



CONTROL DE VERSIONS

**CFGGS Desenvolupament d'aplicacions web
M5:UF3**

Neus Bravo Arias

ÍNDEX

Pregunta 1: La meva història de por.....	3
Pregunta 2: Comparem sistemes de control de versions.....	4
Pregunta 3: Quantes versions guardem?.....	6
Pregunta 4: Configuració global.....	7
Pregunta 5: Ajut d'algunes comandes interessants.....	8
Pregunta 6: Configuració inicial.....	10
Pregunta 7: Resum de comandes.....	11
Pregunta 8: Comptem objectes.....	12
Pregunta 9: Una mica de pràctica.....	16
Pregunta 10: Visualització.....	23
Pregunta 11: L'art de la línia de comandes.....	26
Conclusions finals.....	27

INTRODUCCIÓ

La finalitat d'aquest document és familiaritzar-nos amb els control de versions i fer un recorregut a través de diferents petits exercicis introductoris que ens portaran a conèixer els conceptes més bàsics del sistema que utilitzarem en aquests curss, en aquesta ocasió farem ús de Git.

Git és el software més utilitzat en quant a control de versions i, a pesar que portem des d'inici de curs veient-lo, amb aquest petit treball profunditzarem una mica més en el seu funcionament de cara al món laboral.

El document està dividit en 11 seccions, en cadascuna d'elles anirem coneixent un nou concepte relacionat amb el control de versions.

A la primera secció fem una reflexió en quant a la necessitat d'utilitzar un CVS.

Al segon punt, parlarem dels diferents tipus de cvs que existeixen i veurem les seves característiques. La secció número 3 ens endinsa en el funcionament del sistema. Després començarem amb la configuració global i veurem les primeres comandes i les més bàsiques. A l'apartat 6 comencem a experimentar amb el sistema i acte seguit ens creem una petita cheat-sheet amb les comandes que anem aprenent.

A la part final, posarem en pràctica els conceptes bàsics i registrarem amb captures de pantalla tots els passos dels processos que estem seguint.

Pregunta 1: La meva història de por

Describeix un cas concret de la teva experiència que hagis trobat amb un o més dels problemes descrits a la secció Justificació. Fes-ho en els termes indicats, per exemple, si ha estat entre molts treballadors o de manera telemàtica.

Concreta molt. Per exemple, descriu la convenció de noms que has fet servir per cada versió.

Jo em sento bastant identificada amb l'apartat que parla de les recuperacions de codi ja que és una cosa que em passa sempre, sempre, SEMPRE.

Ara mateix no es difícil de gestionar perquè només sóc jo amb el meu portàtil intentat fer un petit programa, però l'emprenyament és immens cada vegada que començo a modificar el codi per arreglar els errors que em vaig trobant. Sempre que un tros del codi que crec (i dic crec perquè també m'ha passat que aquesta part funciona perfectament abans de començar jo a fer coses estranyes) que és el que m'està donant problemes, tinc la mala mania d'esborrar-ho tot de cop sense pensar perquè dins de mi mateixa tinc la falsa certesa que si ho he fet una vegada podré tornar a escriure-ho com si res una segona vegada si ho necessito. Això em porta a que quan descobreixo que el programa ara em peta inclòs més del que em petava abans, jo ja he esborrat el codi anterior que sí funcionava i per sorpresa meua, ja no sé com havia fet l'anterior. A l'estar ja compilat i testejat no tinc forma de recuperar el que tenia i em trobo tenint que refer les mateixes coses una i altra vegada per les meves dolentes costums de borrar sense pensar.

Pregunta 2: Comparem sistemes de control de versions

Agafa els diferents exemples de controls de versions que hi apareixen a la secció Definició i crea un fitxer per cadascun d'ells amb el màxim que trobis dels següents punts:

- **nom del cvs**
- **url del projecte**
- **llicència**
- **descripció**
- **bondats**
- **impressió**

SCV Centralitzats

La seva pàgina oficial és: <https://www.perforce.com/>

És un software de control de versions que comença sent gratuït i a mida que el teu equip creix et creen un pressupost ajustat a la teva necessitat de llicències segons les funcionalitats que utilitzaran.

"Perforce Helix Core is the leading version control system for teams who need to accelerate innovation at scale. Store and track changes to all your digital assets, from source code to binary to IPs. Connect your teams and empower them to move faster and build better."

Com a avantatges, principalment destaquen que és un software totalment gratuït per petites empreses. També té el benefici que es pot integrar amb Git i amb tot tipus de softwares com IDE's i altres programes, a això s'afegeix el fet que el control de versions es dur a terme al background, per tant no és necessari deixar de treballar en cap moment per comprovar si tot s'està executant de bona manera.

Conclusió: al no conèixer gaire softwares d'aquest tipus i no haver vist molts, m'ha impressionat bastant la funcionalitat que ofereixen i d'aquest en concret m'ha agradat moltíssim la presentació del producte en comparació a altres webs que ho mostren d'una forma més simple i antiga. Sense dubtes em quedaria amb aquest si hagués d'escollir un.

SCV Distribuïts

La pàgina oficial del escollita és: <https://git-scm.com/>

Git és el software per excel·lència dins del món de programes de control de versions. Començant pel fet que és un sistema distribuït, gratis i open-source designat per treballar amb velocitat tant en projectes petits com en projectes d'empreses multinacionals.

"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."

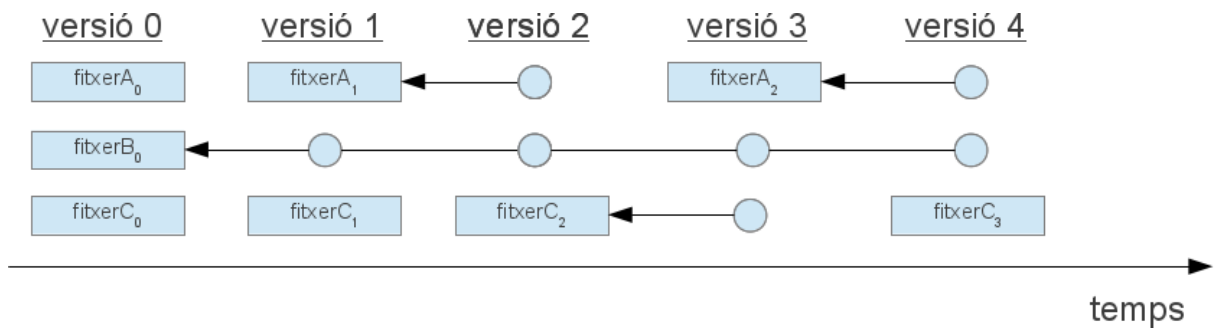
Com a avantatges del sistema, ells mateixos els listen en aquests: Branching and Merging, Small and fast, Distributed, Data assurance, Staging area, Free and open source, Trademark. Es destaca el fet que és un sistema distribuït, free i open source, que compta amb un model criptogràfic per assegurar les dades i que es dur a terme amb una velocitat inigualable degut a la forma en la que actuen les operacions.

Conclusió: si és el més utilitzat al món ha de ser amb motius i aquesta és la sensació que transmet des de que llegeixes la documentació que aporta la pàgina web. El que més m'animaria a escollir-ho és el fet que existeixen una quantitat immensa de rutes d'aprenentatge per aprendre a utilitzar-ho i moltíssims recursos per dominar-ho.

Impressió general: sense dubtes i sense haver utilitzat cap encara, crec que els controls de versions són uns sistemes extremadament necessaris i de gran ajuda per la feina que ens pertoca aviat. Tots tenen les seves avantatges personals però Git es porta la meua confiança si hagués d'escollir un per aprendre.

Pregunta 3: Quantes versions guardem?

Recorda l'esquema que varem veure a la introducció i completa la taula següent, indicant quants fitxers(no enllaços) són guardats realment per Git a cada versió d'aquest exemple.

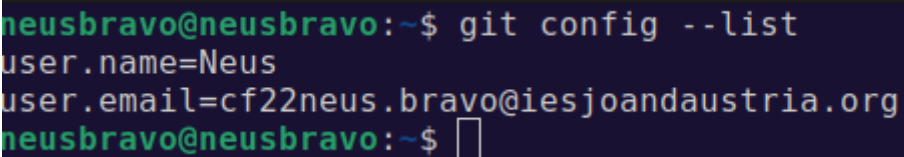


versi ó	nombre de fitxers guardats
0	3
1	2
2	1
3	1
4	1

Pregunta 4: Configuració global

*Un cop disposes de Git instal·lat i configurat al teu sistema, crida la següent instrucció a la consola: **git config --list***

La resposta a aquest exercici és el resultat d'aquesta consola. Revisa que les dades siguin les que esperes.

A screenshot of a terminal window with a dark background. The prompt is 'neusbravo@neusbravo:~\$'. The command 'git config --list' has been entered, and the output is displayed on the next two lines: 'user.name=Neus' and 'user.email=cf22neus.bravo@iesjoandaustria.org'. The prompt 'neusbravo@neusbravo:~\$' is shown again with a cursor at the end.

```
neusbravo@neusbravo:~$ git config --list
user.name=Neus
user.email=cf22neus.bravo@iesjoandaustria.org
neusbravo@neusbravo:~$
```


Pregunta 5: Ajut d'algunes comandes interessants

Obté ajuda per les següents opcions de git:

- **clone**
- **init**
- **add**
- **mv**
- **reset**
- **rm**
- **log**
- **status**
- **checkout**
- **commit**

Per cadascuna d'aquestes opcions, afegeix al document:

- *la comanda per obtenir l'ajut*
- *l'ajut generat per git. Només cal que indiqueu la breu descripció que apareix a la secció 'Name' i la descripció de la secció 'Description'.*

git help init: git-init - Create an empty Git repository or reinitialize an existing one.

This command creates an empty repository - basically a .git directory with subdirectories for objects, refs/heads, refs/tags, and template files.

git help clone: Clone a repository into a new directory. Clones a repository into a newly created directory, creates remotes-tracking branches for each branch in the cloned repository (visible using git branch --remotes), and creates and checks out an initial branch that is forked from the cloned repository's currently active branch.

git help add: Add files content to the index. Updates the index using the current content found in the working tree, to prepare the content stage for the next commit.

git help mv: Move or rename a file, a directory, or a symlink.

git help reset: Reset current head to the specified state. In the first three forms, copy entries from <tree-ish> to the index. In the last form, set the current branch head (HEAD) to <commit>, optionally modifying index and working tree to match. The <tree-ish>/<commit> defaults to HEAD in all forms.

git help rm: Remove files from the working tree and from the index. In the first three forms, copy entries from <tree-ish> to the index. In the last form, set the current branch head (HEAD) to <commit>, optionally modifying index and working tree to match. The <tree-ish>/<commit> defaults to HEAD in all forms.

git help log: Show commit logs. List commits that are reachable by following the parent links from the given commit(s), but exclude commits that are reachable from the one(s) given with a ^ in front of them. The output is given in reverse chronological order by default.

git help status: Show the working tree status. Displays paths that have differences between the index file and the current HEAD commit, paths that have differences between the working tree and the index file, and paths in the working tree that are not tracked by Git (and are not ignored by gitignore(5)).

git help checkout: Switch branches or restore working tree files. Updates files in the working tree to match the version in the index or the specified tree. If no pathspec was given, git checkout will also update HEAD to set the specified branch as the current branch.

git help commit: Record changes to the repository. Create a new commit containing the current contents of the index and the given log message describing the changes.

Pregunta 6: Configuració inicial

*Un cop hakis fet la creació d'un repositori, crida la següent comanda per consola: **git config --list***

1. Quina sortida t'ha generat la comanda anterior?

```
neusbravo@neusbravo:~/AprendiendoGit$ git config --list
user.name=Neus
user.email=cf22neus.bravo@iesjoandaustria.org
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
neusbravo@neusbravo:~/AprendiendoGit$
```

2. Què voldria dir el valor de **core.bare?**

El terme “bare” és un valor boolean que en true fa referència a que un repositori no té un directori de treball associat i per tant no té una còpia dels arxius del projecte al seu sistema de fitxers, s'utilitza més com un directori remot per compartir historial de canvis amb altres repositoris. En false, sí té un directori associat.

Pregunta 7: Resum de comandes

Creat un petit resum de comandes que han aparegut en aquesta introducció a Git. Pensa-ho com una cheat-sheet.

commit -m <<comentaris>>

Registra els canvis a stage associant-los amb un comentari.

git clone <<directory>>

Copia un repositori en un directori concret amb totes les seves branques i historial de versions.

git init

Crea un nou repositori dins d'un directori.

git add <<file>>

Afegeix canvis de arxius al stage pel proper commit

git stash

Emmagatzema temporalment tots els arxius modificats del quals es té almenys una versió guardada.

git status

Enumera tots els arxius nous o modificats els quals es guardaran canvis.

git branch

Enumera totes les branques al repositori actual.

git checkout <<branca>>

Canvia de branca

git merge <<branca>>

Fusiona la branca actual amb la seleccionada

Pregunta 8: Comptem objectes

A partir dels continguts de 'Passar a stage', crea un nou repositori.

Pas 0. Compta el objectes que està guardant Git en aquest moment.

```
neusbravo@neusbravo:~/tmpo$ git init
ayuda: Usando 'master' como el nombre de la rama inicial. Este nombre de rama predeterminado
ayuda: está sujeto a cambios. Para configurar el nombre de la rama inicial para usar en todos
ayuda: de sus nuevos repositorios, reprimiendo esta advertencia, llama a:
ayuda:
ayuda: git config --global init.defaultBranch <nombre>
ayuda:
ayuda: Los nombres comúnmente elegidos en lugar de 'master' son 'main', 'trunk' y
ayuda: 'development'. Se puede cambiar el nombre de la rama recién creada mediante este comando:
ayuda:
ayuda: git branch -m <nombre>
Inicializado repositorio Git vacío en /home/neusbravo/tmpo/.git/
neusbravo@neusbravo:~/tmpo$ git count-objects
0 objects, 0 kilobytes
neusbravo@neusbravo:~/tmpo$
```

Pas 1. Crea un fitxer anomenat 'test.txt' però de moment no l'afegeixis a Git.

```
neusbravo@neusbravo:~/tmpo$ echo "hola" > text.txt
neusbravo@neusbravo:~/tmpo$ git count-objects
0 objects, 0 kilobytes
neusbravo@neusbravo:~/tmpo$
```

Pas 2. Afegeix el fitxer a Git però no facis commit encara.

```
neusbravo@neusbravo:~/tmpo$ git count-objects
1 objects, 4 kilobytes
neusbravo@neusbravo:~/tmpo$
```

Pas 3. Fes el primer commit i compta el objectes guardats.

```
neusbravo@neusbravo:~/tmpo$ git commit -m "Registre inicial"
[master (commit-raíz) 9b23970] Registre inicial
 1 file changed, 1 insertion(+)
 create mode 100644 text.txt
neusbravo@neusbravo:~/tmpo$ git count-objects
3 objects, 12 kilobytes
neusbravo@neusbravo:~/tmpo$
```

Pas 4. Modifica el fitxer i abans de fer commit compta quants objectes hi ha.

```
neusbravo@neusbravo:~/tmpo$ git config --list > text.txt
neusbravo@neusbravo:~/tmpo$ git count-objects
3 objects, 12 kilobytes
neusbravo@neusbravo:~/tmpo$ □
```

Pas 5. Torna a fer commit i compta.

```
neusbravo@neusbravo:~/tmpo$ git commit -am "Canvis"
[master 65747e0] Canvis
 1 file changed, 6 insertions(+), 1 deletion(-)
neusbravo@neusbravo:~/tmpo$ git count-objects
6 objects, 24 kilobytes
neusbravo@neusbravo:~/tmpo$ □
```

Emplena la següent taula:

<i>pa s</i>	<i>objecte s</i>	<i>kilobytes</i>
0	0	0
1	0	0
2	1	4
3	3	12
4	3	12
5	6	24

Pregunta 9: Una mica de pràctica

Practicarem els continguts treballats a 'Branques'.

1. *Personalitza el teu Git.*
2. *Crea un repositori Git.*

```
neusbravo@neusbravo:~$ mkdir Branques
neusbravo@neusbravo:~$ cd Branques/
neusbravo@neusbravo:~/Branques$ git init
ayuda: Usando 'master' como el nombre de la rama inicial. Este nombre de rama predeterminado
ayuda: está sujeto a cambios. Para configurar el nombre de la rama inicial para usar en todos
ayuda: de sus nuevos repositorios, reprimiendo esta advertencia, llama a:
ayuda: git config --global init.defaultBranch <nombre>
ayuda: Los nombres comúnmente elegidos en lugar de 'master' son 'main', 'trunk' y
ayuda: 'development'. Se puede cambiar el nombre de la rama recién creada mediante este comando:
ayuda: git branch -m <nombre>
Iniciado repositorio Git vacío en /home/neusbravo/Branques/.git/
neusbravo@neusbravo:~/Branques$
```

3. *Afegeix al directori un fitxer amb contingut generat per la comanda 'ip a' des de la teva màquina.*

```
neusbravo@neusbravo:~/Branques$ ip a >> ip.txt
neusbravo@neusbravo:~/Branques$ git add ip.txt
neusbravo@neusbravo:~/Branques$
```

4. *Fes el primer commit.*

```
neusbravo@neusbravo:~/Branques$ git commit -m "Registre inicial ip a"
[master (commit-raíz) 8135f38] Registre inicial ip a
1 file changed, 14 insertions(+)
 create mode 100644 ip.txt
neusbravo@neusbravo:~/Branques$
```

5. *Copia el contingut del teu fitxer de configuració '.gitconfig' a un fitxer dins del directori i afegeix-ho al control de versions.*

```
neusbravo@neusbravo:~/Branques$ git add config.txt
neusbravo@neusbravo:~/Branques$
```

6. *Crea un nou fitxer on afegeixis la data i hora del sistema(date), afegeix-lo a stage i comprova l'estat del projecte.*

```
neusbravo@neusbravo:~/Branques$ date >> datahora.txt
neusbravo@neusbravo:~/Branques$ git add datahora.txt
neusbravo@neusbravo:~/Branques$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    nuevos archivos: config.txt
    nuevos archivos: datahora.txt
neusbravo@neusbravo:~/Branques$
```


7. Modifica el primer fitxer tot afegint la data i l'hora i comprova l'estat del projecte.

```
neusbravo@neusbravo:~/Branques$ date > ip.txt
neusbravo@neusbravo:~/Branques$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    nuevos archivos: config.txt
    nuevos archivos: datahora.txt

Cambios no rastreados para el commit:
  (usa "git add <archivo>..." para actualizar lo que será confirmado)
  (usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)
    modificados:    ip.txt

neusbravo@neusbravo:~/Branques$
```

8. Registra el canvis i torna a comprovar l'estat.

```
neusbravo@neusbravo:~/Branques$ git commit -m "Afegit date a fitxer ip"
[master 88d75e1] Afegit date a fitxer ip
2 files changed, 4 insertions(+)
create mode 100644 config.txt
create mode 100644 datahora.txt
neusbravo@neusbravo:~/Branques$ git status
En la rama master
Cambios no rastreados para el commit:
  (usa "git add <archivo>..." para actualizar lo que será confirmado)
  (usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)
    modificados:    ip.txt

sin cambios agregados al commit (usa "git add" y/o "git commit -a")
neusbravo@neusbravo:~/Branques$
```

```
neusbravo@neusbravo:~/Branques$ git add ip.txt
neusbravo@neusbravo:~/Branques$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    modificados:    ip.txt

neusbravo@neusbravo:~/Branques$
```

9. Consulta l'historial de canvis fins el moment.

```

neusbravo@neusbravo:~/Branques$ git log
commit 88d75e1daf3a4ddd9bc93a44ed6e56903c15af6b (HEAD -> master)
Author: Neus Bravo Arias <cf22neus.bravo@iesjoandaustria.org>
Date:   Wed Apr 26 22:14:06 2023 +0200

    Afegit date a fitxer ip

commit 8135f3862640bb04fc6301ce18756941373e7b09
Author: Neus Bravo Arias <cf22neus.bravo@iesjoandaustria.org>
Date:   Wed Apr 26 20:49:10 2023 +0200

    Registre inicial ip a
neusbravo@neusbravo:~/Branques$ █

```

10. Configura el teu projecte de manera que Git no tracti de gestionar el `.class`. Realitza les accions necessàries per comprovar que no ho està fent.

```

neusbravo@neusbravo:~/Branques$ touch .gitignore
neusbravo@neusbravo:~/Branques$ vim .gitignore
neusbravo@neusbravo:~/Branques$ cat .gitignore
*.class
neusbravo@neusbravo:~/Branques$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    modificados:   ip.txt

Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
    .gitignore
neusbravo@neusbravo:~/Branques$ █

```

11. Realitza les accions necessàries per demostrar com es pot veure els canvis realitzats al contingut d'un fitxer que encara no s'ha passat a stage.

```

neusbravo@neusbravo:~/Branques$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    modificados:    ip.txt

Cambios no rastreados para el commit:
  (usa "git add <archivo>..." para actualizar lo que será confirmado)
  (usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)
    modificados:    ip.txt

Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
    .gitignore

neusbravo@neusbravo:~/Branques$ git diff
diff --git a/ip.txt b/ip.txt
index 9db5537..22d0f91 100644
--- a/ip.txt
+++ b/ip.txt
@@ -1,1 +1 @@
-mié 26 abr 2023 22:13:05 CEST
+nou contingut
neusbravo@neusbravo:~/Branques$

```

12. Demostra com comprovar els canvis realitzats a un fitxer respecte el darrer commit.

```

neusbravo@neusbravo:~/Branques$ git diff HEAD
diff --git a/ip.txt b/ip.txt
index 614a546..22d0f91 100644
--- a/ip.txt
+++ b/ip.txt
@@ -1,14 +1 @@
-1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
-   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
-   inet 127.0.0.1/8 scope host lo
-       valid_lft forever preferred_lft forever
-   inet6 ::1/128 scope host
-       valid_lft forever preferred_lft forever
-2: enp3s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
-   link/ether c0:18:50:8f:bb:8a brd ff:ff:ff:ff:ff:ff
-3: wlp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
-   link/ether 8c:b8:7e:fa:55:9b brd ff:ff:ff:ff:ff:ff
-   inet 192.168.18.150/22 brd 192.168.19.255 scope global dynamic noprefixroute wlp2s0
-       valid_lft 86028sec preferred_lft 86028sec
-   inet6 fe80::cbb6:cfb:c599:8fd5/64 scope link noprefixroute
-       valid_lft forever preferred_lft forever
+nou contingut
neusbravo@neusbravo:~/Branques$

```

13. Consulta l'històric de canvis d'un dels fitxers incloent les diferències de cada versió respecte l'anterior.

```

neusbravo@neusbravo:~/Branques$ git log -p ip.txt
commit 8135f3862640bb04fc6301ce18756941373e7b09
Author: Neus Bravo Arias <cf22neus.bravo@iesjoandaustria.org>
Date:   Wed Apr 26 20:49:10 2023 +0200

    Registre inicial ip a

diff --git a/ip.txt b/ip.txt
new file mode 100644
index 0000000..614a546
--- /dev/null
+++ b/ip.txt
@@ -0,0 +1,14 @@
+1: lo: <LOOPBACK,UP,LOWER UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
+   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
+   inet 127.0.0.1/8 scope host lo
+       valid_lft forever preferred_lft forever
+   inet6 ::1/128 scope host
+       valid_lft forever preferred_lft forever
+2: enp3s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 1000
+   link/ether c0:18:50:8f:bb:8a brd ff:ff:ff:ff:ff:ff
+3: wlp2s0: <BROADCAST,MULTICAST,UP,LOWER UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
+   link/ether 8c:b8:7e:fa:55:9b brd ff:ff:ff:ff:ff:ff
+   inet 192.168.18.150/22 brd 192.168.19.255 scope global dynamic noprefixroute wlp2s0
+       valid_lft 86028sec preferred_lft 86028sec
+   inet6 fe80::cbb6:cfb:c599:8fd5/64 scope link noprefixroute
+       valid_lft forever preferred_lft forever
neusbravo@neusbravo:~/Branques$

```

- 14. Realitza les comandes necessàries per demostrat com es pot tornar a l'estat del darrer commit d'un determinat fitxer.**

```

neusbravo@neusbravo:~/Branques$ git checkout ip.txt
Actualizada l ruta desde el índice
neusbravo@neusbravo:~/Branques$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    modificados:   ip.txt

Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
    .gitignore
neusbravo@neusbravo:~/Branques$

```

- 15. Crea una nova branca, modifica un dels fitxers i registra els canvis. Comprova que els canvis hi són a la nova branca però no a la branca mestre. Torna a la branca mestre, fusiona el canvis de la nova branca i elimina la nova branca. Comprova que s'ha eliminat.**

```

neusbravo@neusbravo:~/Branques$ git branch prova
neusbravo@neusbravo:~/Branques$ git checkout prova
M   ip.txt
Cambiado a rama 'prova'
neusbravo@neusbravo:~/Branques$ echo "afegixo nou contingut al fitxer" >> ip.txt
neusbravo@neusbravo:~/Branques$ git add ip.txt
neusbravo@neusbravo:~/Branques$ git commit -m "nous canvis afegits"
[prova 547cf3d] nous canvis afegits
 1 file changed, 2 insertions(+), 14 deletions(-)
 rewrite ip.txt (100%)
neusbravo@neusbravo:~/Branques$ git status
En la rama prova
Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
    .gitignore

no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa "git add" para hacerles seguimiento)
neusbravo@neusbravo:~/Branques$

```

```

neusbravo@neusbravo:~/Branques$ git checkout master
Cambiado a rama 'master'
neusbravo@neusbravo:~/Branques$ git status
En la rama master
Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
    .gitignore

no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa "git add" para hacerles seguimiento)
neusbravo@neusbravo:~/Branques$ git merge prova
Actualizando 88d75e1..547cf3d
Fast-forward
 ip.txt | 16 ++++++
 1 file changed, 2 insertions(+), 14 deletions(-)
neusbravo@neusbravo:~/Branques$ git branch -d prova
Eliminada la rama prova (era 547cf3d).
neusbravo@neusbravo:~/Branques$ git branch
* master
neusbravo@neusbravo:~/Branques$ 

```

16. Crea una nova branca i realitza els canvis necessaris sobre algun dels fitxers, de manera que les dues branques divergeixen. Aconsegueix que en intentar fusionar els canvis a la branca mestre algun dels fitxers modificats sigui fusionat automàticament però algun requereixi modificacions manuals. Finalment hauria de quedar tots els canvis fusionats a la branca mestre i l'altra branca eliminada.

```

neusbravo@neusbravo:~/Branques$ echo "canvi des de master" >> ip.txt
neusbravo@neusbravo:~/Branques$ git commit -am "canvi fet des de branch master"
[prova 264f390] canvi fet des de branch master
 1 file changed, 2 insertions(+)
neusbravo@neusbravo:~/Branques$ git checkout prova
Ya en 'prova'
neusbravo@neusbravo:~/Branques$ git checkout master
Cambiado a rama 'master'
neusbravo@neusbravo:~/Branques$ git commit -am "Canvi desde master de verdad"
En la rama master
Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
    .gitignore

no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa "git add" para hacerles seguimiento)
neusbravo@neusbravo:~/Branques$ echo "canvi desde master de verdad" >> ip.txt
neusbravo@neusbravo:~/Branques$ git commit -am "Canvi desde mster de verdad verdad"
[master 5ae8418] Canvi desde mster de verdad verdad
 1 file changed, 1 insertion(+)
neusbravo@neusbravo:~/Branques$ git merge prova
Auto-fusionando ip.txt
CONFLICTO (contenido): Conflicto de fusión en ip.txt
Fusión automática falló; arregle los conflictos y luego realice un commit con el resultado.
neusbravo@neusbravo:~/Branques$ cat ip.txt
mié 26 abr 2023 22:13:05 CEST
afegeixo nou contingut al fitxer
<<<<<< HEAD
canvi desde master de verdad
=====
línea escrita des de master
canvi des de master
>>>>>> prova
neusbravo@neusbravo:~/Branques$ 

```

```

neusbravo@neusbravo:~/Branques$ git checkout prova
ip.txt: needs merge
error: necesitas resolver tu índice actual primero
neusbravo@neusbravo:~/Branques$ git branch
* master
  prova
neusbravo@neusbravo:~/Branques$ vim ip.txt
neusbravo@neusbravo:~/Branques$ git commit -am "Arreglat conflicte merge"
[master c26f368] Arreglat conflicte merge
neusbravo@neusbravo:~/Branques$ git status
En la rama master
Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
    .gitignore

no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa "git add" para hacerles seguimiento)
neusbravo@neusbravo:~/Branques$ git merge prova
Ya está actualizado.
neusbravo@neusbravo:~/Branques$ git branch -d prova
Eliminada la rama prova (era 264f390).
neusbravo@neusbravo:~/Branques$ 

```

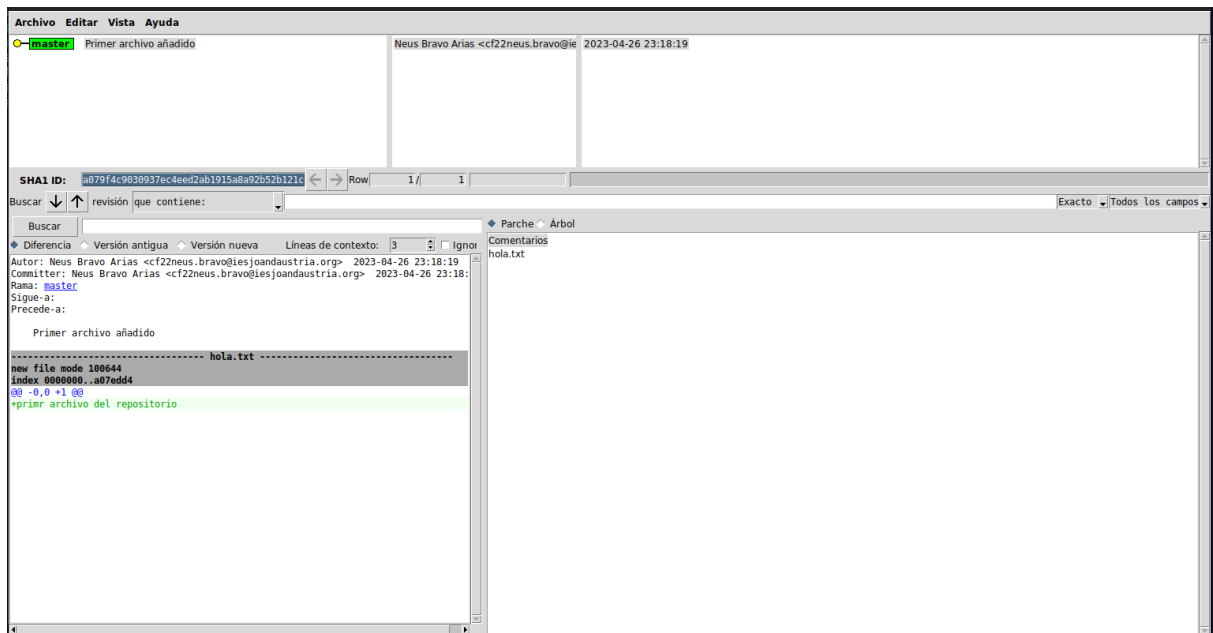
17. Guarda tot el contingut de la branca mestre en un paquet Git.

```
neusbravo@neusbravo:~/Branques$ git branch
* master
neusbravo@neusbravo:~/Branques$ git bundle create paquete master
Enumerando objetos: 19, listo.
Contando objetos: 100% (19/19), listo.
Comprimiendo objetos: 100% (17/17), listo.
Total 19 (delta 8), reusados 0 (delta 0), pack-reusados 0
neusbravo@neusbravo:~/Branques$
neusbravo@neusbravo:~/Branques$ ls
config.txt  datahora.txt  ip.txt  paquete
neusbravo@neusbravo:~/Branques$
```

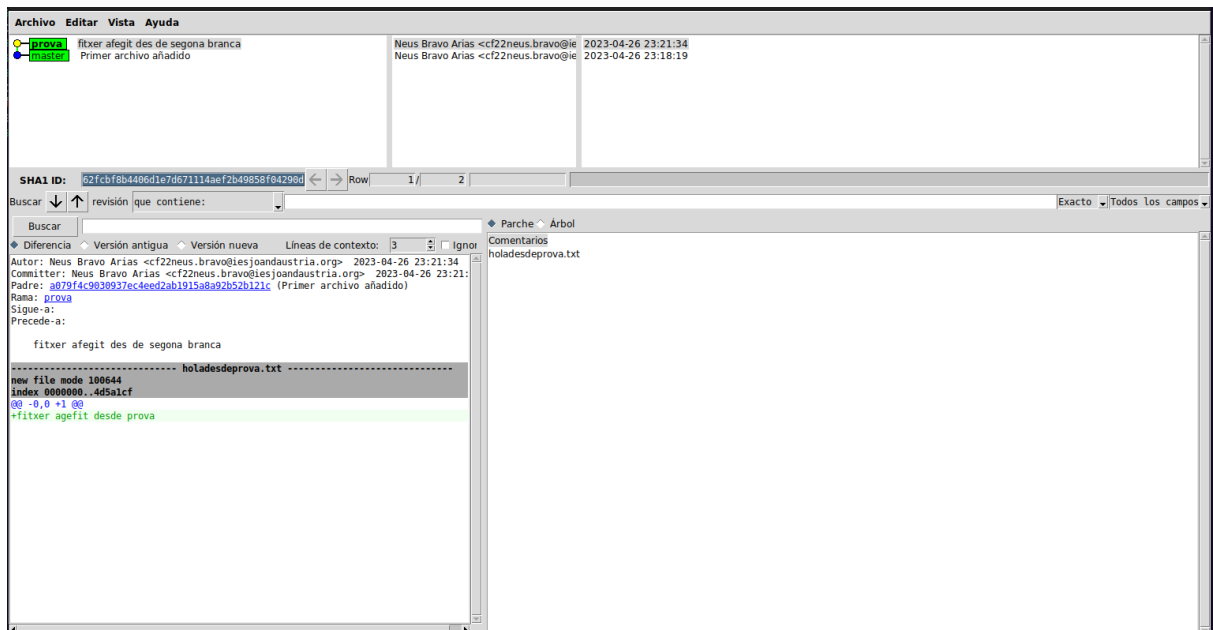
Pregunta 10: Visualització

*Instal·la l'eina **gitk**. Aquestes eines permeten navegar de manera més visual per l'històric de canvis.*

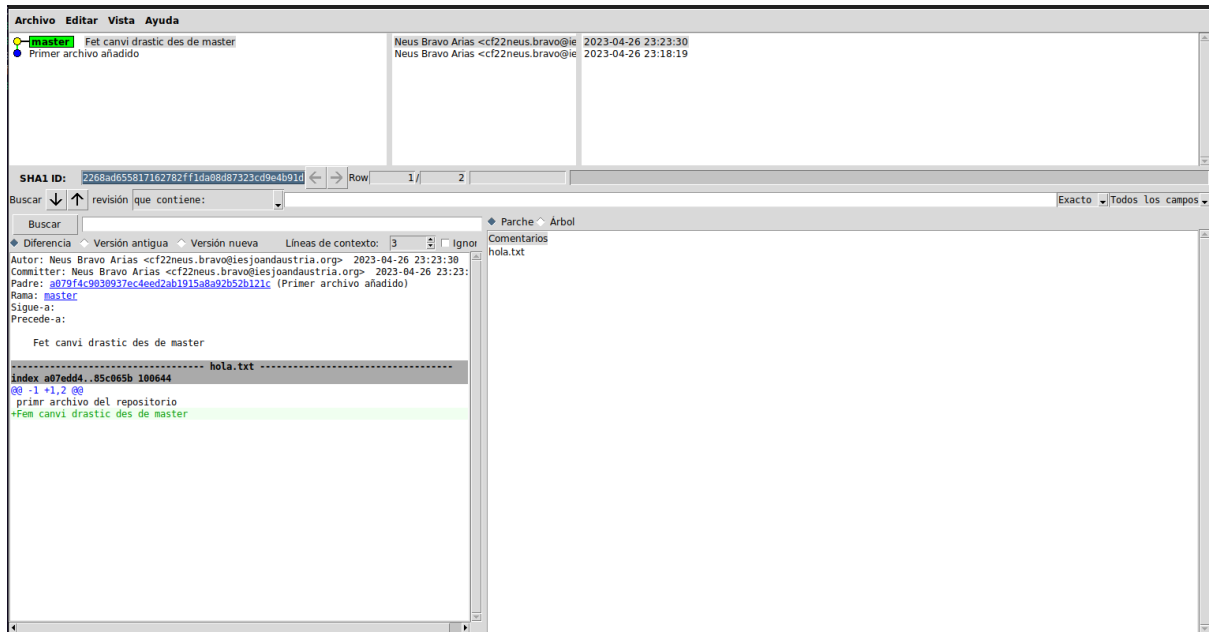
Desenvolupa un petit tutorial de la instal·lació que inclogui captures de pantalla de l'eina amb el projecte anterior.



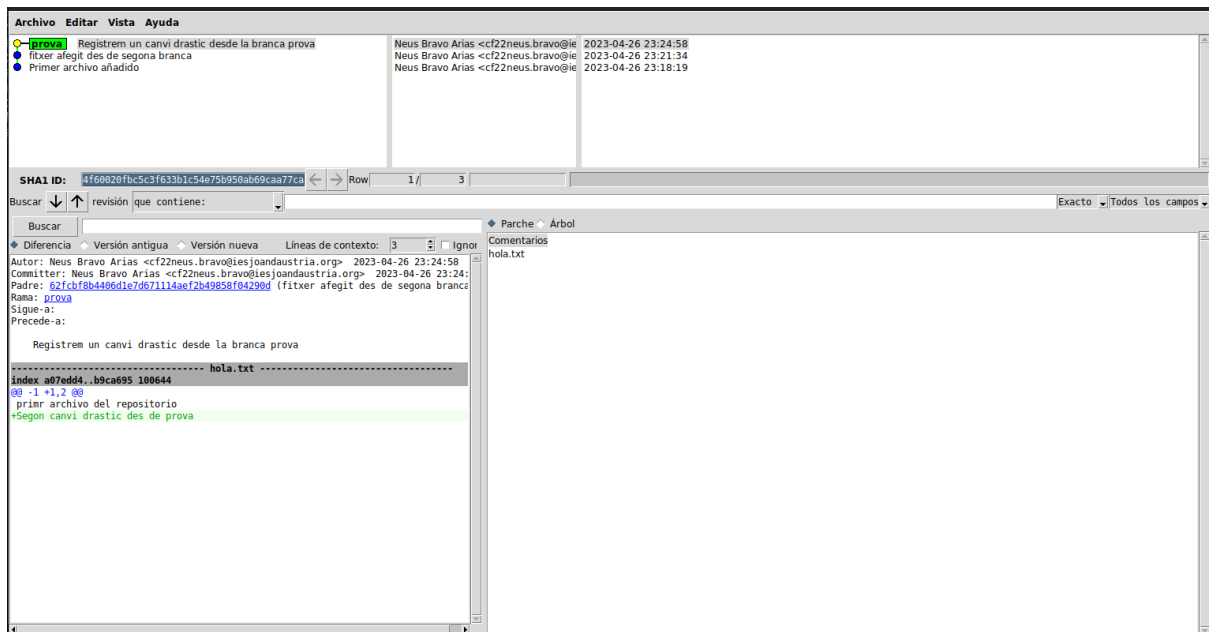
Podem veure l'històric de canvis del repositori amb tots els seus detalls. Hem afegit el primer fitxer a la branca master.



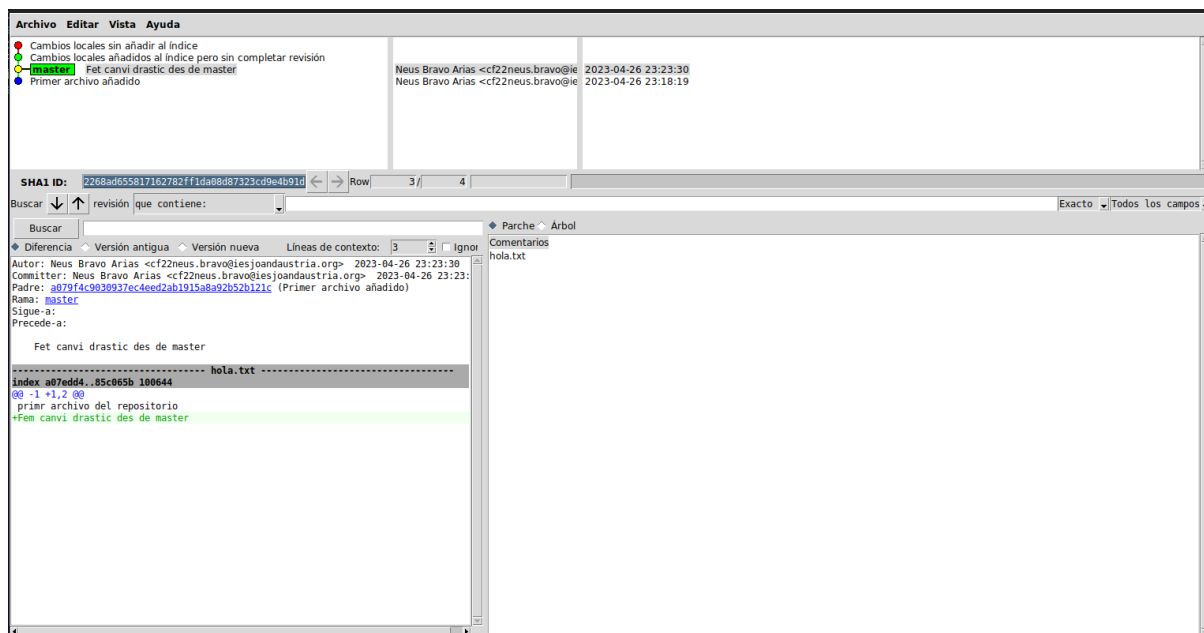
Creem una segona branca i afegim un nou fitxer, podem veure les dues branques diferenciades.



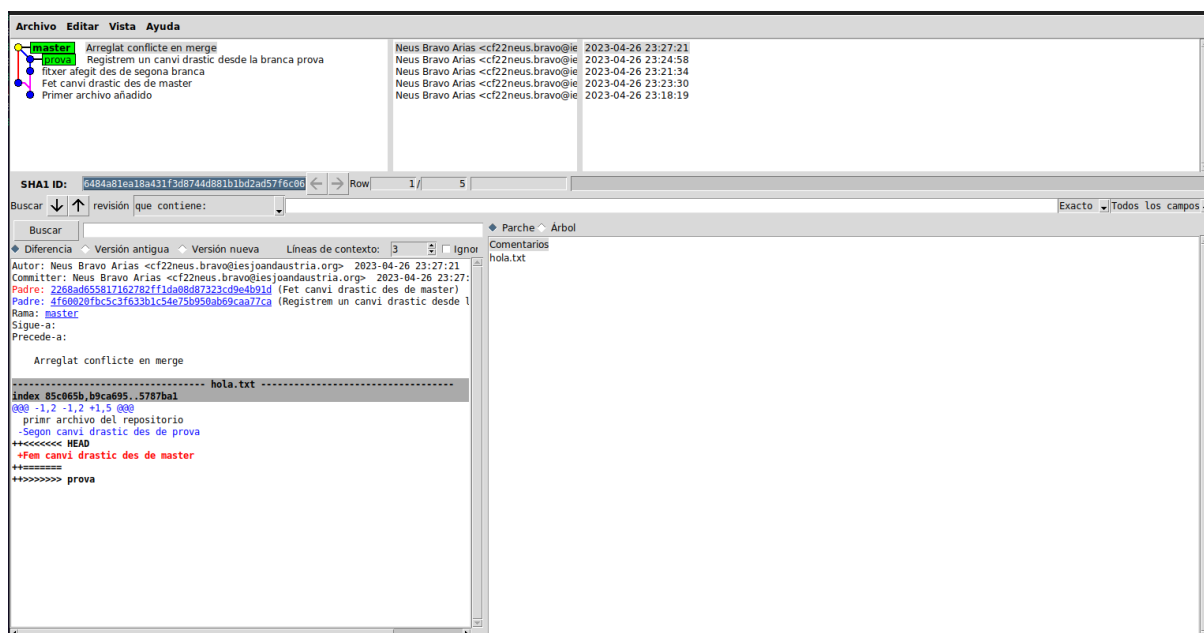
Aquí realitzem el primer canvi des de master a un fitxer.



Canvi realitzat des de la branca prova.



Intentem fusionar les dues branques a les que hem aplicat canvis i veiem com marca el conflicte en vermell.



Arreglem manualment el conflicte de merge i apareix solucionat.

Pregunta 11: L'art de la línia de comandes

Afegim una branca més a un repositori del qual hem fet un fork.

1. Clona'l al teu equip.

```
neusbravo@neusbravo:~$ git clone https://github.com/neusba/the-art-of-command-line.git
Clonando en 'the-art-of-command-line'...
remote: Enumerating objects: 3577, done.
remote: Total 3577 (delta 0), reused 0 (delta 0), pack-reused 3577
Recibiendo objetos: 100% (3577/3577), 2.58 MiB | 2.05 MiB/s, listo.
Resolviendo deltas: 100% (2108/2108), listo.
neusbravo@neusbravo:~$ ls
```

2. Crea una nova branca i anomena-la **catalan**.

```
neusbravo@neusbravo:~/the-art-of-command-line$ git branch catalan
neusbravo@neusbravo:~/the-art-of-command-line$ git branch
catalan
* master
neusbravo@neusbravo:~/the-art-of-command-line$
```

3. Afegeix-hi un nou fitxer anomenat **README-ca.md** amb els continguts del fitxer **README.md**.

<https://github.com/neusba/the-art-of-command-line>

```
neusbravo@neusbravo:~/the-art-of-command-line$ cat README.md >> README-ca.md
neusbravo@neusbravo:~/the-art-of-command-line$
```

4. Comença a fer la traducció dels seus continguts. Com a mínim la línia 27.

```
# TRADUCCIO AL CATALA #
La fluïdesa a la línia de comandes és una habilitat freqüentment oblidada o considerada un misteri, però millora la nostra flexibilitat i productivitat com ingeniers d'una manera obvia i subtil. Això serà una selecció d'apunts i consells per utilitzar la terminal que considerem d'utilitat a l'hora de treballar amb Linux. Alguns consells són bàsics i d'altres més específics i sofisticats. No serà una pàgina llarga, però si som capaços de recordar i saber utilitzar tot el que trobarem aquí, sabrem moltíssim.
# FI TRADUCCIO #
```

5. Fes els commits pertinents i puja-ho al teu repositori remot.

```
neusbravo@neusbravo:~/the-art-of-command-line$ git add README-ca.md
neusbravo@neusbravo:~/the-art-of-command-line$ git commit -m "README-ca.md afegit a la branca"
[master 7de1867] README-ca.md afegit a la branca
1 file changed, 623 insertions(+)
create mode 100644 README-ca.md
neusbravo@neusbravo:~/the-art-of-command-line$ git status
En la rama master
Tu rama està adelantada a 'origin/master' por 1 commit.
(usa "git push" para publicar tus commits locales)

Cambios no rastreados para el commit:
(usa "git add <archivo>..." para actualizar lo que será confirmado)
(usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)
modificados: README.md

sin cambios agregados al commit (usa "git add" y/o "git commit -a")
neusbravo@neusbravo:~/the-art-of-command-line$ git push
Username for 'https://github.com': neusba
Password for 'https://neusba@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/en/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.
fatal: Autenticación falló para 'https://github.com/neusba/the-art-of-command-line.git/'
```

Conclusions finals

Sense dubte he descobert lo potents que arriben a ser els sistemes de control de versions i, pel que hem pogut veure, sobretot Git.

Sembla una mica complexa al principi per totes les comandes que hem de conèixer però crec que per altre banda és també bastant intuïtiu gràcies al fet que el mateix sistema sempre indica les següents passes a seguir.