Facultatea Calculatoare, Informatica si Microelectronica

Universitatea Tehnica a Moldovei

Medii Interactive de Dezvoltare a Produselor Soft

Lucrarea de laborator#1

Version Control Systems

Autor:
Dragoș PALADE

lector asistent:
Irina Cojanu
lector superior:
Svetlana Cojocaru

Laboratory work #2

1 Scopul lucrarii de laborator

Interactionarea cu un sitem VCS, utilizand CLI care il ofera acesta. Dobandirea cunostintelor de baza de lucru cu un sistem VCS(github).

2 Obiective

– Intelegerea si folosirea CLI (basic level)

– Administrarea remote a masinilor linux machine folosind SSH (remote code editing)

– Version Control Systems (git —— bitbucket —— mercurial —— svn)

3 Laboratory work implementation

3.1 Tasks and Points

- Crearea si configurarea unui ssh key
- initializeaza un nou repositoriu
- 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
- folosirea fisierului .gitignore
- merge 2 branches
- rezolvarea conflictelor a 2 branches
- Folosirea tag-urilor pentru marcarea schimbarilor simnificative precum release-ul.

3.2 Analiza lucrarii de laborator

Repository `https://github.com/dragosh1011/MIDPS-laboratories/tree/branch_2/MIDPS/Documentations/LAB_1`link

Primul pas la realizara acestei lucrai de laborator a fost configurarea unei ssh key. Pentru aceasta am folosit documentatia de pe github [1] care ofera pas cu pas informatia cum se realizaza acest lucru. Realizarea unui repositoiu am facut-o pe interfata web a githab. Github ofera comanda git config pentru a configura numele si emailul utilizatorului.

Pentru crearea un branch si schimbrarea imediata pe acesta am utilizat git checkout -b, iar apoi pentru schimbarea pe un branch existent git checkout. Pentru a face commit si push pe remote branch am utilizat git commit -am message si git push origin branch.

git reset HEAD~ ofera posibilitatea de a te anula utlimul commit realizat. File-ul .gitignore se gaseste aproape in orice repozitoriu git, deoarece majoritatea proiectelor in procesul de dezvoltare utilizeaza multe fisiere care nu sunt relevante pentru fiecare dezvoltator sau contine fisiere ce sunt specifice pentru fiecare dezvoltator in parte. git merge branch este comanda care permite sa facem merge intre branchul actual si un branch remote.

Rezolvarea conflictelor este o necesitate des intilnita atunci cand la un proiect lucreaza mai multi dezvoltatori concomitent. Chiar si daca esti un singur dezvoltator, nu esti scutit de riscul de a avea la un moment dat conflicte, daca nu ai tinut cont de unele modificari facute pe alte branchuri. Tagurile sunt intilnite apropae in oricare proiect, deoarece este important de creat o versionare a

produsului care permite urmarirea modificarilor care se fac de la o veriune la alta, dar si revenirea la o versiune mai veche in caz de necesitate.

3.3 Imagini

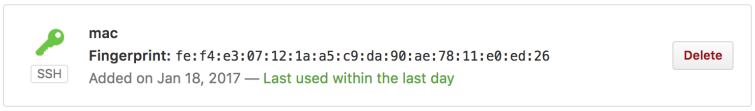

Figure 3.1 – Configured ssh key

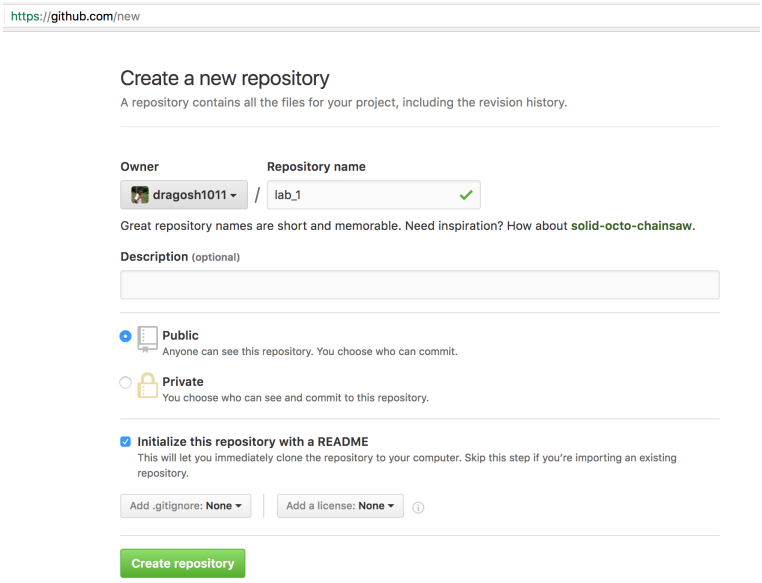

Figure 3.2 – Create new repository

Figure 3.3 – Config github user name and email

```
83:MIDPS-laboratories dpalade$ git checkout -b brach_1
Switched to a new branch 'brach_1'
83:MIDPS-laboratories dpalade$ git status
On branch brach_1
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    MIDPS/Documentations/LAB_1/test.file

83:MIDPS-laboratories dpalade$ git commit -am "remove file"
[brach_1 ebd618b] remove file
 1 file changed, 2 deletions(-)
 delete mode 100644 MIDPS/Documentations/LAB_1/test.file
83:MIDPS-laboratories dpalade$ git push origin brach_1
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 469 bytes | 0 bytes/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To github.com:dragosh1011/MIDPS-laboratories.git
   8cd3698..ebd618b  brach_1 -> brach_1
83:MIDPS-laboratories dpalade$ git checkout -b branch_2
Switched to a new branch 'branch_2'
83:MIDPS-laboratories dpalade$ git status
On branch branch_2
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   MIDPS/Documentations/LAB_1/test

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   MIDPS/Documentations/LAB_1/test

83:MIDPS-laboratories dpalade$ git add .
83:MIDPS-laboratories dpalade$ git commit -am "Add test file"
[branch_2 840d7d8] Add test file
 1 file changed, 2 insertions(+)
 create mode 100644 MIDPS/Documentations/LAB_1/test
83:MIDPS-laboratories dpalade$ git push origin branch_2
Counting objects: 6, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 629 bytes | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To github.com:dragosh1011/MIDPS-laboratories.git
 * [new branch]      branch_2 -> branch_2
83:MIDPS-laboratories dpalade$
```

Figure 3.4– Create 2 branches and do commits on them

```
83:MIDPS-laboratories dpalade$ git status
On branch branch_2
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   MIDPS/Documentations/LAB_1/testr/test.js

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .gitignore

83:MIDPS-laboratories dpalade$
```

Figure 3.5– Use .gitignore

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .gitignore

83:MIDPS-laboratories dpalade$ git commit -am "modify .gitignore"
[branch_2 1590b0e] modify .gitignore
 2 files changed, 6 insertions(+), 1 deletion(-)
 create mode 100644 MIDPS/Documentations/LAB_1/testr/test.js
83:MIDPS-laboratories dpalade$ git status
On branch branch_2
nothing to commit, working tree clean
83:MIDPS-laboratories dpalade$ git reset HEAD~
Unstaged changes after reset:
M       .gitignore
83:MIDPS-laboratories dpalade$ git status
On branch branch_2
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .gitignore

no changes added to commit (use "git add" and/or "git commit -a")
83:MIDPS-laboratories dpalade$
```

Figure 3.6– reset last commit

```
83:MIDPS-laboratories dpalade$ git merge branch_2
Updating ebd618b..f8c6400
Fast-forward
 .gitignore                        | 4 +++-
 MIDPS/Documentations/LAB_1/test | 2 ++
 2 files changed, 5 insertions(+), 1 deletion(-)
 create mode 100644 MIDPS/Documentations/LAB_1/test
83:MIDPS-laboratories dpalade$
```

Figure 3.7 – Merge 2 branches

```
83:MIDPS-laboratories dpalade$ git merge brach_1
Auto-merging MIDPS/Documentations/LAB_1/test
CONFLICT (content): Merge conflict in MIDPS/Documentations/LAB_1/test
Automatic merge failed; fix conflicts and then commit the result.
```

Figure 3.8 – Conflict

```
<<<<<<< HEAD
I'm sure that conflict will be appear
=======
conflict will be appear
>>>>>>> brach_1
```

Figure 3.9 – Conflict file

```
83:MIDPS-laboratories dpalade$ git tag
v0.1.0
83:MIDPS-laboratories dpalade$ git tag -a v0.2.0 -m "Added documentaion"
83:MIDPS-laboratories dpalade$ git tag
v0.1.0
v0.2.0
83:MIDPS-laboratories dpalade$
```

Figure 3.10 – Tag

Concluzie

In urma realizarii acestei lucrari de laborator am reusit sa adaug cateva cunostinte noi la cele acumulate anterioar legate de github. Creara unui tag cu git cli si merge la 2 branchuri din cli sunt actiuni care nu le-am folosit sau nu m-am intilnit cu ele in utlima perioada si care sunt inforamtii noi pentru mine. De asemenea pe parcursul elaborarii lucrarii am aflat si ca un pull request se poate crea la fel din command line. Posibilitatile pe care le ofera git sunt foarte mari, insa atita timp cat existe unele comenzi care sunt utilizate zi de zi, esti mai putin interesant de cele de care ai nevoie foarte rar si pe care de obicei le gasesti atunci cand ai nevoie dupa o cautare pe google.

References

1 Github add ssh key, https://help.github.com/articles/adding-a-new-ssh-key-to-your-github-account/

2 Undo anything, https://github.com/blog/2019-how-to-undo-almost-anything-with-git

3 Tagging, https://git-scm.com/book/en/v2/Git-Basics-Tagging