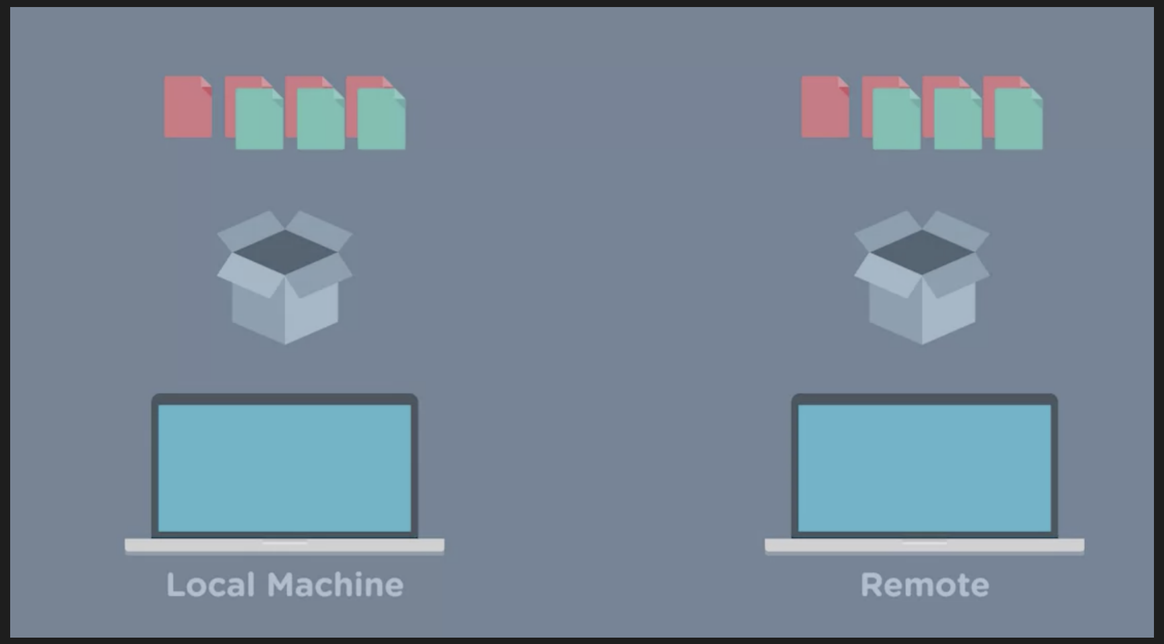
We have worked only on our local machine so far.
At some point, we share or merge our code with others.

Local repository (repo)

Every developer has a repository of the code on her/his local machine

Remote repository (repo)

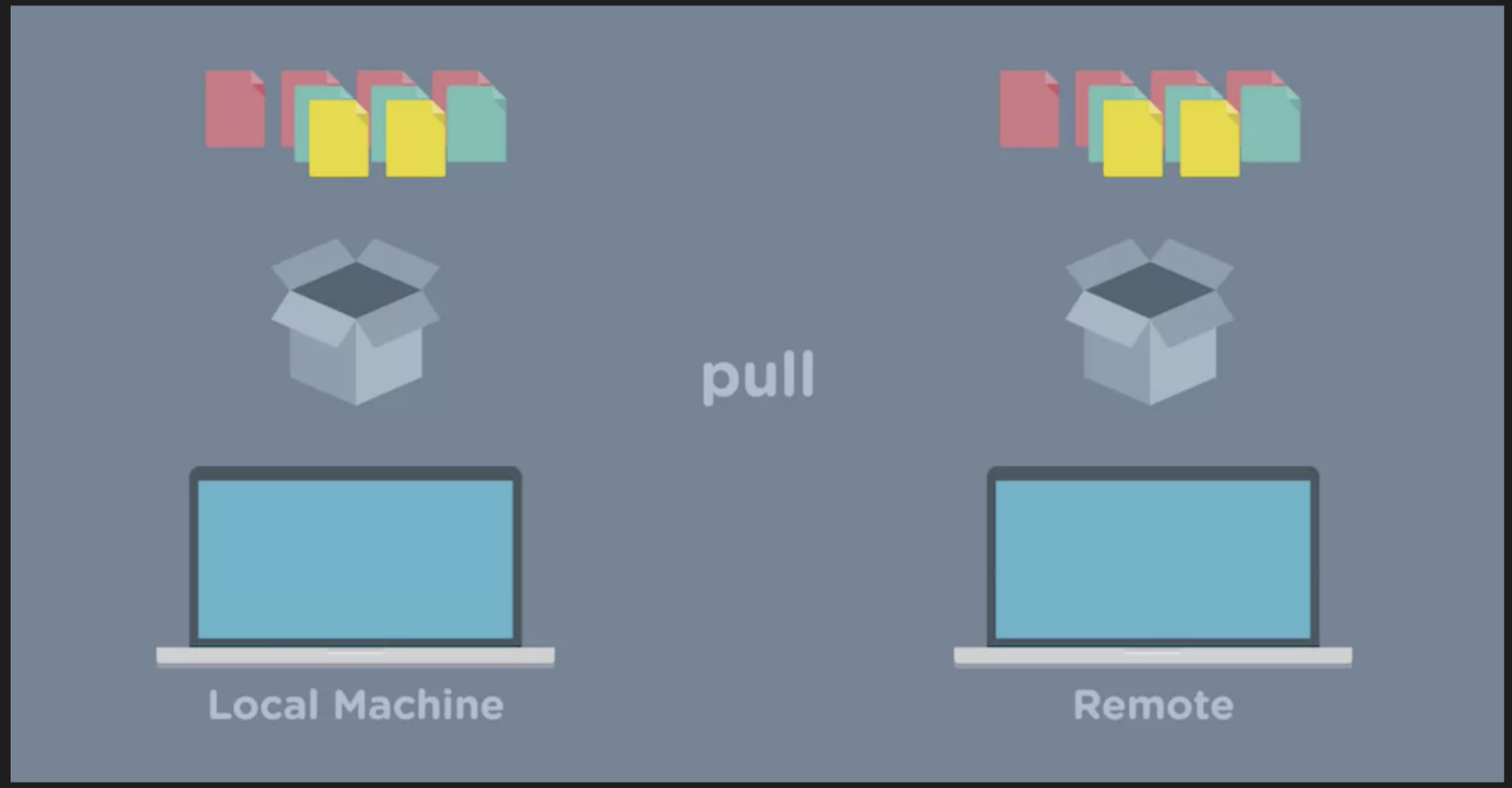
A copy of the local repository based on other server or machine



Github and Other Remote Repositories

If there is a change on remote repository, which does not exist on developers local machine, we need to PULL from the remote repository to get the copy of the changes on the developer's local machine

- **Command to pulling the remote changes on your local machine**
  - git pull



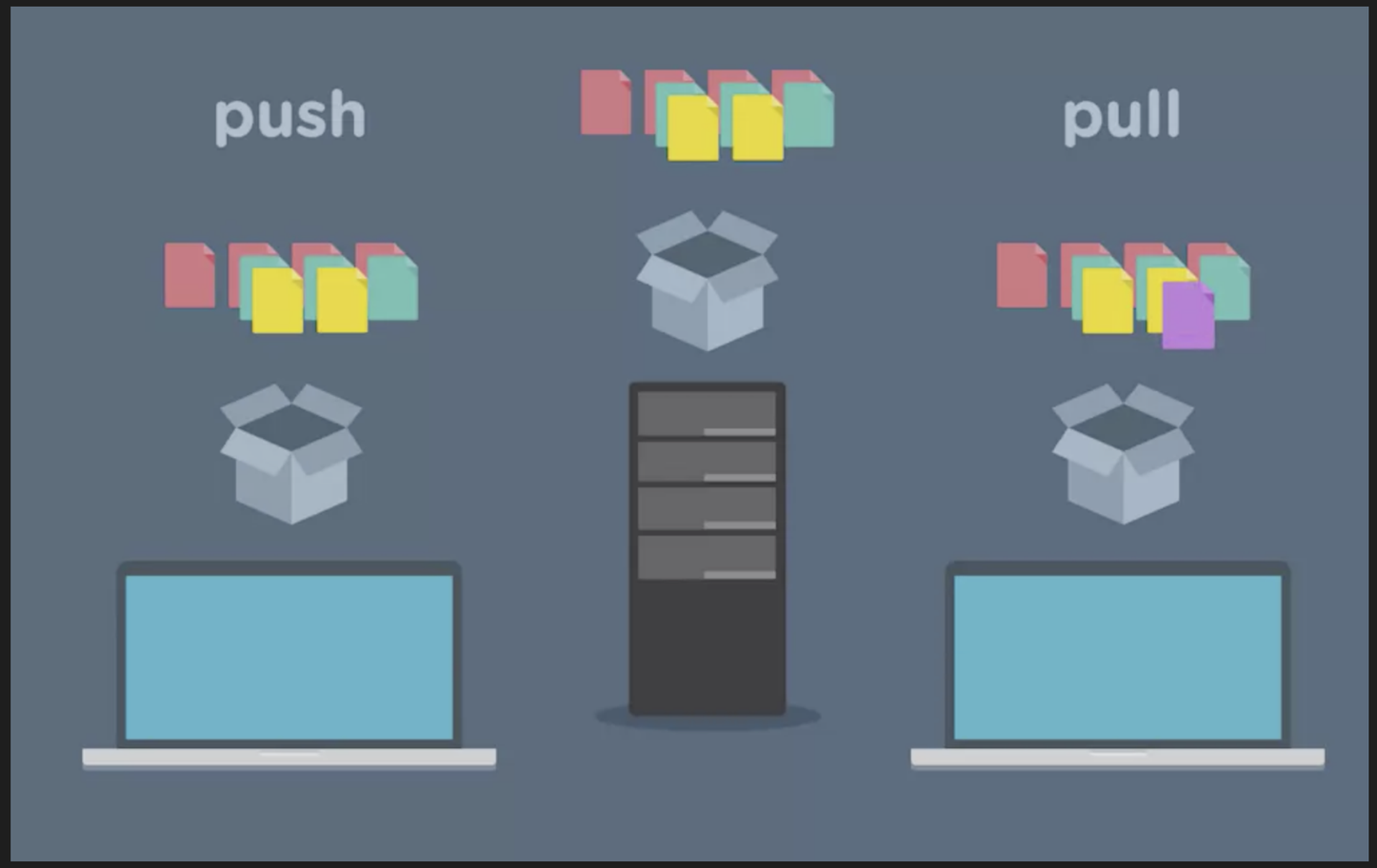
Github and Other Remote Repositories

> In the most cases, we have a remote repository on remote server.
> One developer makes change to the repository and pushes the change to the remote repository and other developer pulls the changes from the developer to have the current repository changes.

> This service will be supported by

- Github
- Gitlab
- Bitbucket
- etc.

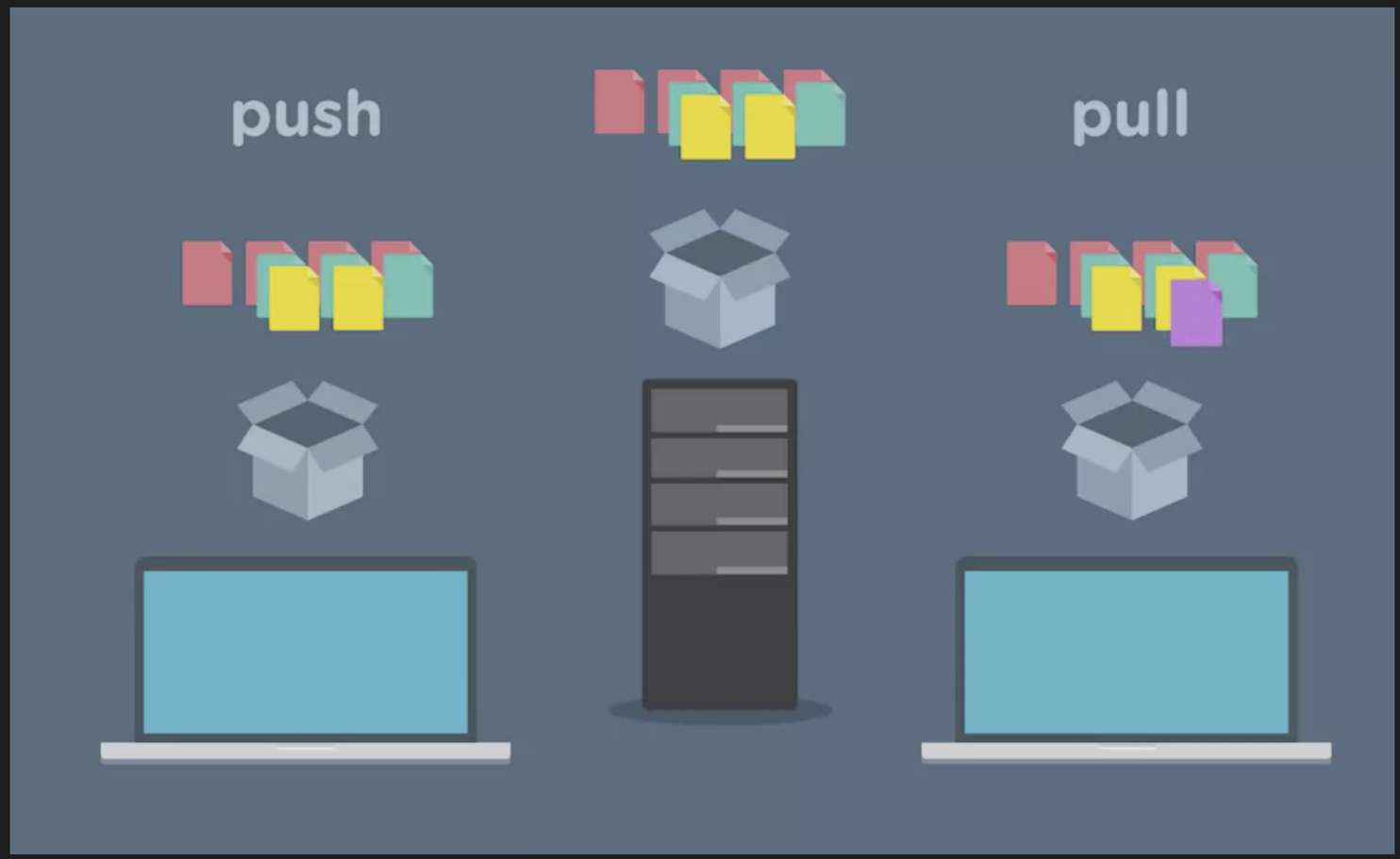
- Command to pushing to the remote repository from your local machine
  - git push



Github and Other Remote Repositories

To get the remote server repository on your local machine

- **git clone REMOTE_REPO_URL**
  - For example:
  - git clone https://github.com/khangaikhuu-mstars/git-and-github
- **git clone REMOTE_REPO_URL my_clone**
  - For example:
  - git clone https://github.com/khangaikhuu-mstars/git-and-github my_clone



Git challenge – remote repository cloning

Task 1

- Let's try cloning a Git repository. But this time, instead of cloning a local repo, we'll clone one from the Internet.
- Here's a terminal on a different computer. Run the "git clone" command.
- Instead of a local directory name to clone from, give it a URL of **"https://github.com/khangaikhuu-mstars/mstars-class-three.git"**.
- By default, this will clone to a local directory named git-and-github, but you can specify a different directory name after the URL, if you want.

Task 2

1. Every student should create a new folder with her/his name
2. Inside your named folder, create a index.html file with basic html document inside in it.

Task 3

Do the staging with the current change in the repository.

Task 4

Do the commit with current staging changes with meaninful messages

Task 5

- Push the most recent commit to the remote server
- If there is error message, you may need to PULL the changes from the remote repository

Github and Other Remote Repositories

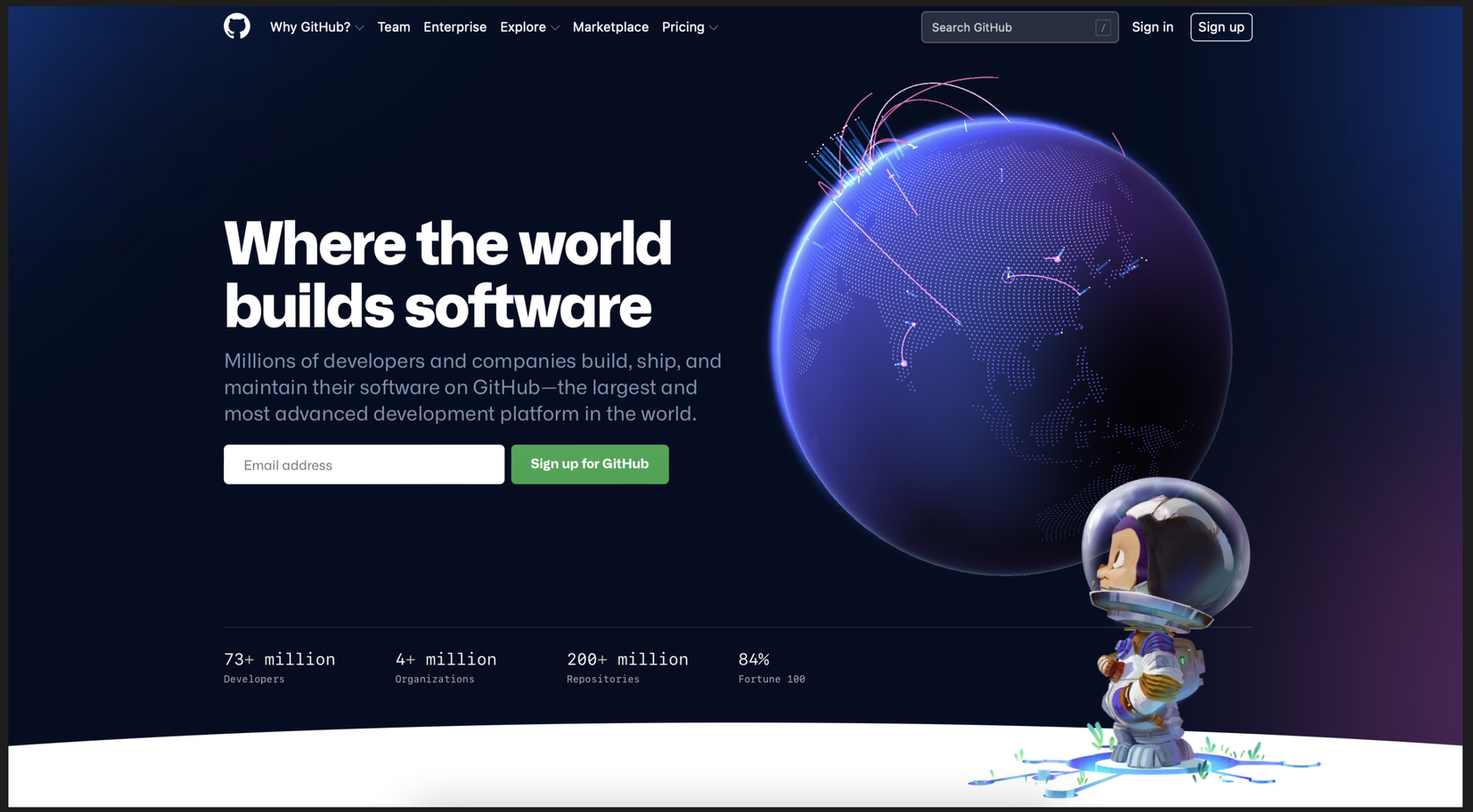
In developer's local repository, if there is no remote repository, a remote repository can be added by the command

- **git remote add origin** https://github.com/khangaikhuu-mstars/mstars-class-three.git

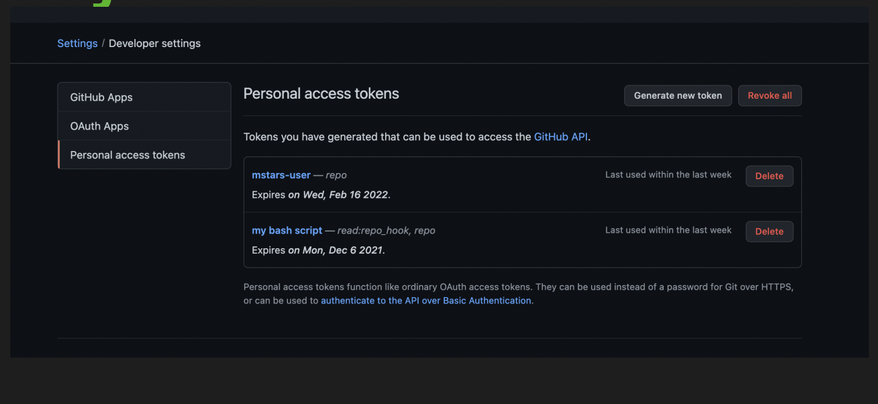
Introduction to Github

Github is a remote git repository hosting server.

1. It has 2 kind of services for deploying git repos
   a. Enterprise or paid
   b. Free
2. Founded by Tom Preston-Werner, Chris Wanstrath and P.J.Hyett in 2008
3. Acquired by Microsoft 2018 for
   a. US$7.5 Billion



Github challenge– create an account and new repository on Github



Task 1

- Create an account in Github
- Create a personal access token and save it safely

Task 2

- Create a new public repository with the name
  - **git-and-github**

Task 3

- Create a new local folder on your desktop

Task 4

- Inside this folder create a new file index.html with some example HTML content

Task 5

- Initialize new git repository inside this folder
- Do staging the changes
- Commit the changes with meaningful message

Task 6

- Add remote Github repository to the local repository folder
  - **git remote add origin YOUR_GITHUB_REPOSITORY_URL**

Task 7

- Push the local repository to the remote Github repository
  - **git push -u origin master**

Task 8

- Check the Github remote repository, whether your local repository pushed to the the remote repository

Github challenge– create new pages repository on Github

Task 1

- Create a new repository your_name.github.io in Github

Task 2

- Add your profile HTML page into the Repository

Task 3

- Stage the file in the repository

Task 4

- Commit the changes in the repository

Task 5

- Push to the remote repository

- For more information
  - https://pages.github.com/

Github challenge– add all your files with your html file into github repository

Task 1

- Clone the repository
  - https://github.com/khangaikhuu-mstars/git-and-github

Task 2

- Add a folder with your name in this repository

Task 3

- Add all your created files and folder of html, css and js exercises into your own named folder

Task 4

- Stage the files in the repository

Task 5

- Commit the changes

Task 6

- Push to the repository
  - Use the github authentication key and username

Recap

Starting a repo
- git init

Managing committed files
- git rm and git mv
- Commit SHA's
- git revert

Committing files
- git add and git commit
- git status
- git log

Working with remotes
- git clone
- git pull
- git add
- git push
- git remote