# EXERCICIS SOBRE EL CONTROL DE VERSIONS

Islam El Omrani Akhazzan
CFGS Desenvolupament Web 1
12/02/22

# ÍNDEX



INTRODUCCIÓ:
L'objectiu d'aquest document és practicar més a fons l'ús dels controls de versions i al principi informar-nos sobre què son i perquè s'utilitzen.

# Pregunta 1. La meva història de por

JUSTIFICACIÓ DE L'ÚS DEL CONTROL DE VERSIONS

**Molts desenvolupadors i treball telemàtic** – Aquests són els motius més grans pels quals hem d'utilitzar control de versions, a l'hora de programar diversos companys en un projecte hauríem trobat impossible organitzar-nos si no haguéssim pogut pujar els canvis que fem a un repositori comú. Crec que hauríem fet cadascú una part, sense poder provar perquè no tenim tot el codi, i després hauríem de quedar per ajuntar-ho tot i començar a fer proves. O anar enviant-nos per correu cadascú la seva feina. En fi, un caos.

**Coexistència de versions i experiments** – És un problema típic. Fem un canvi o una millora i ho guardem a un nou fitxer perquè no volem canviar o afectar al que ja tenim fet, a aquest nou fitxer li agregues al nom 'definitiu'. Després resulta que aquesta versió és millor i així en bucle. Finalment acabes amb 200 fitxers dient-los ...def, ...defiii, ...definitivo, ...DEFINITIVOOOO. Quan més gran el nom més recent la versió.

# Pregunta 2. Comparem sistemes de control de versions

**CVS:**
**URL del projecte:** http://cvs.nongnu.org/
**Llicència:** GNU General Public License, versió 1.0 o posterior.
**Descripció:** CVS is a version control system, an important component of Source Configuration Management (SCM). Using it, you can record the history of sources files, and documents. It fills a similar role to the free software RCS, PRCS, and Aegis packages. CVS is a production quality system in wide use around the world, including many free software projects.
**Bondats:** While CVS stores individual file history in the same format as RCS, it offers the following significant advantages over RCS:

- It can run scripts which you can supply to log CVS operations or enforce site-specific policies.

- Client/server CVS enables developers scattered by geography or slow modems to function as a single team. The version history is stored on a single central server and the client machines have a copy of all the files that the developers are working on. Therefore, the network between the client and the server must be up to perform CVS operations (such as checkins or updates) but need not be up to edit or manipulate the current versions of the files. Clients can perform all the same operations which are available locally.

- In cases where several developers or teams want to each maintain their own version of the files, because of geography and/or policy, CVS's vendor branches can import a version from another team (even if they don't use CVS), and then CVS can merge the changes from the vendor branch with the latest files if that is what is desired.

- Unreserved checkouts, allowing more than one developer to work on the same files at the same time.

- CVS provides a flexible modules database that provides a symbolic mapping of names to components of a larger software distribution. It applies names to collections of directories and files. A single command can manipulate the entire collection.

- CVS servers run on most unix variants, and clients for Windows NT/95, OS/2 and VMS are also available. CVS will also operate in what is sometimes called server mode against local repositories on Windows 95/NT.

**Impressió:** En quant a servidors centralitzats té bona pinta tot i que sembla una mica antic i en desús pel disseny de la web i perquè posa que funciona a Windows 95. El problema és que no es pot canviar el nom dels arxius, l'has d'esborrar i tornar a pujar-lo amb el nom diferent.

**SUBVERSION:**
**URL del projecte: https://subversion.apache.org/**
**Llicència:** Apache License, Version 2.0
**Descripció:** Subversion is an open source version control system. Founded in 2000 by CollabNet, Inc., the Subversion project and software have seen incredible success over the past decade. Subversion has enjoyed and continues to enjoy widespread adoption in both the open source arena and the corporate world.
**Bondats:**
- Se segueix la història dels fitxers i directoris a través de còpies i canvis de nom.
- Les modificacions (incloent canvis a diversos fitxers) són atòmiques.
- La creació de branques i etiquetes és una operació eficient O (1) i no O(n) com el CVS.
- S'envien només les diferències en les dues adreces (amb el CVS sempre s'envien al servidor fitxers complets).
- Es pot servir mitjançant Servidor HTTP Apache sobre WebDAV/DeltaV.
- Gestiona eficientment fitxers binaris (a diferència del CVS que els tracta internament com si fossin text).
**Impressió:**
Com a sistema centralitzat Subversion em dóna més bona vibració, sembla que està tot més treballat i sembla que es segueix utilitzant. Té algunes limitacions que tracten de solucionar però si hagués d'agafar un sistema centralitzat gratuït agafaria aquest.

**PERFORCE:**
**URL del projecte: https://www.perforce.com/**
**Llicència:** Propietaris
**Descripció:** Helix Core, formerly Perforce Helix, is the company's version control software for large scale development environments. The Helix Version Control System manages a central database and a master repository of file versions.

**Bondats:**
- The Perforce system can make part or all of its content available as Git repositories. Users of Git and of other clients can work with the same file content and history.
- Git commits are visible to users of other clients as Perforce changelists, and vice versa. Users submit changed files together in changelists, which are applied as atomic commits.
- The server and client software are released as pre-built executables for Microsoft Windows, macOS, Linux, Solaris, FreeBSD, and other operating systems.

**Impressió:**
Si tingués diners per pagar un sistema centralitzat agafaria aquest sense cap dubte, tenen clients molt potents com Nvidia i Oracle, uns productes amb l'última tecnologia i sembla que són els reis del sector.

**GIT:**

**URL del projecte: http://git-scm.com/**
**Llicència:** GNU GPL v2
**Descripció:**
Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

**Bondats:**
El seu lema ho diu tot, distributed-even-if-your-workflow-isnt.
- Forta incidència en la no-linealitat dels canvis, per a la rapidesa en la gestió de ramificacions i mesclat de diferents versions.
- Gestió distribuïda. Els canvis s'importen com a ramificacions, i poden ser mesclades en la manera en què ho fa una ramificació de l'emmagatzemament en local.
- Els magatzems d'informació poden publicar-se per HTTP, FTP, SSH, rsync o mitjançant un protocol natiu, a part de ser possible emular CVS.
- Gestió eficient de projectes grans, donada la rapidesa de gestió de diferències entre arxius, entre altres millores d'optimització de velocitat d'execució.
- Totes les versions prèvies a un canvi determinat, impliquen la notificació d'un canvi posterior en qualsevol d'elles a aquest canvi (denominat autenticació criptogràfica d'historial). Això existia a Monotone.
- Millora el treball amb afectacions de codi que concorren en operacions similars en diversos arxius.
- Els reanomenats es treballen basant-se en similituds entre fitxers, a part de noms de fitxers, però no es fan marques explícites de canvis de nom basant-se en suposats noms únics de nodes de sistema de fitxers, el qual evita possibles, i possiblement desastroses, coincidències de fitxers diferents en un únic nom.
- Re-emmagatzemament periòdic en paquets (fitxers). Això és relativament eficient per l'escriptura de canvis i relativament ineficient per lectura si el re-empaquetat (basant-se en diferències) no té lloc amb regularitat.

**Impressió:**
Fan total de git, em sembla espectacular el que ha aconseguit aquest projecte i crec que durant tota la meva vida el faré servir, es súper còmode i funcional.

**MERCURIAL:**

**URL del projecte: https://mercurial.selenic.com/**
**Llicència: GPL 2.0 o posterior**
**Descripció:**
Work easier, Work faster
Mercurial is a free, distributed source control management tool. It efficiently handles projects of any size and offers an easy and intuitive interface.

**Bondats:**
It is fast and powerful:
Mercurial efficiently handles projects of any size and kind. Every clone contains the whole project history, so most actions are local, fast and convenient. Mercurial supports a multitude of workflows and you can easily enhance its functionality with extensions.

It is easy to learn:
You can follow our simple guide to learn how to revision your documents with Mercurial, or just use the quick start to get going instantly. A short overview of Mercurial's decentralized model is also available.
And it just works:
Mercurial strives to deliver on each of its promises. Most tasks simply work on the first try and without requiring arcane knowledge.
**Impressió:**
Té bona pinta i m'agradaria probar-lo però dubto que pugui competir amb git.

**BAZAAR:**
**URL del projecte: http://bazaar.canonical.com/en/**
**Llicència:** GPL 2.0 o posterior
**Descripció:** Bazaar is a version control system that helps you track project history over time and to collaborate easily with others. Whether you're a single developer, a co-located team or a community of developers scattered across the world, Bazaar scales and adapts to meet your needs.
**Bondats:**
In contrast to purely distributed version control systems which do not use a central server, Bazaar supports working with or without a central server. It is possible to use both methods at the same time with the same project. The websites Launchpad and SourceForge provide free hosting service for projects managed with Bazaar.

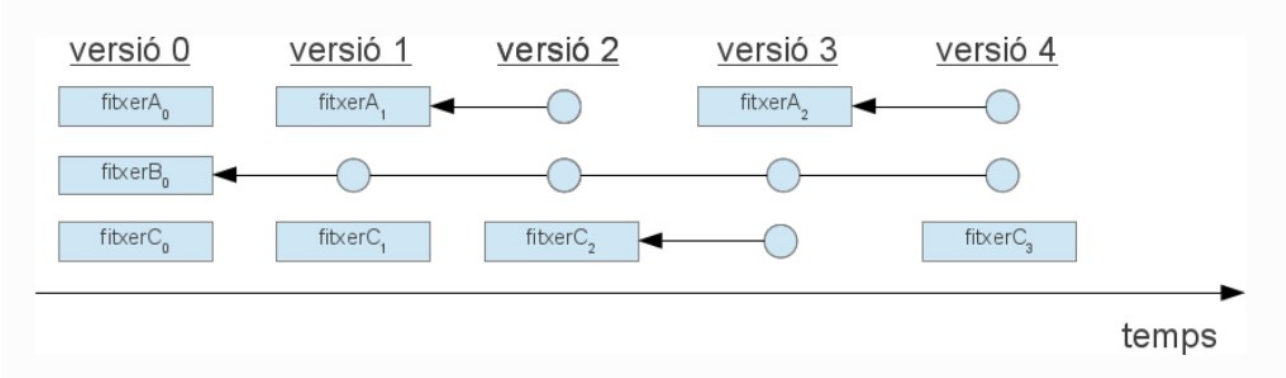
Bazaar has support for working with some other revision control systems. This allows users to branch from another system (such as Subversion), make local changes and commit them into a Bazaar branch, and then later merge them back into the other system. Read-only access is also available for Git and Mercurial. Bazaar also allows for interoperation with many other systems (including CVS, Darcs, Git, Perforce, Mercurial) by allowing one to import/export the history.

Bazaar supports files with names from the complete Unicode set. It also allows commit messages, committer names, etc. to be in Unicode.
**Impressió:**
M'ha agradat la idea de poder fer merges amb branques d'altres sistemes. Sembla un bon sistema de control de versions.

**DARCS:**
**URL del projecte: http://darcs.net/**
**Llicència: GPL 2.0 o posterior**
**Descripció:** Darcs is a free and open source, cross-platform version control system, like git, mercurial or svn but with a very different approach: focus on changes rather than snapshots. Darcs offers a freer way of working, and a simpler user interface. Darcs does not require a central server, and works perfectly in offline mode.

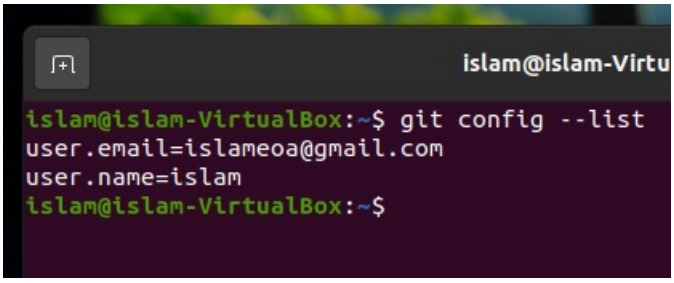
**Bondats:** Segons la seva web que es centren en els canvis en comptes de en les captures com deuen fer la resta. Tenen una interfície més simple i funciona perfectament offline. Segons viquipèdia han estat bastant criticats pel seu algoritme de merge.
**Impressió:**
Hauria de provar-lo però de tots potser es l'últim que provaria.

## Pregunta 3. Quantes versions guardem?




| versió | nombre de fitxers guardats |
|---|---|
| 0 | 3 |
| 1 | 2 |
| 2 | 1 |
| 3 | 1 |
| 4 | 1 |

## Pregunta 4. Configuració global

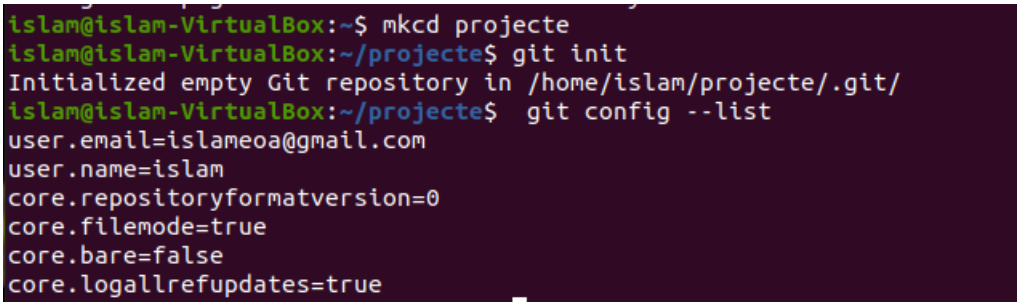


Pregunta 5. Ajut d'algunes comandes interessants

```
islam@islam-VirtualBox:~$ 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
   commit            Record changes to the repository
   merge             Join two or more development histories together
   rebase            Reapply commits on top of another base tip
   reset             Reset current HEAD to the specified state
   switch            Switch branches
   tag               Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch             Download objects and refs from another repository
   pull              Fetch from and integrate with another repository or a local branch
   push              Update remote refs along with associated objects
```

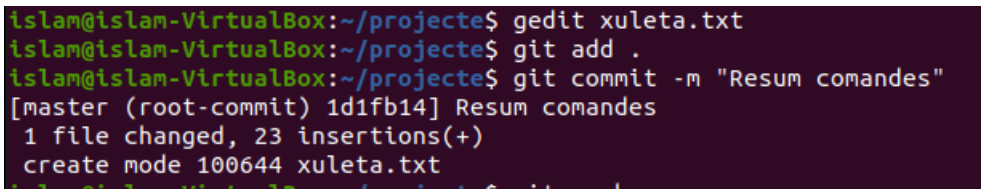
## Pregunta 6. Configuració inicial

1.Sortida:




2. Què voldrà dir el valor de core.bare?
- Si el nostre repositori té directori de treball o no, en aquest cas és false, per tant, no.

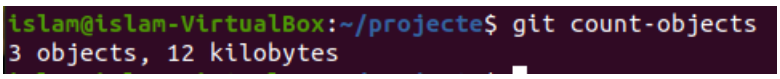
## Pregunta 7. Resum de comandes



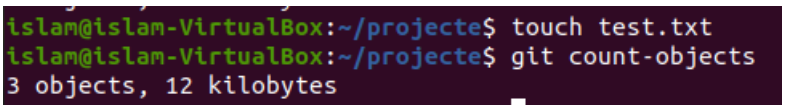

## Pregunta 8. Comptem objectes

S'ha de tenir en compte que començo amb 3 objectes perquè ja he fet commit de la 'xuleta'. Farem com que el 3 és un 0.

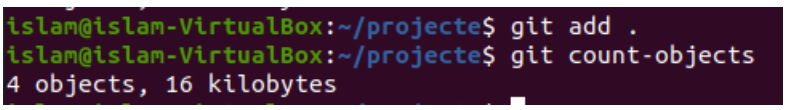
Pas 0:




Pas 1:




Pas 2:



Pas 3:

```
islam@islam-VirtualBox:~/projecte$ git commit -m "Primer commit"
[master 742c555] Primer commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test.txt
islam@islam-VirtualBox:~/projecte$ git count-objects
6 objects, 24 kilobytes
```

Pas 4:
Abans de fer commit:

```
islam@islam-VirtualBox:~/projecte$ git config --list > test.txt
islam@islam-VirtualBox:~/projecte$ git count-objects
6 objects, 24 kilobytes
```

Després de fer commit:

```
islam@islam-VirtualBox:~/projecte$ git count-objects
9 objects, 36 kilobytes
```

Pas 5:

| pas | objectes | kilobytes |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 2 | 1 | 4 |
| 3 | 3 | 12 |
| 4 | 3 | 12 |
| 5 | 6 | 24 |

# Pregunta 9. Una mica de pràctica

1,2,3,4. Creem un fitxer amb l'ip de l'ordinador i fem un commit:

```
islam@islam-VirtualBox:~/projecte$ gedit ip
islam@islam-VirtualBox:~/projecte$ git add .
islam@islam-VirtualBox:~/projecte$ git commit -am "fitxer amb ip afegit"
[master 5101f5f] fitxer amb ip afegit
 1 file changed, 13 insertions(+)
 create mode 100644 ip
```

5,6. Copiar el contingut del .gitconfig, creem un fitxer amb la data del dispositiu i ho afegim tot al control de versions:

```
islam@islam-VirtualBox:~/projecte$ gedit gitconfig
islam@islam-VirtualBox:~/projecte$ gedit date
islam@islam-VirtualBox:~/projecte$ git add .
islam@islam-VirtualBox:~/projecte$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   date
        new file:   gitconfig
```

7.Modifiquem el fitxer gitconfig i comprovem l'estat del projecte:

```
islam@islam-VirtualBox:~/projecte$ gedit gitconfig
islam@islam-VirtualBox:~/projecte$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   date
        new file:   gitconfig

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:   gitconfig
```

8. Registra tots els canvis i comproba l'estat del projecte:

```
islam@islam-VirtualBox:~/projecte$ git add .
islam@islam-VirtualBox:~/projecte$ git commit -am "gitconfig i date afegits"
[master 0a5e849] gitconfig i date afegits
 2 files changed, 5 insertions(+)
 create mode 100644 date
 create mode 100644 gitconfig
islam@islam-VirtualBox:~/projecte$ git status
On branch master
nothing to commit, working tree clean
```

9.Consultar historial de canvis:

```
islam@islam-VirtualBox:~/projecte$ git log
commit 0a5e849c390dd7aa6624edd6b6985870da465ecb (HEAD -> master)
Author: islam <islameoa@gmail.com>
Date:   Sat May 14 18:14:34 2022 +0200

    gitconfig i date afegits

commit 5101f5f1a9ce7018c26d2a6a154209662a9a2204
Author: islam <islameoa@gmail.com>
Date:   Sat May 14 17:57:36 2022 +0200

    fitxer amb ip afegit
```

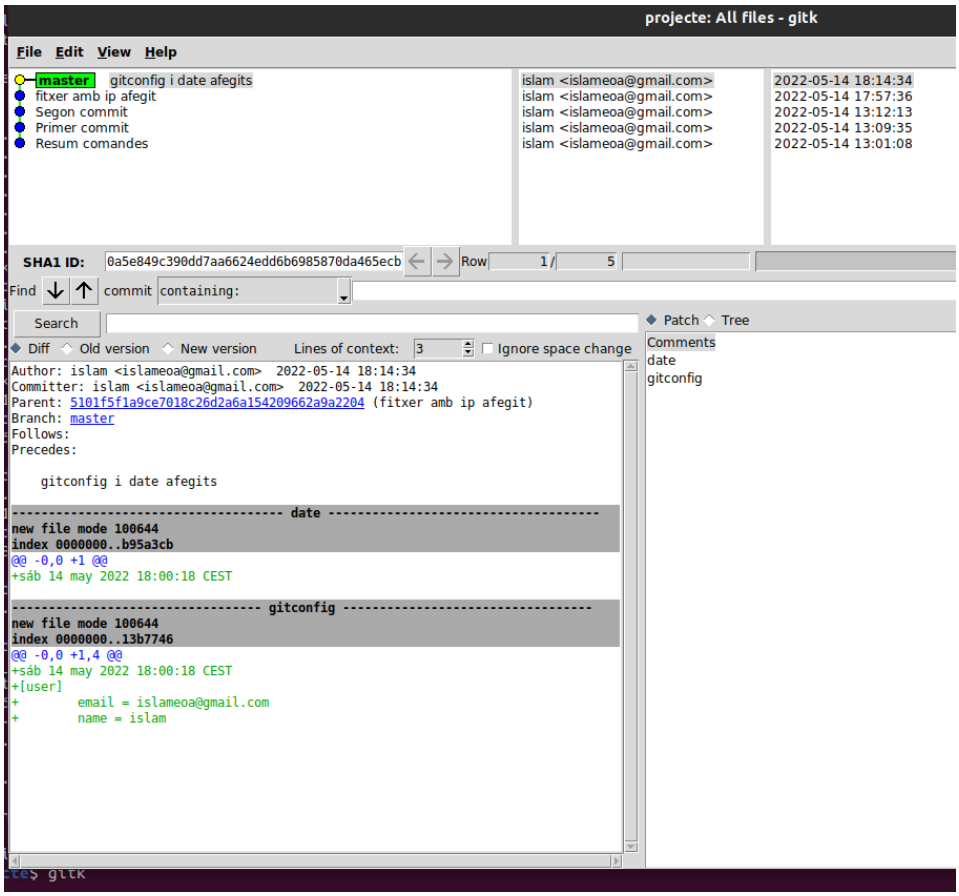
10. Ignorar els .class:

```
islam@islam-VirtualBox:~/projecte$ gedit .gitignore
islam@islam-VirtualBox:~/projecte$ cat .gitignore
target/
bin/
!.mvn/wrapper/maven-wrapper.jar

### STS ###
.apt_generated
.classpath
.factorypath
.project
.settings
.springBeans

### IntelliJ IDEA ###
.idea
*.iws
*.iml
*.ipr

### NetBeans ###
nbproject/private/
build/
nbbuild/
dist/
nbdist/
.nb-gradle/
target/
```

# Pregunta 10. Visualització

```
islam@islam-VirtualBox:~/projecte$ sudo apt-get install gitk
[sudo] password for islam:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-5.11.0-27-generic linux-hwe-5.11-headers-5.11.0-27 linux-image-5.11.0-27-generic linux-modules-5.
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libtcl8.6 libtk8.6 tcl tcl8.6 tk tk8.6
Suggested packages:
  git-doc tcl-tclreadline
The following NEW packages will be installed:
  gitk libtcl8.6 libtk8.6 tcl tcl8.6 tk tk8.6
0 upgraded, 7 newly installed, 0 to remove and 154 not upgraded.
Need to get 1.793 kB of archives.
After this operation, 8.331 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://es.archive.ubuntu.com/ubuntu focal/main amd64 libtcl8.6 amd64 8.6.10+dfsg-1 [902 kB]
```

Posem la comanda sudo i quan ens demani confirmació, escrivim y i donem intro. Després escrivim la comanda gitk i ja tenim l'aplicació:




CONCLUSIONS: Per falta de temps no he pogut aprofundir en aquests exercicis, m'hauria encantat perquè considero que el control de versions és una eina crucial per a la vida d'un programador, però considero que amb aquesta pinzellada podré defensar-me en qualsevol situació en que em faci falta utilitzar control de versions i amb això marxo content.