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 lucrarii 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 repositoriu
Configurea VCS
Folosirea fisierului .gitignore
Crearea branch-urilor
Commit, Push pe branch-uri
Revenire la versiunele 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
Rezolvarea conflictelor
Folosirea tag-urilor.

3.2 Analiza lucrarii de laborator

Inainte de a incepe indeplinirea sarcinelor s-au efectuat citiva pasi aditionali, ce tin de pregatirea calculatorului pentru a putea utiliza acest sistem de versionare a codului. Am ales un sistem descentralizat deoarece acesta ne ofera posibilitati ample, referitor la controlul distribuit al versiunelor, gestionarea si revizuirea codului cit si vizualizarea activitatii proectului fata de sistemul centralizat. In special, acesta fiind un model de source control ce permite partajarea surselor in mod distribuit intre membrii echipelor fara a depinde de un repozitoriu central.

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

Utilizind serviciile prestate de reteaua sociala (GitHub) pentru proiecte cu versionarea bazata pe Git, am creat un cont pe *www.github.com* care detine repozitoriul laboratorului efectuat.

 Link la repozitoriu: `https://github.com/cyberti/MIDPS`

Pentru a asigura o comunicare securizata a datelor intre doua statii am utilizat protocolul de retea SSH. Acesta se foloseste de criptarea cu ***chei asimetrice*** ce ofera un mod mai sigur de logare (comunicare) intr-un server cu SSH decit folosind o parola obisnuita. La generarea acestor perechi de chei se utilizeaza un mecanism special care ne ofera doua siruri lungi de caractere: o cheie publica si o cheie privata.

Intr-un prim pas se creaza perechea de chei pe masina 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 fisierilor 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 nostu 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 direcțiile de dezvoltare vor diverge. Desigur, poate aparea caz contrar, in care este nevoie de alipirea(unirea) ramurelor 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 concret rezultatul pe care dorim sal obtinem in final.

In continuare sunt atasate rezultatele obtinute in urma indeplinirei sarcinelor.

3.3 Imagini

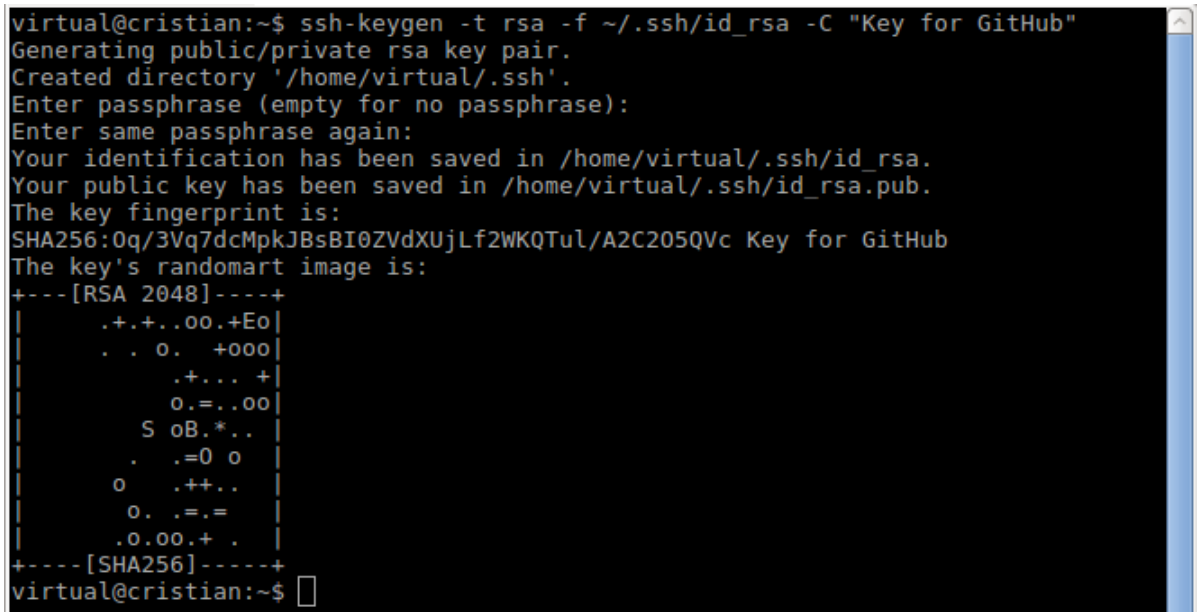

Figure 3.1 – Generarea ssh-key

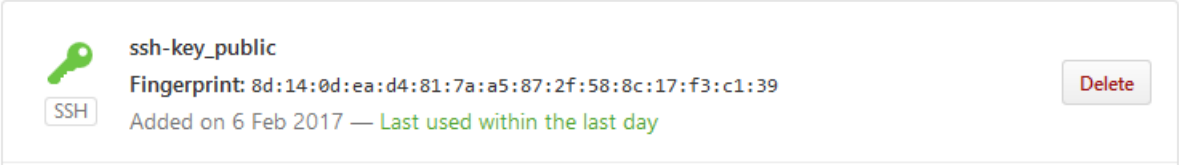

Figure 3.2 – Adaugarea ssh-key pe server

```
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

```
virtual@cristian:~/MIDPS2$ git init
Initialized empty Git repository in /home/virtual/MIDPS2/.git/
virtual@cristian:~/MIDPS2$
```

Figure 3.4– Crearearea unui repository local

```
cristian:~/MIDPS2$ git remote add origin git@github.com:cyberti/MIDPS.git
cristian:~/MIDPS2$ git fetch origin
From github.com:cyberti/MIDPS
 * [new branch]      master     -> origin/master
cristian:~/MIDPS2$ git branch --set-upstream-to=origin/master master
Branch master set up to track remote branch master from origin.
cristian:~/MIDPS2$
```

Figure 3.5– Set to track a remote branch

```
cristian:~/MIDPS2$ git tag v0.1
cristian:~/MIDPS2$ git add README.md
cristian:~/MIDPS2$ git commit --amend -m "Initial commit"
[master 3e11684] Initial commit
 Author: cyberti <fex500@yandex.ru>
 Date: Mon Feb 20 11:42:46 2017 +0300
 2 files changed, 263 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 README.md
cristian:~/MIDPS2$ git push origin --tags
Total 0 (delta 0), reused 0 (delta 0)
To git@github.com:cyberti/MIDPS.git
 * [new tag]         v0.1 -> v0.1
cristian:~/MIDPS2$
```

Figure 3.6– Snapshot and push on remote server

```
cristi:~/MIDPS2$ git reset --hard c4a7f13b3fc9ec
HEAD is now at c4a7f13 lab.1: Add some screen
cristi:~/MIDPS2$
```

Figure 3.7– Resetarea la commit-ul anterior

```
cristian:~/MIDPS2$ git merge
fatal: You have not concluded your merge (MERGE_HEAD exists).
Please, commit your changes before you merge.
cristian:~/MIDPS2$ git commit -m "Lab.1: Fix README file"
[master 6b185a8] Lab.1: Fix README file
cristian:~/MIDPS2$ git merge
Already up-to-date.
cristian:~/MIDPS2$ git push origin master
Counting objects: 25, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (25/25), done.
Writing objects: 100% (25/25), 75.14 KiB | 0 bytes/s, done.
Total 25 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To git@github.com:cyberti/MIDPS.git
   05b58a7..6b185a8  master -> master
cristian:~/MIDPS2$
```

Figure 3.8– Unirea a doua branch-uri

```
cristi:~/MIDPS2$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
cristi:~/MIDPS2$ git merge dev_branch
Auto-merging lab1/TEX template/Chapter_1.tex
CONFLICT (content): Merge conflict in lab1/TEX template/Chapter_1.tex
Removing lab1/MIDPS_LAB1.md
Automatic merge failed; fix conflicts and then commit the result.
cristi:~/MIDPS2$
```

Figure 3.9– Conflict la unirea a 2 branch-uri

```
153 <<<<<<< HEAD
154 \clearpage
155 =======
156 \begin{figure}[htb]
157 >---\begin{center}
158 >--->---\centering
159 >--->---\includegraphics[scale = 0.9]{img/git_reset.png}
160 >--->---\caption{Resetarea la commit-ul anterior}%
161 >--->---\label{fig:git_reset}
162 >---\end{center}
163 \end{figure}
164
165 \begin{figure}[htb]
166 >---\begin{center}
167 >--->---\centering
168 >--->---\includegraphics[scale = 0.9]{img/git_merge.png}
169 >--->---\caption{Unirea a doua branch-uri}%
170 >--->---\label{fig:gitmerge}
171 >---\end{center}
172 \end{figure}
173
174
175 \clearpage
176 >>>>>>> dev_branch
~
```

Figure 3.10– Rezolvarea conflictelor

Concluzie

Efectuind lucrarea de laborator nr. 1 am cercetat sistemul de versionare a codului "**Git**". Am opservat ca Git este o unealtă destul de complexă, la fel și GitHub. Consider ca o echipa de dezvoltatori pote să colaboreze mult mai eficient utilizind serviciile prestate de github.com. Aici au posibilitatea de a stoca in remote proiectele, si apoi sa lucreze paralel asupra lor. Am constatat ca sistemul ofera o flexibilitate foarte extinsa in ceea ce priveste module de folosire. Fiind destul de faimos acesta poate fi integrat in foarte multe medii de dezvoltare a softului.
Recunosc faptul ca in aceasta lucrare de laborator nu am atins tot intregul functionalitatilor oferite de Git. Defapt, abia am atins suprafața a ceea ce e posibil cu aceste unelte. Sunt bucuros ca inginerii IT (*Linus Trovalds*) au realizat un astfel de soft pe care il studiez si utilizez din ce in ce mai frecvent.

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 Lars Vogel , *Git - Tutorial*, http://www.vogella.com/tutorials/Git/article.html

4 Catalin Gosman , *Versioning (SVN)*, http://elf.cs.pub.ro/idp/laboratoare/lab-02#git