

Facultatea Calculatoare, Informatics is
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:

Madiudin Radu

A verificat:

Gojin Victor

Lucrarea de laborator nr. 1

1. Scopul lucrării de laborator

De a se învăța utilizarea unui Version Control System și modul de setare a unui server.

2. Obiective

Studierea Version Control Systems (git).

3 Mersul lucrării de laborator

3.1 Cerințele

Initializarea unui nou repository.

Configurarea VCS.

Commit, Push pe branch.

Folosirea fișierului .gitignore.

Revenire la versiunile anterioare.

Crearea branch-urilor noi.

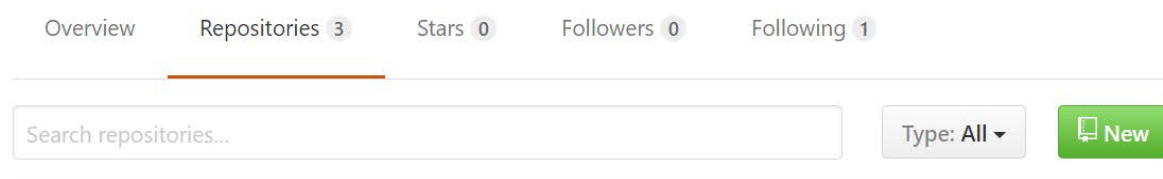
Commit pe ambele branch-uri.

Merge la 2 branchuri.

Rezolvarea conflictelor.

3.2 Analiza Lucrării de laborator

Una din metodele de initializare a unui repository pe github este să deschidem pagina noastră pe github, să alegem repositories și să apăsăm butonul new.



Configurarea gitului constă în mai multe etape. La început vom configura numele și emailul. Scriem următoarele comenzi:

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

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

```
Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (master)
$ git config --global user.name "Hiruine"

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (master)
$ git config --global user.email "radu.madiudin@yahoo.com"

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (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
help.format=html
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
diff.astextplain.textconv=astextplain
rebase.autosquash=true
credential.helper=manager
user.email=radu.madiudin@yahoo.com
user.name=Hiruine
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
remote.origin.url=git@github.com:Hiruine/MIDPS.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.master.remote=origin
branch.master.merge=refs/heads/master
```

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 repozitoriul nostru cu un fisier README.md si un .gitignore. In fisierul README.md vom adauga niste informatie pentru cei care se vor folosi de repozitoriu iar in fisierul .gitignore vom adauga toate fisierele ce trebuiesc ignorate (adica sa nu fie incarcate).

Vom adauga fisierele noi create pe repozitoriul nostru. Pentru aceasta vom avea nevoie de urmatoarele comenzi :

git add . - comanda indexeaza toate fisierele.

git commit -m - comanda face un snapshot la toate schimbarile noastre.

git push origin master - comanda incarca toate fisierele indexate pe git.

Pentru a ne asigura ca am facut totul bine si nu avem probleme vom utiliza **:git status**

```

Hiruline@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (master)
$ git add *

Hiruline@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (master)
$ git commit -m "file creation"
[master cefe81e] file creation
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 lab_1/amazing.txt

Hiruline@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (master)
$ git push origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 300 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To github.com:Hiruline/MIDPS.git
8308ce4..cefe81e master -> master

Hiruline@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean

Hiruline@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (master)
$ git show
commit cefe81ed0ce9dd3aad2583f9d04e4de8f873751c
Author: Hiruline <radu.madiudin@yahoo.com>
Date: Sun Feb 26 19:09:12 2017 +0200

    file creation

diff --git a/lab_1/amazing.txt b/lab_1/amazing.txt
new file mode 100644
index 0000000..e69de29

```

Una dintre caracteristicile principale a unui VCS este faptul ca ne permite sa revenim la o versiune mai veche. Aceasta poate fi efectuata cu ajutorul comenzii `git reset --TYPE "codul comitului"`. Exista diferenta intre `--soft` si `--hard`, cind facem `soft reset` indexurile ramin neschimbate. Iar in cazul cind facem `hard reset`, pierdem indexurile.

```

Hiruline@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (master)
$ git reset --hard cefe81e
HEAD is now at cefe81e file creation

```

VCS ne permite sa avem mai multe branchuri. Din traducere branch semnifica "creanga". Branchurile sunt foarte comod de folosit cind dorim sa lucram paralel la un proiect si apoi dorim sa unim toate modificarile.

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

```
Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (master)
$ git branch "branch_1"

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (master)
$ git branch
  branch_1
* master

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (master)
$ git checkout -b "branch_2"
M       lab_1/amazing.txt
Switched to a new branch 'branch_2'

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (branch_2)
$ git checkout "branch_1"
M       lab_1/amazing.txt
Switched to branch 'branch_1'

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (branch_1)
$ |
```

git checkout "nume" - face switch la branchul "nume".

git branch -u upstream/name - face track la branchul indicat din branchul curent.

git branch -u upstream/name "nume" - face track din branchul "nume" la branchul indicat.

git branch --track "name" upstream/name - creeaza branchul "name" si ii face track la branchul indicat.

git branch --unset-upstream - scoate trackingul la branchul in care ne aflam.

Putem avea conflicte in cazul cind dorim sa facem merge la 2 branchuri si unele rinduri sunt diferite. In asa caz ne vin in ajutor un mergetool.

```
Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (branch_2)
$ git checkout branch_1
Switched to branch 'branch_1'

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (branch_1)
$ git merge branch_2 branch_2
Auto-merging lab_1/amazing.txt
CONFLICT (content): Merge conflict in lab_1/amazing.txt
Automatic merge failed; fix conflicts and then commit the result.

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (branch_1|MERGING)
$ |
```

Pentru a rezolva problema vom utiliza taguri pe a marca comiturile. Sunt doua tipuri de taguri in git: annotated si lightweight . Un tag lightweight este asemeni unui branch care nu se schimba – este doar un pointer la un commit specific. Tagurile Annotated , sunt pastrate ca obiecte in baza de date a gitului. Pentru a crea un annotated tag utilizam **git tag -a v1.1 -m "Message"**, unde v.1 este numele tagului

Conflictul

```
Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (g2)
$ git add .

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (g2)
$ git commit -m "cauza 2"
[g2 73140a0] cauza 2
1 file changed, 1 insertion(+), 1 deletion(-)


Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (g2)
$ git checkout g1
Switched to branch 'g1'

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (g1)
$ git add .

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (g1)
$ git add .

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (g1)
$ git commit -m "cauza 1"
[g1 f73f5a1] cauza 1
1 file changed, 1 insertion(+), 1 deletion(-)

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (g1)
$ git merge g1 g2
Auto-merging lab_1/amazing.txt
CONFLICT (content): Merge conflict in lab_1/amazing.txt
Automatic merge failed; fix conflicts and then commit the result.
```

 amazing.txt - Notepad

File Edit Format View Help

```
<<<<<<< HEAD
modificare nr 111
=====
modificarea nr 222
>>>>>>> g2
```

```
Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (g1|MERGING)
git add .

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (g1|MERGING)
git commit -m "Merge with g2"
[g1 448bdf9] Merge with g2

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (g1)
|
```

```

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (g1)
$ git checkout master
Your branch is up-to-date with 'origin/master'.
Switched to branch 'master'

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (master)
$ git tag -a ver.1 -m "Before merge with g1"

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (master)
$ git tag
v1
v1.1
ver.1

Hiruine@DESKTOP-BDQA38V MINGW64 /C/MIDPS/lab_1 (master)
$ git show ver.1
tag ver.1
Tagger: Hiruine <radu.madiudin@yahoo.com>
Date: Sun Feb 26 22:02:46 2017 +0200

Before merge with g1

commit 83bdbfd47a7619062bfa9715241bb1d37d0b8ec8
Author: Hiruine <radu.madiudin@yahoo.com>
Date: Sun Feb 26 21:42:36 2017 +0200

    again

diff --git a/lab_1/amazing.txt b/lab_1/amazing.txt
index 3af50b3..74b8100 100644
--- a/lab_1/amazing.txt
+++ b/lab_1/amazing.txt
@@ -1,8 +1 @@
 prima modificare pu a testa revenirea la o versiune mai veche
-<<<<<<< HEAD
-commit 1
-commit 2 ups
-=====
-commit 2, commit2
-commit 1, modificarea
->>>>>> branch_2

```

4 Concluzii

In lucrarea data am studiat lucrul cu VCS. Am luat cunostinta cu platforma github. Toate lucrurile, comenzile le-am indeplinit in Git Bash. Sunt foarte multe plusuri in folosirea VCS. Fara el producerea produselor soft ar fi foarte lenta si problematica. El ne permite lucrul paralel, menajarea versiunelor, revenire la versiuni anterioare.