

Sisteme de versionare software

• • •

George Popa
Gemini Solutions

Sisteme de versionare - *version control systems (VCS)*

Definiție

- Instrument de dezvoltare software utilizat în gestionarea multiplelor versiuni ale fișierelor și dependențelor unei aplicații, înregistrând toate stările acestora, inclusiv modificări, autori și comentarii privind fiecare modificare.

Necesitatea unui sistem de versionare al proiectului

- Securitate - *repository* securizat, backup;
- Lucrul în echipă - permite colaborarea prin distribuirea modificărilor;
- Istoria proiectului - permite revizuirea stărilor anterioare ale proiectului;
- Integrare cu project trackers - modificările și comentariile vor apărea în tracker.

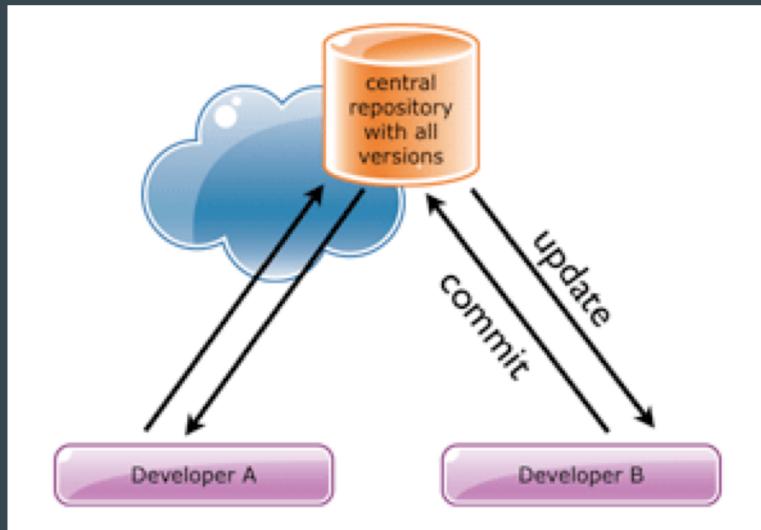
Exemple de sisteme de versionare

- Apache Subversion (SVN) - <https://subversion.apache.org/>
- CVS - <http://www.nongnu.org/cvs/>
- Git - <https://git-scm.com/>
- Mercurial - <https://www.mercurial-scm.org/>
- Perforce - <http://www.perforce.com/>

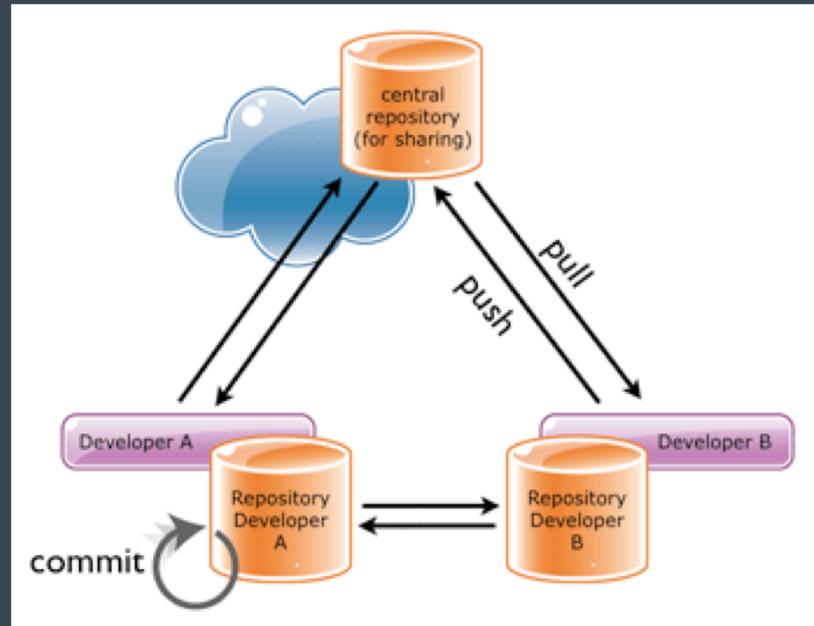
Concepțe VCS

- **Repository** - server de fișiere (bază de date) unde sunt stocate datele proiectului software
- **Working copy** - versiunea curentă a proiectului
- **Commit** - modificări efectuate asupra unor fișiere, publicate în *repository*
- **Revision** - versiune a unui fișier, ca urmare a unui commit
- **Branch** - copie separată a proiectului, ce poate conține modificări individuale
- **Merge** - operație de combinare a modificărilor din *branch-uri* separate
- **Checkout / fetch / pull** - operațiunea de descărcare modificări de pe *repository*
- **Push** - încărcarea modificărilor de pe un *repository* local pe un *repository* remote.

Arhitecturi ale sistemelor de versionare



Arhitectura nedistribuită
(exemplu: SVN)



Arhitectura distribuită
(exemplu: Git)



git – Source Control Management

Sistem open source, descentralizat, dezvoltat de Linus Torvalds (autorul Linux kernel).

Particularități:

- Salveaza patches (diferente) pentru fiecare commit
- Local branches folosite intensiv
- Operatiuni de merge, rebase, fork mult mai facile decat in SVN
- Repository principal poate fi schimbat
- Fork & pull request

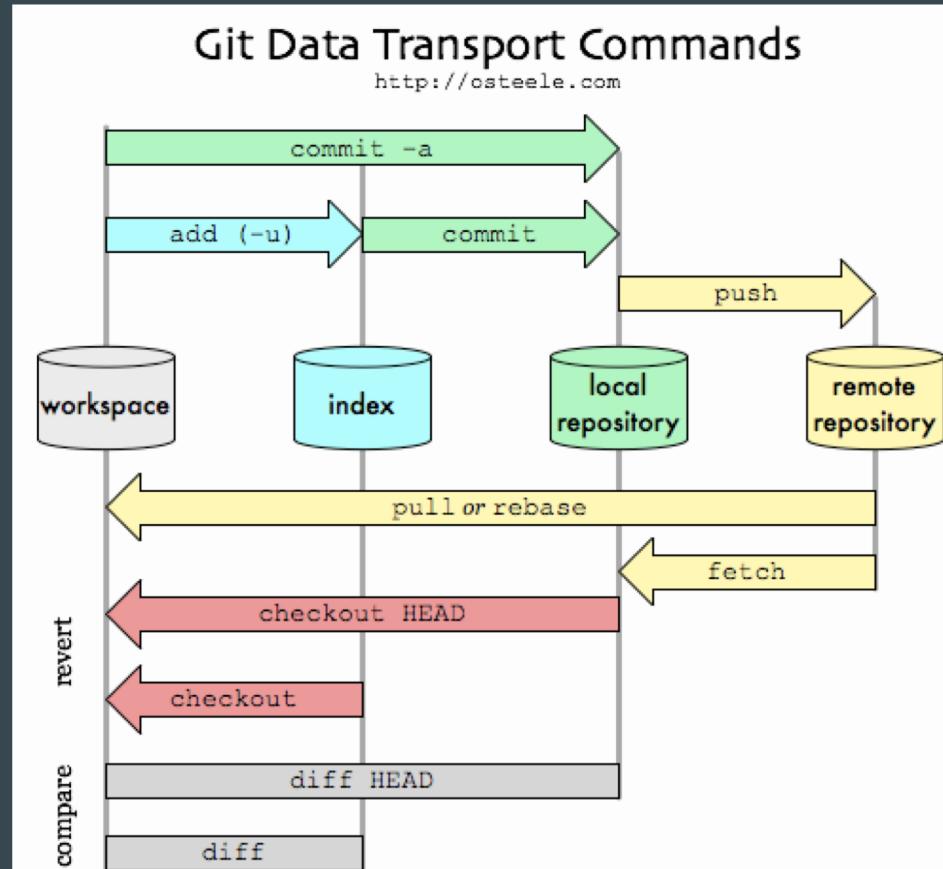
Documentatie: <https://git-scm.com/doc>

Git – Source Control Management

Comenzi uzuale:

- git config user.name / user.email
- git init
- git add [file1, file2...]
- git commit -m “First commit”
- git remote add origin url.git
- git push origin master

- git clone url.git
- git pull origin master
- git rebase origin master



Git GUI tools – gitk

<https://git-scm.com/downloads/guis>

The screenshot shows the gitk graphical interface for Git. The main window displays a commit history for the 'tube' repository. The commits are listed in chronological order from top to bottom:

- Local uncommitted changes, not checked in to index
- master-sync** Merge branch 'master' of github.com:ideo/tube into master-sync
- remotes/origin/master** Hot patch to old production code. Needed way to download headshots by
- remotes/origin/master-sync** [Changes for #106759546] Fix projects read, create, update from p
- [Changes for #106759546] Fix projects read, create, update from p
- master** Merge tag '4.4.0_RC4' - Putting new Resources pages into production.
- 4.4.0_RC4** [Fix for #91701744] Invalidate URLs starting with mailto and ftp (for now)
- Small fix for section height computing when editing a section
- Revert for Gemfile.lock
- Updated UAT deployment to point to 4.4.0_RC3
- 4.4.0 RC3** [Changes for #92925588] Limit the number of characters a user can enter for the re

Below the commit list, there is a search bar with the SHA1 ID: `dc0dfdecc2632a0a5cae5241d1eae61876c6197`. The interface also includes a 'Find' button, a 'commit containing:' dropdown, and a 'Search' input field.

On the right side of the commit list, there is a detailed view of the commit for '4.4.0 RC3'. It shows the author and committer information, the parent and child commit IDs, branches, follows, and precedes relationships. It also lists the files affected by the commit:

- Comments
- app/assets/javascripts/resource_entries.js
- app/assets/stylesheets/resource_entries.css.scss

At the bottom of the interface, a diff viewer shows the changes made in the file `app/assets/javascripts/resource_entries.js` between the commit `index 423d322..c05ae37 100644` and the previous commit. The diff highlights the addition of a function `removeResourceInfoTooltip()` and the modification of the `setSectionHeight` function.

```
----- app/assets/javascripts/resource_entries.js -----
index 423d322..c05ae37 100644
@@ -552,11 +552,10 @@ function removeResourceInfoTooltip() {
}

function setSectionHeight(section) {
- if ($(section).find(".row-1").length) {
+ if ($(section).find(".row-1").length || $(section).hasClass("table")) {
    var element = $(section).find(".section-container");
    var rowHeight = $(section).find(".row-1").outerHeight();
    var height = $(element).find("ul.resource-list").height();
-   height = $(section).hasClass("table") && $(section).hasClass("active") ? height + ro
    if ($(section).hasClass("active"))
      $(element).height(height);
  }
}
```

Getting started!

(part 1)

Creați-vă un cont pe BitBucket (<http://bitbucket.org>), iar un membru al echipei (Administrator) poate crea un proiect nou.

Configurați git local:

- `git config --global user.name john-doe`
- `git config --global user.email johndoe@example.com`
- Comunicați `user.name` către repository Administrator pentru a vă acorda acces la proiect.

Descărcați proiectul de test:

- `git clone https://bitbucket.org/\[administrator\]/project.git`
- `git status`



Getting started!

(part 2)

Creați un fișier nou în cadrul proiectului:

- touch nume_prenume.txt
- git status

Urcați fișierul nou creat pe *repository* bitbucket.com și descărcați fișierele colegilor:

- git add nume_prenume.txt
- git commit -m “Upload nume_prenume.txt”
- git pull origin master
- git push origin master
- git status



Git conflicts

- Inevitabile in munca in echipa!
- Apar de regulă când sunt modificări simultane în aceleași linii de cod.
- Nu există tool-uri automate care să rezolve toate conflictele.
- Rezolvarea conflictelor implică decizii luate de echipă, specifice logicii aplicației.

The image shows a Java code editor window titled "TestClass.java" and a "Merge Conflicts Resolver" dialog box.

TestClass.java:

```
package com.sap.text;

public class TestClass {

    <<<<< OURS
    void doSomethingElse() {
    =====
        void doSomething() {
    >>>>> THEIRS

    }
}
```

Merge Conflicts Resolver:

Conflict: 1 of 1, Unresolved: 1

Local Working File	Remote file
13 /** 14 * @param args the command line arguments 15 */ 16 public static void main(String[] args) { 17 // conflict 1 18* } 19 }	13 /** 14 * @param args the command line arguments 15 */ 16 public static void main(String[] args) { 17 // conflict 2 18* } 19 }

Buttons: Accept, Accept & Next, OK, Cancel, Help

Getting started!

(part 3)

- În echipe de câte două persoane, modificați aceleasi linii din acelasi fisier:
 - vi george.popa.txt
- Comiteți modificările create și apoi efectuați *pull* de pe <https://bitbucket.com>
 - :
 - git add george.popa.txt
 - git commit -m "Rename & repair greeting"
 - git pull origin master
 - **### CONFLICT ### TODO: resolve!**
 - git add george.popa.txt
 - git commit -m "Resolve conflict"
 - git push origin master



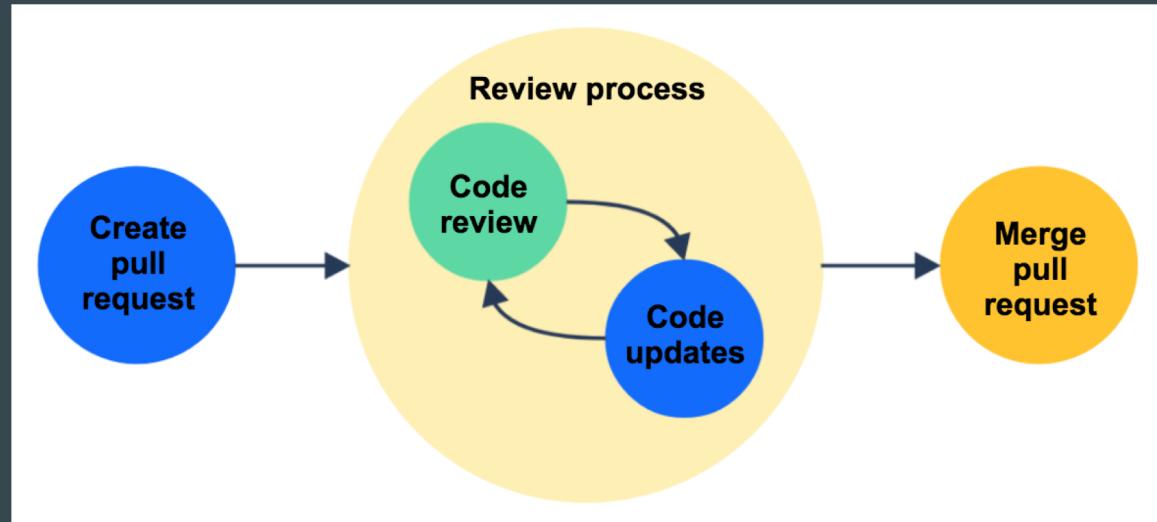
Git branches



- Utilizatorii git lucrează deregulă pe multiple branch-uri locale;
- Operatii cu branch-uri:
 - git checkout -b new_feature
 - git rebase master
 - git checkout master
 - git merge new_feature
 - git push origin master
- Pull requests

Code review

GitHub ofera
functionalitatea
Pull requests



- permite informarea echipei ca exista o noua functionalitate / bug fix, care asteapta a fi integrat in *master*;
- accepta feedback si permite aplicarea de modificari asupra commit-ului initial, inainte de integrare;
- integrarea cu alte instrumente (CI builds, etc...), permite validarea unitara, functionala sau a altor metriki inainte de integrare.

Code review

Gerrit ofera posibilitatea acordarii de punctaj atat instrumentelor automate CI, cat si review-erilor.

De asemenea poate conditiona operatiunea de integrare in functie de:

- validarii testelor automate (unitare sau de integrare);
- conditii de dependenta fata de alte componente;
- scoring pe baza nivelelor de acces sau experientei review-erului.

The screenshot shows a Gerrit code review interface. At the top, there's a navigation bar with tabs: All, My, Projects, People, Documentation, Changes, Drafts, Draft Comments, Edits, Watched Changes, Starred Changes, and Groups. The 'My' tab is selected. Below the navigation is a header for 'Change 1017 - Needs Quality-Assurance' with a 'Reply...' button. A 'Change-ID' field contains 'I93dae94ba1961bc9cd5a7fed4ae3b46ad5917adc'. To the right, there's an 'Add...' button and a 'Cannot Merge' message with a gear icon. A summary section includes 'Branch: develop', 'Topic', 'Strategy: Merge if Necessary', and 'Updated 90 minutes ago'. Below this are buttons for 'Cherry Pick', 'Rebase', 'Abandon', and 'Follow-Up'. A horizontal line separates this from a 'Code-Review' section. Under 'Code-Review', there are status indicators: 'Continuous-Integration +1' (green), 'Continuous Integration' (grey), and 'Quality-Assurance' (grey). The main content area shows commit details: Author, Committer, Commit hash, Parent(s) hash, and a note 'Char Not current - rebase possible'. It also shows file lists, an 'Open All' button, a 'Diff against: Base' dropdown, and an 'Edit' button. At the bottom, a 'Comments Size' table shows two rows: one with value '1' and another with value '8', each accompanied by a green progress bar.