

Ministerul Educației al Republicii Moldova
Universitatea Tehnică a Moldovei
Catedra Automatică și Tehnologii Informaționale

RAPORT

Lucrare de laborator Nr.1
la PR

Tema: (LL1) Versionarea codului sursă utilizind GIT

A elaborat :

st.gr. TI-142 : Chicu Roman

A verificat :

lector superior
lector superior

Chișinău 2017

Tema: (LL1) Versionarea codului sursă utilizind GIT

Scop și Obiectiv: Lucrarea de laborator are ca scop studiul și înțelegerea principiilor de funcționare și utilizare a sistemului distribuit de control al versiunilor numit GIT.

Obiectiv: Crearea unui repozitoriu distant, localizat de serviciul gitlab.ati.utm.md, și sincronizarea tuturor modificărilor efectuate asupra repozitoriului local.

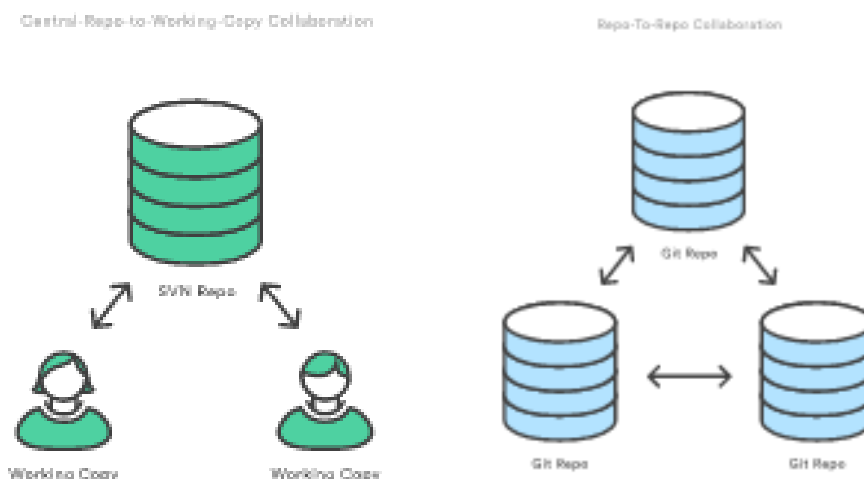
Note teoretice

Sistemele de versionare (VCS, Version Control Systems - eng.) servesc la gestionarea versiunilor multiple ale fișierelor incluse într-un proiect colaborativ. Fiecare modificare efectuată asupra elementului de proiect se memorizează împreună cu autorului schimbării. Important de menționat că în orice moment de timp se poate reveni la o versiune anterioară a entității.

Motivatia principala consta in posibilitatea ca diferiti membri ai echipei, aflatii eventual in spatii geografice indepartate, sa poata lucra simultan la proiect, urmand ca, la final, modificarile lor sa fie reunite in noi versiuni ale proiectului. De asemenea, exista si alte avantaje. Cand se observa un bug, se poate reveni la o versiune anterioara, in vederea determinarii momentului introducerii acestuia in program. In acelasi timp, se poate urma o dezvoltare pe ramuri (branches), in care se lucreaza, in paralel, la multiple versiuni ale proiectului - de exemplu, una in care se doreste inlaturarea bug-urilor, iar cealalta, in care se urmareste adaugarea de noi functionalitati, inaintea slefuirii celor existente.

Exista doua modele de VCS-uri:

- centralizat (ex: SVN): codul sursa este situat pe un server central, de unde clientii pot obtine variante de lucru pe masina locala (working copy). Dupa efectuarea locala a modificarilor, dezvoltatorul solicita actualizarea variantei de pe server.
- distribuit (ex: Git): nu exista un server central, procesul de sincronizare desfasurandu-se la nivel peer-to-peer.



Terminologie

- *repository* - pe server, conține ierarhia de fișiere și informațiile de versiune;
- *working copy* - varianta locală, obținută de la server, pe care se fac modificările;
- *revision* - o versiune a unui document. (v1, v2, v3...).
- *checkout* - aducerea pe masina locala a versiunii de pe server, sub forma unei working copy

- *update/pull*: actualizarea repozitoriului local în funcție de modificările survenite, între timp, pe server. Se aduc doar fișierele modificate;
- *commit* - înregistrează o nouă versiune a fișierului (fișierelor) modificat în repozitoriu.
- *commit message* - un mesaj asociat unei acțiuni *commit* care descrie schimbările făcute în noua versiune.
- *changelog* - o listă a versiunilor (commit-urilor) unui fișier/proiect de obicei însoțită de mesajele asociate fiecărui *commit*.
- *diff*: Afișează diferențele dintre două versiuni a unui fișier sau dintre fișierul modificat local (pe working copy) și o versiune de pe repository.
- *revert* - renunțarea la ultimele modificări (locale) făcute într-un fișier din *working copy*, și revenirea la ultima versiune aflată în repozitoriu sau la o versiune la alegere.
- *branch* - creează o “copie” a unui fișier/proiect pentru modificări „în paralel” fără a afecta starea actuală a unui proiect.
- *merge* - aplică ultimele modificări dintr-o versiune a unui fișier peste alt fișier;
- *conflict* - situația în care un *merge* nu se poate executa automat și modificările locale sunt în conflict cu modificările din repozitoriu.
- *resolve*: rezolvarea (de obicei manuală) a conflictelor apărute într-un fișier după un *merge*.

Mersul Lucrării

1. Instalăm sistemul de control al versiunilor

- GitBash - Downloads, <https://git-scm.com/downloads>

2. Creăm cont <https://github.com/>

- Vă adresați profesorului pentru acest punct
- Adăugați cheia publică generată de ssh-keygen (în git-bash terminal) în profilul utilizatorului. (Atenție! Cheia publică se regăsește în conținutul fișierului generat cu extensia *.pub*; cheile fiind localizate în directorul ascuns *.ssh* al utilizatorului de sistem: *./../Users/NumeUtilizator/.ssh/id_rsa*)

The image shows a Windows terminal window and a screenshot of the GitHub website. In the terminal, the command `ssh-keygen` is executed, generating a public/private RSA key pair. The prompt asks for the file name to save the key, and the user enters `/c/Users/Roman-PC/.ssh/id_rsa`. The terminal output shows the key pair generated successfully.

The screenshot of the GitHub website shows the 'SSH keys' section of a user's profile. It lists several SSH keys associated with the account, including 'GitHub Desktop - Roman-Pc' and 'Home-Pc'. Each key entry displays the key type (SSH), the fingerprint, and the date it was added. A 'Delete' button is provided for each key.

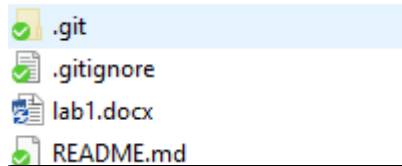
3. Creăm un director local de lucru în care se vor afla fișierele de versionat
 - Puteți utiliza comanda **mkdir** pentru crearea directorului în linia de comandă a terminalului, iar navigarea o puteți realiza prin: **cd NumeDirector**.
4. Inițializați repozițoriul GIT în acest director

- **git init**

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical University of Moldova/s  
mestru 2/PR/lab1 (master)  
$ git init  
Reinitialized existing Git repository in C:/Users/Roman-PC/OneDrive - Technical  
University of Moldova/simestru 2/PR/lab1/.git/
```

5. Modificați conținutul directorului

- adăugăm 1-2 fișiere...



```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical University  
mestru 2/PR/lab1 (master)  
$ touch .gitignore
```

6. Modificăm repozițoriul local, ori de câte ori se modifică directorul de lucru:

- STAGE - **git add .**
- HEAD - **git commit -m "First commit"**

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive  
mestru 2/PR/lab1 (master)  
$ git add .  
  
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical Un  
mestru 2/PR/lab1 (master)  
$ git commit -m "Init file"  
[master 2d198d6] Init file  
2 files changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 .gitignore  
create mode 100644 lab1.docx
```

7. Adăugăm repozițoriul distant

- **git remote add origin [git@github.com:logan11116/lab1.git](https://github.com/logan11116/lab1.git)**

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical Universi  
mestru 2/PR/lab1 (master)  
$ git remote add origin git@github.com:logan11116/lab1.git
```

8. Reînnomim repozițoriul distant **origin**, ramura **master**

- **git push -u origin master**

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical Universi  
mestru 2/PR/lab1 (master)  
$ git push -u origin master  
Enter passphrase for key '/c/Users/Roman-PC/.ssh/id_rsa':  
Counting objects: 4, done.  
Delta compression using up to 4 threads.  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (4/4), 200.30 KiB | 0 bytes/s, done.  
Total 4 (delta 0), reused 0 (delta 0)  
To github.com:logan11116/lab1.git  
36c2122..2d198d6 master -> master  
Branch master set up to track remote branch master from origin.
```

9. Utilizăm frecvent comanda **git status** pentru a vedea starea directorului de lucru și repozițoriului local

```
Roman-PC@DESKTOP-A10I0VQ MINGW32 ~/OneDrive - Technical University of Moldova,
mestru 2/PR/lab1 (Roman)
$ git status
On branch Roman
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        lab.cpp

nothing added to commit but untracked files present (use "git add" to track)
```

10. Creăm o ramură nouă de dezvoltare **Roman**, *numebranch*

- `git branch Roman`

```
Roman-PC@DESKTOP-A10I0VQ MINGW32 ~
mestru 2/PR/lab1 (master)
$ git branch Roman
```

11. Trecem în ramura nou creată **Roman**

- `git checkout Roman`
- ultimele două operații de creare și migrare la branch pot fi combinate prin: `git`

```
checkout -b Roman
Roman-PC@DESKTOP-A10I0VQ MINGW32 ~
mestru 2/PR/lab1 (master)
$ git checkout Roman
Switched to branch 'Roman'

Roman-PC@DESKTOP-A10I0VQ MINGW32 ~
mestru 2/PR/lab1 (Roman)
$ |
```

12. Modificăm directorul de lucru și repozitoriul local

- Adăugați un nou fișier pe lângă cele existente
- Reînnomim repozitoriul local (similar p. 6)

```
Roman-PC@DESKTOP-A10I0VQ MINGW32 ~/OneDrive - Tech
mestru 2/PR/lab1 (Roman)
$ git add .

Roman-PC@DESKTOP-A10I0VQ MINGW32 ~/OneDrive - Tech
mestru 2/PR/lab1 (Roman)
$ git commit -m "Adaugam un file cpp"
[Roman 14c8a24] Adaugam un file cpp
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 lab.cpp
```

13. Verificăm dacă sunt modificări în repozitoriul distant

- `git pull`
- studiem care alte comenzi `git` cuprinde această comandă!

```
Roman-PC@DESKTOP-A10I0VQ MINGW32 ~/OneDrive - Technical University of Moldova,
mestru 2/PR/lab1 (Roman)
$ git pull
Enter passphrase for key '/c/Users/Roman-PC/.ssh/id_rsa':
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.

        git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

        git branch --set-upstream-to=origin/<branch> Roman
```

14. Reînnomim repozitoriul distant, aflându-vă pe ramura **Roman**

- `git push -u origin Roman`






```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical University of
mestru 2/PR/lab1 (Roman)
$ git push -u origin Roman
Enter passphrase for key '/c/Users/Roman-PC/.ssh/id_rsa':
Counting objects: 2, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 229 bytes | 0 bytes/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To github.com:logan11116/lab1.git
 * [new branch]      Roman -> Roman
Branch Roman set up to track remote branch Roman from origin.
```

15. Verificam dacă sunt modificări în repozitoriul distant *origin* (similar p. 13)

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical Univ
mestru 2/PR/lab1 (Roman)
$ git pull origin
Enter passphrase for key '/c/Users/Roman-PC/.ssh/id_rsa':
Already up-to-date.
```





16. Trecem pe ramura principală de dezvoltare *master*

- `git checkout master`
- Observați conținutul directorului de lucru.
- Opțional puteți modifica directorul de lucru și repozitoriul pe ramura *master* (pp. 5-6)

| | | |
|---|------------|------|
|  | .git | 2/21 |
|  | .gitignore | 2/21 |
|  | lab.cpp | 2/21 |
|  | lab1.docx | 2/21 |
|  | README.md | 2/14 |

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Te
mestru 2/PR/lab1 (Roman)
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.






Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Te
mestru 2/PR/lab1 (master)
$ |
```

| | |
|---|------------|
|  | .git |
|  | .gitignore |
|  | lab1.docx |
|  | README.md |

17. Unificam ramurile *master* și *Roman*

- `git merge Roman`

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Tech
mestru 2/PR/lab1 (master)
$ git merge Roman
Updating 2d198d6..14c8a24
Fast-forward
 lab.cpp | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 lab.cpp
```




| | |
|--|---------------|
|  .git | 2/20/2017 2: |
|  .gitignore | 2/20/2017 1: |
|  lab.cpp | 2/20/2017 2: |
|  lab1.docx | 2/20/2017 1: |
|  README.md | 2/14/2017 10: |

18. Reînnom repozitoriul distant **origin**, ramura **master** (similar p. 8)

👤 Roman (8 minutes ago) 🔗 Compare & pull request

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

📁 logan11116 Init file Latest commit 2d198d6 17 minutes ago





| | | |
|--|-----------|----------------|
|  .gitignore | Init file | 17 minutes ago |
|  README.md | init | 6 days ago |
|  lab1.docx | Init file | 17 minutes ago |

📄 README.md

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical Universit
mestru 2/PR/lab1 (master)
$ git push -u origin master
Enter passphrase for key '/c/Users/Roman-PC/.ssh/id_rsa':
Total 0 (delta 0), reused 0 (delta 0)
To github.com:logan11116/lab1.git
 2d198d6..14c8a24 master -> master
Branch master set up to track remote branch master from origin.
```

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾







📁 logan11116 Adaugam un file cpp Latest commit 14c8a24 12 minutes ago

| | | |
|--|---------------------|----------------|
|  .gitignore | Init file | 18 minutes ago |
|  README.md | init | 6 days ago |
|  lab.cpp | Adaugam un file cpp | 12 minutes ago |
|  lab1.docx | Init file | 18 minutes ago |

19. Observa modificările realizate în proiectul localizat în <https://github.com/logan11116/lab1>

20. Compilăm un fișier cpp în git bash

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical Univer
mestru 2/PR/lab1 (master)
$ gcc lab.cpp -o lab
lab.cpp: In function 'int main()':
lab.cpp:8:16: error: 'getch' was not declared in this scope
return 0; getch();
               ^
```

| |
|--|
|  .git |
|  .gitignore |
|  lab.cpp |
|  lab.exe |
|  lab1.docx |
|  README.md |

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/On
mestru 2/PR/lab1 (master)
$ ./lab
4
Enter an integer: You entered: 4
```

```
C:\Users\Roman-PC\OneDrive - Technical Univer
Enter an integer: 4
```

Concluzie: Ni-am făcut familiari cu mediul github și în special am lucrat cu consola git bash în care am creat un depozitoriu local și l-am importat într-un extern. Am studiat diverse comenzi git și importanța lor într-un mediu unix. Am compilat un cod sursă în terminal

Bibliografie

1. Scott Chacon, Pro Git, July 29, 2009 <http://git-scm.com/book>
2. Lars Vogel, Git - Tutorial, actualizat
- 14.12.2014, <http://www.vogella.com/tutorials/Git/article.html>
3. Git How To, <http://githowto.com/>
4. Atlassian, Git Tutorials, <https://www.atlassian.com/git/tutorial>
5. Vincent Driessen, A successful Git branching model, January 05, 2010, <http://nvie.com/posts/a-successful-git-branching-model/>
6. [Linux.conf.au 2013] - Git For Ages 4 And Up,
Youtube, <https://www.youtube.com/watch?v=1ffBJ4sVUb4>
7. Visualizing Git Concepts with D3, <http://onlywei.github.io/explain-git-with-d3/>
8. tryGit, <https://try.github.io>
9. Learn Git Branching, <http://pcottle.github.io/learnGitBranching/>
10. Code School, Git Real, Free preview, <https://www.codeschool.com/courses/git-real>
11. Code School, Git Real 2, Free preview, <https://www.codeschool.com/courses/git-real-2>