

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:

Cristian GODONOAGA

lector asistent:

Irina COJANU

lector superior:

Radu MELNIC

Laboratory work #1

1 Scopul lucrării de laborator

Studierea unui sistem de control a versiunelor (**Version Control System**) utilizind linia de comanda (**Command Line Interface**).

2 Obiective

- Insusirea modului de utilizare a celor mai importante componente a unui VCS descentralizate.

3 Laboratory work implementation

3.1 Tasks and Points

- Initializarea unui repository
- Configurea VCS
- Folosirea fisierului .gitignore
- Crearea branch-urilor
- Commit, Push pe branch-uri
- Revenire la versiunile anterioare
- Track a remote branch
- Resetarea unui branch la commit-ul anterior
- Salvarea temporara a schimbarilor (stash)
- Folosirea fisierului .gitignore
- Unirea a doua branch-uri
- Rrezolvarea conflictelor
- Folosirea tag-urilor.

3.2 Analiza lucrarii de laborator

Înainte de a începe îndeplinirea sarcinilor s-au efectuat câțiva pași adiționali, ce țin de pregătirea calculatorului pentru a putea utiliza acest sistem de versionare a codului. Am ales un sistem descentralizat deoarece acesta ne oferă posibilități ample, referitor la controlul distribuit al versiunilor, gestionarea și revizuirea codului cit și vizualizarea activității proiectului față de sistemul centralizat. În special, acesta fiind un model de source control ce permite partajarea surselor în mod distribuit între membrii echipelor fără a depinde de un repository central.

Instalarea și setarea sistemului Git sa efectuat conform manualului oferit de situl oficial git-scm.com.

Utilizând serviciile prestate de rețeaua socială (GitHub) pentru proiecte cu versionarea bazată pe Git, am creat un cont pe www.github.com care detine repositoryul laboratorului efectuat.



Link la repository: <https://github.com/CristianGodonoaga/MIDPS>

Pentru a asigura o comunicare securizată a datelor între două stații am utilizat protocolul de rețea SSH. Acesta se folosește de criptarea cu **chei asimetrice** ce oferă un mod mai sigur de logare (comunicare) într-un server cu SSH decât folosind o parolă obișnuită. La generarea acestor perechi de chei se utilizează un mecanism special care ne oferă două siruri lungi de caractere: o cheie publică și o cheie privată.

Într-un prim pas se creează perechea de chei pe mașina client:

```
$ssh-keygen -t rsa
```

Dupa ce am introdus comanda de generare a cheii, am primit citeva intrebari ce tine de marirea securitatii prin protejarea cheii private, cu o parola de acces. Odata ce perechea de chei este generata plasam cheia publica pe serverul GitHub.

Pentru a incepe utilizarea sistemului Git, mai intii este nevoie de setarea configuratiei de baza. Aceste configurari pot fi facute in trei modalitati, ele vor determina cit de amplu va fi aplicarea acestora. Utilizind comenzile:

```
$git config --global user.name "Godonoaga C."  
$git config --global user.email "some@email.com"
```

am setat numele si posta electronica la nivel de utilizator, ce ne permite utilizarea acestor date in viitor pentru oarecare alt proiect al acestuia fara a le specifica in parte. Deoarece aceste setari depind foarte mult de necesitatea utilizatorului nu exista o varianta exacta ce tine de setarea sistemului Git.

Pentru a da start sistemului si anume pentru a monitoriza schimbarile in proiectul necesar se utilizeaza comanda `$git init`, aceasta va insemna ca noi avem nevoie de monitorizarea schimbarilor a tuturor fisierelor ce se afla in folderul curenta (creareaza un "depozit" numit repository).

Din acest moment putem incepe versionarea proiectului nostru insa mai este un pas ce tine de setare, nu toate fisierele ce sunt in proiectul nostru au nevoie de versionare, unele fiind executabilele generate de procesul de compilare sau efectiv parametrii de configurare ai aplicatiei. Pentru aceasta fisierele sau directoarele ce se doresc a fi ignorate de sistemul de versionare se noteaza intr-un fisier numit `.gitignore`. Voi plasa acest fisier in radacina proiectului pentru a nu cauta foarte mult path-urile ignorate, deoarece acesta este evaluat in mod recursiv si putem avea mai multe fisiere `.gitignore`.

Salvarea starii fisierelor (snapshot) sau intregul proiect se efectueaza folosind comenzile de baza pe care le contine Git. Ele ne permit gestionarea procesului de versionare. La indeplinirea lucrarii de laborator sunt utilizate doar o parte, insa ele contin o multime destul de vasta si este nevoie de practica pentru a percepe modul de functionare si pentru a capata deprinderea de utilizare.

Un alt mecanism ce ne ofera sistemul Git este ramificarea acestui proces de versionare. Posibilitatile oferite de acesta sunt multe iar baza fiind clonarea starii unei ramuri numite "branch", unde pot fi aplicate modificari care nu afecteaza ramura parinte. Adica dintr-un anumit punct al dezvoltarii produsului nostru, stiva de commit-uri se poate ramifica si apoi directiile de dezvoltare vor diverge. Desigur, poate aparea caz contrar, in care este nevoie de alipirea(unirea) ramurilor sau a unei componente. Deci Git dispune si de acest mecanism, insa el este unul din cele mai complexe si lipsa cunostintelor de a utiliza efectiv aceasta comanda poate provoca conflicte. In mod normal aceasta unire este efectuata automat, insa mecanismul nu este capabil sa alipeasca unu si acelasi document prezent in ambele ramuri cu context diferit. Aici apelam la rezolvarea acestora in mod manual indicind varianta corecta.

In continuare am atasat imagini care contin desfasurarea lucrarii

3.3 Imagini

```
virtual@cristian:~$ ssh-keygen -t rsa -f ~/.ssh/id_rsa -C "Key for GitHub"
Generating public/private rsa key pair.
Created directory '/home/virtual/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/virtual/.ssh/id_rsa.
Your public key has been saved in /home/virtual/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:0q/3Vq7dcMpkJBsBI0ZVdXUjLf2WKQTul/A2C205QVc Key for GitHub
The key's randomart image is:
+---[RSA 2048]---+
|      .+.+.00.+Eo|
|      . . 0.  +000|
|      .+. . . +|
|      0.=..00|
|      S oB.*..|
|      . . =0 o |
|      o  .+..|
|      o. . =.|
|      .0.00.+ .|
+-----[SHA256]-----+
virtual@cristian:~$
```

Figure 3.1 – Generarea ssh-key

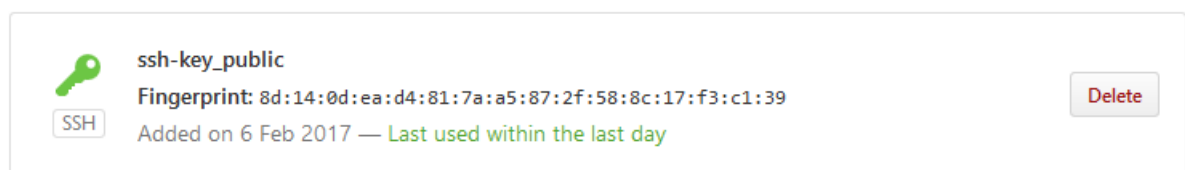
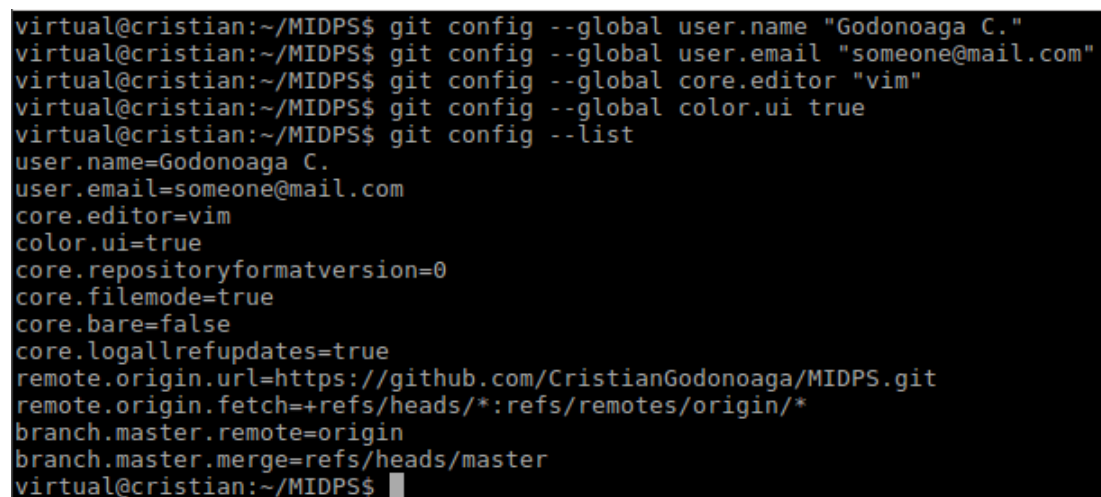


Figure 3.2 – Adaugarea ssh-key pe server

A terminal window with a black background and white text. The prompt is 'virtual@cristian:~/MIDPS\$'. The user enters four commands: 'git config --global user.name "Godonoaga C."', 'git config --global user.email "someone@mail.com"', 'git config --global core.editor "vim"', and 'git config --global color.ui true'. The fifth command is 'git config --list', which outputs a list of configuration settings. The settings include: 'user.name=Godonoaga C.', 'user.email=someone@mail.com', 'core.editor=vim', 'color.ui=true', 'core.repositoryformatversion=0', 'core.filemode=true', 'core.bare=false', 'core.logallrefupdates=true', 'remote.origin.url=https://github.com/CristianGodonoaga/MIDPS.git', 'remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*', 'branch.master.remote=origin', and 'branch.master.merge=refs/heads/master'. The prompt returns to 'virtual@cristian:~/MIDPS\$' at the bottom.

```
virtual@cristian:~/MIDPS$ git config --global user.name "Godonoaga C."
virtual@cristian:~/MIDPS$ git config --global user.email "someone@mail.com"
virtual@cristian:~/MIDPS$ git config --global core.editor "vim"
virtual@cristian:~/MIDPS$ git config --global color.ui true
virtual@cristian:~/MIDPS$ git config --list
user.name=Godonoaga C.
user.email=someone@mail.com
core.editor=vim
color.ui=true
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
remote.origin.url=https://github.com/CristianGodonoaga/MIDPS.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.master.remote=origin
branch.master.merge=refs/heads/master
virtual@cristian:~/MIDPS$
```

Figure 3.3– Configurare VCS

Concluzie

Git este o unealtă destul de complexă și la fel și GitHub, așa că nu avem nicio șansă să acoperim totul într-un singur capitol. Defapt, abia am atins suprafața a ceea ce e posibil cu aceste unelte.

timilie pentru Versionare codului

conostinta cu Git

mobilitate (ceia ce tine de lucru echipei la distanta)

Aici trebuie sa fie concluzia ta.

References

- 1 Pro Git Manual, *official page*, <https://git-scm.com/book/ru/v1>
- 2 Learnprogrammfree, *youtube page*, https://www.youtube.com/playlist?list=PLpY_9m7gHQDibfa9MJfnEQd2UWI5gztp8
- 3 Timo Ojala, *Multiresolution gray-scale and rotation invariant texture classification with local binary patterns*, 2002
- 4 Biometric, www.biometricupdate.com/201501/history-of-biometrics