Facultatea Calculatoare, Informatica si Microelectronica

Universitatea Tehnica a Moldovei

Medii Interactive de Dezvoltare a Produselor Soft Lucrarea de laborator#1

Version Control Systems si modul de setare a unui server

Autor: Dilan Nicolae

lector asistent: Irina Cojanu

Chișinău 2017

Lucrarea de laborator #1

**0.1 Scopul lucrarii de laborator**

Studierea principiilor de creare si de lucru cu Git .

**0.2 Obiective**

Version Control Systems (git || bitbucket || mercurial || svn).

**0.3 Efectuarea lucrarii de laborator**

**0.3.1 Tasks and Points**

Basic Level (nota 5 || 6) :

-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)

Normal Level (nota 7 || 8):

-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 -salvarea temporara a schimbarilor care nu se vor face commit imediat.

-folosirea fisierului .gitignore

Advanced Level (nota 9 || 10):

-merge 2 branches

-rezolvarea conflictelor a 2 branches

-comezile git care trebuie cunoscute

Bonus Point (+1):

-Tags. Folosirea tag-urilor pentru marcarea schimbarilor simnificative precum release-ul.

**0.3.2 Analiza lucrarii de laborator**

Linkul la repozitoriu: <https://github.com/dumitritag/MIDPS-lab>

1.Initializarea unui repositoriu si configurarea VCS

Initial a fost creat un cont pe GitHub care contine repozitoriul laboratorului efectuat. Am creat repozitoriul meu MIDPS−lab. A fost stabilita conexiunea cu serverul prin generarea keygen-ului SSH prin instructiunea ssh-keygen, ea fiind adaugata in setari. Dupa ce am clonat repozitoriul, folosind comanda git clone shhlik , am configurat contul cu comenzile git config −global user.name ”YourName” si git config −global user.email ”youremail@domain.com” Am creat fisierele .gitignore si README in repozitoriu si cu ajutorul comenzii git pull le-am adaugat in masina locala. In masina locala am creat 5 mape, fiecare 2 continind cite un fiser README si le-am adaugat in repozitoriu cu comenzile git add . , git commit si git push . Commiturile sunt schimbarile salvate in poiectu nostru. Fiecare commit are un mesaj asociat, in care descrim ce schimbare am facut. In mapa Lab1 din masina locala am creat un fisier myfile.txt cu continutul ”Welcome!” si l-am adaugat in repozitoriu.

2. Crearea branch-urilor

Branch-ul implicit in Git este master, comanda git init il creeaza implicit. Am creat 2 branch-uri noi: branch1 si branch2 cu ajutorul comenzii git branch ¡branchname¿. Aceasta creeaza un nou pointer la ultimul commit. Pentru a comuta la un branch nou am folosit comanda git checkout ¡branchname¿. Am modificat fisierul myfile.tx, l-am adaugat la branch-urile si am facut primele commituri in noile branch-uri.

3. Track remote origin

Am creat un nou branch newbranch to track remote origin, folosind comanda git branch –track ¡branchname¿ origin/master. Am creat un nou fisier new.txt, l-am adaugat, am facut commit si cu comanda git push origin ¡branchname¿ am adaugat commitul in newbranch iar cu comanda git push origin HEAD:master in master branch.

4. Resetarea unui branch la commitul anterior

Pentru rescrierea commitului anterior am folosit comanda git commit – amend. Pentru resetarea unui branch la commitul anterior am folosit comanda git revert HEAD.

5. Salvarea temporara a schimbarilor

Uneori facem schimbari care nu dorim sa le facem commit dar dorim sa le salvam, asta putem face cu comanda git stash.

6. Folosirea fisierului .gitignore

Pentru a vedea cum influenteaza fisierul .gitignore, am adaugat in fisierul .gitignore \*.txt si am creat un nou fisier de tip txt, am aplicat comanda git add si acest fisier a fost ignorat.

7. Merge 2 branches

Merge 2 branches inseamna a le integra intr-un singur branch. Pentru aceasta am folosit comanda git merge ¡branchname¿.

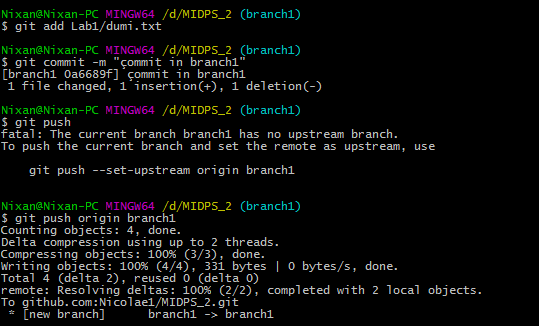
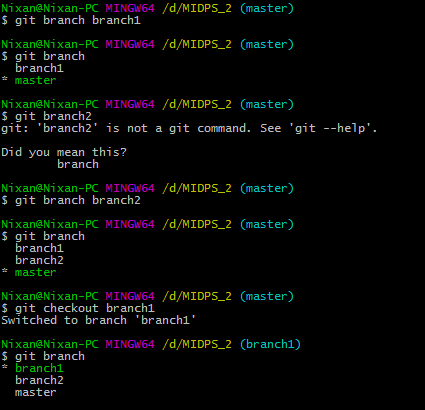
8. Rezolvarea conflictelor a 2 branch-uri

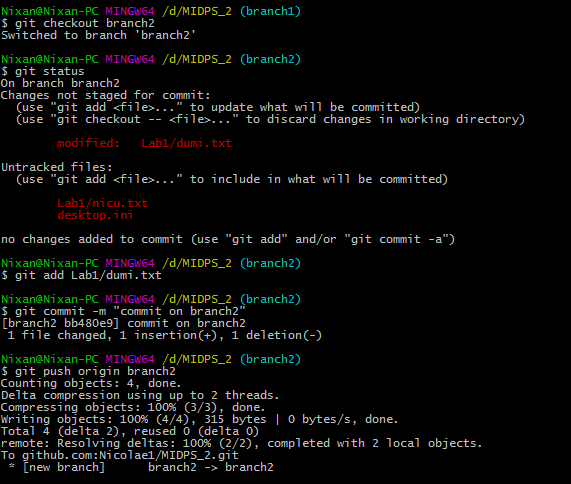
Daca avem 2 branches, in ambele a fost schimbat aceeasi parte a unui fisier si dorim sa facem merge la ambele va aparea un conflict. Acest conflict putem sa-l rezolvam manual, modificind fisierul care a cauzat conflictul. O alta metoda este prin folosirea comenzii git rebase master.

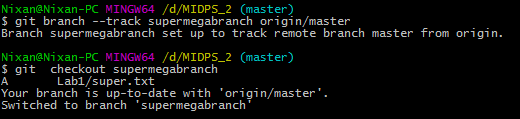
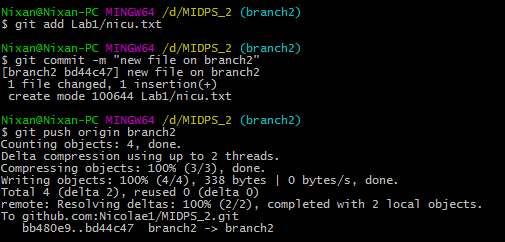
9. Folosirea tag-urilor

Un tag este folosit pentru a eticheta si a marca un commit. Sunt 2 tipuri de tag-uri lightweight, folosim comanda git tag ¡tagname¿ si annotated, comanda folosita este git tag -am ”message” ¡tagname¿.

**0.3.3 Imagini**







**Concluzie**

Un sistem de control al versiunilor reprezinta un produs software ce ajuta dezvoltatorii dintr-o echipa software sa colaboreze in cadrul diferitelor proiecte si de asemenea pastreaza un jurnal complet al muncii fiecaruia. Un sistem de control al versiunilor ( SCV ) are trei scopuri principale :

1. Sa ofere posibilitatea muncii simultane, nu seriale

2. Atunci cand mai multe persoane lucreaza in acelasi timp se asigura ca modificarile realizate de acestia nu intra in conflict unele cu altele

3. Sa ofere o arhiva a fiecarei versiuni continand informatii despre cine, unde si din ce motiv a fost facuta fiecare modificare.

Sunt 2 forme de organizarea a sistemelor de control al versiunilor: centralizata si distribuita. Principiul de baza al sistemelor centralizate se bazeaza pe relatia clientserver. Un depozit (repository) este situat intr-un singur loc iar mai multi clienti au acces la el. Toate modificarile utilizatorilor si toate informatiile legate de aceste modificari (utilizator, data, revizie) sunt transmise si preluate de la un depozit (repository) central. Sistemele de control al versiunilor distribuite sunt o optiune mai noua. In cadrul acestora fiecare utilizator are propria copie a intregului repository, nu doar fisiere, ci intregul jurnal. Aceasta abordare foloseste modelul peer-to-peer spre deosebire de modelul client-server folosit de sistemele centralizate. In acest caz sincronizarea dintre repository-uri este realizata prin schimbul de changeseturi sau patch-uri dintre statii. Doua dintre cele mai folosite SCV-uri de acest fel sunt Git si Mercurial.

Principalele beneficii ale sistemelor Git sunt :

- O urmarire a schimbarilor mai avansata si mai detaliata, lucru care duce la mai putine conflicte

- Lipsa necesitatii unui server-toate operatiile cu exceptia schimbului de informatii intre repository-uri se realizeaza local

- Operatiile de branching si merging sunt mai sigure, si prin urmare folosite mai des

- Rapiditate mai mare a operatiilor datorita lipsei necesitatii comunicarii cu serverul

Principalele dezavantaje ale sistemelor Git :

- Modelul distribuit este mai greu de inteles

- Nu exista asa de multi clienti GUI datorita faptului ca aceste sisteme sunt mai noi - Reviziile nu sunt numere incrementale, lucru ce le face mai greu de referentiat

- Riscul aparitiei de greseli este mare daca modelul nu este familiar

**Bibliography**

[1] <https://backlogtool.com/git-guide/en/stepup/stepup71.html>

[2] <https://www.siteground.com/tutorials/git/commands.htm>

[3] <https://git-scm.com/book/en/v2>