Versionare - GIT

ALIN ZAMFIROIU

Controlul versiunilor - necesitate

- Caracterul colaborativ al proiectelor;
- Backup pentru codul scris
- Istoricul modificarilor

Terminologie și concepte

- **VCS** – Version Control Software;
- **SCM** – Source Control Management;
- **repository** – componenta server ce conține informații privind ierarhia de fișiere și reviziile asupra acestora;
- **checkout** – preluarea în mediul local a unei anumite revizii publicate pe server (în repository);
- **working copy** – versiunea locala a proiectului; versiunea în care lucrează programatorul;
- **commit** – cerere de publicare în repository-ul local a unor modificări realizate în working copy;
- **pull** – acțiunea de actualizare (update) a informațiilor locale cu cele de pe server;

Terminologie și concepte

- **conflict** – apare atunci când mai mulți utilizatori au realizat modificări în aceleași fișiere din proiect; sistemul de aplicare a versiunilor diferite nu poate îmbina modificările și astfel este nevoie de intervenția umană pentru a realiza merge;
- **merge** – procesul de unire a două sau mai multe versiuni de lucru;
- **branch** – ramuri secundare de dezvoltare a proiectului, pe lângă master;
- **revert** – revenirea la o versiune anterioară pe un anume fir de dezvoltare (branch);
- **Stash** – arhivă locală pentru un set de modificări.

Istoricul GIT

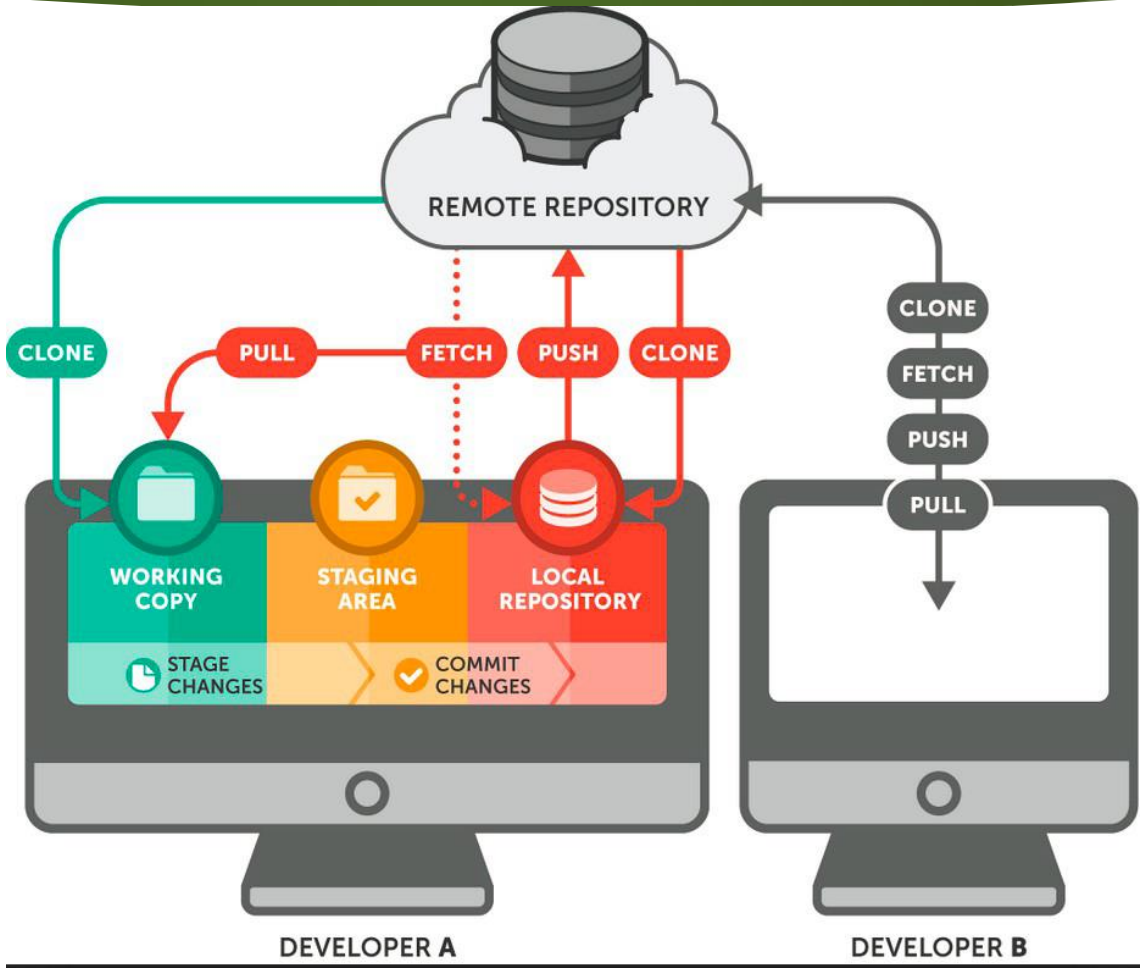
- Dezvoltat de Linus Torvalds pentru a gestiona proiectul de dezvoltare a kernel-ului de Linux în anul 2005, după un conflict cu BitKeeper, vechiul sistem de versionare folosit pentru kernelul de Linux

- Open source code

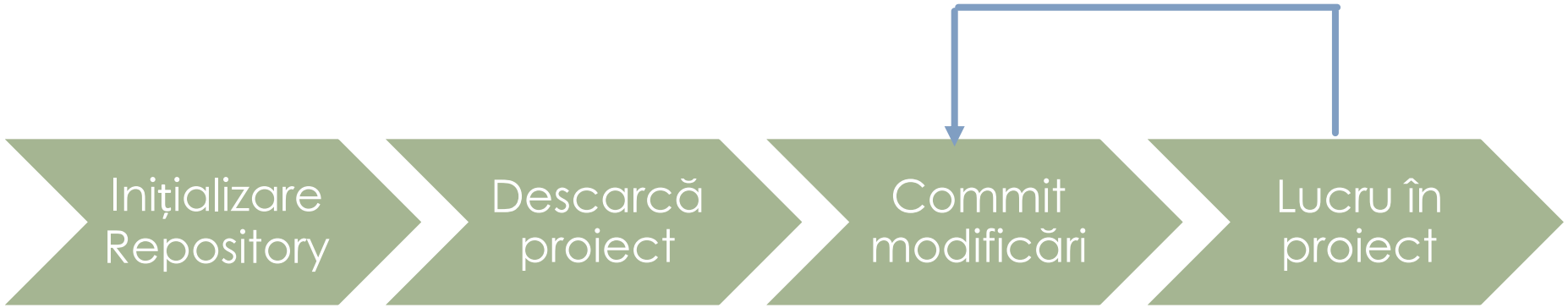
GIT

- Este un sistem **distribuit** de versionare;
- Fiecare programator lucrează pe mașina sa și are o copie a repository-ului pe mașina proprie.
- Toți programatorii au acces la istoricul modificărilor.

GIT - diagramă



Flux GIT



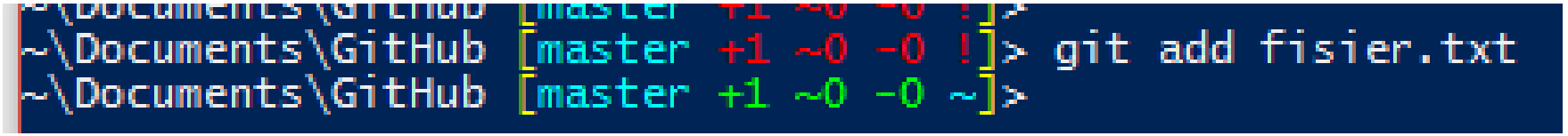
Tutorial GIT

- 1. Pentru inițializarea unui Repository se folosește comanda: **git init**.
- 2. Pentru verificarea statusului proiectului se folosește comanda: **git status**.
  - Daca nu aveți nimic în repository, răspunsul comenzii va fi ca nu aveți nimic pentru commit.
  - Dacă aveți fișiere modificate sau adăugate acestea apar listate pentru a fi adăugate.

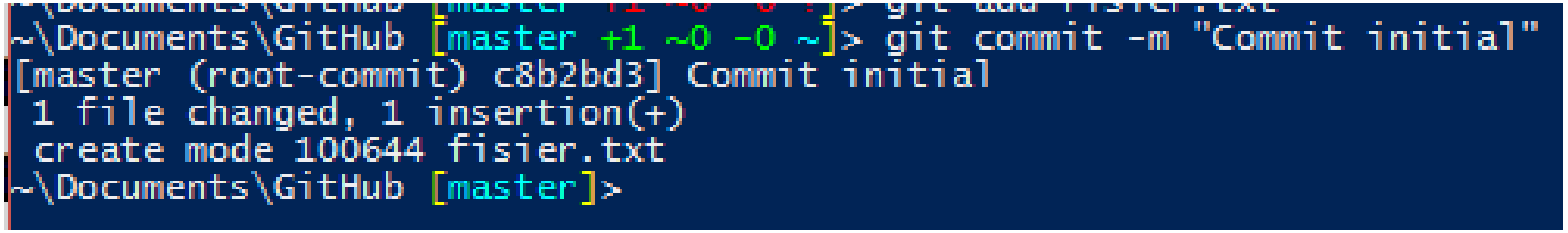
```
~\Documents\GitHub [master]> git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        fisier.txt

nothing added to commit but untracked files present (use "git add" to track)
~\Documents\GitHub [master +1 ~0 -0 !]>
```

Tutorial GIT

- 3. Adăugarea noului fișier în track se face cu comanda **git add**:



- 4. Commiterea acestuia se face prin comanda **git commit**, și cu un mesaj **–m**:



- 5. Pentru trimite se folosește comanda **git push**.

Tutorial GIT

- 6. Clonarea. Ne mutăm pe un nou folder și clonăm un repository existent:
  - https://github.com/zamfiroiu/CursuriCTS.git
- 7. Comenzile date:
  - **Git init**;
  - **Git remote add origin** ……;
  - **Git clone** …….

```
and the repository exists.
~\Documents\GitHub [master]> cd CTS
~\Documents\GitHub\CTS [master]> git init
Initialized empty Git repository in C:/Users/alinZ/Documents/GitHub/CTS/.git/
~\Documents\GitHub\CTS [master]> git remote add origin https://github.com/zamfiroiu/CursuriCTS.git
~\Documents\GitHub\CTS [master]> git clone https://github.com/zamfiroiu/CursuriCTS.git
Cloning into 'CursuriCTS'...
remote: Counting objects: 293, done.
remote: Total 293 (delta 0), reused 0 (delta 0), pack-reused 293
Receiving objects:  79% (232/293), 28.01 KiB | 26.00 KiB/s
Receiving objects: 100% (293/293), 39.31 KiB | 26.00 KiB/s, done.
Resolving deltas: 100% (47/47), done.
~\Documents\GitHub\CTS [master +1 ~0 -0 !]>
```

Tutorial GIT

- 8. Actualizarea versiunii curente se face cu comanda **git pull** <<branch>>;
  - **git pull origin**;
- 9. După ce se realizează modificările necesare se realizează commit și push.

Tutorial GIT

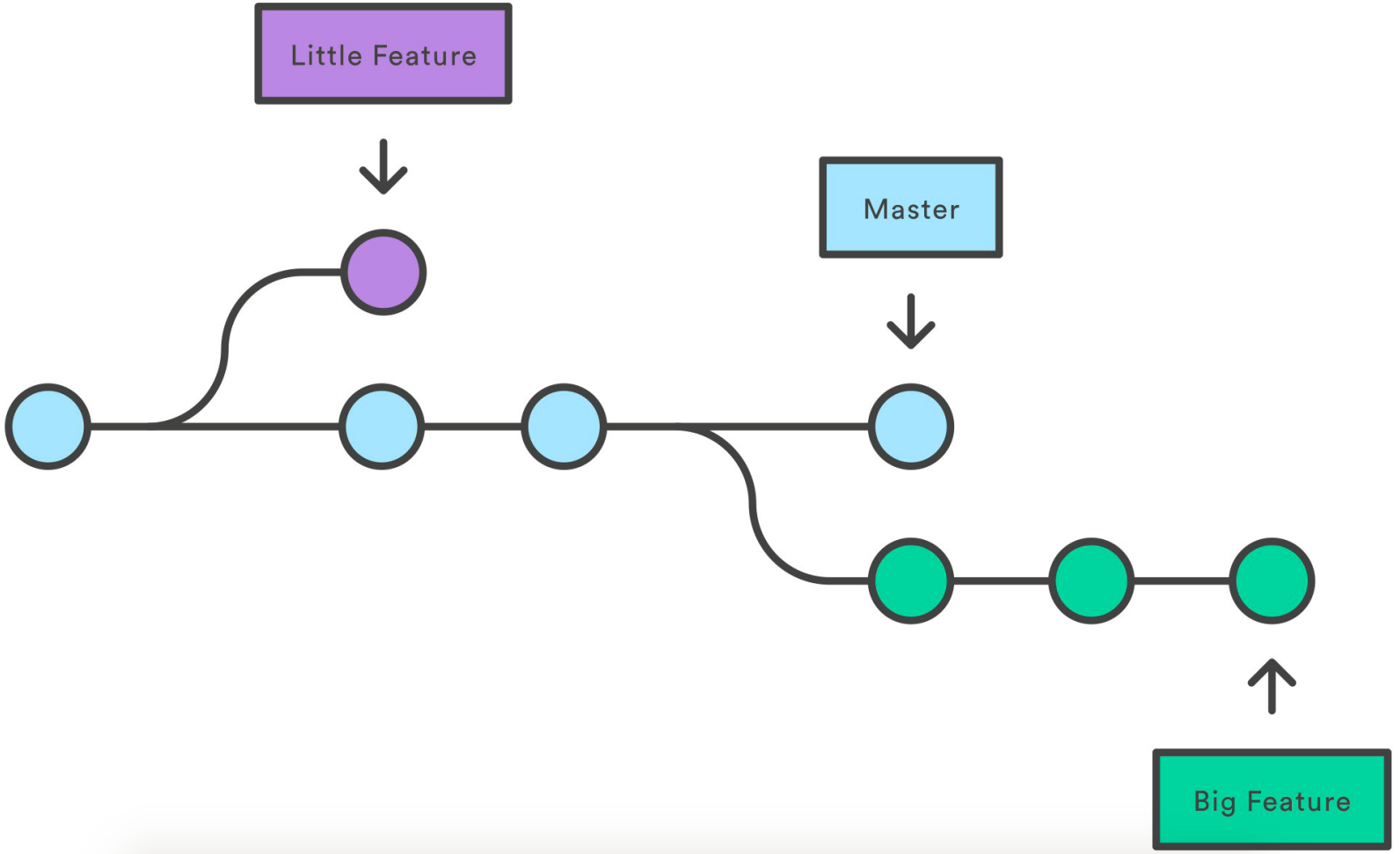
- 10. Lucrul cu branch-uri
  - **git branch** – afișează branch-urile locale.
  - **git branch –a** – afișează branch-urile locale și pe cele de pe repository.
  - **git checkout <<*branch_local*>>** – se schimbă branch-ul pe care se lucrează.
  - **git branch –b <<*new_branch*>>** - se creaza un nou branch pe baza celui curent.
  - **git branch –b <<*new_branch*>> <<*branch_sursa*>>** - creaza un nou branch sincronizat cu branch-ul sursa din repository.
  - **git branch –D <<*branch*>>** - sterge branch-ul local si de pe repository. Dacă se folosește **D**, ștergerea se face chiar dacă există modificări ne-merge-uite, dacă se folosește **d**, și există modificări, ștergerea nu se face.

Tutorial GIT

- 11. Merge-uirea branch-urilor
  - **git merge <<*branch_cu_modificari*>>**

- Aplică modificările existente în branch-ul cu modificări pe branch-ul curent.

Versionarea pe branch-uri



Referințe

- https://ro.wikipedia.org/wiki/Controlul_versiunilor
- https://try.github.io/levels/1/challenges/1
- https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control
- http://www.vogella.com/tutorials/EclipseGit/article.html

GIT