

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:
Anna Anghiloglu

lector asistent:
Victor Gojin

lector superior:
Radu Melnic

Chisinau 2017

Lucrare de laborator Nr.1

1. Scopul lucrării de laborator – Invatarea unui Version Control System si a modului de setare a unui server
2. Obiective – Studierea a Version Control Systems (git)
3. Implementarea lucrării de laborator
 - 3.1. Sarcini si Obiective – initializeaza un nou repository – 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 – salvarea temporara a schimbarilor care nu se vor face commit imediat. – folosirea fisierului .gitignore – merge 2 branches – rezolvarea conflictelor a 2 branches – comenzile git care trebuie cunoscute – Tags. Folosirea tag-urilor pentru marcarea schimbarilor semnificative precum release-ul.

3.2. Analiza lucrării de laborator

1. Basic Level

– initializeaza un nou repository La prima etapa cream un nou repository , urmind pasii din ghidul de utilizare de pe platforma github, accesind butonul ” + ” si optiunea create new repository. Exista mai multe modalitati de a initializa un repository pe github. Putem crea o mapa goala in care vom plasa gitul nostru prin intermediul comenzii git init , crearea unui nou repository este posibil datorita comenzii curu ’USER’ `https://api.github.com/user/repos -d”name”:”NUME”`

– configureaza-ti VCS Esential este configuram accountul git si repository. Esential este generarea cheii SSH(Secure Shell). Daca scriem in repository CLI sshkeygen, iar cheia obtinuta o copiem in setarile noastre de pe git. necesar sa unim gitul nostru gol cu repository creat folosind comanda git remote add origin ”Linkul catre repository”Pentru a clona repository in mapa locala de pe memoria calculatorului, utilizam linkul HTTP al acestuia, care se va copia de pe repository creat prin comanda ”Copy to clipboard”. Pentru aceasta ar trebui sa introducem in linia de comanda gitbash urmatoarele caractere: git config --global user.name "YourName" git config --global user.email "youremail@domain.com Putem adauga fisiere in repository cu ajutorul comenzii: git add si sa verificam starea acestuia prin: git status

– crearea branch-urilor (creeaza cel putin 2 branches) Pentru a crea o ramura in linia de comanda se executa urmatoarele comenzi: git checkout -b branch1 cream ramura git branch -a vedem toate ramurile existente Introducem schimbari in ramura respectiva;

– commit pe ambele branch-uri (cel putin 1 commit per branch) Pentru a face commit se utilizeaza comanda: git commit -am "new feature branch1" cream commit-ul git push origin branch1 actualizam si trimitem fisierul branch1 in repository de pe GitHub git checkout master accesam iarasi ramura initiala

2. Normal Level (nota7||8):

– seteaza un branch to track a remote origin pe care vei putea sa faci push (ex. Github, Bitbucket or custom server) Pentru a seta ca o ramura sa urmareasca originul prin care am putea face push pe Github. Putem face track remote origin

direct de pe ramura master git push sau putem crea alta ramura RemoteBranch si executa comanda git push --set-upstream origin Imaginile respective le puteti gasi in urmatorul subpunct, unde am utilizat extensia --all-gits pentru a observa schimbarile efectuate

– reseteaza un branch la commit-ul anterior Daca vrem sa ne intoarcem la un commit anterior, ramura noastra trebuie sa contina citeva commituri care pot fi introduse cu comenzile: git add . git commit -m "new feature branch1" cream commit-ul var.1 git commit -am "new feature branch1" cream commit-ul var.2 Astfel oricare commit anterior va putea fi resetat cu ajutorul a 3 tipuri de reseturi, in functie de scopul urmarit, insa intii si intii vom efectua comanda git log pentru a primi o lista de commituri cu codurile respective: git reset --soft care poate contine 6-9, pe care il vom copia din git log resetarea soft, fisierele commitului pot fi accesate git reset resetarea moderata, fisierele se sterg insa sunt inca in working stage git reset --hard resetarea puternica, sterge toate fisierele complet, pot fi desi niste fisiere untracked ramase daca ele au existat in commit, insa ele pot fi usor eliminate git clean -df stergerea fisierelor neurmarite/untracked

– salvarea temporara a schimbarilor care nu se vor face commit imediat. Schimbarile pot fi salvate temporar. Aceasta este eficient atunci cind lucram asupra a mai multor taskuri. Avind mai multe fisiere neterminate insa la care putem lucra in viitor. Nu este rezonabil sa facem commit la un asemenea fisier deoarece ceea la ce facem commit ar trebui sa fie executabil. Respectiv avem nevoie de o modalitate prin care sa introducem modificariile intro parte a memoriei si sa o accesam dupa ce am facut careva modificari in alta ramura si am hotarit sa continuam lucru asupra unei parti de program, respectiv utilizam un stash/un ascunzis

git stash save " Am lucrat asupra unui fis" salvam in stash deja cind efectuem git diff facem diferenta dintre server si mapa locala git status verificam starea directoriului ramurii, care de regula ar trebui sa contina un fisier modificat de tip untracked dar cere nu contine nimic deja deoarece acesta a fost mutat in stash/ascunzis Daca dupa anumit timp dorim sa accesam fisierul modificat, putem proceda astfel, apelind : git stash apply stash@ {0} accesam elementul din stash, fara al sterge din acesta git stash pop accesam elementul din stash, si il autodistrugem

– folosirea fisierului .gitignore Un fisier .gitignore poate fi creat direct de pe mapa locala MIDPS sau cu ajutorul comenzii touch .gitignore .gitignore-ul este util pentru a specifica tipurile de fisier care vor fi neglijate/ignoreate, pentru a demonstra utilitatea am indicat in interiorul acestui fisier un *.log, care va ignora toate fisierele cu extensia .log. Pentru a introduce aceste modificari se ruleaza comenzile git add . adaugam git commit -am "adding ignore file" cream commit-ul

Resultat:

The screenshot shows the GitHub repository page for 'anghiloglu / MIDPS'. The repository has 3 commits, 3 branches, 0 releases, and 1 contributor. The 'branch1' branch is selected, showing it is 2 commits ahead of master. The latest commit is 'second changes from branch1' by anghiloglu, with the latest commit hash 'a0ca3ac' on 5 Mar. The commit history shows two commits: '.gitignore' (Initial commit, 3 months ago) and 'README.md' (second changes from branch1, 3 months ago). The file list shows 'README.md'.

Fig 1. Branchul 1

The screenshot shows the GitHub repository page for 'anghiloglu / MIDPS'. The repository has 3 commits, 3 branches, 0 releases, and 1 contributor. The 'branch2' branch is selected, showing it is 2 commits ahead of master. The latest commit is 'second changes from branch2' by anghiloglu, with the latest commit hash 'aa4e8b8' on 5 Mar. The commit history shows two commits: '.gitignore' (Initial commit, 3 months ago) and 'README.md' (second changes from branch2, 3 months ago). The file list shows 'README.md'.

Fig 1. Branchul 2

```
posh~git ~ MIDPS [branch2]
~\Documents\GitHub\MIDPS [branch2 =>] git status
On branch branch2
Your branch is up-to-date with 'origin/branch2'.
nothing to commit, working tree clean
~\Documents\GitHub\MIDPS [branch2 =>] git branch
branch1
* branch2
master
```

Fig 3. Comenzi

```
~\Documents\GitHub\MIDPS [branch2 ≡]> git log
commit aa4e8b874bd8690e645c71b7bcd1696ce39c1a6c
Author: anna.anghiloglu <anghiloglu@gmail.com>
Date: Sun Mar 5 22:20:10 2017 +0200

    second changes from branch2

commit 45948f423eae5ba17a20dc36ce57b4a14bb09633
Author: anna.anghiloglu <anghiloglu@gmail.com>
Date: Sun Mar 5 22:19:24 2017 +0200

    first changes from branch2

commit 4de80d49b09fbc71de304b4c42362348709b7278
Author: anna.anghiloglu <anghiloglu@gmail.com>
Date: Sun Mar 5 21:49:22 2017 +0200

    Initial commit
~\Documents\GitHub\MIDPS [branch2 ≡]>
```

Fig 4. Comenzi

Concluzie: In urma realizarii acestei lucrari de laborator am reusit sa adaug cunostinte esentiale in ceea ce priveste platforma github. Crearea unui repository, asocierea acestui repository cu un fisier local, transmiterea prin comanda push a unor elemente in repository. Posibilitatile care le ofera git sunt foarte mari, insa unicul neajuns pe care l-am gasit eu este ca acesta opereaza cu foarte multe comenzi, si daca acestea nu sunt utilizate zi de zi, exista mereu necesitatea de a face search pe google in cautarea comenzii necesare.