

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:
Lungu Dan

lector asistent:
Victor Gojin

lector superior:
Radu Melnic

1. Scopul lucrarii de laborator

De a se invata utilizarea unui Version Control System si modul de setare a unui server.

2. Obiective

Studierea Version Control Systems (git).

3. Mersul lucrarii de laborator

3.1 Cerințele

- 1.Initializarea unui nou repositoryu.
- 2.Con_gurarea VCS.
- 3.Commit, Push pe branch.
- 4.Folosirea _sierului .gitignore.
- 5.Revenire la versiunile anterioare.
- 6.Crearea branch-urilor noi.
- 7.Commit pe ambele branch-uri.
- 8.Merge la 2 branchuri.
- 9.Rezolvarea conictelor.

3.2 Analiza Lucrarii de laborator

Linkul la repositoryu <https://github.com/lungudan/MIDPS>

Sunt mai multe modalitati de a initializa un repositoryu pe github. Putem crea o mapa goala in care vom plasa gitul nostru prin intermediul comenzii

`git init.`

Urmatorul pas este crearea insusi a noului repositoryu pe care il vom crea utilizind urmatoarea comanda `curl -u 'USER' https://api.github.com /user/repos -d '{"name":"NUME"}`g'. Unde cuvintele scrise cu CAPS se vor inlocui cu numele utilizatorului si numele repositoryului.

Dupa aceasta este necesar sa unim gitul nostru gol cu repositoryul creat.
Vom

folosi urmatoarea comanda git remote add origin "Linkul la repository"

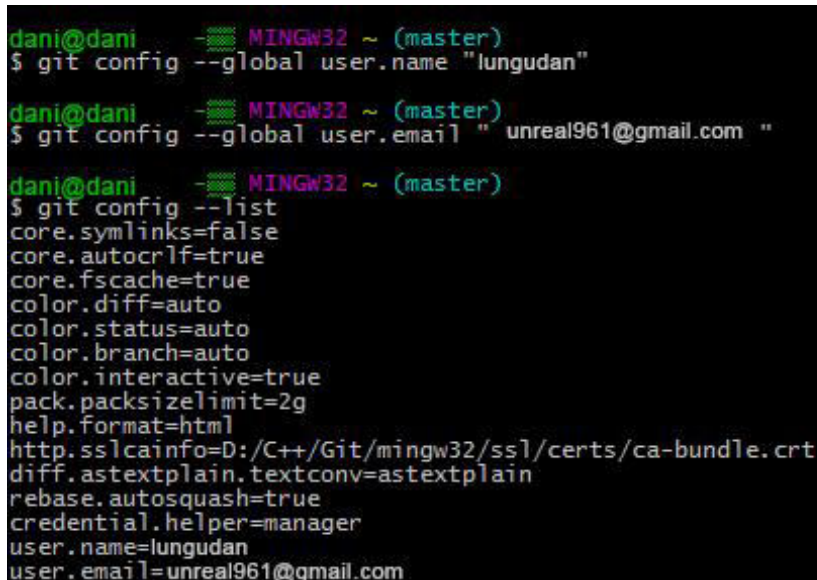
O alta metoda de a crea un repository este cea online. Pentru aceasta este nevoie sa deschidem pagina noastra pe github , sa alegem repositories

si sa apasam butonul new.

Configurarea gitului consta in mai multe etape. La inceput vom configura numele si emailul. Scriem urmatoarele comenzi:

```
git config --global user.name "NUMELE"
```

```
git config --global user.email EMAIL
```



```
dani@dani - MINGW32 ~ (master)
$ git config --global user.name "lungudan"

dani@dani - MINGW32 ~ (master)
$ git config --global user.email "unreal961@gmail.com"

dani@dani - MINGW32 ~ (master)
$ git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
pack.packsizeLimit=2g
help.format=html
http.sslCAinfo=D:/C++/Git/mingw32/ssl/certs/ca-bundle.crt
diff.astextplain.textconv=astextplain
rebase.autosquash=true
credential.helper=manager
user.name=lungudan
user.email=unreal961@gmail.com
```

Urmatorul pas consta in generarea la cheia SSH (Secure Shell). Scriem in CLI ssh-keygen, iar cheia obtinuta o copiem in setarile noastre de pe git. Este de dorit sa initializam repositoryul nostru cu un _sier README.md si un .gitignore. In _sierul README.md vom adauga niste informatie pentru cei care se vor folosi de repository iar in _sierul .gitignore vom adauga toate _sierele ce trebuiesc ignorate (adica sa nu _e incarcate).

Dupa ce am generat keygen-ul,clonam repositoryul pe masina locala.

Pentru a adauga _siere pe repozitoriu,vom folosi urmatoarele comenzi: git add * - comanda indexeaza toate _sierele. git commit -m - comanda face un snapshot la toate schimbarile noastre. git push origin master - comanda

incarca toate _sierele indexate pe git.Totodata vom folosi git status si git show pentru a ne asigura ca _sierele au fost adaugate in repozitoriu.

Revenirea la o versiune mai veche poate _efectuata cu ajutorul comenzii git reset TYPE codul comitului. Exista diferenta intre soft si hard , cind facem soft reset indexurile ramn neschimbate. Iar in cazul in care facem hard reset , pierdem indexurile. Am creat un _sier nou text.txt in versiunea 1.Dupa care l-am sters si am facut commit la versiunea 2 in care am sters _sierul test.txt.Dorim sa revenim la versiunea1.La inceput vom lansa comanda git log care ne arata logul de commituri si codul pentru _ecare commit. Vom avea nevoie de primele 7 cifre la commitul anterior.

Acum folosim comenzile git reset -hard si git reset -soft.

VCS ne permite sa avem mai multe branch-uri.

Branch-urile sunt comod de folosit cind dorim sa lucram paralel la un proiect si apoi dorim sa unim toate

modi_carile. git branch name - creeaza un branch nou cu numele name. git branch - vizualizarea branch-urilor (* indica branch-ul curent). git branch -d nume - sterge branch-ul nume. git checkout -b name - creeaza un branch nou cu numele name si face switch la el.

Vom lucra cu 2 branch-uri - master si nou. Vom crea in _ecare branch cite un _sier to mer,dar continutul _ecaruia va _diferit.

4. Concluzie

În urma efectuării lucrării de laborator numărul 1 la MIDP am studiat VCS. Mi-am aprofundat cunoștințele în folosirea platformei GitHub. Am analizat initializarea unui nou repository și am învățat cum să creez branch-uri noi. A fost o experiență nouă pentru mine, deoarece anterior nu am folosit git-ul. Astfel am aflat că git-ul este un sistem open-source de control a versiunilor conceput de Linus Torvalds, persoana care a creat sistemul de operare Linux. Git este în fapt asemănător cu alte astfel de sisteme de control a versiunilor, precum Mercurial, Subversion sau CVS. Git este efectiv o unealtă bazată pe linii de comandă, însă locul în care se centralizează toate datele și în care are loc stocarea proiectelor este efectiv Hub-ul, mai exact GitHub.com. Aici dezvoltatorii pot adăuga și stoca proiecte la care lucrează împreună cu alți pasionați.