Ministerul Educației al Republicii Moldova
Universitatea Tehnică a Moldovei

RAPORT

Lucrarea de laborator nr. 2
Medii Interactive de dezvoltare a produselor software
Tema: Version Control Systems şi modul de setare a unui server

A efectuat:
st. gr. TI-142 Dragutan Andrei

A verificat:
lect. univ. Irina Cojan

Chișinău 2016

Tema: Version Control Systems şi modul de setare a unui server.
Scopul lucrării: Studierea bazelor lucrului cu VCS. Obiectivele lucrării:

1. Înţelegerea şi folosirea CLI (basic level)
2. Administrarea remote a maşinilor linux machine folosind SSH (remote code editing)
3. Version Control Systems (git || mercurial || svn)
4. Compileaza codul C/C++/Java/Python prin intermediul CLI, folosind compilatoarele gcc/g++/javac/python

Modul de lucru: Paşii lucrării:

- conecteaza-te la server folosind SSH
- compileaza cel putin 2 sample programs din setul HelloWolrdPrograms folosind CLI
- executa primul commit folosind VCS ○ initializeaza un nou repositoriu ○ configureaza-ti VCS
- crearea branch-urilor (creeaza cel putin 2 branches) ○ commit pe ambele branch-uri (cel putin 1 commit per branch) ○ seteaza un branch to track a remote origin pe care vei putea sa faci push (ex. Github, Bitbucket or custom server) ○

reseteaza un branch la commit-ul anterior

- merge 2 branches ○ rezolvarea conflictelor a 2 branches

Efectuarea Lucrarii:

- ✓ A fost creat un repositoriu pe github.com.
- ✓ A fost stabilită conexiunea cu serverul prin generarea keygen-ului SSH prin instrucţiunea ssh-keygen, ea fiind adăugată în setări.
- ✓ A fost creat un fişier şi iniţializat git-ul prin instrucţiunea git init.
- ✓ A fost introdus .gitignore şi README.md prin instrucţiunea git add, git commit şi git push.
- ✓ Au fost create două programe, una în C şi alta în C++, cu extensia respectivă .c şi .cpp
- ✓ Cu ajutorul la Command Prompt, ele au fost compilate cu instrucţiunea:

gcc Hello_C.c –o Hello_C şi g++ Hello_C++.cpp –o Hello_C++ .

- ✓ Respectiv, au fost executate fişierele noi cu extensia .exe în Command Prompt, apelînd la ele prin: /Hello_C şi /Hello_C++.
- ✓ A fost creat un branch nou, unde au fost încărcate fişierele compilate în CLI, împreună cu imaginile respective şi fişierele originale .c şi .cpp.
- ✓ Am făcut merge dintre branch-ul master şi NBranch, unde se aflau fişierele compilate.
- ✓ Am creat o situaţie de conflict, creînd un fişier Conflict.txt şi încărcîndu-l pe branch master.
- ✓ Am creat un branch nou ConflictBranch, în care am modificat conţinutul fişierului Conflict.txt, astfel făcînd opţiunea merge imposibilă.
- ✓ Am rezolvat conflictul şi am făcut merge între branch-uri.

Screen-uri ale executarii comenzilor:

- Crearea unui nou Branch:

```
Andrei@Andriusa MINGW32 /e/MIDPSS/MIDPSS (master)
$ git branch Andrei
```

- Trasnferul la Branch-ul nou:

```
Andrei@Andriusa MINGW32 /e/MIDPSS/MIDPSS (master)
$ git checkout Andrei
A       1234.cpp
A       1234.exe
A       1234.o
Switched to branch 'Andrei'
```

- Crearea si executarea Hello_C.c şi Hello_C++.cpp:

```
C:\Users\Andrei\Desktop\MIDPSS>g++ 1234.cpp -o 1234

C:\Users\Andrei\Desktop\MIDPSS>g++ 1234.cpp -o 1234
```

```
Andrei@Andriusa MINGW32 /e/MIDPSS/MIDPSS (Andrei)
$ ./1234
hello, world
```

-

Adaugarea unui fisier .txt, care v-a fi resetat:

```
Andrei@Andriusa MINGW32 /e/MIDPSS/MIDPSS (Andrei)
$ git push --set-upstream origin Andrei
Branch Andrei set up to track remote branch Andrei from origin.
Everything up-to-date

Andrei@Andriusa MINGW32 /e/MIDPSS/MIDPSS (Andrei)
$ git checkout Andrei1
error: Your local changes to the following files would be overwritten by checkout:
        example.txt
Please, commit your changes or stash them before you can switch branches.
Aborting

Andrei@Andriusa MINGW32 /e/MIDPSS/MIDPSS (Andrei)
$ git branch Andrei2

Andrei@Andriusa MINGW32 /e/MIDPSS/MIDPSS (Andrei)
$ git checkout Andrei2
M       example.txt
Switched to branch 'Andrei2'
```

```
Andrei@Andriusa MINGW32 /e/MIDPSS/MIDPSS (Andrei2)
$ git push --set-upstream origin Andrei2
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 249 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To git@github.com:AndreiDragutan/MIDPSS.git
 * [new branch]      Andrei2 -> Andrei2
Branch Andrei2 set up to track remote branch Andrei2 from origin.

Andrei@Andriusa MINGW32 /e/MIDPSS/MIDPSS (Andrei2)
$ git checkout Andrei
Switched to branch 'Andrei'
Your branch is up-to-date with 'origin/Andrei'.
```

Resetarea la commit-ul precedent:

```
Andrei@Andriusa MINGW32 /e/MIDPSS/MIDPSS (Andrei)
$ git reset --hard
HEAD is now at 6f650d6 Exemplu
```

- Merge între două branch-uri:

```
Andrei@Andriusa MINGW32 /e/MIDPSS/MIDPSS (Andrei)
$ git merge Andrei2
Updating 6f650d6..8b4e0cd
Fast-forward
 example.txt | 2 ++
 1 file changed, 2 insertions(+)
```

- Afişarea situaţiei de conflict:

```
$ git merge MBranch
CONFLICT (rename/delete): Lab#2/Used_Files/Branch1.txt deleted in
 MBranch and renamed in HEAD. Version HEAD of Lab#2/Used_Files/Br
anch1.txt left in tree.
Automatic merge failed; fix conflicts and then commit the result.
```

Concluzie

In urma efectuarii acestei lucrari de laborator au fost obtinute deprinderi practice de lucru cu git – unul din cele mai populare DVCS. Am ajuns la concluzia ca un VCS este un instrument indispensabil lucrului in echipa. Acest VCS este un instrument ce verifică versiunea proiectului, permitîndu-ne să evităm conflicte Am observat ca in git, lucru cu branch-urile se face cu mult mai simplu decit in alte VSC-uri cum ar fi SVN. Un lucru placut despre git este viteza si eficienta lui. Repositoriile git-ului ocupa mai putin loc decit cele ale lui SVN, si ca git este distribuit, deci nu este novoie de a fi mereu conectat la retea.