Curs GIT, Virtualizare, Containerizare, Jenkins

Curs 3. Git1

Coordonator:
dr. conf. ing. Serban Obreja

Autori:
ing. Cosmin Cimpoeru, ing. Ciprian Chende

Versiuni

Versiune	Cine	Data	Descriere schimbare
v3	Ciprian Chende	2023/03/14	Adaugare extra explicatii la branching. Adaugare ‘fork’ la Lab
v2	Ciprian Chende	2023/02/25	Completat continut. Varianta completa.
v1	Ciprian Chende	2023/02/18	Cuprins si continut

Curs 3
Git 1

Curs 3 GIT 1

- Configuratie necesara pentru curs
- Sisteme de versionare. De ce?
- Sisteme de versionare. Exemple.
- Sisteme de versionare. Reprezentare schematica.
- Git – de ce GIT?
- Git local pe PC-ul de dezvoltare
  - Adaugare director de lucru ‘in / sub’ git.
    Comenzi `git: add, commit, log, diff, help`
- Comenzi Git de baza – reprezentare schematica
- Prezentare GitHub – link, cont, nume repository
- Cum clonez un repository GIT
- Cum pun pe GIT codul la care lucrez
- Flux standard de dezvoltare
  - Creez directoare / fisiere si le adaug pe GIT.
    `git pull, git push`

Configuratie necesara pentru curs

- Laptop cu Linux sau masina virtuala linux
- Acces la internet pentru instalare
- Cont GitHub: `https://github.com`
- Editor cod: Linux: vi / vim, gedit (`www.gedit.org`) (se pot instala cu apt sau din Synaptic),
  Windows: notepad, notepad++ (Windows)
  Linux, Windows, Mac:
  Visual Studio Code: `https://code.visualstudio.com`
  (Download fisier `.deb`; Instalare `cu apt install <fisier>`)

Sisteme de versionare (de ce)

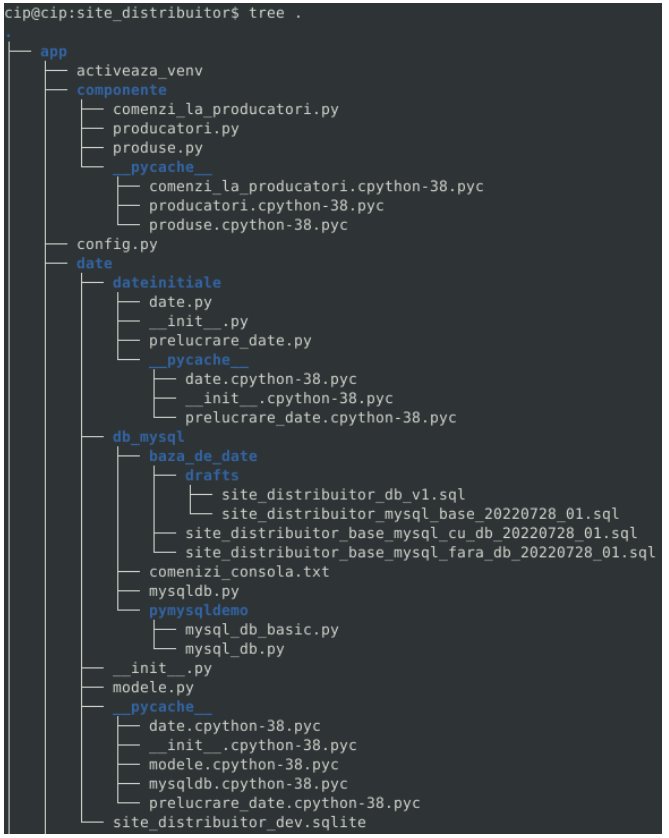
- Dezvoltarea aplicatiilor software presupune:
  - Organizare cod in mai multe fisiere si directoare
  - Mai multi dezvoltatori care lucreaza simultan la aplicatie

    (pentru aplicatiile mici – un dezvoltator, un fisier nu este nevoie de sistem de versionare. Programatorul poate tine evidenta evidenta schimbarilr singur, atasand la nume v1 / v2 / data modificare etc.
    Cand proiectul este mare si daca lucreaza la el mai multi dezvoltatori, practic, nu se poate lucra fara ajutorul unui sisteme de versionare)
- Pentru proiecte cu mai multe fisiere si directoare si atunci cand mai multi dezvoltarori lucreaza la un proiect, este nevoie de un sistem care:
  - sa inregistreze modificarile aduse de fiecare utilizator
  - sa permita revenirea la o versiune anterioara
  - sa permita ca mai multi dezvoltatori sa dezvolte functionatilati aplicatie in paralel, fara sa se afecteze reciproc
  - sa se poata face ‘review’ de cod (modificarile facute de un programator sa poate fi verificate de)
  - sau, punand lucrurile sub o alta forma: **sa permita pratajare de resurse si cloaborarea**
- Acronime: SCM – Source Code Management (folosit in Jenkins) / SV - Sistem de versionare

Sisteme de versionare (de ce) (exemplu)

- Exemplu cod aplicatie web simpla – site_distribuitor
- Aplicatia contine mai multe directoare si fisiere
- Chiar si in cazul in care un singur dezvoltator lucreaza la proiect, este greu sa se gestioneze codul fara un sistem de versionare

NOTA: De observat directorarele __pycache__, generate automat pe care nu vrem sa le gestioneze sistemul de versionare

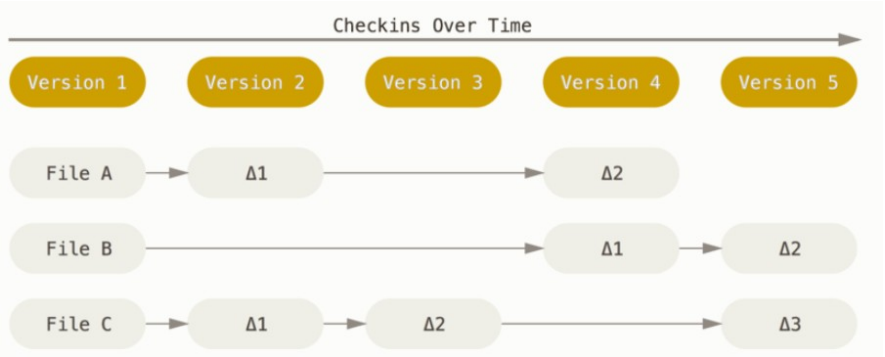


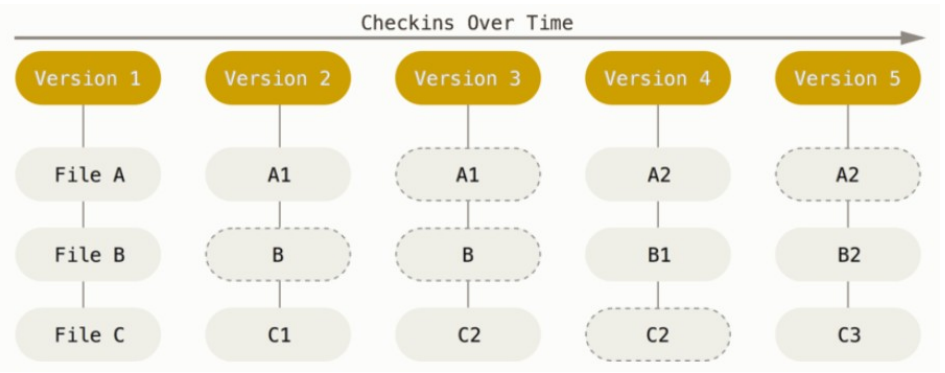
Sisteme de versionare. Exempe

- Sisteme de versionare fara repository local, cu branch-uri care presupun copierea efectiva a codului:
  - **CVS**: Control Versioning System
  - **SVN**: Subversion (Apache)
  - **Perforce**
  - **Clear Case**
- Sisteme de versionare cu repository local si cu branch-uri care nu presupun copierea efectiva a codului
  - **Git**. Sistem de versionare distribuit, dezvoltat de **Linus Torsvald.** Inseamna persoana dificila - “unpleasant person” in limbaj jargon britanic.
  - Exista varianta de client si de server care se pot instala atat pe Linux cat si pe Windows
  - Servere Git publice:
    - GitHub (compania Microsoft): https://github.com/
    - BitBucket (compania Atlasian), GitLab (compania GitLab)

Sisteme de versionare. Organizare informatie

Diferente intre GIT si alte sisteme de versionare din punct de vedere al organizarii si stocarii informatiei: https://git-scm.com/book...





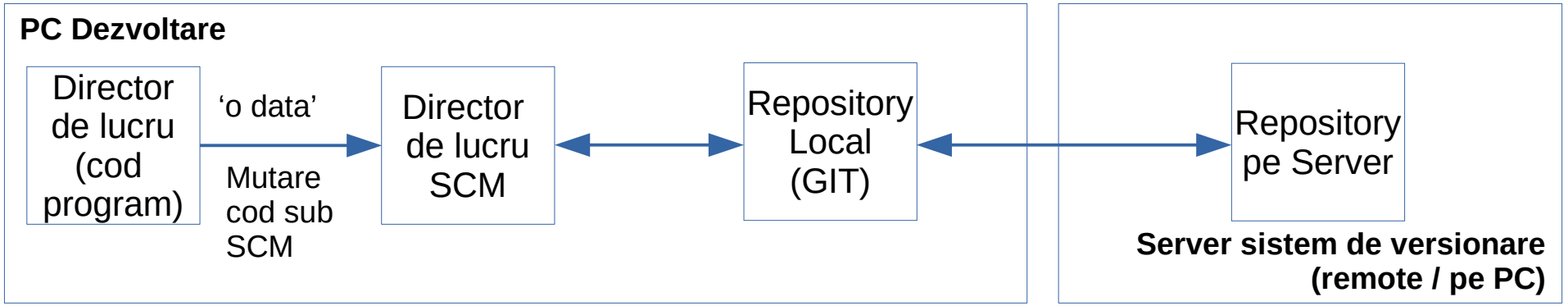
A) **Delta** / Diferente

Sisteme de versionare care stocare date sub forma de Delta,
Varianta initiala a fiecarui fisier si Delta, la fiecare modificare.
Sistem folosit de CVS, SVN, Perforce

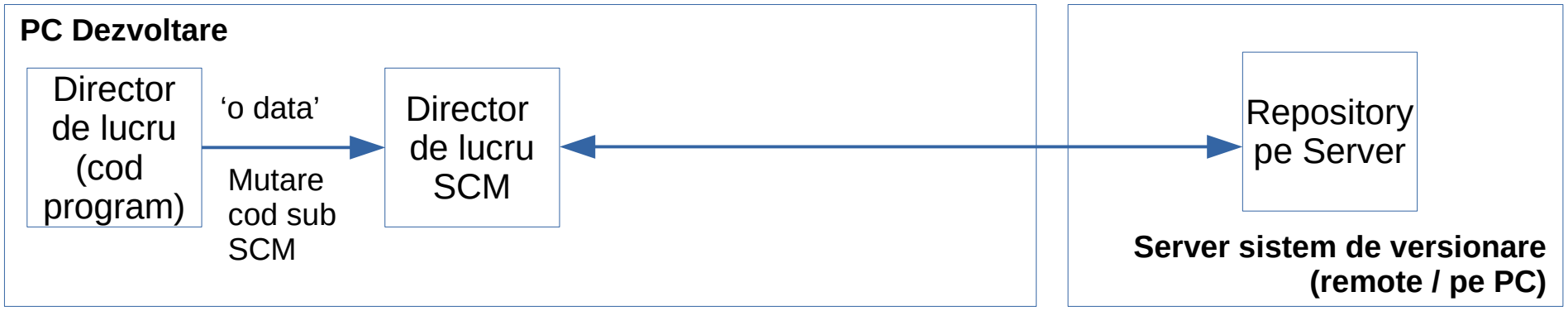
B) **“Snapshot”** Stocare date ca secvente de ‘poze’ ale proiectului la diverse momente

Sistem folosit de GIT
La fiecare moment GIT face o ‘poza’ a intregului repository si stocheaza o referinta la aceasta.
Pentru eficienta, pentru fisierele nemodificate se face legatura cu fisierul anterior, identic, care a fost deja stocat pe disc

Sisteme de versionare. Reprezentare schematica



A. Git



B. CSV, SVN, Perforce, ...

De ce Git?

- Majoritatea companiilor folosesc Git sau tranzitioneaza la Git
- Nu are nevoie de server
- Functioneaza la fel de bine si cu server central pentru a partaja cod si colaborare intre mai multi dezvoltatori
- Ofera:
  - Viteza
  - Design simplu
  - Suport pentru dezvoltare neliniara – mii de branch-uri in paralel
  - Capabil sa gestioneze proiecte mari, cum ar fi kernel-ul de Linux

Git local, pe PC-ul de dezvoltare (1)

LAB

- Git trebuie sa fie instalat pe masina virtuala Linux folosita pentru curs
- Creere director pe care vrem sa-l gestionam cu Git: `dir_local_cu_git'
- Executare comanda 'git init' pentru a-i comunica git-ului ca vrem sa gestioneze directorul de lucru
- Comanda de mai sus, creaza in dir_local_cu_git, directorul **.git** care va fi folosit de GIT pentru a tine evidenta fisierelor, versiunile etc.
  - git config -list -show-origin # comanda penru vizualizarea configuratie repository-ului
- Creere si adaugare fisier in repository-ul local, (nu pe server)
  - touch test_git.txt # Creere fisier test_git.txt
  - git status # pentru a vedea starea repository-ului
  - git add # pentru adaugare fisier in zona de 'stage'
  - git commit -m "test" # adaugare in repository, cu mesajul 'test'. de observat eroarea
  - Configurare git:
    - git config --global user.email "you@example.com" # configuratie necesara pentru commit
    - git config --global user.name "Your Name" #
  - repetare git commit
  - executie git status
  - git diff # diferente intre versiuni
  - git log # vizualizare commit-uri
  - git help, git help -a, … # comenzi help

Git local, pe PC-ul de dezvoltare (2)

LAB

- Creere director
- Creere repository local cu acest director
- Adaugare fisier `test_git.txt`
- Director `'.git'` si listare configuratie

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ ls -la
total 12
drwxrwxr-x 3 cgit cgit 4096 feb 24 18:16 .
drwxrwxr-x 3 cgit cgit 4096 feb 24 18:37 ..
drwxrwxr-x 7 cgit cgit 4096 feb 24 18:39 .git
-rw-rw-r-- 1 cgit cgit    0 feb 24 18:16 test_git.txt
cgit@cip:~/GITLOCAL/dir_local_cu_git$ cat test_git.txt
cgit@cip:~/GITLOCAL/dir_local_cu_git$
```

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git config --list --show-origin
file:.git/config        core.repositoryformatversion=0
file:.git/config        core.filemode=true
file:.git/config        core.bare=false
file:.git/config        core.logallrefupdates=true
```

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ tree .git
.git
├── branches
├── config
├── description
├── HEAD
├── hooks
│   ├── applypatch-msg.sample
│   ├── commit-msg.sample
│   ├── fsmonitor-watchman.sample
│   ├── post-update.sample
│   ├── pre-applypatch.sample
│   ├── pre-commit.sample
│   ├── pre-merge-commit.sample
│   ├── prepare-commit-msg.sample
│   ├── pre-push.sample
│   ├── pre-rebase.sample
│   ├── pre-receive.sample
│   └── update.sample
├── index
├── info
│   └── exclude
├── objects
│   ├── 23
│   │   └── 1b9f1f28e3f58df139918052d8efdd82b3c158
│   ├── e6
│   │   └── 9de29bb2d1d6434b8b29ae775ad8c2e48c5391
│   ├── info
│   └── pack
└── refs
    ├── heads
    └── tags

11 directories, 19 files
```

Git local, pe PC-ul de dezvoltare (3)

LAB

- Verificare stare, inainte si dupa add, configurare user si email, commit

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test_git.txt

nothing added to commit but untracked files present (use "git add" to track)
```

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git add test_git.txt
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   test_git.txt
```

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git commit -m "adaugare fisier test_git"

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'cgit@cip.(none)')
```

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git config --global user.email <email>
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git config --global user.name  <nume>
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git commit -m "adaugare fisier test_git"
[master (root-commit) 4936f4c] adaugare fisier test_git
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test_git.txt
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git status
On branch master
nothing to commit, working tree clean
cgit@cip:~/GITLOCAL/dir_local_cu_git$
cgit@cip:~/GITLOCAL/dir_local_cu_git$
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git config --list --show-origin
file:/home/cgit/.gitconfig      user.email= <email>
file:/home/cgit/.gitconfig      user.name=  <nume>
file:.git/config        core.repositoryformatversion=0
file:.git/config        core.filemode=true
file:.git/config        core.bare=false
file:.git/config        core.logallrefupdates=true
```

Git local, pe PC-ul de dezvoltare (4)

LAB

- Modificare fisier, adaugare si commit in git, git log

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ echo linia 1 > test_git.txt
cgit@cip:~/GITLOCAL/dir_local_cu_git$ cat test_git.txt
linia 1
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   test_git.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git add test_git.txt
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git commit -m "Adaugare continut - linia 1"
[master 04b4be5] Adaugare continut - linia 1
 1 file changed, 1 insertion(+)
cgit@cip:~/GITLOCAL/dir_local_cu_git$
```

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ echo linia 2 >> test_git.txt
cgit@cip:~/GITLOCAL/dir_local_cu_git$ cat test_git.txt
linia 1
linia 2
cgit@cip:~/GITLOCAL/dir_local_cu_git$
```

- Repetare operatie – modificari, adaugare, commit …

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git log
commit cced64d761c0bf620b603f39a61ad091bd7b3df9 (HEAD -> master)
Author: Ciprian Chende <cip_chende@yahoo.com>
Date:   Sat Feb 25 00:55:04 2023 +0200

    linia 2

commit 04b4be5bbd0b75e9006c2eacb03888b3f94683a9
Author: Ciprian Chende <cip_chende@yahoo.com>
Date:   Sat Feb 25 00:47:34 2023 +0200

    Adaugare continut - linia 1

commit 4936f4c608d11b7a803f42dc6aaf1da09c8be812
Author: Ciprian Chende <cip_chende@yahoo.com>
Date:   Sat Feb 25 00:00:16 2023 +0200

    adaugare fisier test_git
```

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git log --oneline --graph
* cced64d (HEAD -> master) linia 2
* 04b4be5 Adaugare continut - linia 1
* 4936f4c adaugare fisier test_git
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git log --oneline
cced64d (HEAD -> master) linia 2
04b4be5 Adaugare continut - linia 1
4936f4c adaugare fisier test_git
```

Git local, pe PC-ul de dezvoltare (5)

LAB

- `git diff, git diff help`

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git log --oneline
cced64d (HEAD -> master) linia 2
04b4be5 Adaugare continut - linia 1
4936f4c adaugare fisier test_git
cgit@cip:~/GITLOCAL/dir_local_cu_git$
cgit@cip:~/GITLOCAL/dir_local_cu_git$
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git diff 04b4be5
diff --git a/test_git.txt b/test_git.txt
index 5f21e1d..32bfdc3 100644
--- a/test_git.txt
+++ b/test_git.txt
@@ -1 +1,2 @@
 linia 1
+linia 2
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git diff 4936f4c
diff --git a/test_git.txt b/test_git.txt
index e69de29..32bfdc3 100644
--- a/test_git.txt
+++ b/test_git.txt
@@ -0,0 +1,2 @@
+linia 1
+linia 2
cgit@cip:~/GITLOCAL/dir_local_cu_git$
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git diff 04b4be5 --compact-summary
 test_git.txt | 1 +
 1 file changed, 1 insertion(+)
```

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git diff --help
```

```
GIT-DIFF(1)                                Git Manual                                GIT-DIFF(1)

NAME
       git-diff - Show changes between commits, commit and working tree, etc

SYNOPSIS
       git diff [<options>] [<commit>] [--] [<path>...]
       git diff [<options>] --cached [<commit>] [--] [<path>...]
       git diff [<options>] <commit> <commit> [--] [<path>...]
       git diff [<options>] <blob> <blob>
       git diff [<options>] --no-index [--] <path> <path>

DESCRIPTION
       Show changes between the working tree and the index or a tree, changes between the index and a tree, changes between
       two trees, changes between two blob objects, or changes between two files on disk.

       git diff [<options>] [--] [<path>...]
           This form is to view the changes you made relative to the index (staging area for the next commit). In other words,
           the differences are what you could tell Git to further add to the index but you still haven't. You can stage these
           changes by using git-add(1).

       git diff [<options>] --no-index [--] <path> <path>
           This form is to compare the given two paths on the filesystem. You can omit the --no-index option when running the
           command in a working tree controlled by Git and at least one of the paths points outside the working tree, or when
           running the command outside a working tree controlled by Git. This form implies --exit-code.

       git diff [<options>] --cached [<commit>] [--] [<path>...]
           This form is to view the changes you staged for the next commit relative to the named <commit>. Typically you would
           want comparison with the latest commit, so if you do not give <commit>, it defaults to HEAD. If HEAD does not exist
           (e.g. unborn branches) and <commit> is not given, it shows all staged changes. --staged is a synonym of --cached.

       git diff [<options>] <commit> [--] [<path>...]
           This form is to view the changes you have in your working tree relative to the named <commit>. You can use HEAD to
           compare it with the latest commit, or a branch name to compare with the tip of a different branch.

       git diff [<options>] <commit> <commit> [--] [<path>...]
           This is to view the changes between two arbitrary <commit>.
```

Git local, pe PC-ul de dezvoltare (6)

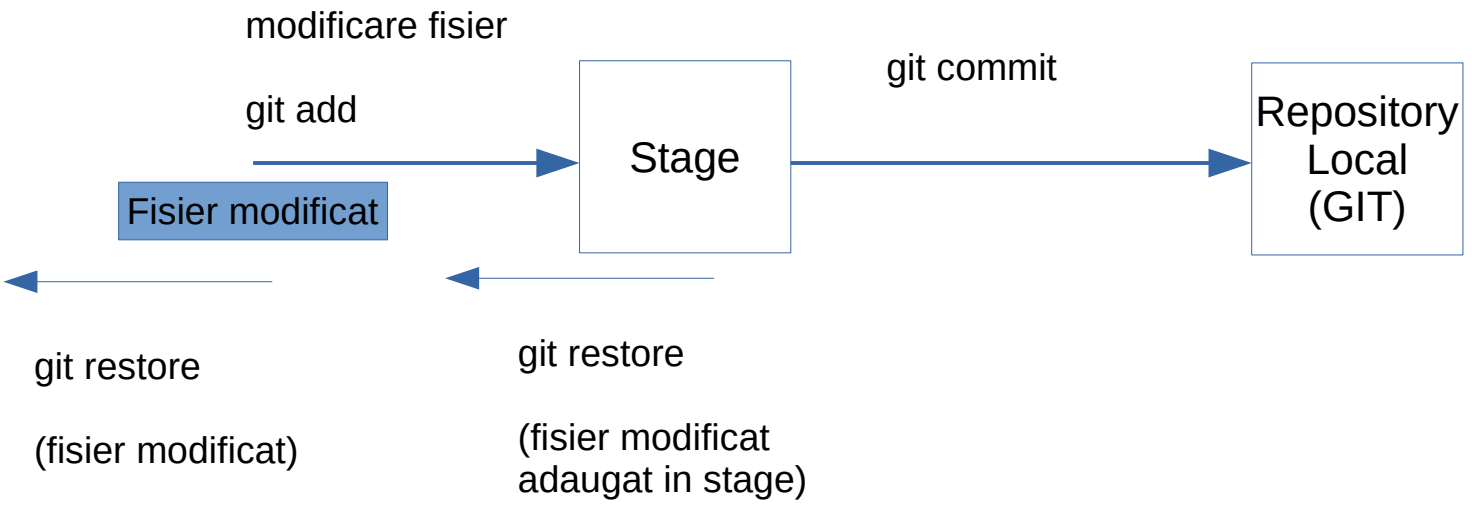
LAB

- `git help, git help -a`

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone             Clone a repository into a new directory
   init              Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add               Add file contents to the index
   mv                Move or rename a file, a directory, or a symlink
   restore           Restore working tree files
   rm                Remove files from the working tree and from the index
   sparse-checkout   Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
   bisect            Use binary search to find the commit that introduced a bug
   diff              Show changes between commits, commit and working tree, etc
   grep              Print lines matching a pattern
   log               Show commit logs
   show              Show various types of objects
   status            Show the working tree status

grow, mark and tweak your common history
   branch            List, create, or delete branches
```

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git help -a
See 'git help <command>' to read about a specific subcommand

Main Porcelain Commands
   add                  Add file contents to the index
   am                   Apply a series of patches from a mailbox
   archive              Create an archive of files from a named tree
   bisect               Use binary search to find the commit that introduced a bug
   branch               List, create, or delete branches
   bundle               Move objects and refs by archive
   checkout             Switch branches or restore working tree files
   cherry-pick          Apply the changes introduced by some existing commits
   citool               Graphical alternative to git-commit
   clean                Remove untracked files from the working tree
   clone                Clone a repository into a new directory
   commit               Record changes to the repository
   describe             Give an object a human readable name based on an available ref
   diff                 Show changes between commits, commit and working tree, etc
   fetch                Download objects and refs from another repository
   format-patch         Prepare patches for e-mail submission
   gc                   Cleanup unnecessary files and optimize the local repository
   gitk                 The Git repository browser
   grep                 Print lines matching a pattern
   gui                  A portable graphical interface to Git
   init                 Create an empty Git repository or reinitialize an existing one
   log                  Show commit logs
   merge                Join two or more development histories together
   mv                   Move or rename a file, a directory, or a symlink
   notes                Add or inspect object notes
   pull                 Fetch from and integrate with another repository or a local branch
   push                 Update remote refs along with associated objects
   range-diff           Compare two commit ranges (e.g. two versions of a branch)
   rebase               Reapply commits on top of another base tip
   reset                Reset current HEAD to the specified state
   restore              Restore working tree files
   revert               Revert some existing commits
   rm                   Remove files from the working tree and from the index
```

Comenzi git de baza. Reprezentare schematica



Creere ramuri (branches) de dezvoltare.
`git branch, git checkout`

LAB

- in branch-ul main (master), creati fisierul ‘`fisier1.py`’ (`touch fisier1.py`) si continutul `print(‘fisier1’)`
  (`echo fisier1 > fisier1.py`
  (echo este cea mai rapida varianta de adaugare continut de dimensiuni mici intr-un fisier. Alternativa ar fi sa deschidem un editor vim/vscode etc, sa editam si sa salvam)
- adaugati-l in repository-ul local cu git add si git commit
- creere branch devel: `git branch devel`,
- Trecere pe branch-ul devel: `git checkout devel`
- Vizualizati continutul fisierului (`cat fisier1.py`). Are acelasi continut ca si in master (fig. de mai jos)
- Modificare `fisier1.py pe devel`: edidati in ‘vim’ modificati prima linie in: `print(‘fisier1 – devel’)`, adaugare modificare si commit pe devel
- Trecere pe master: `git checkout master`; de observat ca fisierul este in forma lui initiala
  - Modificare fisier pe master, modificare linie initiala in fisier: `print(‘fisier1 – master’)`

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git branch devel
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git checkout devel
Switched to branch 'devel'
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git branch
* devel
  master
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git status
On branch devel
nothing to commit, working tree clean
cgit@cip:~/GITLOCAL/dir_local_cu_git$ cat fisier1.py
print('fisier1')
cgit@cip:~/GITLOCAL/dir_local_cu_git$
```

Integrare cod din ramuri de dezvoltare diferite optiunea 1
`git merge`

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git merge devel
Auto-merging fisier1.py
CONFLICT (content): Merge conflict in fisier1.py
Automatic merge failed; fix conflicts and then commit the result.
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   fisier1.py

no changes added to commit (use "git add" and/or "git commit -a")
cgit@cip:~/GITLOCAL/dir_local_cu_git$ cat fisier1.py
<<<<<<< HEAD
print('fisier1 - master')
=======
print('fisier1 - devel')
>>>>>>> devel
cgit@cip:~/GITLOCAL/dir_local_cu_git$
cgit@cip:~/GITLOCAL/dir_local_cu_git$
cgit@cip:~/GITLOCAL/dir_local_cu_git$ python3 fisier1.py
  File "fisier1.py", line 1
    <<<<<<< HEAD
    ^
SyntaxError: invalid syntax
```

LAB

- Integrare modificare din devel in master: `git merge devel`
- Conflict la `'merge'` in ambele fisiere sunt modificarei in aceeasi zona – in cazul de fata pe aceeasi linie. Git-ul nu stie ce sa aleaga si-i comunica programatorului ca trebuie sa rezolve problema.
- Dupa rezolvarea problemei, noul fisier trebuie adaugat pe git cu add si commit.

Integrare cod din ramuri de dezvoltare diferite optiunea 1 `git merge`

LAB

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ cat fisier1.py
print('fisier1 - master')
print('fisier1 - devel')
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git log --oneline --graph
*   a1e74b5 (HEAD -> master) Integrare devel in branch
|\
| * 4a8ae60 (devel) branch devel - print('fisier1 - devel')
* | feb5f4a branch master - print('fisier1 - master')
|/
* eae1be0 adaugare fisier1.py care contine print('fisier1))
* cced64d linia 2
* 04b4be5 Adaugare continut - linia 1
* 4936f4c adaugare fisier test_git
```

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git checkout devel
Switched to branch 'devel'
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git merge master
Updating 4a8ae60..a1e74b5
Fast-forward
 fisier1.py | 1 +
 1 file changed, 1 insertion(+)
```

```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git log --oneline --graph
*   a1e74b5 (HEAD -> master, devel) Integrare devel in branch
|\
| * 4a8ae60 branch devel - print('fisier1 - devel')
* | feb5f4a branch master - print('fisier1 - master')
|/
* eae1be0 adaugare fisier1.py care contine print('fisier1))
* cced64d linia 2
* 04b4be5 Adaugare continut - linia 1
* 4936f4c adaugare fisier test_git
```

- Mutare pe branch-ul master: `git checkout master`
- Rezolvare conflict – stergere linii adaugate de git: cele care incep cu `<<<<<< / >>>>>>`. Vizualizare fisier modificat cu comanda `'cat'`
- Integrare modificare din `devel` in `master` (branch activ - master):

  `git merge devel`
- Vizualizare grafica dupa merge, branch-ul ‘devel’ ramane la commit-ul: `4a8ae60;`
- Pe ‘master’ dupa commit-ul care integreaza versiunea de pe master cu cea de pe devel, noul commit este: `a1e74b5`
- Daca vrem sa refolosim branch-ul devel, este recomandat sa-l sincronizam cu master-ul. Aceasta se rezolva prin:
  - `git checkout devel`
  - `git merge master`
- Merge-ul este de tip fast-forward, doar mutarea inainte a branch-ului `‘devel’` pe commit-ul `a1e74b5`
- Acum putem reutiliza branch-ul devel pentru a adauga o noua functionalitate si apoi sa facem merge in master
- Este recomandata aceasta metoda de dezvoltare, cu modificari pentru noi functionalitati adaugate in devel
- ‘fix’-uri pentru bug-uri pe codul distribuit la client – luat de obicei din master/main – se recomanda sa se faca in branch-uri din master ‘fix / hotfix branch’ si apoi sa se integreze in master.
- Pentru simplitate, aici am facut modificarea direct in master.

Vizualizare grafica modificare in master si devel inainte de merge.

LAB

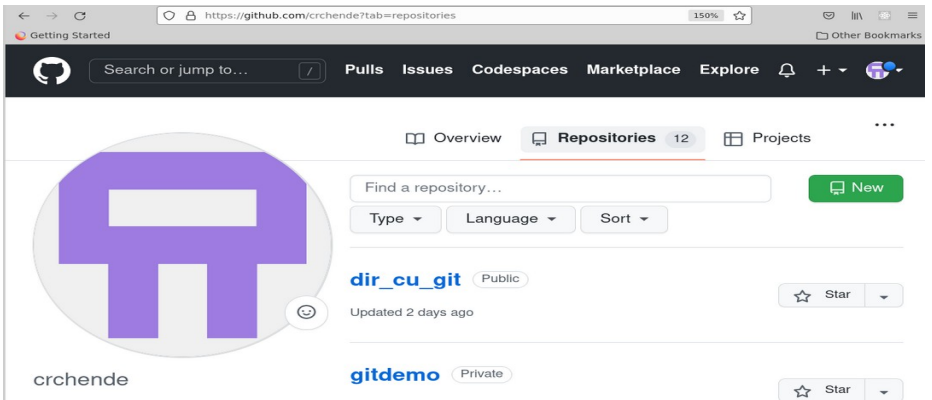
```
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git status
On branch master
nothing to commit, working tree clean
cgit@cip:~/GITLOCAL/dir_local_cu_git$
cgit@cip:~/GITLOCAL/dir_local_cu_git$ git log --oneline --graph --all
* 0cd0c7c (HEAD -> master) brach master - modificare 1
| * 183a2bf (devel) branch devel - modificare 1
|/
*   a1e74b5 Integrare devel in branch
|\
| * 4a8ae60 branch devel - print('fisier1 - devel')
* | feb5f4a branch master - print('fisier1 - master')
|/
* eae1be0 adaugare fisier1.py care contine print('fisier1))
* cced64d linia 2
* 04b4be5 Adaugare continut - linia 1
* 4936f4c adaugare fisier test_git
cgit@cip:~/GITLOCAL/dir_local_cu_git$
```

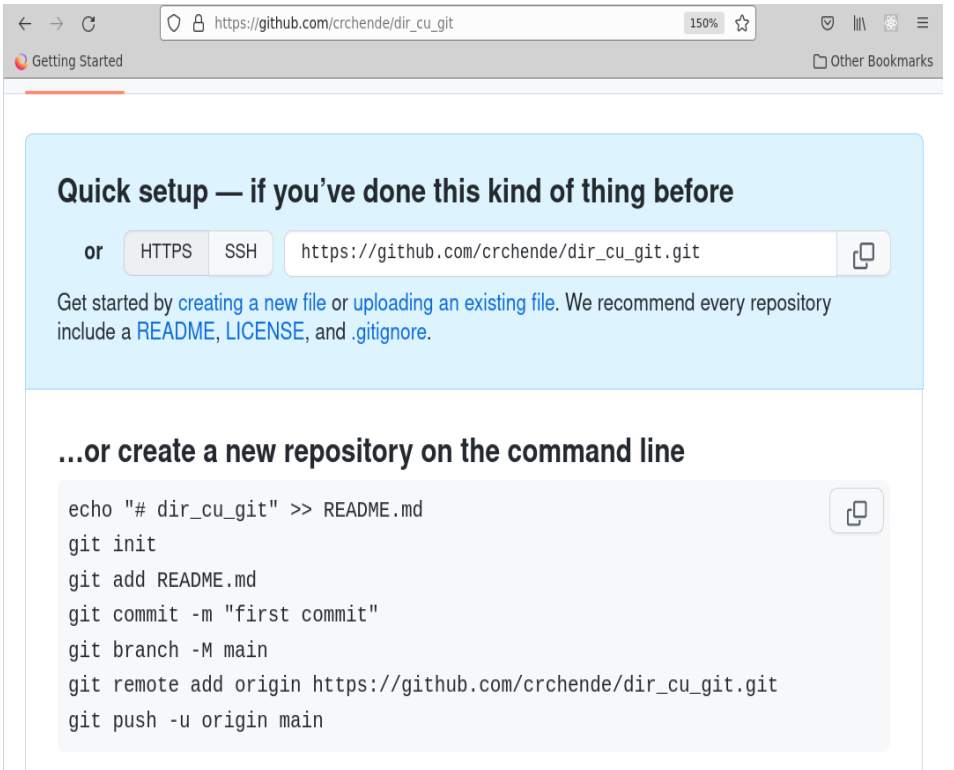
- Checkout devel, modificare fisier
- Checkout master, modificare fisier
- Vizualizare grafica cu:
  `git log –oneline –graph –all`
- Git log prezinta doar modificarile de pe un branch, pentru a vedea modificarile de pe toate branch-urile este nevoie de optiunea `–all`.
- De observat branch-urile `devel` si `master`, faptul ca pe ambele sunt modificari.
- Nu s-a facut merge intre branch-uri. Acest lucru se vede in reprezentarea grafica.

Utilizare GitHub. Creere repository pe GitHub

LAB

- Login pe github.com
- Creere repository ‘dir_cu_git’
- Accesare pagina repository
- Copiere URL repository
- Clonare locala cu comanda
  - `git clone <URL>`
- In noul repository care are conexiune la github.com, se pot face toate operatiunile anterioare – creere fisier, adaugare, commit, creere branch, modificare, commit etc.
- Pentru a pune modificarile pe github.com se foloseste comanda:
  - `git push`
- Modificarile de pe server (putem edita in pagina web, pentru a simula ca un alt dezvoltator lucreaza la repository, sau putem sa invitam un coleg sa fie contribuitor la proiectf) pot fi aduse local cu comenzile:
  - `git fetch  # modificarile sunt aduse dar nu se face si merge`
  - `git pull   # se aduc local modificarile de pe server si se`
    `# face merge`





Clonare si legatura ‘local’ ↔ ‘remote’

Clonare repository de pe server. Comanda `git clone <URL>`

```
cgit@cip:~/DIR_PT_GIT$ git clone https://github.com/crchende/dir_cu_git.git
Cloning into 'dir_cu_git'...
warning: You appear to have cloned an empty repository.
cgit@cip:~/DIR_PT_GIT$ ll
total 16
drwxrwxr-x 4 cgit cgit 4096 feb 26 23:33 ./
drwxr-xr-x 4 cgit cgit 4096 feb 26 23:32 ../
drwxrwxr-x 3 cgit cgit 4096 feb 26 23:33 dir_cu_git/
```

LAB

Explicare ‘origin’ – alias / nume dat URL-ului repository-ului ‘remote’, de pe server.

Comenzi utilizate: `git remote –v, git config --list--show-origin`

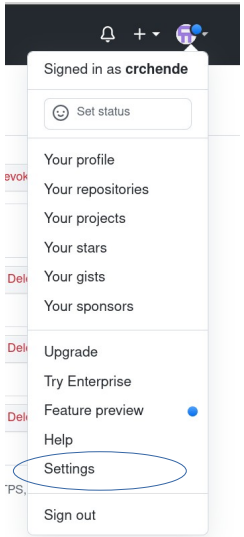
```
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ # Director GIT local, cu repository local in .git
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ ls -al
total 16
drwxrwxr-x 3 cgit cgit 4096 feb 28 00:57 .
drwxrwxr-x 4 cgit cgit 4096 feb 26 23:33 ..
-rw-rw-r-- 1 cgit cgit   31 feb 28 00:57 fisier1.py
drwxrwxr-x 8 cgit cgit 4096 feb 28 01:13 .git
cgit@cip:~/DIR_PT_GIT/dir_cu_git$
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ # Vizualizare 'remote' si 'origin
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ git remote -v
origin  https://github.com/crchende/dir_cu_git.git (fetch)
origin  https://github.com/crchende/dir_cu_git.git (push)
cgit@cip:~/DIR_PT_GIT/dir_cu_git$
```

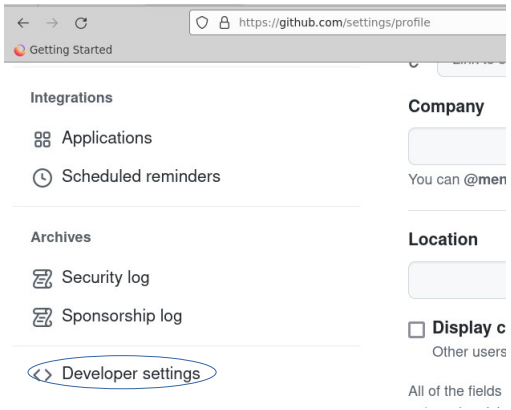
```
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ git config --list --show-origin
file:/home/cgit/.gitconfig      user.email=cip_chende@yahoo.com
file:/home/cgit/.gitconfig      user.name=Ciprian Chende
file:.git/config        core.repositoryformatversion=0
file:.git/config        core.filemode=true
file:.git/config        core.bare=false
file:.git/config        core.logallrefupdates=true
file:.git/config        remote.origin.url=https://github.com/crchende/dir_cu_git.git
file:.git/config        remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
file:.git/config        branch.master.remote=origin
file:.git/config        branch.master.merge=refs/heads/master
```

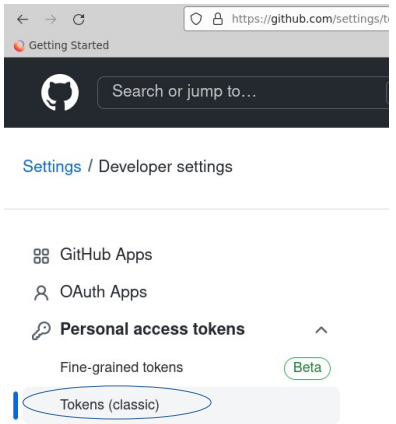
Adaugare modificari pe GitHub,
configurare server pentru a permite modificarile pe server

LAB

- Modificarile locale trebuie adaugate in repository-ul local cu:
  - `git add <fisiere>`
  - `git commit (git commit –m “<mesaj>”)`
- Pentru a adauga modificarile si pe serverul ‘remote’ trebuie utilizata comanda:
  - `git push`
- Accesul la ‘remote’ necesita autentificare, cu ‘**`user`**’ si ‘**`personal access token`**’







- Token-ul se genereaza in pagina care apare la selectarea optiunii **‘Tokens (clasic)’**
- Token-ul / cheia generata trebuie copiata local, dupa creere.
- Daca se iese din pagina de creere tocken acesta nu mai poate fi vizualizat.
- Username-ul si tocken-ul sunt cerute la operatiunile de scriere (push)
- Pentru fiecare token se configureaza nivele de acces.
  De cele mai multe ori este suficient sa se permita doar: ‘**repo**’ – pentru operatiuni standard pe repo.

Adaugare modificari pe GitHub, `git push`

LAB

- Fisierul creat local, dupa add si commit, poate fi adaugat pe server cu git push
- `git push,` duce pe server modificarile locale.
- In cazul de fata, fiind primul fisier nu sunt conflicte

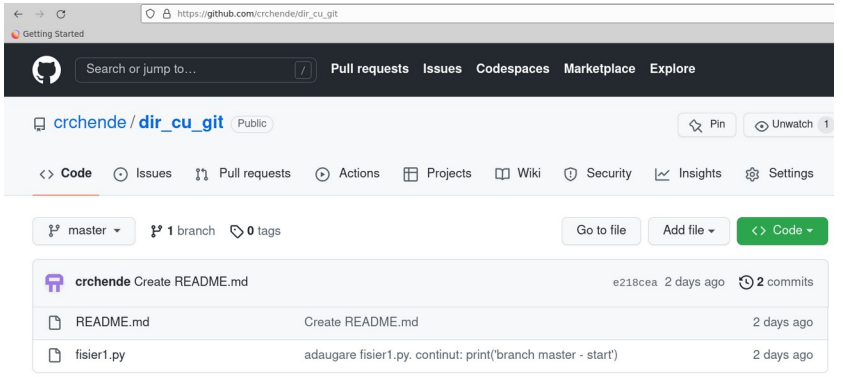
```
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ ls -la
total 16
drwxrwxr-x 3 cgit cgit 4096 feb 28 00:57 .
drwxrwxr-x 4 cgit cgit 4096 feb 26 23:33 ..
-rw-rw-r-- 1 cgit cgit   31 feb 28 00:57 fisier1.py
drwxrwxr-x 8 cgit cgit 4096 mar  1 23:04 .git
```

```
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ git push
Username for 'https://github.com': crchende
Password for 'https://crchende@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 285 bytes | 285.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/crchende/dir_cu_git.git
 * [new branch]      master -> master
```

Adaugare modificari pe GitHub,
fac modificare pe server si o observ local

LAB

- Adaugare pe server, din browser a fisierului README.md
- Pe client, din linia de comanda:
  - Vizualizare ‘remote’ si ‘local’
  - Se observa diferente intre ID-ul pentru master pe server si local.
  - Comanda git remote show origin ne indica faptul ca nu s-a facut sincronizarea cu serverul.




```
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ git ls-remote
From https://github.com/crchende/dir_cu_git.git
e218cea34c61528ad1ca05979df12a1cdb49fc72        HEAD
e218cea34c61528ad1ca05979df12a1cdb49fc72        refs/heads/master
cgit@cip:~/DIR_PT_GIT/dir_cu_git$
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ git log --oneline --all
c3901d5 (HEAD -> master, origin/master) adaugare fisier1.py. continut: print('branch master - start')
cgit@cip:~/DIR_PT_GIT/dir_cu_git$
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ git log --all
commit c3901d5ef49debfe495cfd47b56315b4fbf4dcb3 (HEAD -> master, origin/master)
Author: Ciprian Chende <cip_chende@yahoo.com>
Date:   Tue Feb 28 00:59:34 2023 +0200

    adaugare fisier1.py. continut: print('branch master - start')
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ ls
fisier1.py
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ git branch -a
* master
  remotes/origin/master
```

```
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ git remote show origin
* remote origin
  Fetch URL: https://github.com/crchende/dir_cu_git.git
  Push  URL: https://github.com/crchende/dir_cu_git.git
  HEAD branch: master
  Remote branch:
    master tracked
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
    master pushes to master (local out of date)
cgit@cip:~/DIR_PT_GIT/dir_cu_git$
```

Sincronizare ‘local’ cu ‘remote’, `git pull`

LAB

- Dupa creerea fisierului README.md pe ‘remtote’, ‘local’ nu mai este sincronizat (nu mai are ultimele modificari depe server)
- Acestea pot fi aduse cu `git pull`. Aceasta comanda aduce local modificarile de pe server si face si ‘merge’ daca este nevoie
- Se observa dupa pull, ca ID-ul comitului, atat pe local cat si pe remote este la fel.

```
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ git remote show origin
* remote origin
  Fetch URL: https://github.com/crchende/dir_cu_git.git
  Push  URL: https://github.com/crchende/dir_cu_git.git
  HEAD branch: master
  Remote branch:
    master tracked
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
    master pushes to master (up to date)
```

```
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 664 bytes | 332.00 KiB/s, done.
From https://github.com/crchende/dir_cu_git
   c3901d5..e218cea  master     -> origin/master
Updating c3901d5..e218cea
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ ls -la
total 20
drwxrwxr-x 3 cgit cgit 4096 mar  1 23:25 .
drwxrwxr-x 4 cgit cgit 4096 feb 26 23:33 ..
-rw-rw-r-- 1 cgit cgit   31 feb 28 00:57 fisier1.py
drwxrwxr-x 8 cgit cgit 4096 mar  1 23:25 .git
-rw-rw-r-- 1 cgit cgit   13 mar  1 23:25 README.md
```

```
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ git ls-remote
From https://github.com/crchende/dir_cu_git.git
e218cea34c61528ad1ca05979df12a1cdb49fc72        HEAD
e218cea34c61528ad1ca05979df12a1cdb49fc72        refs/heads/master
cgit@cip:~/DIR_PT_GIT/dir_cu_git$
cgit@cip:~/DIR_PT_GIT/dir_cu_git$ git log --all
commit e218cea34c61528ad1ca05979df12a1cdb49fc72 (HEAD -> master, origin/master)
Author: crchende <57460107+crchende@users.noreply.github.com>
Date:   Tue Feb 28 01:08:46 2023 +0200

    Create README.md

commit c3901d5ef49debfe495cfd47b56315b4fbf4dcb3
Author: Ciprian Chende <cip_chende@yahoo.com>
Date:   Tue Feb 28 00:59:34 2023 +0200

    adaugare fisier1.py. continut: print('branch master - start')
```

Bibliografie

Git	https://git-scm.com/book/en/v2
	https://learngitbranching.js.org/
	https://github.com

LABORATOR

Instalare Git pe Linux

- Verificare daca git este instalat
  - In terminal, executati comanda: `git`
    - daca ‘git’ este instalat se va afisa ‘help’-ul comenzii:

```
cip@cipasus:~$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
```

    - Altfel, se va afisa un mesaj de eroare si se va sugera cum trebuie instalat git-ul:

```
cip@cip-rst-vm:/mnt/cdrom$ git
Command 'git' not found, but can be installed with:
sudo apt install git
```

- **Pentru instalare, executati comanda:**

```
sudo apt install git-all
```

- Se pot vizualiza pachetele disponibile pentru a fi instalate, si cele instalate folosind comanda apt:
  - `apt list | grep “^git”  # afiseaza toate pachetele care incep cu cuvantul ‘git’`
  - `apt list –installed “^git” # afiseaza toate pachetele instalate care incep cu ‘git’`

```
cip@cip-rst-vm:/mnt/cdrom$ apt list | grep "^git"

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

git-absorb/jammy 0.6.6-2build2 amd64
git-all/jammy-updates,jammy-updates,jammy-security,jammy-security 1:2.34.1-1ubuntu1.8 all
git-annex-remote-rclone/jammy,jammy 0.6-1 all
git-annex/jammy 8.20210223-2ubuntu2 amd64
```

```
cip@cip-rst-vm:/mnt/cdrom$ apt list --installed | grep "^git"

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

cip@cip-rst-vm:/mnt/cdrom$
```

LAB GIT

1. **Lucru cu git local**
   - Creere director
   - Initializare git – ‘adaugare director in git’
   - Operatiuni standard – adaugare continut in director (creere / modificare fisiere), adaugare in git cu ‘add’ si ‘commit’
   - Utilizare branch-uri de dezvoltare
   - Integrare cod dintr-un branch in alt branch prin merge
   - Vizualizare istorie modificari in git, integrari etc. Vizualizare ID-uri commit, vizualizare grafica in terminal
   - Vizualizare si rezolvare conflicte merge
   - Comenzi git utilizate:
     - `git init, git config –list –show-origin, git add, git commit,`
     - `git log, git help, git config,`
     - `git branch, git merge, git diff`

LAB GIT

1. **Lucru cu ‘remote’ pe serverul ‘GitHub’ – partea 1**
   - Conectare la GitHub
   - Creere repository pe server
   - Clonare repository pe PC-ul de dezvoltare
   - Vizualizare director de lucru clonat de pe server
   - Intelegere ‘origin’, si vizualizare informatii despre ‘remote’
   - Configurare ‘token’ de dezvoltare pentru a putea face modificari pe server
   - Adaugare modificari locale pe server prin ‘git push’
   - Adaugare fisier pe server (poate fi orice alta modificare), vizualizare ‘local’ si a faptului ca nu este sincronizat
   - Aducem modificarile de pe server – ‘remote’, pe PC-ul unde lucram – ‘local’ cu `git pull`
   - Comenzi git utilizate:
     - `git clone`
     - `git push, git pull,`
     - `git remote -v, git ls-remote`
   - ‘**fork repository**’ – aducere in spatiul personal de pe GitHub a unui repository la care avem acces. Ex: https://github.com/crchende/sysinfo
     - Login pe GitHub. Acces link de mai sus. In partea dreapta sus a pagini GitHub exista optiunea ‘Fork’
     - Dupa clone:
       - Rulare aplicatie CLI – executia fisierului sysinfocli.py (python3 sysinfocli.py)
       - Rulare site web flask – `source activeaza_venv` si `source ruleaza_aplicatia`
         - `Posibil sa fie nevoie sa instalati mai multe pachete. Vedeti mesajele de eroare la executia scriptului activeaza_venv`