Git & Github

Git este un sistem de control al versiunilor distribuit (VCS) care permite dezvoltatorilor să urmărească modificările, să colaboreze la proiecte și să gestioneze codul sursă într-un mod eficient. A fost creat de Linus Torvalds, aceeași persoană care a dezvoltat sistemul de operare Linux.

Iată câteva concepte și caracteristici cheie ale Git:

**1. Controlul Versiunilor:**

- Git urmărește modificările fișierelor și directoarelor în timp, permițând dezvoltatorilor să revină la versiuni anterioare, să compare modificările și să colaboreze eficient.

- Oferă o istorie completă a tuturor modificărilor făcute în codul sursă.

**2. Sistem Distribuit:**

- Git este un VCS distribuit, ceea ce înseamnă că fiecare dezvoltator are o copie completă a întregului repository, inclusiv istoricul acestuia.

- Acest lucru permite dezvoltatorilor să lucreze offline, să facă commit-uri locale și să se sincronizeze cu repository-urile remote atunci când au o conexiune la internet.

**3. Branching și Merging:**

- Git facilitează crearea de branch-uri, care sunt linii independente de dezvoltare.

- Dezvoltatorii pot crea branch-uri pentru a lucra la noi funcționalități, rezolvarea de bug-uri sau experimente fără a afecta codul principal.

- Branch-urile pot fi integrate înapoi în branch-ul principal (de obicei numit `master` sau `**main**`) atunci când modificările sunt gata.

**4. Colaborare și Repository-uri Remote:**

- Git facilitează colaborarea, permițând mai multor dezvoltatori să lucreze simultan la același proiect.

- Dezvoltatorii pot clona un repository remote pe mașina lor locală, să facă modificări și să trimită acele modificări înapoi în repository-ul remote.

- Git oferă mecanisme pentru gestionarea conflictelor atunci când mai mulți dezvoltatori fac modificări la același fișier.

**5. Staging și Commit-uri:**

- Git utilizează o zonă de staging (cunoscută și sub numele de index) pentru a selecta și pregăti modificările pentru commit.

- Dezvoltatorii pot alege selectiv care modificări să fie incluse într-un commit, facilitând crearea de commit-uri logice și atomice.

**6. Integrare cu Platforme de Găzduire:**

- Git se integrează perfect cu platforme populare de găzduire, cum ar fi GitHub, GitLab și Bitbucket.

- Aceste platforme oferă funcționalități suplimentare, cum ar fi urmărirea problemelor, pull request-uri, revizuirea codului și integrarea continuă.

Git este larg utilizat în industria dezvoltării software datorită flexibilității, vitezei și funcționalităților puternice pe care le oferă. Permite colaborare eficientă, control al versiunilor și gestionarea proiectelor, fiind o unealtă esențială pentru dezvoltatorii care lucrează la proiecte de orice dimensiune.

**Terminologie:**

* **master** - branch-ul principal al repository-ului. În funcție de fluxul de lucru, este branch-ul pe care oamenii lucrează sau branch-ul în care are loc integrarea.
* **clone** - copiază un repository Git existent, de obicei de la o locație remote în mediul tău local.
* **commit** - adaugă fișiere în repository (cel local); în alte VCS-uri este adesea denumit "checkin".
* **fetch** sau **pull** - este similar cu "update" sau "get latest" în alte VCS-uri. Diferența dintre fetch și pull este că pull combină ambele acțiuni, adică aduce codul cel mai recent dintr-un repository remote și realizează și îmbinarea.
* **push** - este utilizat pentru a trimite codul într-un repository remote.
* **remote** - acestea sunt locațiile "remote" ale repository-ului tău, de obicei pe un server central.
* **SHA** - fiecare commit sau nod în arborele Git este identificat printr-o cheie SHA unică. Le poți utiliza în diverse comenzi pentru a manipula un nod specific.
* **head** - este o referință către nodul către care punctează spațiul nostru de lucru al repository-ului în prezent.
* **branch** - este similar cu alte VCS-uri, cu diferența că un branch în Git nu este altceva decât o etichetă particulară pe un anumit nod. Nu este o copie fizică a fișierelor, așa cum se întâmplă în alte VCS-uri populare.

**Cele mai importante comenzi Git:**

1. `git init`: Inițializează un nou repository Git în directorul curent.

2. `git clone <repository>`: Clonează un repository Git existent de la o locație remote în mediul tău local.

3. `git add <file>`: Adaugă un fișier specific în zona de staging pentru a fi pregătit pentru commit.

`git add .` : adauga toate fisierele..

4. `git commit -m "<message>"`: Creează un commit cu fișierele din zona de staging și un mesaj de descriere.

5. `git status`: Afișează starea curentă a repository-ului, inclusiv fișierele modificate, adăugate sau șterse.

6. `git log`: Afișează istoricul commit-urilor, inclusiv mesajele, autorii și timestamp-urile.

7. `git pull`: Aduce ultimele modificări de la un repository remote și le îmbină cu branch-ul curent.

8. `git push`: Trimite commit-urile locale către un repository remote.

9. `git branch`: Afișează lista branch-urilor existente și indică branch-ul curent.

10. `git checkout <branch>`: Comută la un alt branch existent sau creează unul nou.

11. `git merge <branch>`: Îmbină modificările dintr-un branch specific în branch-ul curent.

12. `git remote add <name> <url>`: Adaugă o locație remote cu un nume specific pentru a face referire la repository-ul remote.

**Resurse:**

<https://www.youtube.com/watch?v=2ReR1YJrNOM&ab_channel=ProgrammingwithMosh>

<https://juristr.com/blog/2013/04/git-explained>

<https://www.youtube.com/watch?v=xvwBtODV0ms>