Sisteme de versionare

Prezentare GIT - linie de comanda si integrarea cu IDE-ul Eclipse

Prezentare realizata de Cioropina Gabriel

Obiective
Prezentarea isi propune insusirea comenzilor de baza a sistemului de versionare GIT, atat din linie de comanda, precum si integrarea cu IDE-ul Eclipse. (plugin folosit: Egit)

Sisteme de versionare – scurta introducere
Sistemele de versionare permit gestionarea versiunilor multiple a unor fisiere, precum si asigurarea lucrului colaborativ asupra acestor fisiere. Fiecare modificare va fi inregistrata continand atat diferentele *noii versiuni (revision)*, precum si autorul acestei noi versiuni.
Aceste sisteme au fost concepute pentru a permite membrilor mai multor echipe sa opereze modificari pe acelasi proiect, aceste modificari urmand a fi reunite intr-o noua versiune a proiectului.

Terminologie

- **repository** – componenta server ce contine informatii privind ierarhia de fisiere si reviziile
- **checkout** – preluarea in mediul local a unei anumite revizii publicate pe server
- **working copy** – versiunea locala a proiectului
- **commit** – cerere de publicare pe server a unor modificari
- **pull** – actiunea de actualizare (update) a informatiilor locale cu cele de pe server
- **conflict** – apare atunci cand mai multi utilizatori vor sa publice modificari aplicate acelorasi fisiere din proiect, insa sistemul de aplicare a versiunilor diferite nu poate imbina modificarile
- **revert** – revenirea la o versiune anterioara pe un anume fir de dezvoltare (branch)
- **branch** – ramuri secundare de dezvoltare a proiectului
- **tag** – branch “read-only” ce nu mai permite modificari ulterioare (folosit uneori pentru versiunile stabile si deriva dintr-un branch)

Mod de functionare
Pentru a fi optimizat procesul de stocare a modificarilor survenite de fisierele din proiect, in momentul in care aceste modificari sunt publicate, ele sunt stocate sub forma unui patch (diff) ce urmeaza a fi aplicat in noua versiune – procedeu numit *delta compression*.

Exemplu de diff intre 2 posibile versiuni:

```
gabriel ~/Documents/test 10:14 PM
> diff EntityUser.java EntityUserNew.java
9c9
< public class EntityUser {
---
> public class EntityUser implements Cloneable {
20a21,29
>
> /**
> * Clone card method.
> * @override
> */
> protected Object clone() throws CloneNotSupportedException {
> return super.clone();
> }
>
```

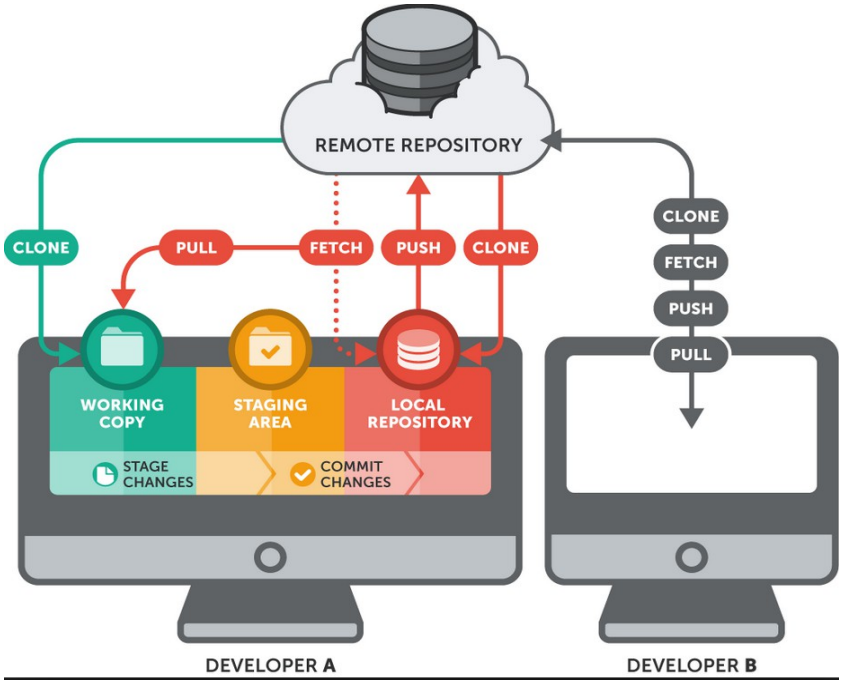
GIT – VCS

GIT este un sistem de versionare distribuit ce ruleaza, sub licenta GNU, pe majoritatea sistemelor de operare existente: **Linux, OSX, Windows**. Este ideal pentru cazul in care membrii unei echipe ce lucreaza la un proiect actioneaza intr-un mod oarecum independent; asadar, sistemul ofera o flexibilitate foarte extinsa in ceea ce priveste modurile de folosire.

Autorul lui este nimeni altul decat *Linus Torvalds*, parintele Linux, cel care l-a conceput in 2005 ca urmare a conflictului aparut cu *BitKeeper,* vechiul sistem de versionare folosit pentru kernelul de Linux.

Proiecte majore ce folosesc GIT: linux-kernel, debian, eclipse, Fedora, GNOME, GTK, rsync, Ruby on Rails.

Diagrama functionare GIT



Instalare

Linux (caz studiat: *Debian*):

apt-get install git

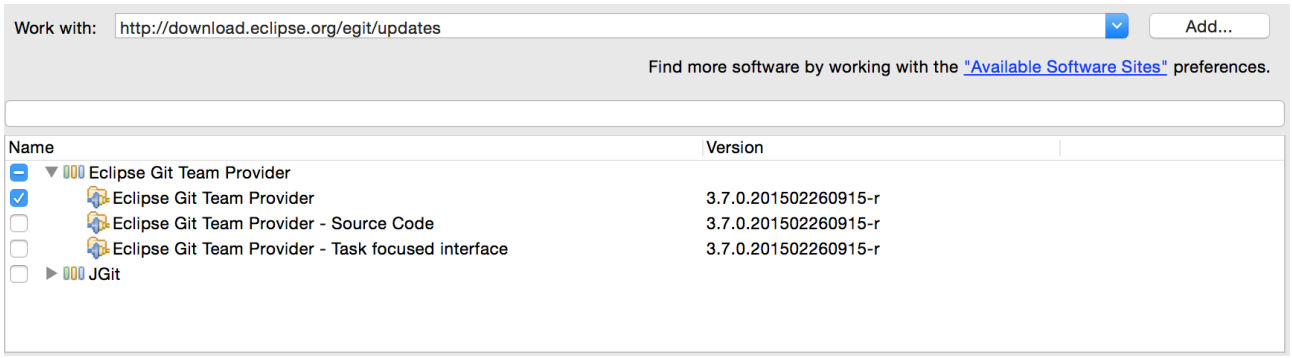
Windows / OSX: se poate descarca kit-ul de instalare de pe http://git-scm.com

Pe Windows, odata instalat git-scm, va fi pus la dispozitia utilizatorului atat un GUI pentru GIT, cat si o emulare a terminalului de Linux ce va permite executarea comenzilor din linia de comanda. Acest terminal este diferit de cel nativ Windows. (cmd)

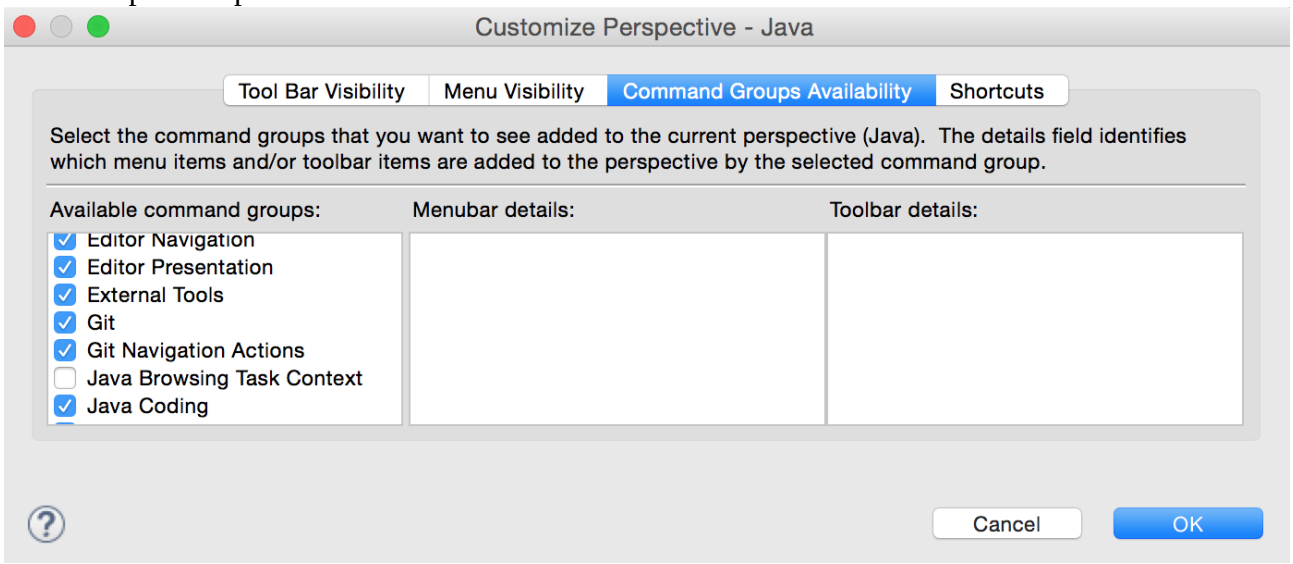
Instalare plugin Egit in Eclipse si configurarea lui

Din meniu: Help → Install New Software.

Se adauga repository-ul http://download.eclipse.org/egit/updates si se selecteaza pachetele ca in exemplul urmator:



Apoi, se va naviga catre Window → Customize Perspective → Command Groups Availability si se vor bifa optiunile pentru GIT:

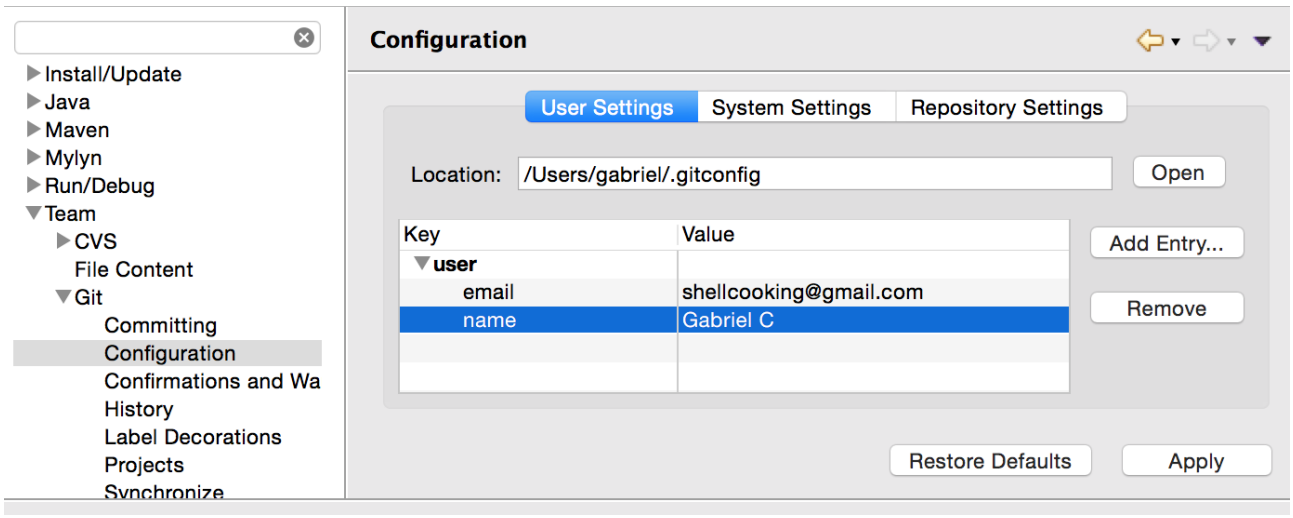


Identificarea utilizatorului curent

Comenzi linie de comanda:

- **`git config user.name "Nume Utilizator"`**
- **`git config user.email "gabriel@lazycoder.ro"`**

In Eclipse, pentru a seta detaliile utilizatorului curent, precum si alte optiuni, avem de parcurs urmatorii pasi:
Eclipse → Preferences → Team → GIT
De aici putem seta detaliile utilizatorului curent (username si email)



Ignorarea anumitor fisiere

Nu intotdeauna dorim sa versionam anumite fisiere, intrucat acestea pot contine executabile generate de procesul de compilare sau efectiv parametrii de configurare ai aplicatiei. (parole catre baze de date)
Fisierele sau directoarele ce se doresc a fi ignorate de sistemul de versionare pot fi trecute intr-un fisier numit *.gitignore*. Acesta este evaluat in mod recursiv, putand astfel avea mai multe fisiere .gitignore in folderele proiectului nostru; cu toate acestea, pentru a nu cauta foarte mult path-urile ignorate, se foloseste in mod conventional un singur astfel de fisier plasat in radacina proiectului.

Exemplu de fisier gitignore:

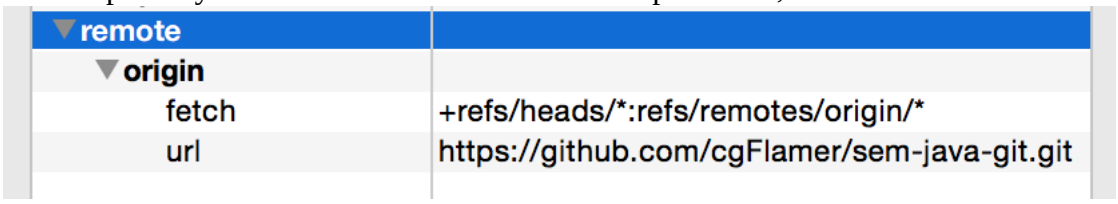
```
bin/ # va ignora toate fisierele din folderul bin
target/ # va ignora toate fisierele din folderul target
*.log # va ignora toate fisierele cu extensia .log
```

Initializarea si sincronizarea cu un repository public

Comenzi linie de comanda:

- **`git init`** (initializare workspace pentru GIT)
- **`git remote`** (asigura management-ul conexiunilor realizate cu repository-urile remote; comenzi aditionale: add, -v [listare], rm, rename)
- **`git clone <remote>`** (cloneaza un repository remote)
- **`git fetch`** (importa commit-urile de pe un repository remote in repository-ul local; commit-urile sunt stocate sub forma unor branch-uri remote)
- **`git pull <remote>`** (combina git fetch cu git merge)

Eclipse:
Click dreapte pe proiect, apoi Team → Share project → GIT. Alegem sa folosim sau sa cream un repository in directorul proiectului curent.
Apoi, Window → Show view → Other → Git Repositories si Git Staging. Vom dori sa modificam proprietatile repository-ului nostru local din tab-ul Git Repositories, creand urmatoarea structura:



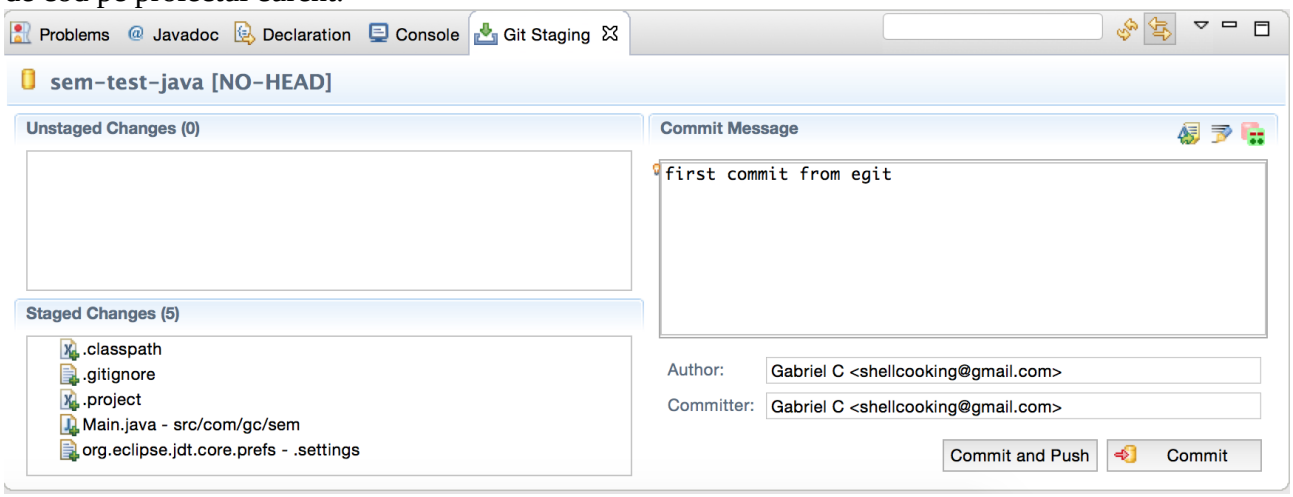
In cazul prezentat, link-ul repository-ului “origin” este https://github.com/cgFlamer/sem-java-git.git

“*Commit-erea*” si publicarea modificarilor

Comenzi linie de comanda:

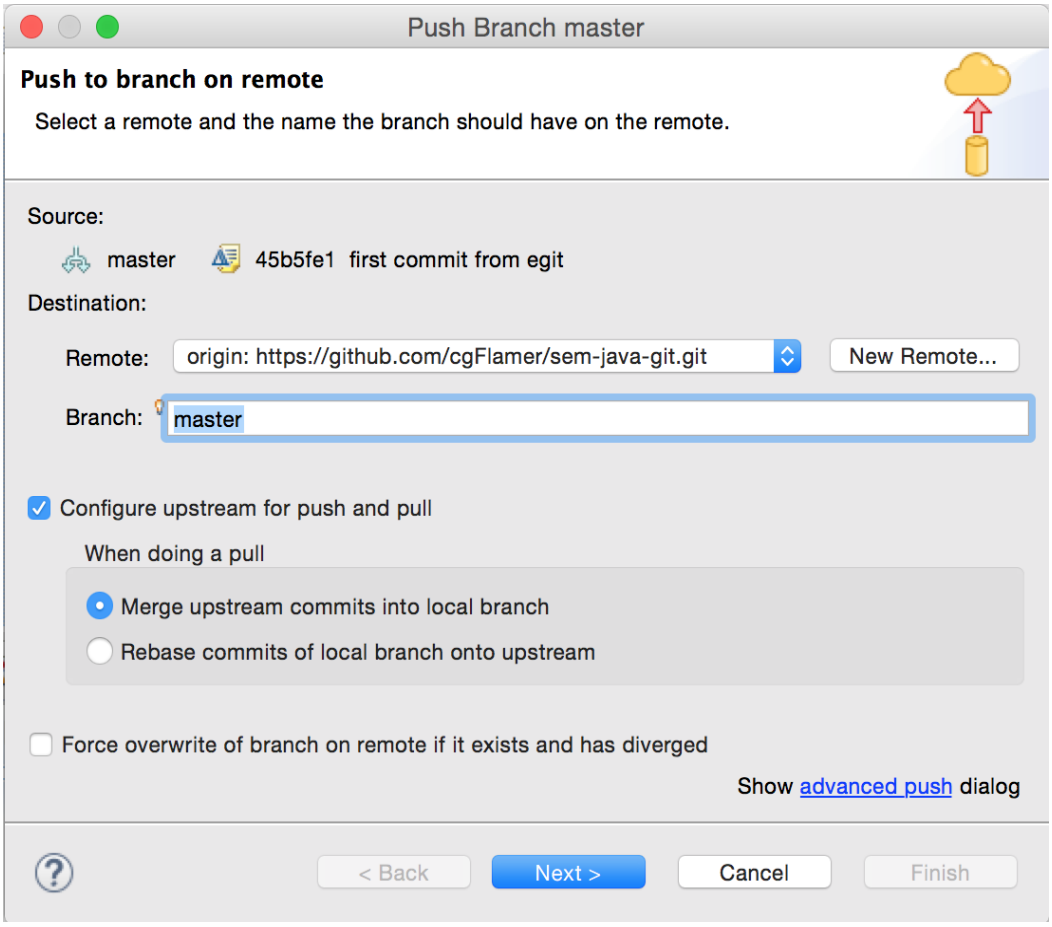
- **`git status`** (afiseaza o lista cu fisierele modificate fata de ultima versiune sincronizata)
- **`git add file1 file2 <...>`** (adaugarea de fisiere in cadrul versiunii ce se intentioneaza a fi publicata)
- **`git commit -m “mesaj”`** (demarcarea cu ajutorul unui mesaj a noi versiuni ce urmeaza a fi publicata; commit-ul va avea atasat o versiune a reviziei [un hash], precum si detaliile utilizatorului [identitatea lui])

In Eclipse vom urmari datele oferite de Git Staging, urmand a selecta precum in imaginea de mai jos modificarile pe care dorim sa le publicam. (trecerea cu drag& drop a fisierelor din “Unstaged changes” in “Staged changes”) In cazul de fata, modificarile vor consemna de altfel prima publicare de cod pe proiectul curent.



Pentru a publica pe repository-ul remote modificarile, vom actiona Commit and Push.

Optiunile de publicare se prezinta ca in imaginea de mai jos, noi putand in acest pas sa mentionam repository-ul unde vrem sa publicam, precum si branch-ul ales de noi.



La publicare, ne vor fi cerute credentialele noastre.

Diferentele dintre versiunile publicate

Comenzi linie de comanda:

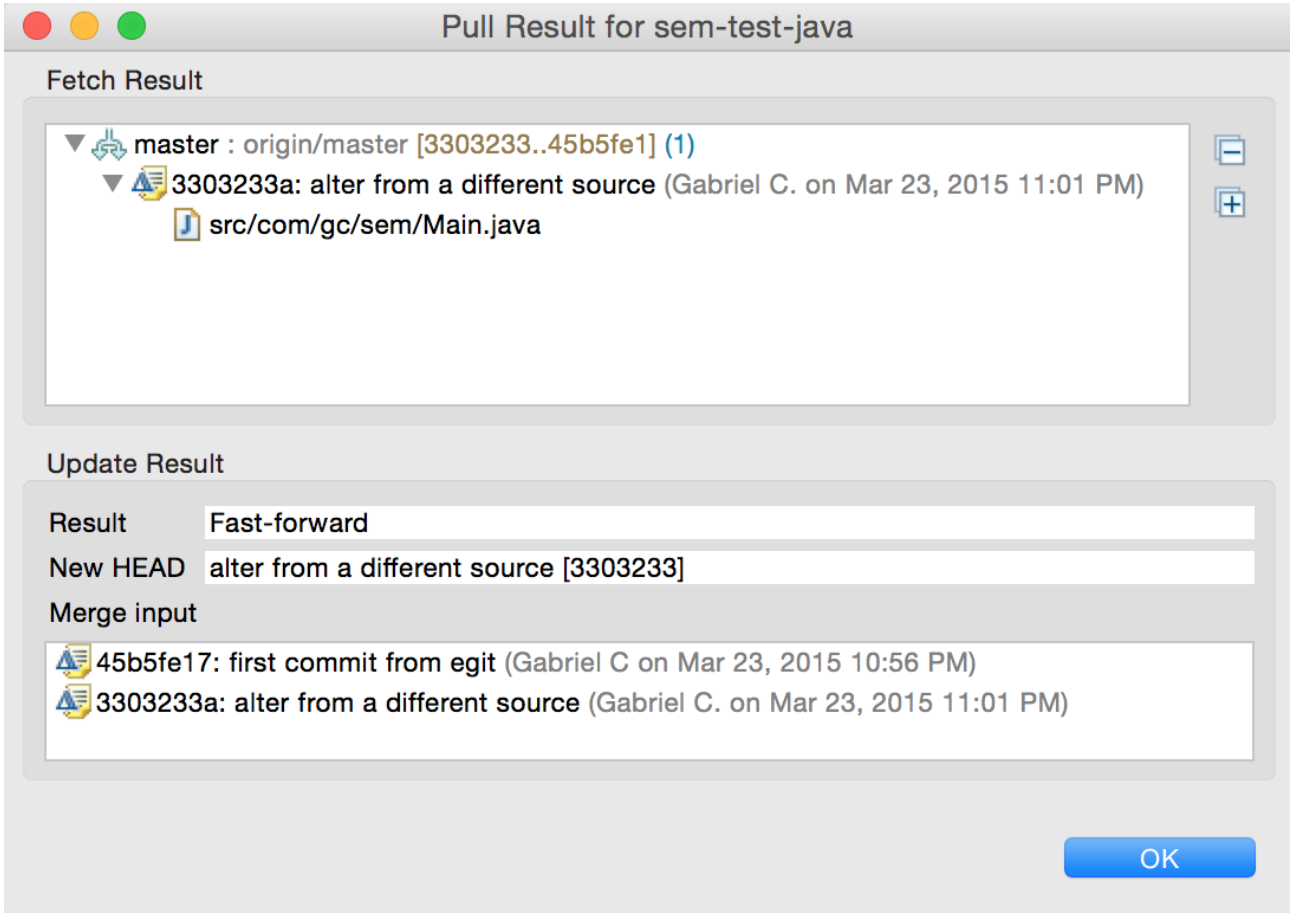
- **`git log <optional file1>`** (afiseaza versiunile publicate si sincronizate pe firul curent de versionare; daca se specifica un fisier, vor fi afisate toate reviziile ce au modificat acel fisier)
- **`git diff <optional file1>`** (arata diferentele inregistrate intre versiunea curenta si versiunea de la care s-a pornit, mai precis ultima revizie sincronizata)

Actualizarea versiunii curente a unui branch

Comenzi linie de comanda:

- **`git pull origin <branch>`**

Pentru a actualiza modificarile suferite de proiect pe un anumit branch folosindu-ne de Egit, va trebui sa actionam pe repository-ul afisat in perspectiva “Git repositories” optiunea “Pull”. Vom fi informati daca exista modificari ce nu au fost sincronizate pentru moment:



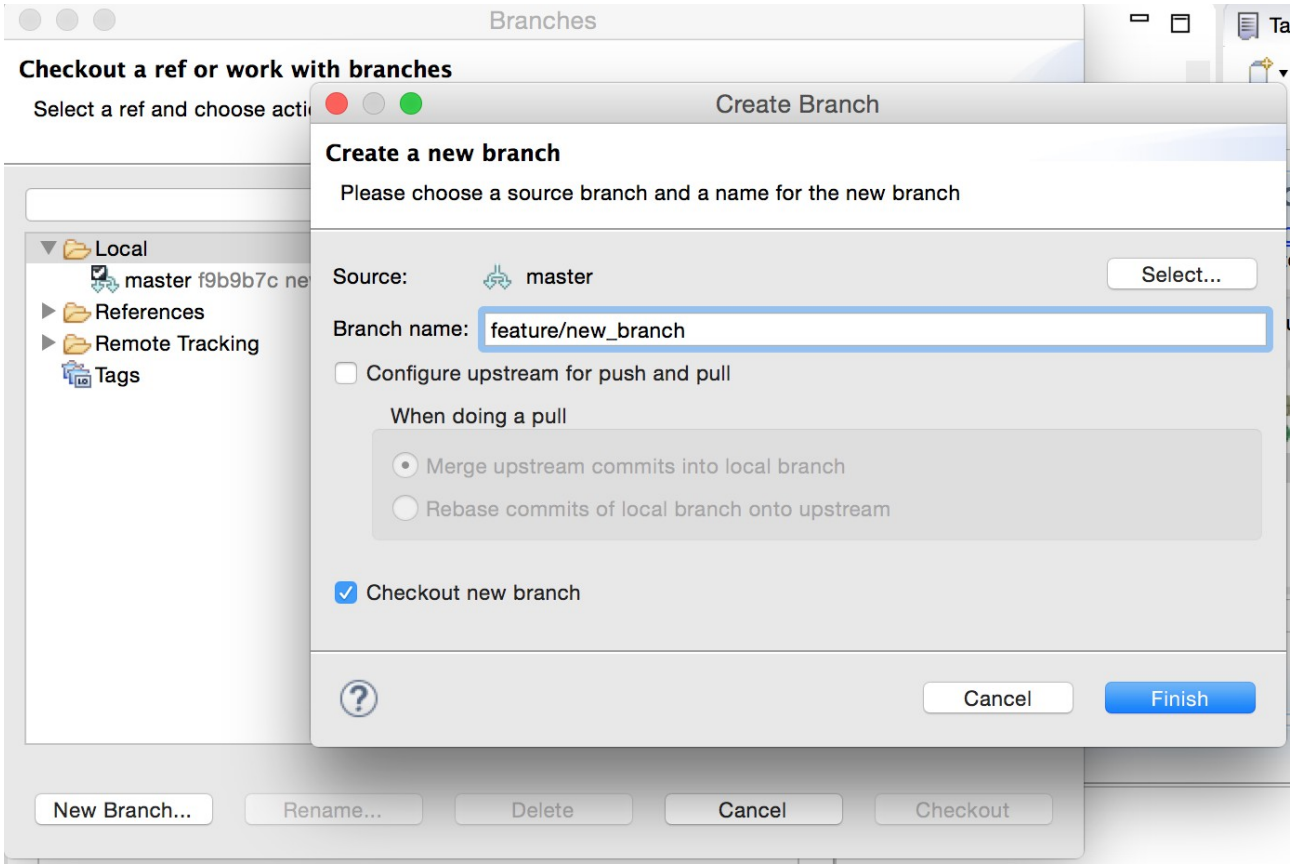
Codul acum este actualizat cu noua varianta publicata.

Migrarea intre branch-uri si lucrul cu acestea

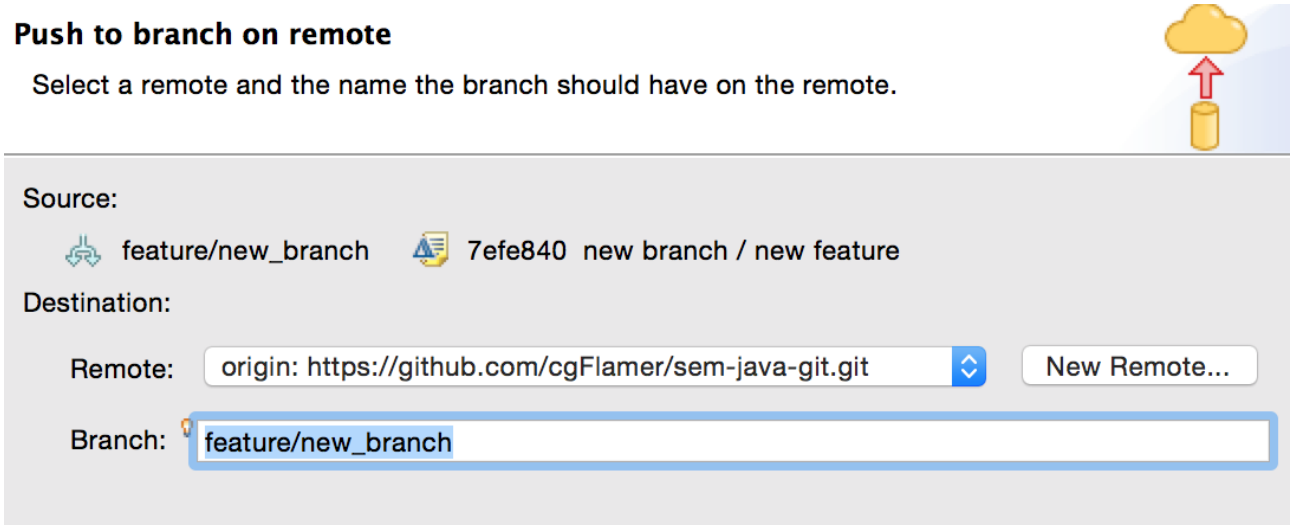
Comenzi linie de comanda:

- **`git branch`** (va afisa branch-urile locale, precum si branch-ul curent pe care ne aflam)
- **`git branch -a`** (va afisa atat branch-urile locale, precum si branch-urile de pe repository)
- **`git checkout <branch_local>`** (va schimba branch-ul curent in cel specificat, migrandu-se astfel pe un nou fir al versionarii)
- **`git checkout -b <branch>`** sau **`git branch <branch>`** (va crea un branch nou pe baza celui curent)
- **`git checkout -b <branch> <sursa>`** (va crea un branch nou ce va fi sincronizat cu un branch-sursa, din repository)
- **`git branch -D <branch>`** sau **`git branch -d <branch>`** (va sterge branch-ul de pe repository, cat si local; parametrul “d” nu va permite aceasta operatiune daca exista modificari ce nu au fost merge-uite, pe cand “D” forteaza operatiunea)

In Eclipse, crearea unui branch si publicarea codului pe aceasta noua “ramura” este foarte simpla, totul putand fi accesat din meniul GIT → Switch to. Apoi vom alege un nou branch creat in repository-ul local, avand ca punct de referinta (plecare) unul din branch-urile sincronizate.



Vom putea acum crea un nou pachet pentru aplicatia noastra, urmand a-l publica pe acest branch nou.



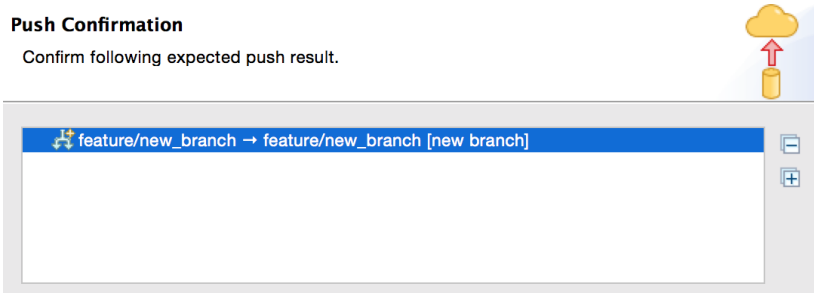
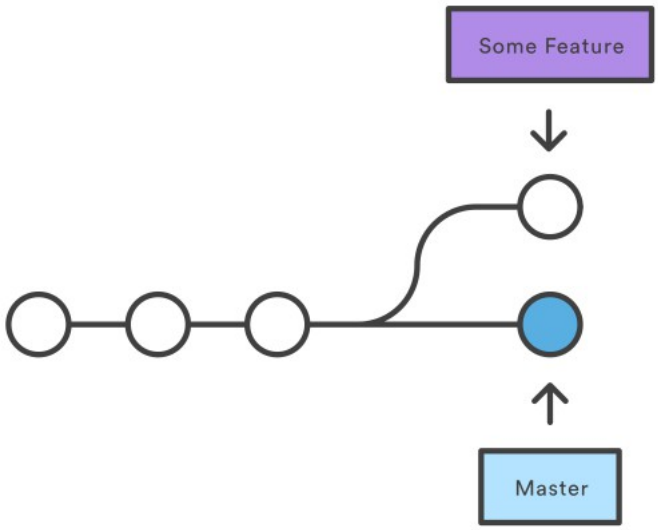


Diagrama exemplificare creare a unui branch nou:

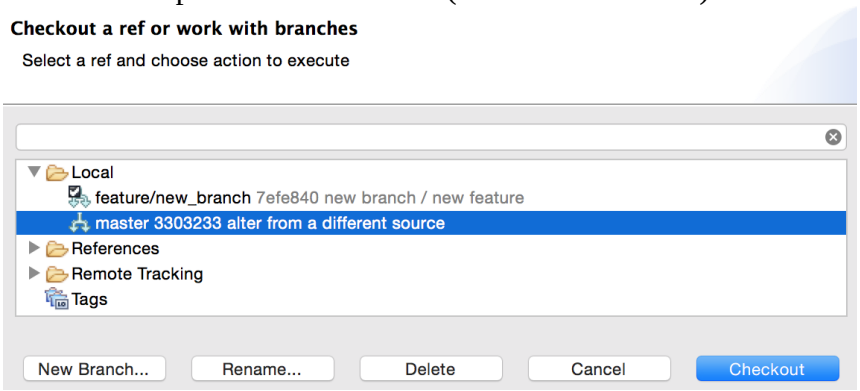


Unirea a doua branch-uri

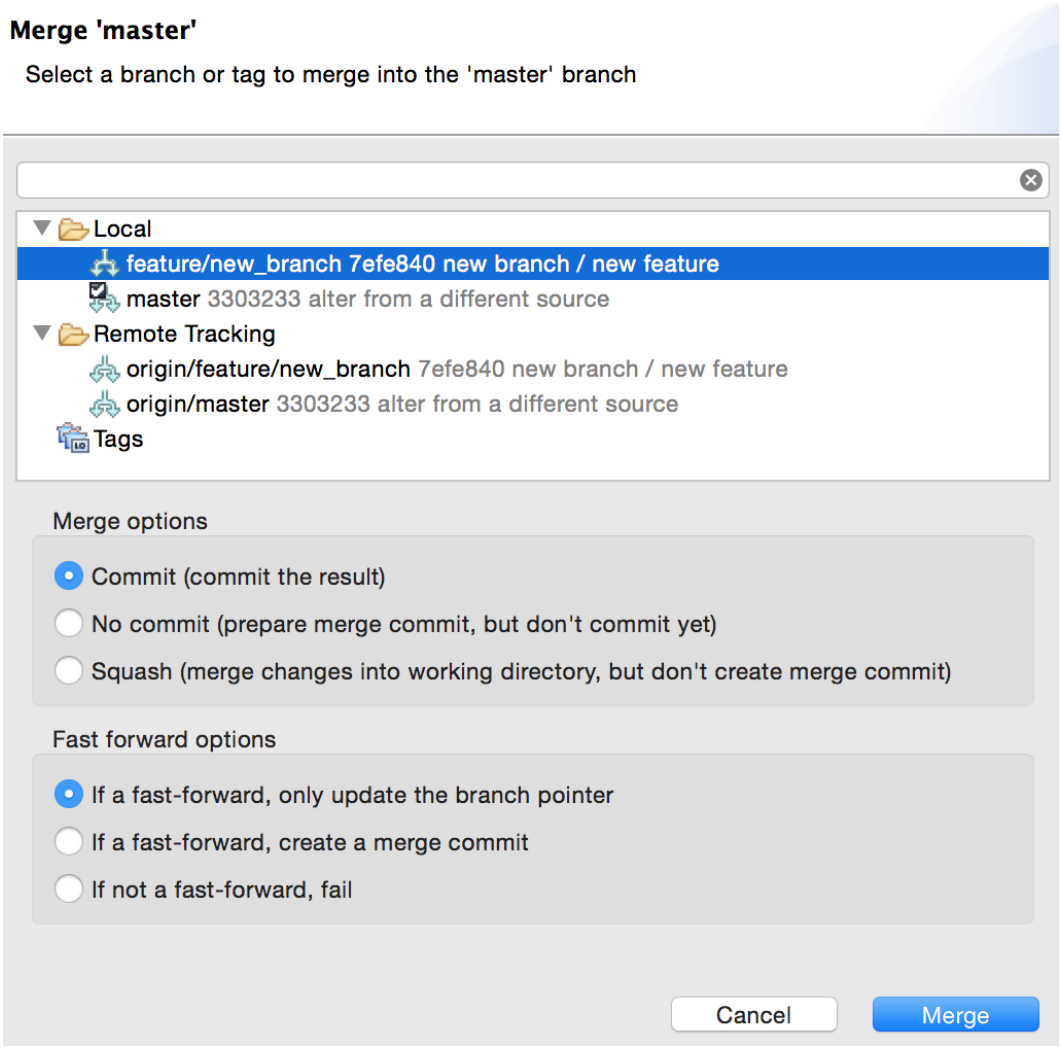
Comenzi linie de comanda:

- **`git merge`** **<branch_ce_va_fi_aplicat>** (va aplica modificarile din branch-ul specificat branch-ului curent, daca acestea exista; branch-ul de unit poate fi atat local, cat si remote, deci nesincronizat local)

In Eclipse, sa facem checkout pe branch-ul master: (Git → Switch to…)



Apoi, din acelasi meniu, alegem unul din punctele de referinta (branch-uri) pe care vrem sa le unim in branch-ul master:



Daca procesul de merging functioneaza corect si nu exista conflicte, putem publica modificarile. Altfel, in cazul existentei conflictelor, acestea vor fi rezolvate local, iar modificarile vor fi atasate unui nou commit.
Publicarea merge-ului se face folosind meniul GIT → Push to upstream.

Diagrama exemplificare proces de unire (*merging*):

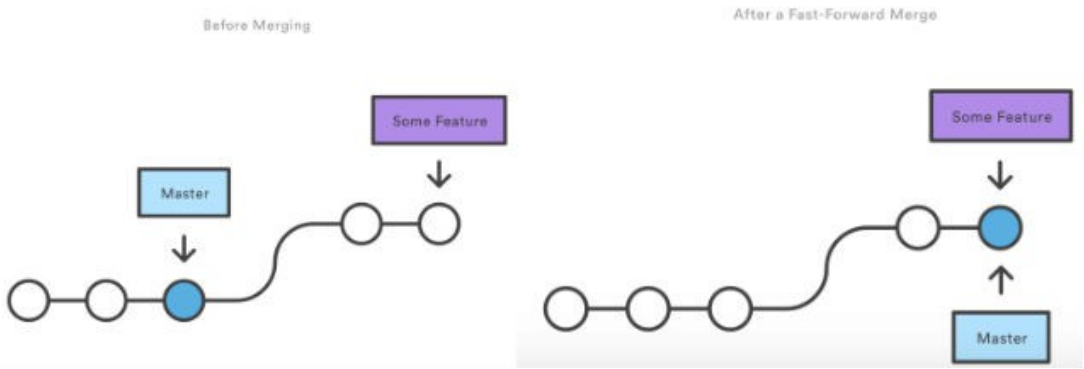
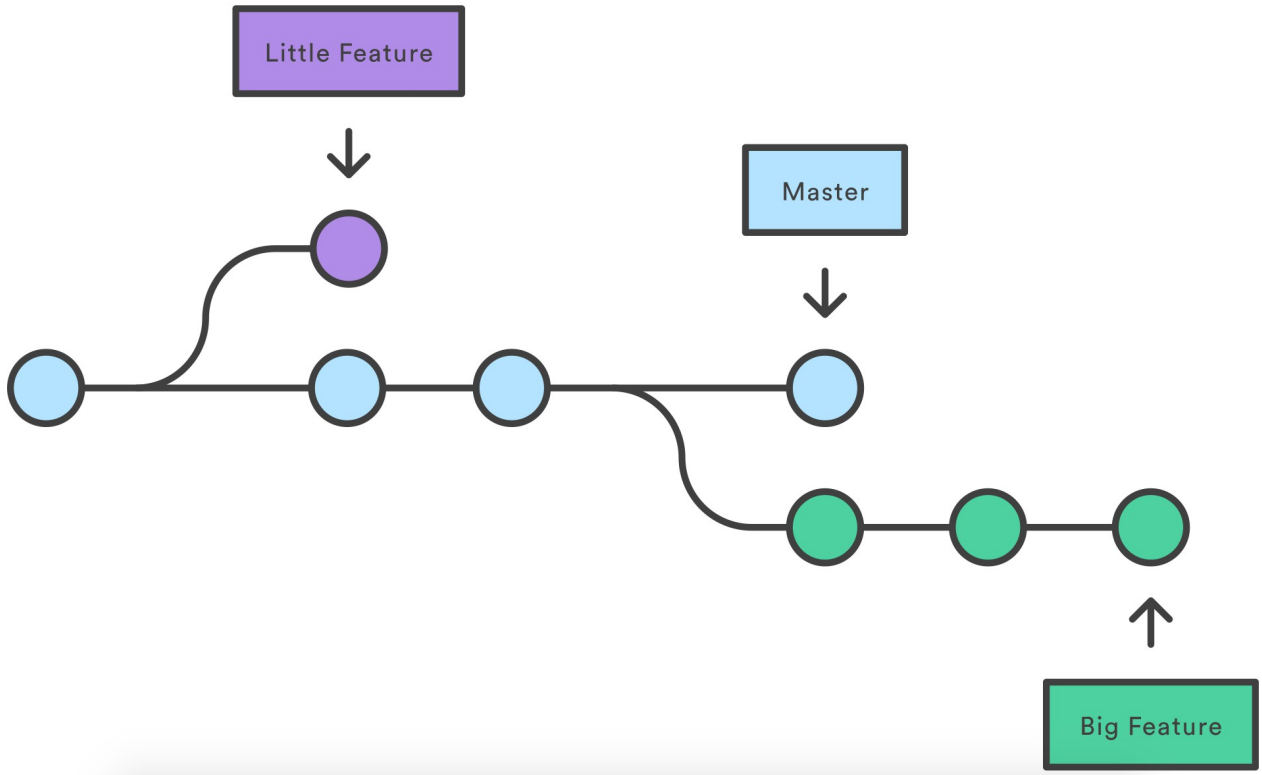


Diagrama uzuala a activitatii de versionare



Modele de folosire a versionarii

- Versionare locala

Face tracking doar in mediul local a versiunilor unui proiect. Practic, procesul este asemanator celui descris mai sus, insa singura diferenta consta in faptul ca modificarile / versiunile nu sunt publicate intr-un repository remote, impartit cu alti potentiali utilizatori.

- Versionare remote

In lucrul cu GIT, se folosesc in mod absolut uzual diferite tehnici de management al branch-urilor. Fiecare astfel de model de versionare cauta sa suplineasca anumite nevoi, putand astfel ca aceste workflow-uri extrem de raspandite sa fie modificate ad-hoc in functie de necesitatile proiectului.

Dintre cele mai cunoscute:
- workflow centralizat
- feature branch
- gitflow (master + develop | feature branches)
- forking workflow

Resurse utile

http://git-scm.com/book/en/v2/Getting-Started-About-Version-Control
http://www.vogella.com/tutorials/EclipseGit/article.html