Ce este un sistem de versionare?

Un sistem de versionare este un mod de management al fișierelor care permite păstrarea istoricului tuturor modificărilor aduse fișierelor urmărite. O introducere a tipurilor de sisteme de versionare poate fi citită aici: git-scm.com

De ce controlul versiunilor? Avantajele folosirii unui sistem de versionare?

- ✓ Proiectele software se dezvoltă în regim colaborativ, codul fiind controlat de mai mulți programatori
- ✓ Ai nevoie de o soluție de back-up independent de mașina pe care se face dezvoltare
- ✓ Ai nevoie de o platformă care să permită lucru colaborativ (fiecare lucrează pe mașina locală si apoi modificările sunt încărcate în sistem pentru a fi disponibile tuturor membrilor echipei)
- ✓ Ai nevoie de o platformă care să iți permită accesul la istoricul modificărilor
- ✓ Toata lumea folosește o soluție de control al versiunii (soluție de versionare)

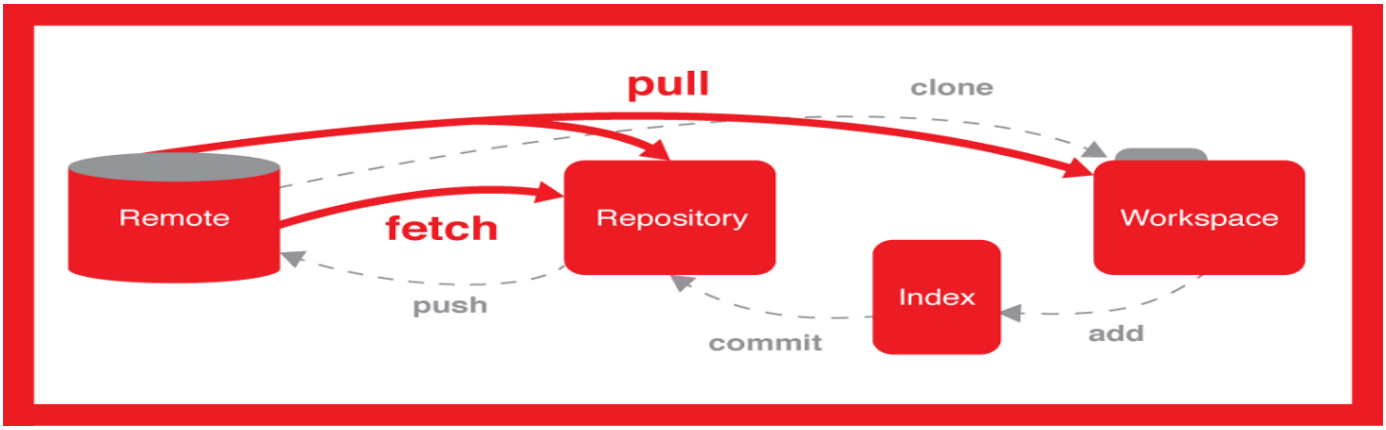
Ce este Git ?

Git este un sistem de versionare a codului și a fost scris de Linux Torvalds, creatorul kernelului Linux.

Ce este github.com?

GitHub este un serviciu de hosting al proiectelor git (un proiect git este numit repository). Acesta permite păstrarea unei copii a proiectului online și vizualizarea tuturor fișierelor și a modificărilor aduse acestora.

Dicționar de termini



git - sistem de versionare

repository (repo) - un „proiect” un “folder” git ce conține toate fișierele și istoricul modificărilor

git init Inițializare repository

git status putem observa ce fisierele sunt in staging.

git add . – adaugă la lista internă de fișiere pe care trebuie să le versioneze

git commit –m 'mesaj' - un set de modicări/adăugări/ștergeri a unui sau a mai multor fișiere din cadrul repository-ului

git push origin master- încarcă/trimite modificările în repo

git pull origin master-descarcă/primește modificările în repo

I. Instalare GIT

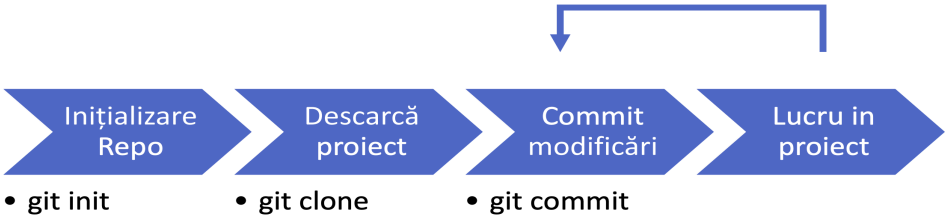
Metoda I: Folosind GIT în terminal/ consolă (RECOMANDAT). Instalare client Git (Git Bash) pe masina de lucru: https://git-scm.com/downloads

Tutorial aici: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
Sau aici: https://www.youtube.com/watch?v=B5t6i-2HWgw

Metoda II: Folosind aplicația GitHub Desktop
https://desktop.github.com/
După instalare va trebui să vă autentificați în aplicație cu username-ul/adresa de mail și parola alese la crearea contului de GitHub.

Operatii de baza Git

- ✓ Inițializare repository
- ✓ Adaugă fișiere noi/modifica fișiere existente
- ✓ Commit – salvează modificările pe server
- ✓ Istoric - vizualizare istoric modificări
- ✓ Share – distribuie modificările
- ✓ Update – actualizează proiectul cu ultimele modificari
- ✓ Revert - anulează modificări cu revenire la o versiune anterioara
- ✓ Branch – creare ramura (proiect secundar) de dezvoltare locala



II. Crearea contului GitHub

Primul pas este crearea unui cont GitHub (https://github.com/), completând un username, adresa de email și o parolă.
La următorul pas va trebui să selectați tipul de cont dorit. Implicit este selectat contul gratuit, care permite crearea de repository-uri publice sau private.
Apăsați „Finish sign up”.
Acesta este un moment bun pentru a valida adresa de email aleasă: tot ce trebuie să faceți este să accesați link-ul din interiorul email-ului primit de la GitHub.

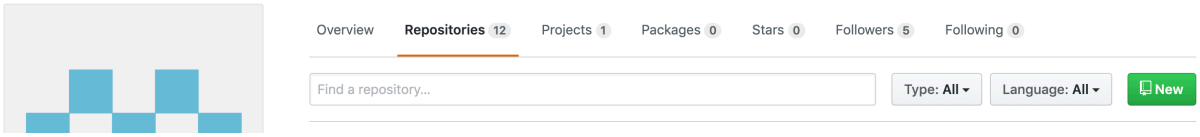
Pas cu pas folosire Git si GitHub:

Pas 1.verificăm dacă s-a instalat git, în consolă rula comanda:

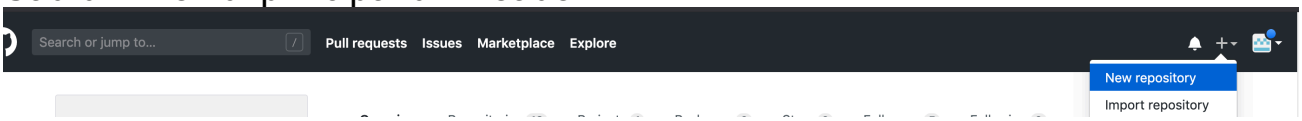
```
git --version
```

Pas 2. Creare cont pe https://github.com sau pe https://bitbucket.org/

Pas 3. Crearea unui repository pe GitHub
Noi ne vom axa pe crearea unui repository. Pentru aceasta dați click pe + **New Repository** sau **New**



Sau din meniul principal din header:



Următorul pas este alegerea unui nume pentru repository.
În câmpul „*Description*” puteți adăuga o scurtă descriere a proiectului. Nu uitați să bifați „Initialize this repository with a README”.

Opțional puteți alege, în partea de jos a paginii, adăugarea unui fișier .gitignore și/sau a unei licențe.
Fișierul .gitignore este folosit de git pentru a ignora fișierele pe care nu le doriți în repository, de exemplu: fișiere generate la compilare, fișiere private, etc. Mai multe detalii puteți găsi aici: help.github.com.

Licența folosită determină condițiile în care o altă persoană poate folosi proiectul vostru. Un ghid alegerea unei licențe poate fi găsit aici: choosealicense.com.

După apăsarea butonului „**Create Repository**” veți ajunge pe pagina repository-ului nou creat.
Alegeti una din cele 2 opțiuni si von rula aceste comenzi local in pasul urmator (Pas 4.):

…or create a new repository on the command line

```
echo "# CursPHP" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/aadiaconitei/proiectlink2022.git
git push -u origin master
```

Daca aveti deja local un “repo”

…or push an existing repository from the command line

```
git remote add origin https://github.com/aadiaconitei/proiectlink2022.git
git push -u origin master
```

Pas 4. Creare director/folder repo local in: **xampp/htdocs/cursphp**

Metoda I:
Inițializare un nou repo, local, cu git init

```
git init
```

Metoda II:
Clonam (crea o copie locală) un repo existent pe server (GitHub). Descărcare fișiere din repo-ul master (se creează o copie locala)

```
git clone https://github.com/aadiaconitei/proiectlink2022.git
```

Pas 5 (Opțional). Configurare client (local) GIT

```
git config --global user.name "Numele tau”
git config --global user.email adresatadeemail@email.com
```

Pas 6. Definește repo-ul master

```
git remote add origin https://github.com/aadiaconitei/proiectlink2022.git
```

Pas 7. Modifică fișiere locale si încarcă-le pe server

```
git status
git add .
git commit –m “Modificari fisiere”
git push origin master
```

Pașii de la 1 la 6 se fac o singură dată la începutul fiecărui proiect. Pasul 7 se repetă de fiecare dată când facem modificări la proiect.

Comenzii utile:

Acum puteți verifica dacă s-au salvat configurările cu:

```
git config --list
```

Afiseaza statusul repository-ului unde iti arata ce fisiere au fost editate, ce fisiere au fost sterse si ce fisiere au fost create.

```
git status
```

Pentru a vedea commit-urile nu scriem git status ci git log:

```
git log
```

Domeniul principal se numește master. Pentru a crea un nou domeniu se folosește:

```
git branch "Nume Domeniu"
git checkout -b "Nume Domeniu"
```

(exp: git branch experimental)

Pentru a afișa domeniile se folosește:

```
git branch
```

Pentru a schimba domeniul se utilizează:

```
git checkout experimental
```

După ce se revine la master, cele două domenii se pot uni folosind:

```
git merge experimental
```

(în caz că există diferențe între domenii apare un mesaj corespunzător)

Pentru a șterge un domeniu se folosește:

```
git branch -d experimental
```

(se efectuează în cazul în care diferențele sunt în domeniul curent)

```
git branch -D experimental
```

(șterge domeniul fără a actualiza diferențele)

Resurse:

https://www.youtube.com/watch?v=SWYqp7iY_Tc
https://www.youtube.com/watch?v=J_Clau1bYco
https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf
https://education.github.com/git-cheat-sheet-education.pdf
https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control
https://code.visualstudio.com/docs/editor/versioncontrol#_vs-code-as-git-editor
https://scotch.io/tutorials/git-integration-in-visual-studio-code
http://www.codebind.com/linux-tutorials/basic-git-commands-list/
https://speakerdeck.com/rstankov/git-workflows