Facultatea Calculatoare, Informatica si Microelectronica

Universitatea Tehnica a Moldovei

Medii Interactive de Dezvoltare a Produselor Soft

Lucrarea de laborator Nr.1

Version Control Systems si modul de setare a unui server

Efectuat : **Tabuncic Vladislav**

Verificat : lector asistent : **Cojanu Irina**

2017

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).

Intelegerea si aplicarea comenzilor GIT.

3. Mersul lucrarii de laborator

3.1 Cerintele :

* Initializare unui nou repositoriu.

* Configurarea VCS.

* Crearea branch-urilor si commit pe ambele branch-uri

* Resetarea branch-urilor la commit-urile anterioare

* Merge la 2 branchuri.

* Folosirea fisierului .gitignore..

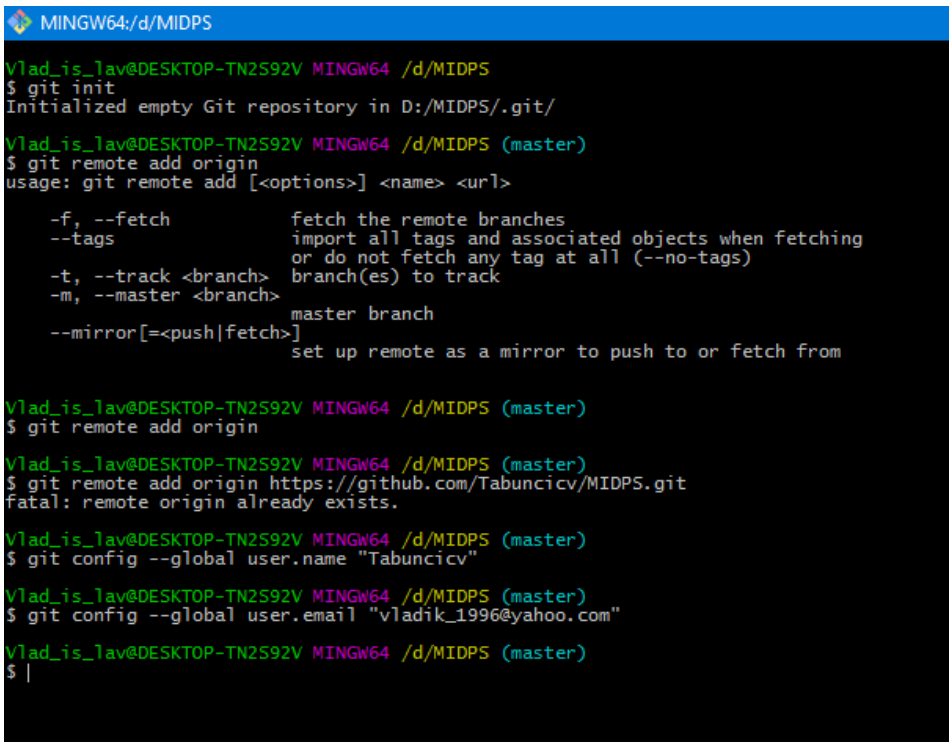
* Rezolvarea conflictelor.

3.2 Analiza lucrarii de laborator :

Linkul repositoriului **https://github.com/Tabuncicv/MIDPS.git**

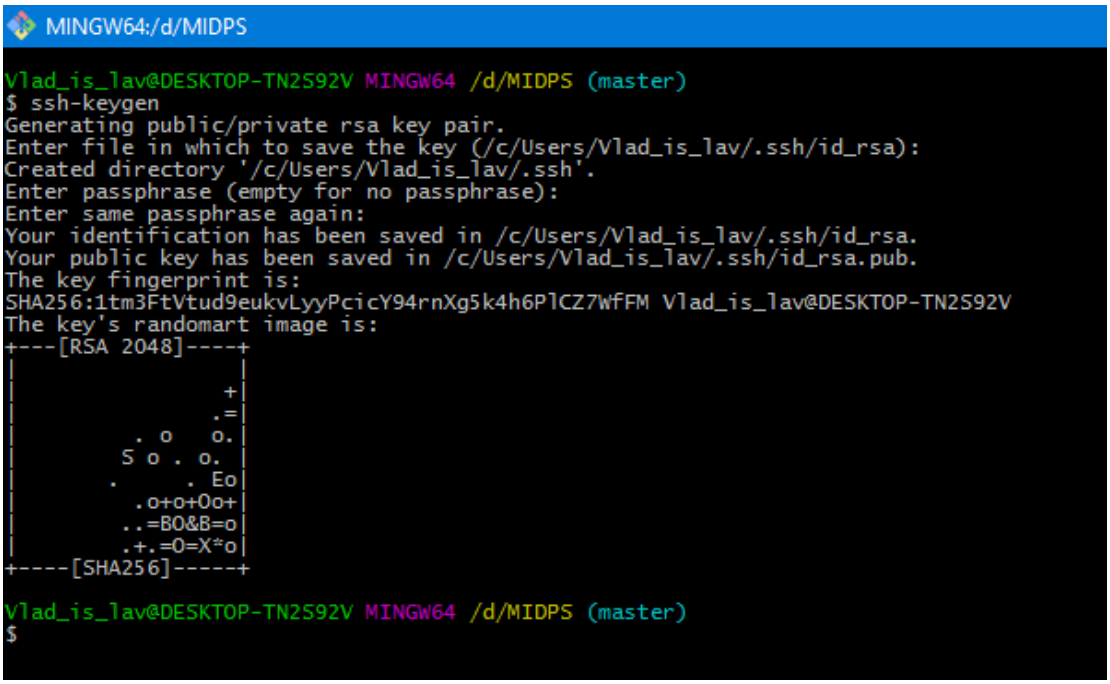
Sunt mai multe modalitati de a initializa un repozitoriu pe github. Putem crea o mapa goala in care vom plasa gitul nostru prin intermediul comenzii **git init.**

Urmatorul pas este crearea a noului repozitoriu pe care il vom crea utilizind urmatoarea comanda **curl – u 'USER' https.//api.github.com/user/repos/ -d '{"name":"NUME"}'.** Unde cuvintele scrise cu CAPS se vor inlocui cu numele utilizatorului si numele repozitoruui. Dupa aceasta este necesar sa unim gitul nostru gol cu repozitoriul creat. Vom folosi urmatoare camonda **git remote add origin "Linkul la repozitoriul nostru"**



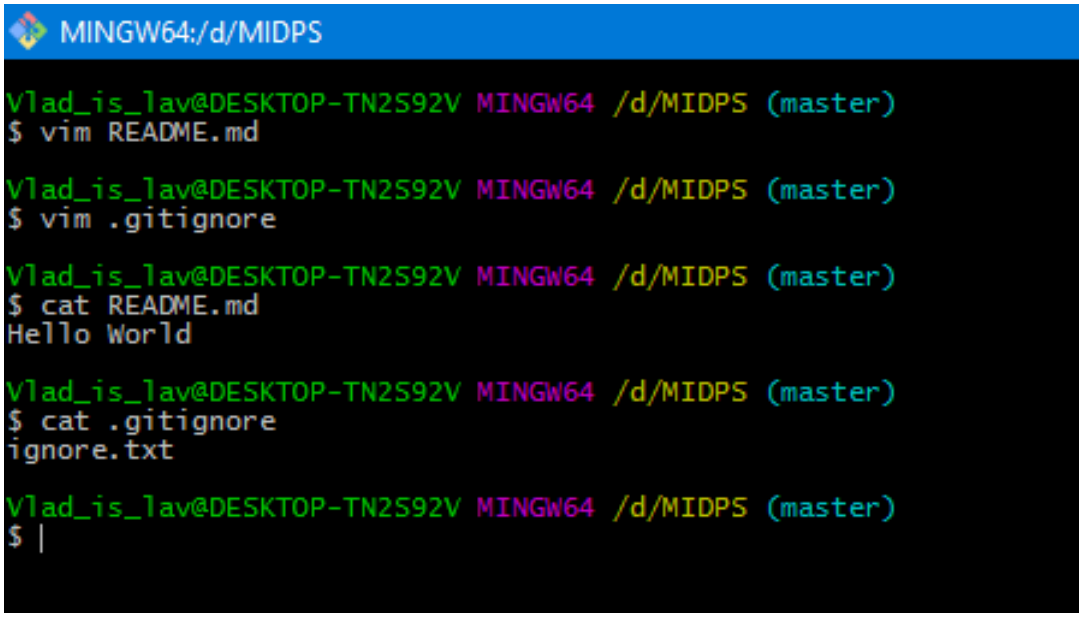
Configurarea gitului consta in mai multe etape. La inceput vom configura numele si emailul prin intermediul urmatoarelor comenzi :

git config –global user.name “Numele”
git config –global user.email “Email”

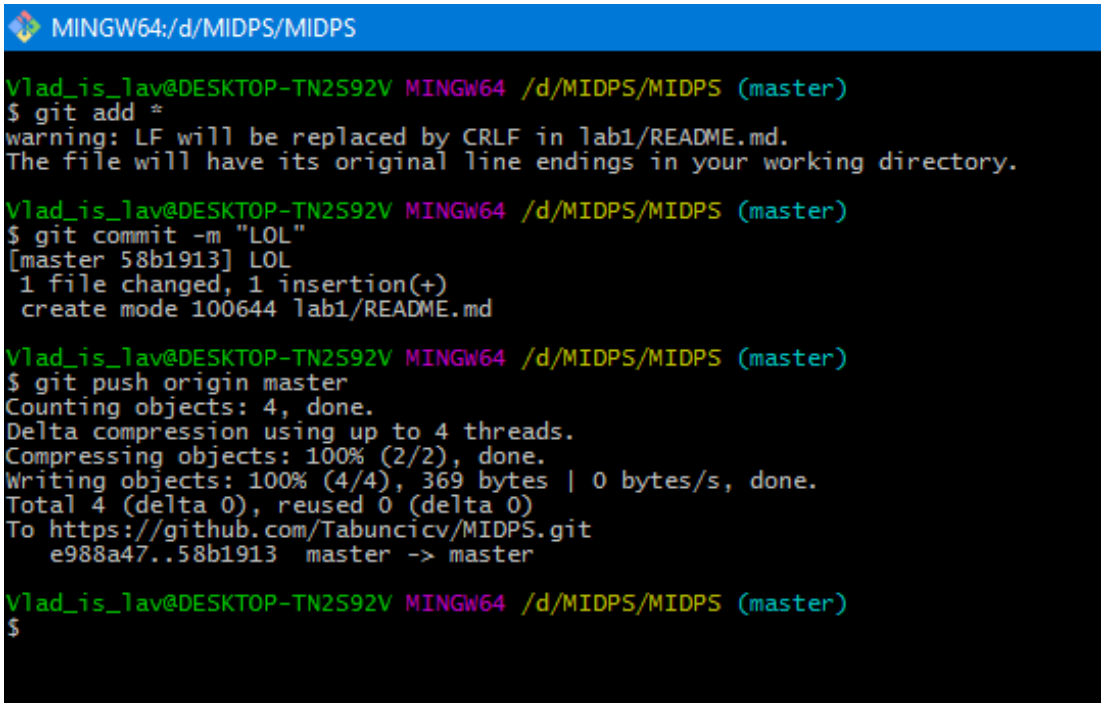


Urmatorul pas consta in generarea **SSH** key**.** Scriem **ssh-keygen,** iar cheia (publica) obtinuta o copiem in setarile noastre de pe github.com.

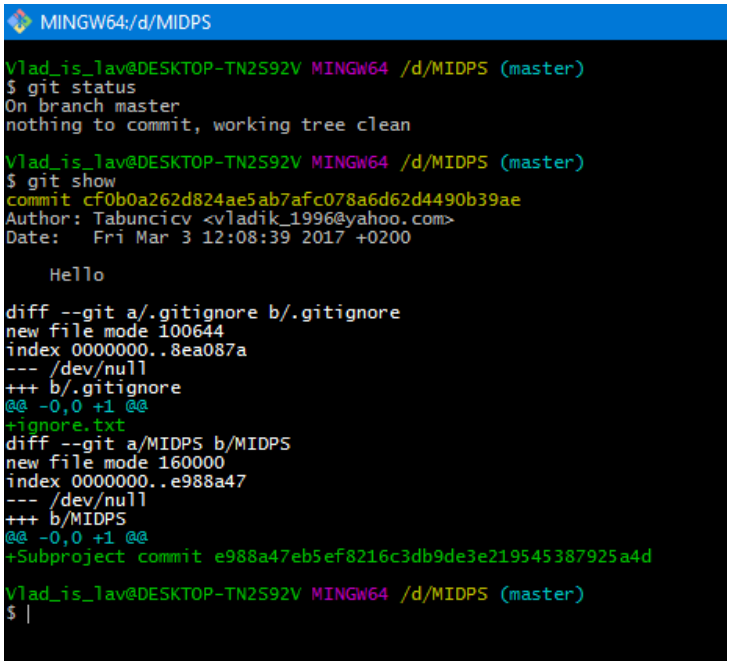
Cum e mentinut si in conditiile laboratorului, este de dorit sa initializam repozitorul nostru cu un fisier **README.md** si un **.gitignore.** In fisierul README.md vom adauga informatii pentru cei care se vor folosi de repozitoriu iar in fisierul .gitignore vom adauga toate fisierele ce trebuiesc ignorate (adica sa nu fie incarcate la moment).



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 "TEXT" – comanda face un snapshot la toate schimbarile noastre.
git push origin master – comanda incarca toate fisierele indexate pe **github.com**

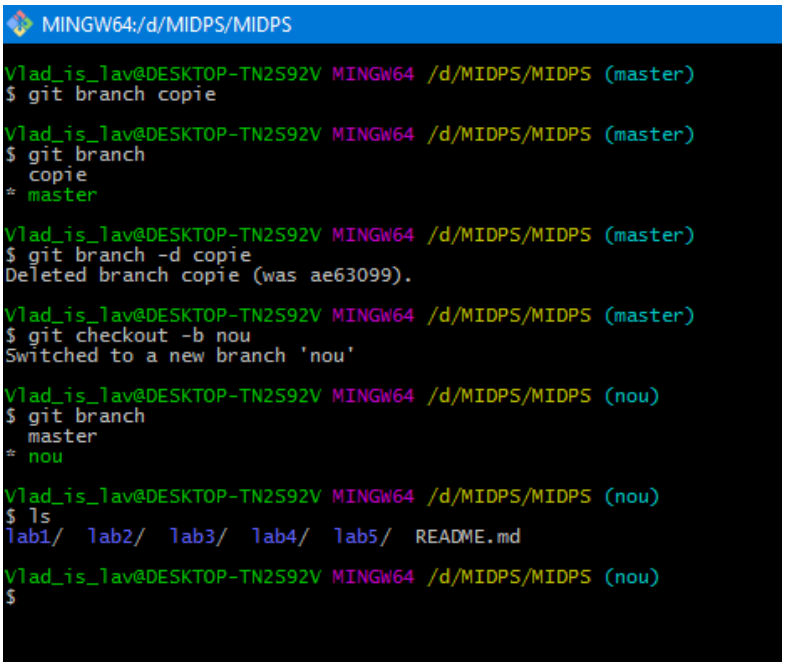


Pentru a ne asigura ca am facut totul bine si nu avem probleme utilizam urmatoarele comenzi git:
***git status**
***git show**

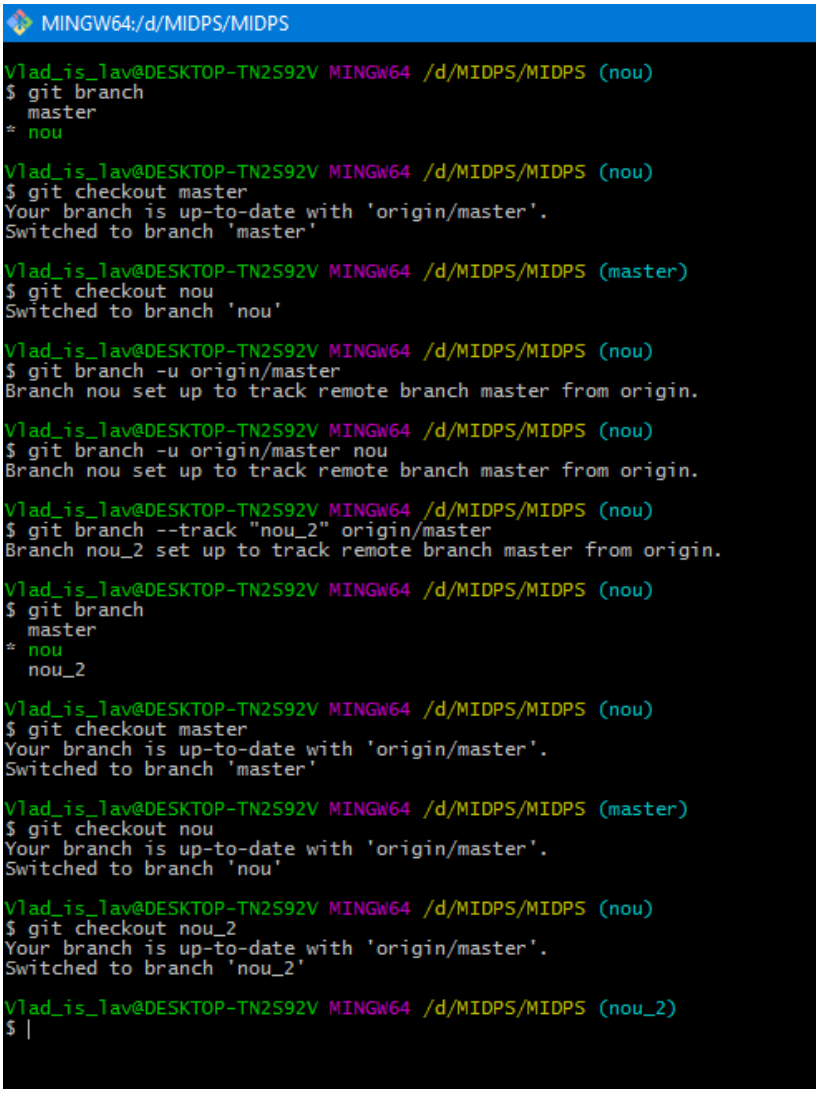


VCS ne permite sa avem mai multe **branchuri.** Din ENG branch semnifica “creanga”. Branch-urile sunt utilizate cind lucram paralel la un proiect si apoi dorim sa combinam toate modificarile.

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

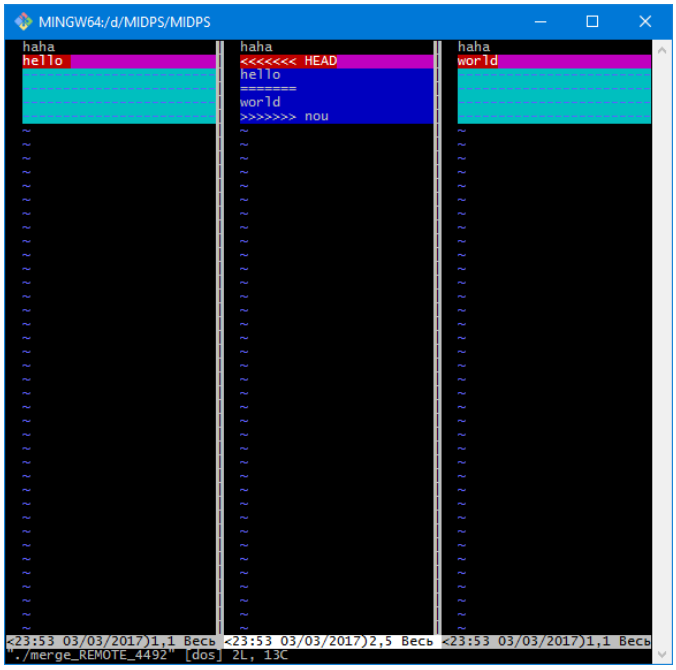


git checkout “name” – face switch la branchul “name”.
git branch –u upstream/name – face track la branchul indicat din branchul curent.
git branch –u upstream/name “name” – face track din branchul “name” la branchul indicat.
git branch –track “name” upstream/name – creeaza branchul “name” si ii face track la branchul indicat.
git branch –unset-upstream – scoate trackingul la branchul in care ne aflam.

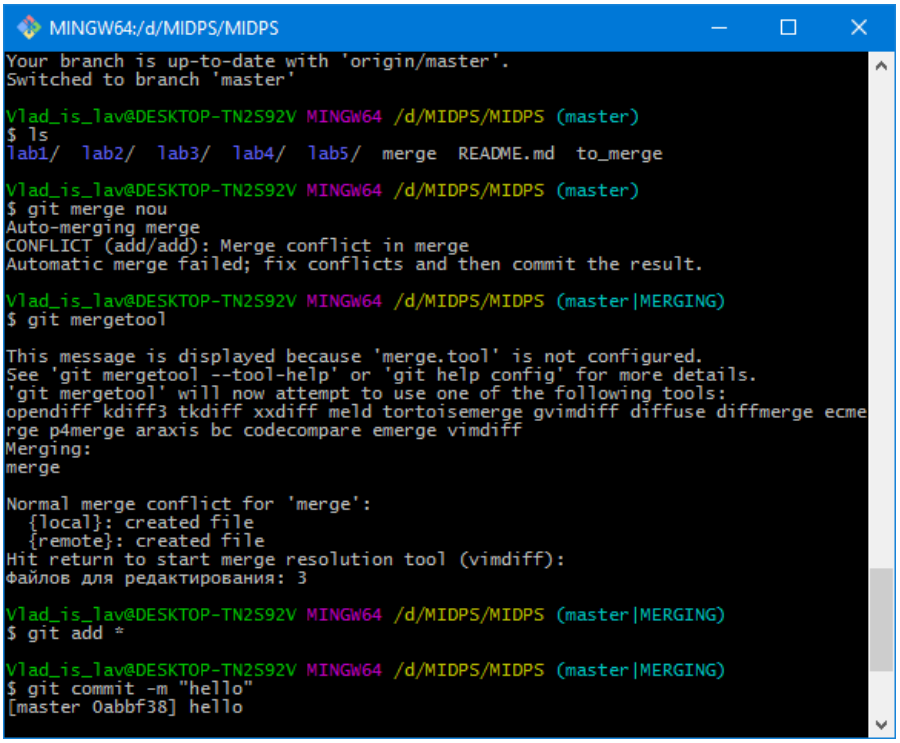


Pot aparea conflicte in cazul cind dorim sa facem **merge** la 2 branch-uri si unele rinduri sunt diferite. In asa caz,pentru a elimina conflictele, folosim **mergetool**. Drept mergetool am ales **kdiff3.** Pentru kdiff3, in mod implicit folosim comanda : **git config –global merge.tool kdiff3.**

In continuare vom lucra cu 2 branchuri – “master” si “nou”. Vom crea in fiecare branch cite un fisier “tomerge” continutul caruia va fi diferit.



In continuare facem merge si incercam sa rezolvam acest conflict cu ajutorul **kdiff3.**



4. Concluzie

In lucrarea nr.1 la MIDPS am studiat lucrul cu **VCS**. Am facut cunostinta cu o alta fata a **github-ului** (eu il facusem doar pentru freecodecamp, fara a-i studia functiile).
Toate comenzile git le-am indeplinit in terminal pe Windows. Fara VCS elaborarea si partajarea produselor soft ar fi foarte lenta si problematica. El ne permite lucrul paralel intre proiecte, menajarea versiunelor, revenire la versiuni anterioare. In lucrare am practicat majoritatea comenzilor esentiale. Companiile de success recomanda cunoasterea unui VCS, deci e si el parte din cheia succesului. El contribuie nu doar la dezvoltarea hard si soft-skillurilor dar si personala, in cazul meu. : In sfirsit am invatat sa folosesc Google, cautind multimea notiunilor si comenzilor mai variate decit cele de baza, acesta fiind si motivul pentru care am intirziat, dar cred ca e mult mai bine in comparatie cu devenirea unui (citez) "*bîdlo-coder*".

5. Referinte :

1.https://github.com/BestMujik/MIDPS-labs/blob/master/MIDPS_LAB%231.md

2. https://github.com/Ernest96/MIDPS/blob/master/LAB1/Lab%231.pdf

3. https://www.atlassian.com/git/tutorials/git-merge

4. http://acs.ase.ro/Media/Default/documents/cts/SeminarZamfiroiu/GIT.pdf

5. https://www.google.com/ //recomand ^_^