



Programación I

Ing. Carlos R. Rodríguez



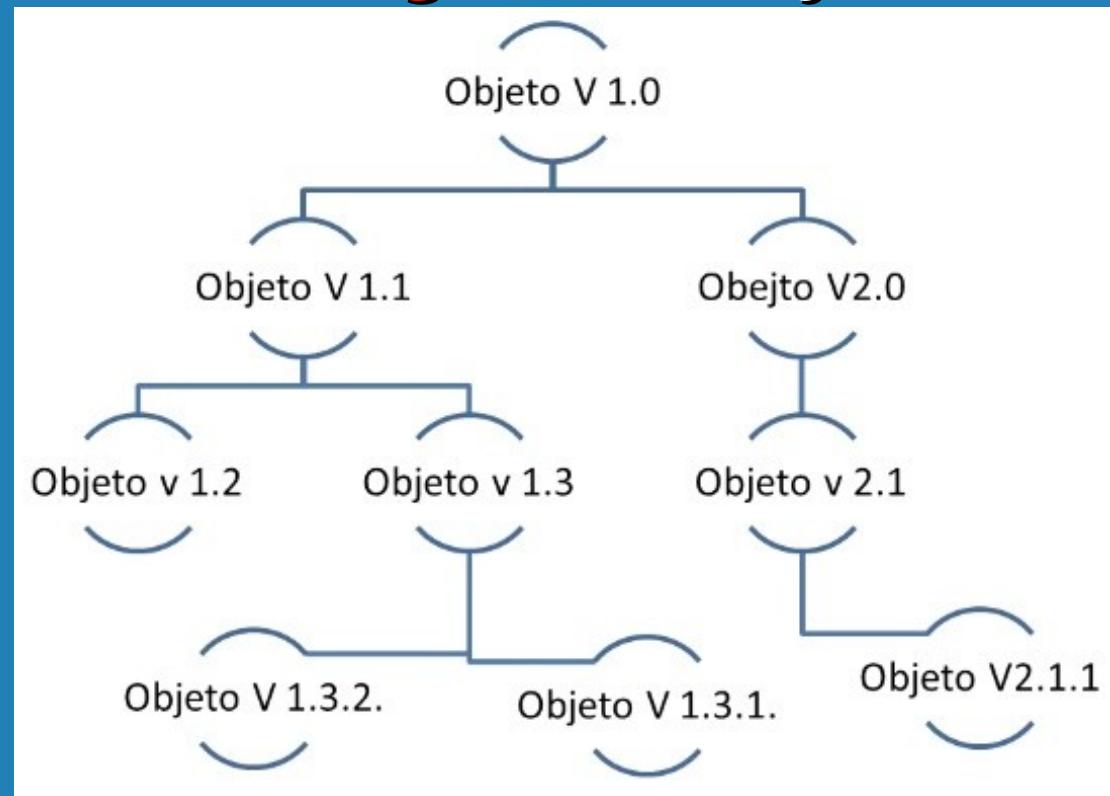
UNIVERSIDAD TECNOLÓGICA NACIONAL
Facultad Regional Mendoza

**Tecnicatura
Superior en
Programación**



¿Qué es?

Es un grupo de aplicaciones que inicialmente gestionaban **cambios en el código fuente** y permitían *revertirlos*. Actualmente incluye todo archivo digital perteneciente a un **proyecto colaborativo**.



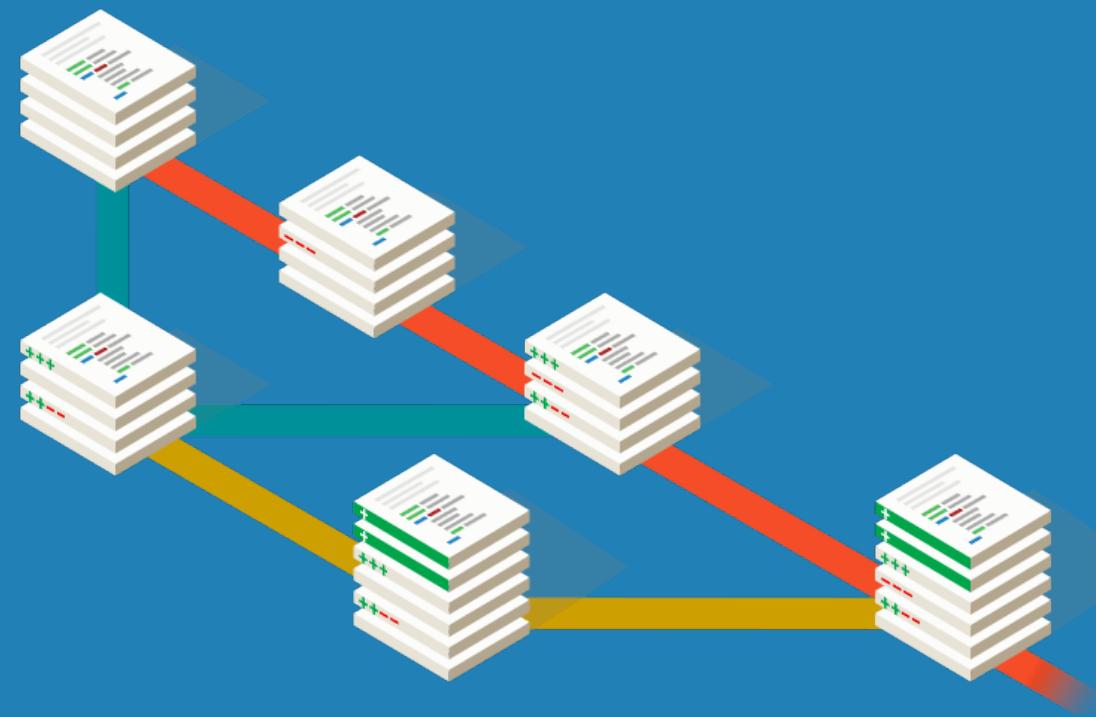
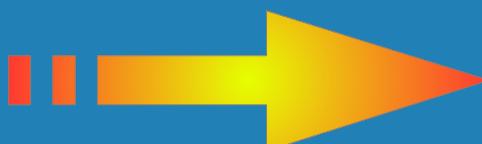


¿Cuál usar?

Debe ser multiplataforma y funcionar desde un entorno simple, pero a la vez debe poder manejar proyectos complejos.

Debe ser gratuito y preferiblemente de código abierto.

Debe ser usado ampliamente.

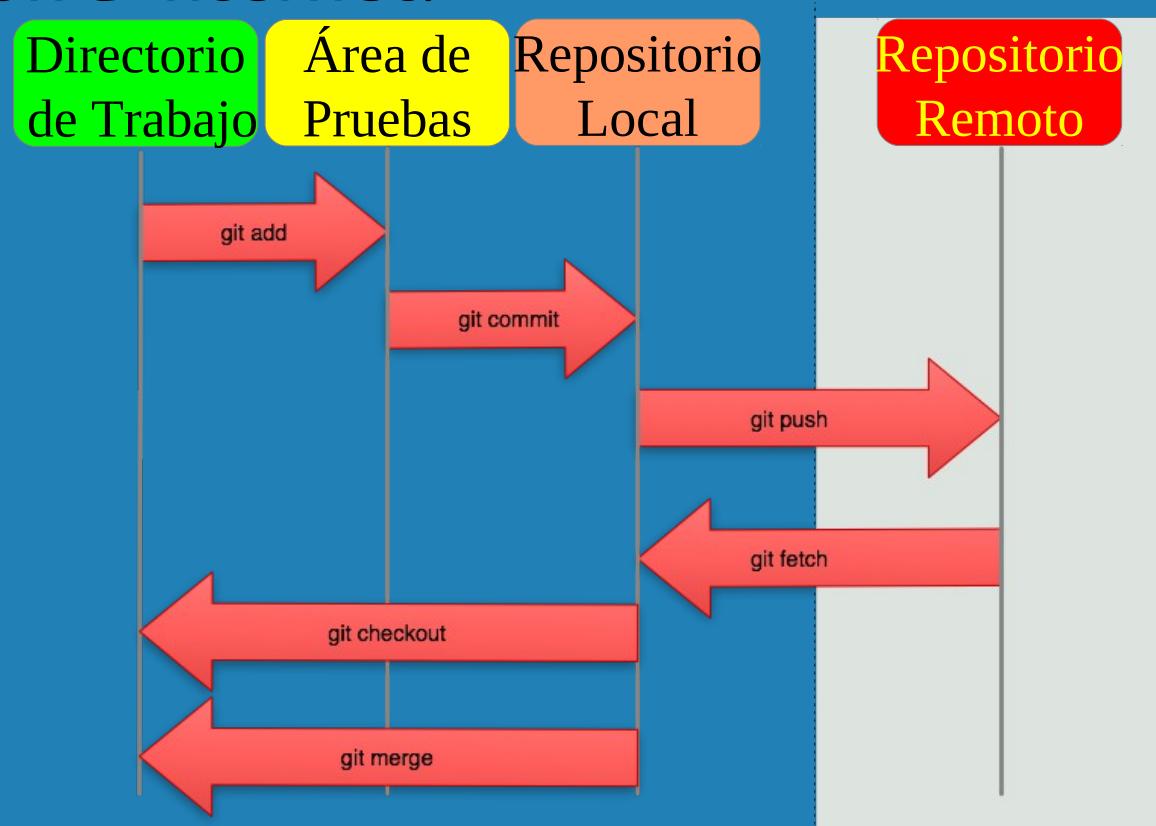




¿Por qué GIT?

Porque es distribuído, y así no bloquea el trabajo ante una caída de la conexión a Internet.

Está pensado para grupos de trabajo heterogéneos trabajando sobre proyectos complejos.



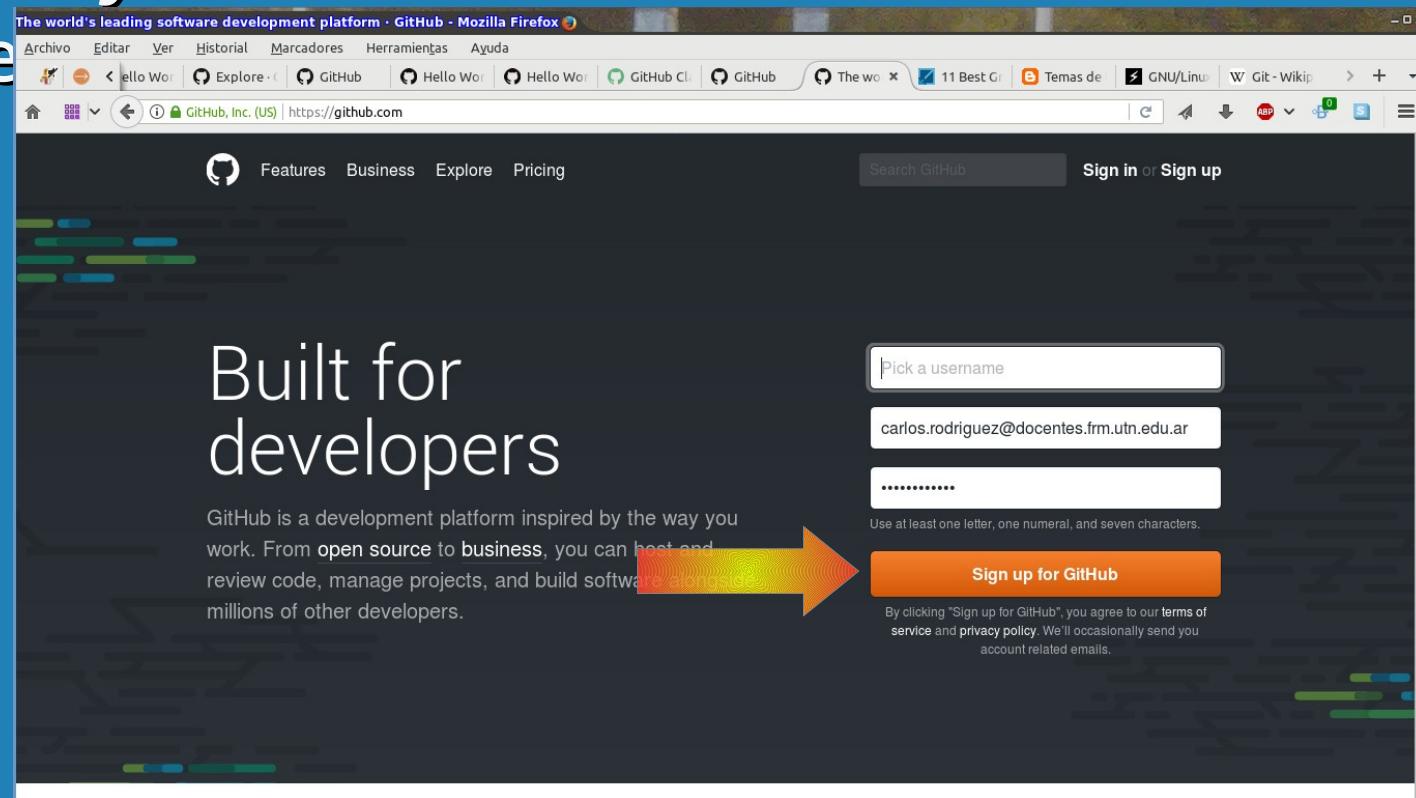


¿Cómo se usa GIT?

Se accede al sitio: <https://github.com/>

La primera vez hay
que registrarse

- Correo.
- Clave.

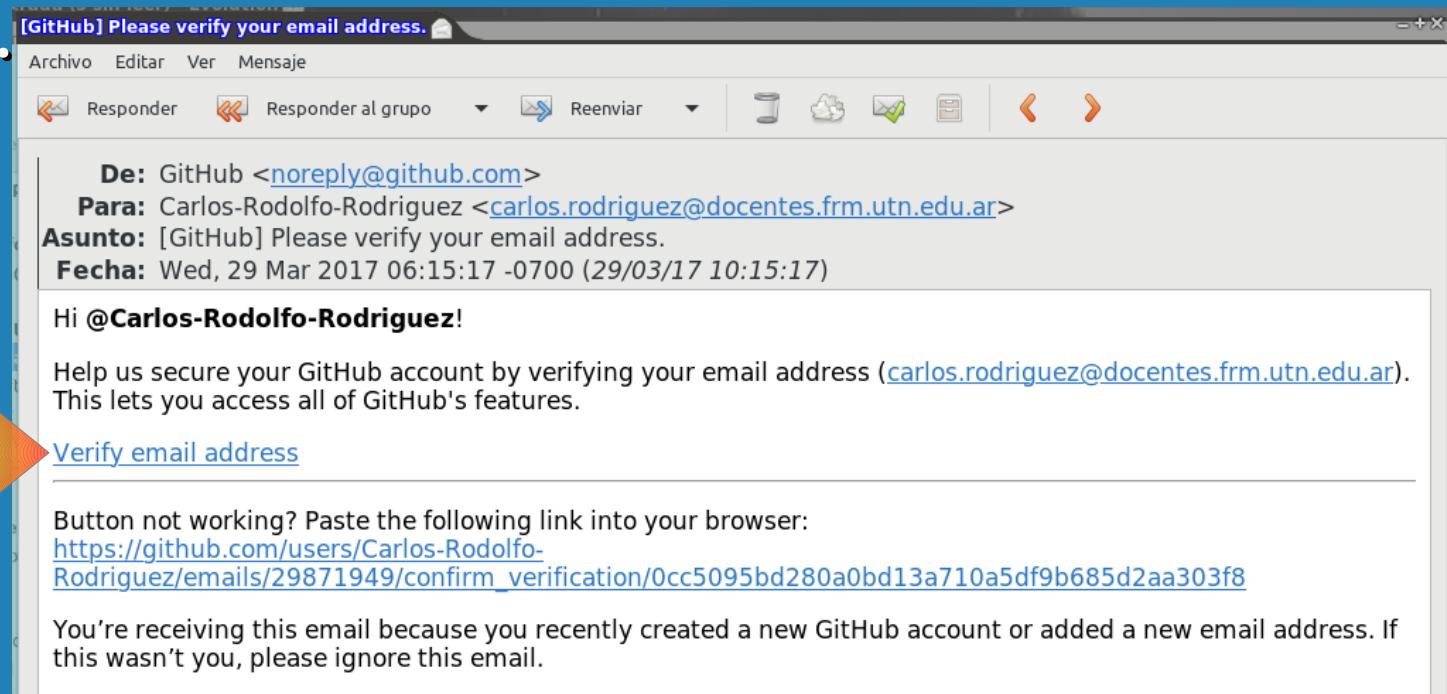




¿Cómo se usa GIT?

Llega un mensaje de verificación a la dirección de correo registrada.

- Contiene un enlace.
- Se pulsa sobre él.
- Listo.



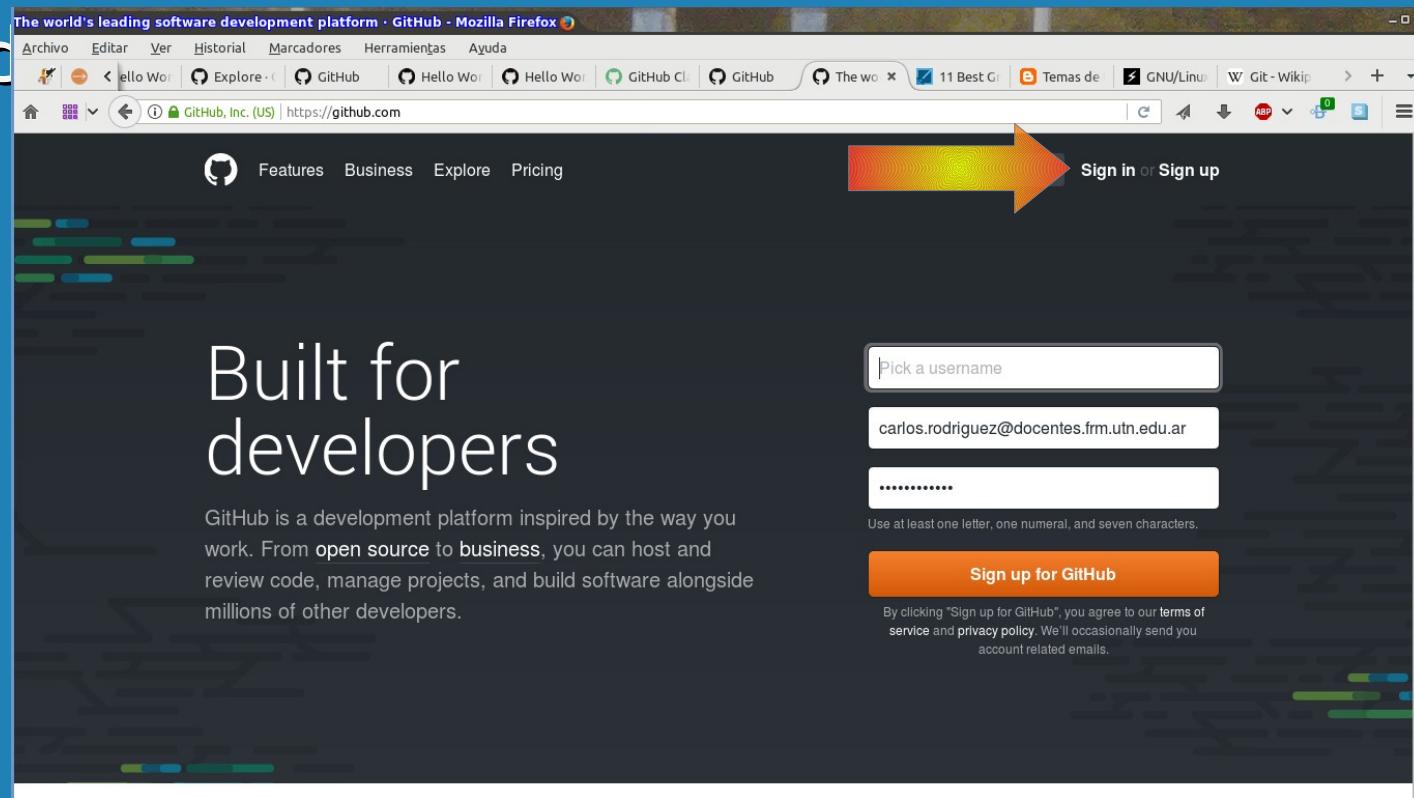


¿Cómo se usa GIT?

Se vuelve al sitio: <https://github.com/>

De ahora en más
se puede acceder

Sign in.





¿Cómo se usa GIT?

Se ingresa con usuario y clave.

Sign in to GitHub

Username or email address

Password [Forgot password?](#)

Sign in

Mozilla Firefox

Sign in to GitHub

Username or email address

Password [Forgot password?](#)

Sign in

New to GitHub? [Create an account.](#)

Terms Privacy Security Contact GitHub



¿Cómo se instala GIT?

Desde la consola, como superusuario (root):

- Se instalan
 - ◊ git
 - ◊ git-gui
 - ◊ gitk

The screenshot shows a terminal window titled "Terminal". The menu bar includes "Archivo", "Editar", "Ver", "Terminal", "Pestañas", and "Ayuda". The toolbar below has icons for new file, new folder, open file, clipboard, search, calendar, editor, back, forward, and help. The command line at the bottom shows the root user executing an apt command to install three Git packages: "root@Lenovo-G50-80:~# apt install git git-gui gitk". A green rectangular selection box highlights the word "gitk" in the command.

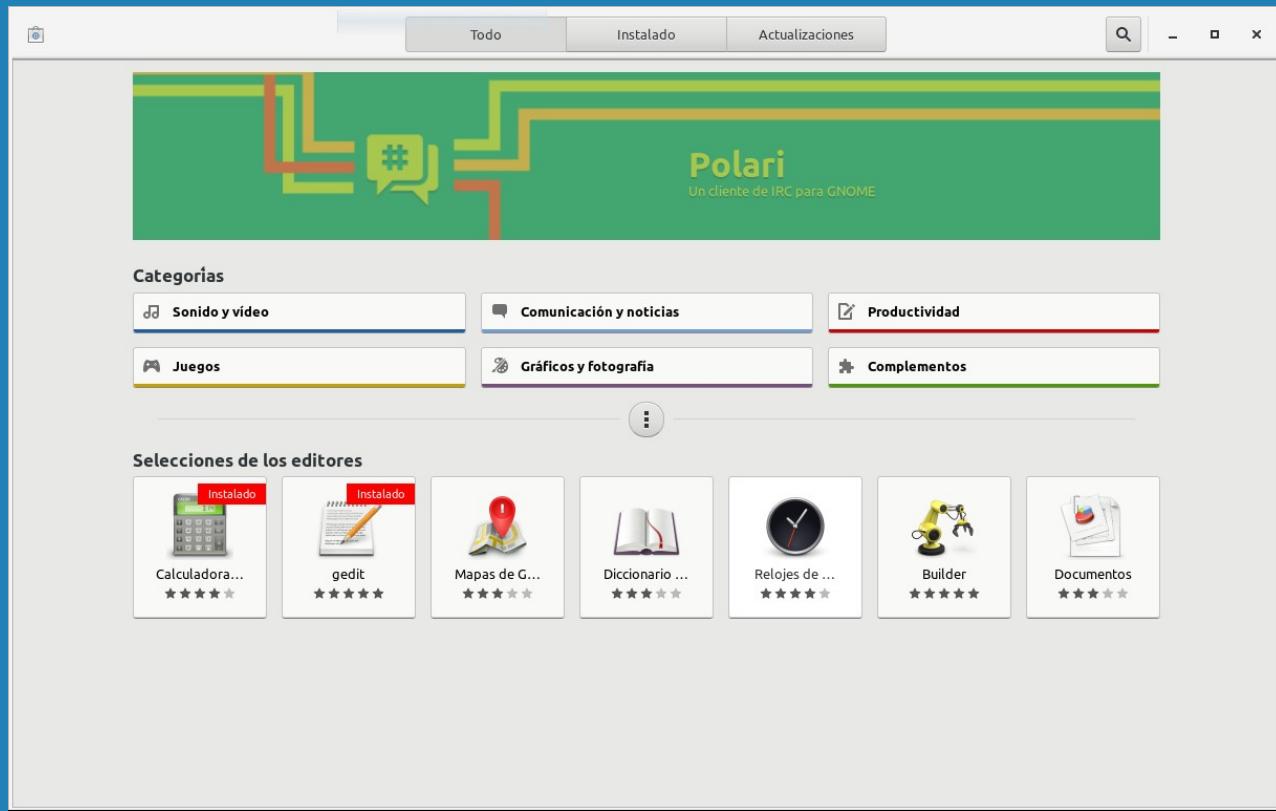
```
root@Lenovo-G50-80:~# apt install git git-gui gitk
```



¿Cómo se instala GIT?

Desde el catálogo gnome-software, como superusuario (root):

- Se instalan
 - ◊ git
 - ◊ git-gui
 - ◊ gitk





¿Cómo se instala GIT?

Desde el catálogo appgrid, como superusuario (root):

Se instalan

- ◊ git
- ◊ git-gui
- ◊ gitk





Sobre el versionador...





¿Cómo se usa GIT?

La primera vez:

- git config --global user.name "Nombre usuario"
- git config --global user.email "usuario@dominio"
- git config --list | grep user

```
carlos@Lenovo-G50-80:~/Zinjai/Adventofcode$ git config --list|grep user
user.name=carlos.rodolfo.rodriguez
user.email=carlos.rodriguez@docentes.frm.utn.edu.ar
```



Uso de GIT

En cualquier momento podemos pedir ayuda:

- git help
- git --help

```
carlos@Lenovo-G50-80:~/Zinjai/Adventofcode$ git help
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

Estas son órdenes Git usadas en diferentes situaciones

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         Mueva o renombre su archivo a un directorio o cree un enlace simbolico
  reset     Restablecer la cabecera al estado actual especificado
  rm         Eliminar archivos del árbol de trabajo y del indice

examine the history and state (see also: git help revisions)
  bisect    Use binary search to find the commit that introduced a bug
  grep      Print lines matching a pattern
  log       Show commit logs
  show      Mostrar los diversos tipos de objetos
  status    Mostrar el estado de los árboles de trabajo

grow, mark and tweak your common history
  branch   List, create, or delete branches
  checkout Cambiar de rama o restaurar los archivos del árbol de trabajo
  commit   Record changes to the repository
  diff     Show changes between commits, commit and working tree, etc
  merge   Join two or more development histories together
  rebase   Reapply commits on top of another base tip
  tag     Crear, listar, eliminar o verificar un objeto de etiqueta firmado con GPG

collaborate (see also: git help workflows)
  fetch   Download objects and refs from another repository
  pull    Debe integrar con otro repositorio o una sucursal local
  push    Actualización de refs remotas junto con objetos asociados

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
```



La ayuda podría ser más completa:

□ git help -a

```
carlos@Lenovo-G50-80:~/Zinjai/Adventofcode$ git help -a
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [-bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

órdenes git disponibles en «/usr/lib/git-core»

add           config          grep            merge-ours      remote-ext    status
add--interactive count-objects  gui             merge-recursive  remote-fd     stripspace
am            credential      gui--askpass    merge-resolve   remote-ftp    submodule
annotate      credential-cache hash-object    merge-subtree   remote-ftps   submodule--helper
apply         credential-cache--daemon help           merge-tree     remote-http   subtree
archive       credential-store  http-backend   mergetool      remote-https  symbolic-ref
bisect        daemon          http-fetch    mktag          remote-testsvn tag
bisect--helper describe        http-push     mktree         repack
blame         diff            imap-send    name-rev      replace
branch        diff-files      index-pack   notes          request-pull
bundle        diff-index     init          pack-objects  rerere
cat-file      diff-tree      init-db      pack-redundant
check-attr    difftool       instaweb    pack-refs     reset
check-ignore  difftool--helper interpret-trailers
check-mailmap fast-export    log          patch-id      rev-list
check-ref-format fast-import   ls-files     prune
checkout      fetch          ls-remote    prune-packed
checkout-index fetch-pack    ls-tree     pull
cherry        filter-branch mailinfo    push
cherry-pick   fmt-merge-msg mailsplit   quiltimport
citool        for-each-ref  merge        read-tree
clean         format-patch  merge-base   rebase
clone         fsck           merge-file  rebase--helper
column        fsck-objects   merge-index receive-pack
commit        gc              merge-octopus
commit-tree   get-tar-commit-id merge-one-file
                           remote      reflog
                           stash

órdenes git disponibles en otra parte en su $PATH

cola          dag          deborig      ftp          remote-bzr

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
```



La ayuda podría venir como una lista de guías:

- git help -g

```
carlos@Lenovo-G50-80:~/Zinjai/Adventofcode$ git help -g
```

The common Git guides are:

attributes	Defining attributes per path
everyday	Everyday Git With 20 Commands Or So
glossary	A Git glossary
ignore	Specifies intentionally untracked files to ignore
modules	Defining submodule properties
revisions	Specifying revisions and ranges for Git
tutorial	A tutorial introduction to Git (for version 1.5.1 or newer)
workflows	An overview of recommended workflows with Git

'git help -a' and 'git help -g' list available subcommands and some concept guides. See 'git help <command>' or "git help <concept>" to read about a specific subcommand or concept.



La ayuda podría venir como una guía específica:

- git help <guia>

GITTUTORIAL(7) ◇ git help tutorial Git Manual GITTUTORIAL

NAME
gittutorial - A tutorial introduction to Git

SYNOPSIS
git *

DESCRIPTION
This tutorial explains how to import a new project into Git, make changes to it, and share changes with other developers.
If you are instead primarily interested in using Git to fetch a project, for example, to test the latest version, you may prefer to start with the first two chapters of [The Git User's Manual\[1\]](#).
First, note that you can get documentation for a command such as `git log --graph` with:

\$ man git-log
or:

\$ git help log
With the latter, you can use the manual viewer of your choice; see `git-help(1)` for more information.
It is a good idea to introduce yourself to Git with your name and public email address before doing any operation. The easiest way to do so is:

\$ git config --global user.name "Your Name Comes Here"
\$ git config --global user.email you@yourdomain.example.com

IMPORTING A NEW PROJECT
Assume you have a tarball project.tar.gz with your initial work. You can place it under Git revision control as follows.

\$ tar xzf project.tar.gz
\$ cd project
\$ git init

Git will reply

Initialized empty Git repository in .git/

You've now initialized the working directory—you may notice a new directory created, named ".git".
Next, tell Git to take a snapshot of the contents of all files under the current directory (note the `_`), with `git add`:

Manual page gittutorial(7) line 1 (press h for help or q to quit)



La ayuda podría ser más específica:

□ git help

◊ git help add

GIT-ADD(1) Git Manual GIT-ADD(1)

NAME
git-add - Add file contents to the index

SYNOPSIS
`git add [--verbose | -v] [--dry-run | -n] [--force | -f] [--interactive | -i] [--patch | -p]
[--edit | -e] [--[no-]all | --[no-]ignore-removal | [--update | -u]]
[--intent-to-add | -N] [--refresh] [--ignore-errors] [--ignore-missing]
[--chmod=(+|-)x] [--] [<pathspec>...]`

DESCRIPTION
This command updates the index using the current content found in the working tree, to prepare the content staged for the next commit. It typically adds the current content of existing paths as a whole, but with some options it can also be used to add content with only part of the changes made to the working tree files applied, or remove paths that do not exist in the working tree anymore.

The "index" holds a snapshot of the content of the working tree, and it is this snapshot that is taken as the contents of the next commit. Thus after making any changes to the working tree, and before running the commit command, you must use the `add` command to add any new or modified files to the index.

This command can be performed multiple times before a commit. It only adds the content of the specified file(s) at the time the add command is run; if you want subsequent changes included in the next commit, then you must run `git add` again to add the new content to the index.

The `git status` command can be used to obtain a summary of which files have changes that are staged for the next commit.

The `git add` command will not add ignored files by default. If any ignored files were explicitly specified on the Manual page `git-add(1)` line 1 (press `h` for help or `q` to quit)



Empezar un repositorio nuevo desde la computadora:

- cd <carpeta>
- git init

```
carlos@Lenovo-G50-80:~$ cd ~/.Zinjai/HackerRank/  
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ ls -lR  
.:  
total 5464
```

```
-rw-rw-r-- 1 carlos carlos 597824 sep 14 2017 datos2.dat  
-rw-rw-r-- 1 carlos carlos 2166595 sep 10 2017 datos.dat  
-rw-rw-r-- 1 carlos carlos 2168629 ago 9 2017 datos.dat~  
-rw-rw-r-- 1 carlos carlos 21711 ago 9 2017 datPrueba2.dat  
-rw-rw-r-- 1 carlos carlos 21717 ago 9 2017 datPrueba2.dat~  
-rw-rw-r-- 1 carlos carlos 35 ago 9 2017 datPrueba.dat  
-rw-rw-r-- 1 carlos carlos 544 jul 2 2017 Ejercicio 01.cpp  
-rw-rw-r-- 1 carlos carlos 395 jul 3 2017 Ejercicio 02.cpp  
-rw-rw-r-- 1 carlos carlos 658 jul 3 2017 Ejercicio 03.cpp  
-rw-rw-r-- 1 carlos carlos 419 jul 4 2017 Ejercicio 04.cpp  
-rw-rw-r-- 1 carlos carlos 672 jul 4 2017 Ejercicio 05.cpp  
-rw-rw-r-- 1 carlos carlos 936 jul 10 2017 Ejercicio 06.cpp  
-rw-rw-r-- 1 carlos carlos 1551 jul 13 2017 Ejercicio 07.cpp  
-rw-rw-r-- 1 carlos carlos 1255 jul 13 2017 Ejercicio 08.cpp  
-rw-rw-r-- 1 carlos carlos 1469 jul 13 2017 Ejercicio 09.cpp  
-rw-rw-r-- 1 carlos carlos 1063 jul 19 2017 Ejercicio 10.cpp
```

```
-rw-rw-r-- 1 carlos carlos 1124 jul 19 2017 Ejercicio 11.cpp  
-rw-rw-r-- 1 carlos carlos 1152 jul 20 2017 Ejercicio 12.cpp  
-rw-rw-r-- 1 carlos carlos 2718 jul 21 2017 Ejercicio 13.cpp  
-rw-rw-r-- 1 carlos carlos 1588 jul 21 2017 Ejercicio 14.cpp  
-rw-rw-r-- 1 carlos carlos 1277 jul 25 2017 Ejercicio 15.cpp  
-rw-rw-r-- 1 carlos carlos 1252 ago 2 2017 Ejercicio 16.cpp  
-rw-rw-r-- 1 carlos carlos 965 ago 6 2017 Ejercicio 17.cpp  
-rw-rw-r-- 1 carlos carlos 1949 ago 30 2017 Ejercicio 18.cpp  
-rw-rw-r-- 1 carlos carlos 1191 sep 5 2017 Ejercicio 19.cpp  
-rw-rw-r-- 1 carlos carlos 1290 sep 6 2017 Ejercicio 20.cpp  
-rw-rw-r-- 1 carlos carlos 4075 sep 14 2017 Ejercicio 21b.cpp  
-rw-rw-r-- 1 carlos carlos 1995 sep 13 2017 Ejercicio 21b.cpp~  
-rw-rw-r-- 1 carlos carlos 1504 sep 10 2017 Ejercicio 21.cpp  
-rw-rw-r-- 1 carlos carlos 2535 sep 14 2017 Ejercicio 22.cpp  
-rw-rw-r-- 1 carlos carlos 1572 nov 2 18:55 Ejercicio 23.cpp  
-rw-rw-r-- 1 carlos carlos 147399 sep 14 2017 resultados2.txt  
-rw-rw-r-- 1 carlos carlos 10 sep 14 2017 resultados.txt  
-rw-rw-r-- 1 carlos carlos 180120 ago 9 2017 resultados.txt~  
-rw-rw-r-- 1 carlos carlos 0 ago 28 2017 resul.txt  
-rw-rw-r-- 1 carlos carlos 147399 sep 14 2017 salida.sal  
-rw-rw-r-- 1 carlos carlos 265 jul 2 2017 solucion01.i  
-rw-rw-r-- 1 carlos carlos 226 jul 3 2017 solucion02.i  
-rw-rw-r-- 1 carlos carlos 414 jul 3 2017 solucion03.i  
-rw-rw-r-- 1 carlos carlos 223 jul 4 2017 solucion04.i  
-rw-rw-r-- 1 carlos carlos 603 nov 2 18:57 solucion.i  
-rw-rw-r-- 1 carlos carlos 604 nov 2 18:57 solucion.i~  
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git init  
Initialized empty Git repository in /home/carlos/.Zinjai/HackerRank/.git/  
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ █
```

```
-rw-rw-r-- 1 carlos carlos 603 nov 2 18:57 solucion.i  
-rw-rw-r-- 1 carlos carlos 604 nov 2 18:57 solucion.i~  
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git init  
Initialized empty Git repository in /home/carlos/.Zinjai/HackerRank/.git/  
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ █
```



Por computadora:

- En `/etc/gitconfig`
 - ◊ Se crea la primera vez que se usa, y se edita con:
 - ▷ `git config --system -e`





Por usuario:

- En `~/.gitconfig`
 - ◊ Siendo `~` la carpeta raíz del usuario.
 - ◊ Se edita con:
 - ▷ `git config --global -e`

```
/home/carlos/.gitconfig
m.utn.edu.ar
```

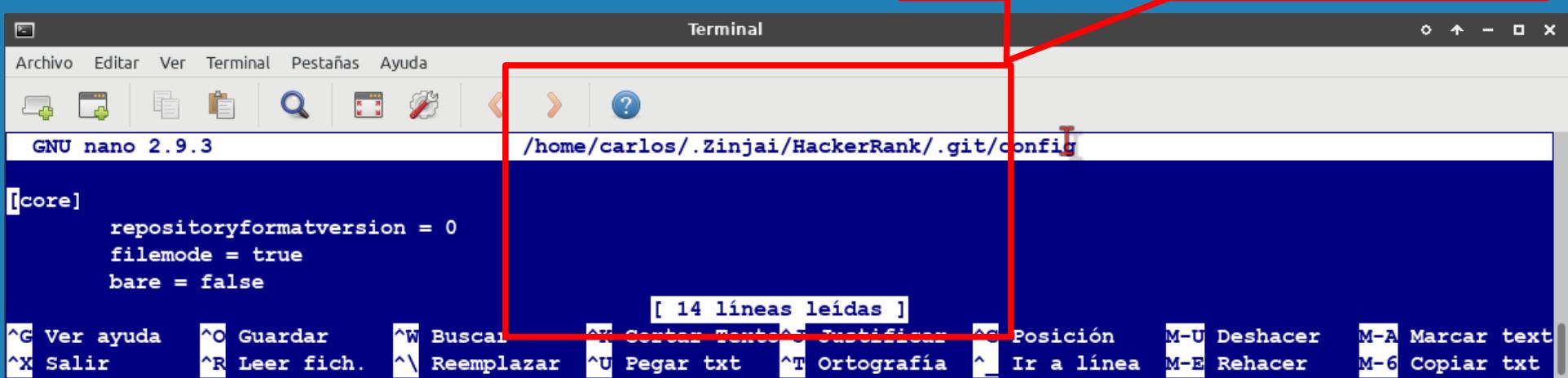
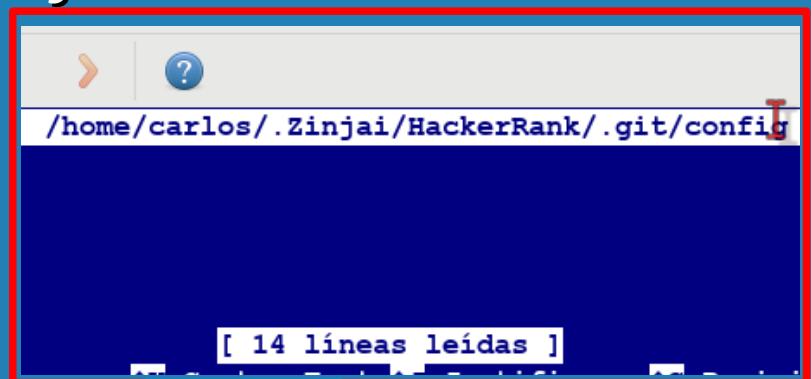
Terminal

```
GNU nano 2.9.3
[user]
    name = carlos.rodolfo.rodriguez
    email = carlos.rodriguez@docentes.frm.utn.edu.ar
[merge]
[ 12 líneas leidas ]
^G Ver ayuda      ^O Guardar      ^W Buscar      ^K Cortar Texto ^J Justificar   ^C Posición   M-U Deshacer   M-A Marcar texto
^X Salir         ^R Leer fich.  ^\ Reemplazar  ^U Pegar txt   ^T Ortografía ^_ Ir a línea M-E Rehacer   M-6 Copiar txt
```



Por directorio de trabajo:

- En .git/config
 - ◊ Siendo . la carpeta de trabajo.
 - ◊ Se edita con:
 - ▷ **git config -e**





Cada nivel **sobrescribe** los valores del nivel anterior, por lo que los valores de `.git/config` tienen **preferencia** sobre los de `/etc/gitconfig`.

- El editor asumido se cambia con:
 - ◊ `git config --global core.editor <editor>`

The screenshot shows a terminal window and a gedit text editor side-by-side. The terminal window displays the command `git config --global core.editor gedit` being run, followed by a message from gedit asking for confirmation to close the file. The gedit window shows the contents of the `.git/config` file, which includes the new configuration entry for the editor.

```
[core]
repositoryformatversion = 0
filemode = true
bare = false

carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git config --global core.editor gedit
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git config -e
ayuda: Esperando que tu editor cierre el archivo ...

[core]
repositoryformatversion = 0
filemode = true
bare = false
logallrefupdates = true
[remote "HackerRank"]
url = https://github.com/Carlos-Rodolfo-Rodriguez/HackerRank.git
fetch = +refs/heads/*:refs/remotes/HackerRank/*
[submodule "Seudocódigo"]
url = https://github.com/Carlos-Rodolfo-Rodriguez/Seudocodigo.git
active = true
[branch "prueba"]
remote = https://github.com/Carlos-Rodolfo-Rodriguez/Seudocodigo.git
merge = refs/heads/prueba
```



Para ver toda la configuración actual aplicada:

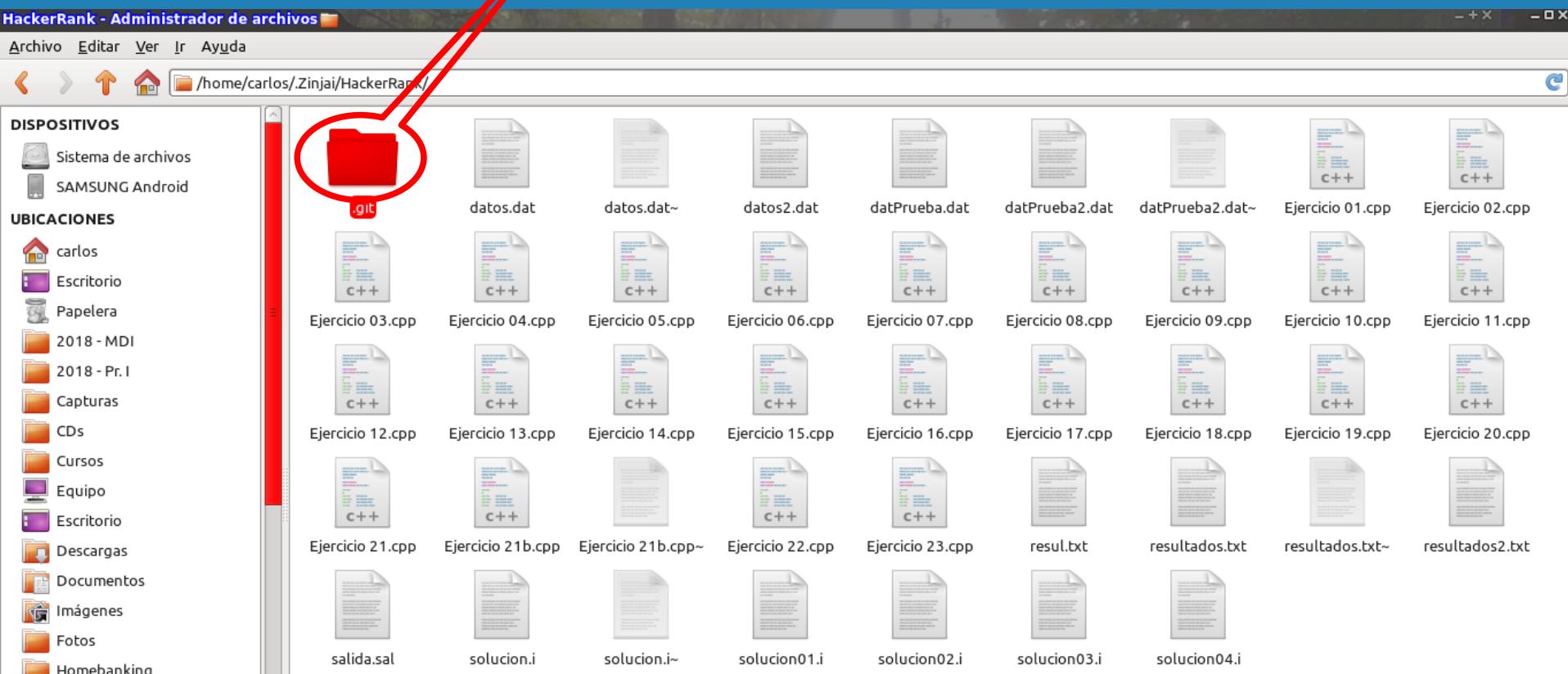
- git config –list**
 - ◊ Pueden verse claves repetidas.
 - ◊ Vale la última.

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git config --list
user.name=carlos.rodolfo.rodriguez
user.name=carlos.rodolfo.rodriguez
user.name=carlos.rodolfo.rodriguez
user.email=carlos.rodriguez@docentes.frm.utn.edu.ar
merge.tool=meld
gui.spellingdictionary=none
```



git init crea una carpeta .git

Borrando esa carpeta se anula el uso de git.





Se añaden los archivos a incluir (no serán todos)...

□ git add <archivos>

- ◊ git add *.cpp
- ◊ git add *.dat
- ◊ git add *.txt
- ◊ git add *.i

□ Se marcan para incluir en el repositorio con el próximo commit.

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git add *.cpp
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git add *.dat
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git add *.txt
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git add *.i
```



Estado:

- git status
- git rm --cached
o git reset son
lo opuesto a git
add.

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git status
En la rama master

No commits yet

Cambios para hacer commit:
(use «git rm --cached <archivo>...» para sacar del stage)

    nuevo archivo: Ejercicio 01.cpp
    nuevo archivo: Ejercicio 02.cpp
    nuevo archivo: Ejercicio 03.cpp
    nuevo archivo: Ejercicio 04.cpp
    nuevo archivo: Ejercicio 05.cpp
    nuevo archivo: Ejercicio 06.cpp
    nuevo archivo: Ejercicio 07.cpp
    nuevo archivo: Ejercicio 08.cpp
```

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git status
En la rama master
```

No commits yet

Cambios para hacer commit:
(use «git rm --cached <archivo>...» para sacar del stage)



Veamos el estado...

□ git status

◊ Avisa cuáles archivos están sin seguimiento

- ▷ Se pueden dejar así o...
- ▷ Se puede añadirlos con git add

```
nuevo archivo: Ejercicio 21.cpp
nuevo archivo: Ejercicio 21b.cpp
nuevo archivo: Ejercicio 22.cpp
nuevo archivo: Ejercicio 23.cpp
nuevo archivo: datPrueba.dat
nuevo archivo: datPrueba2.dat
nuevo archivo: datos.dat
nuevo archivo: datos2.dat
nuevo archivo: resul.txt
nuevo archivo: resultados.txt
nuevo archivo: salida.sal
nuevo archivo: solucion.i
nuevo archivo: solucion01.i
nuevo archivo: solucion02.i
nuevo archivo: solucion03.i
nuevo archivo: solucion04.i
```

Archivos sin seguimiento:

(use «git add <archivo>...» para incluir en lo que se ha de confirmar)

```
Ejercicio 21b.cpp~
datPrueba2.dat~
datos.dat~
resultados.txt~
salida.sal
solucion.i~
```

carlos@Lenovo-G50-80:~/Zinjai/HackerRank\$ █



Primer respaldo.

□ `git commit -m <"mensaje">`

◊ Identifica el respaldo.

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git commit -m "Primer respaldo"
[master (root-commit) 36392ae] Primer respaldo
36 files changed, 140515 insertions(+)
create mode 100644 Ejercicio 01.cpp
create mode 100644 Ejercicio 02.cpp
create mode 100644 Ejercicio 03.cpp
create mode 100644 Ejercicio 04.cpp
create mode 100644 Ejercicio 05.cpp
create mode 100644 Ejercicio 06.cpp
create mode 100644 Ejercicio 07.cpp
create mode 100644 Ejercicio 08.cpp
create mode 100644 Ejercicio 09.cpp
create mode 100644 Ejercicio 10.cpp
```

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git commit -m "Primer respaldo"
[master (root-commit) 36392ae] Primer respaldo
36 files changed, 140515 insertions(+)
create mode 100644 Ejercicio 01.cpp
```

```
create mode 100644 Ejercicio 18.cpp
create mode 100644 Ejercicio 19.cpp
create mode 100644 Ejercicio 20.cpp
create mode 100644 Ejercicio 21.cpp
create mode 100644 Ejercicio 21b.cpp
create mode 100644 Ejercicio 22.cpp
create mode 100644 Ejercicio 23.cpp
create mode 100644 datPrueba.dat
create mode 100644 datPrueba2.dat
```



Veamos el estado ahora...

□ git status

- ◊ Avisa sobre los archivos sin seguimiento.
 - ▷ Nada que agregar.
 - ▷ Si queremos que ignore algunos archivos, se usa el archivo especial **.gitignore**

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git status
En la rama master
Archivos sin seguimiento:
(use «git add <archivo>...» para incluir en lo que se ha de confirmar)

Ejercicio_21b.cpp~
datPrueba2.dat~
datos.dat~
resultados.txt~
salida.sal
solucion.i~

no se ha agregado nada al commit pero existen archivos sin seguimiento (use «git add» para darle seguimiento)
carlos@lenovo-G50-80:~/Zinjai/HackerRank$ █
```



Archivos a ignorar con .gitignore

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ cat .gitignore
*.*~
```

```
salida.sal
```

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git status
```

```
En la rama master
```

```
Archivos sin seguimiento:
```

```
(use «git add <archivo>...» para incluir en lo que se ha
```

```
.gitignore
```

```
no se ha agregado nada al commit pero existen archivos sin
```

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git add .gitign
```

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git status
```

```
En la rama master
```

```
Cambios para hacer commit:
```

```
(use «git reset HEAD <archivo>...» para sacar del stage)
```

```
nuevo archivo: .gitignore
```



Segundo respaldo:

- git add -A añade todo lo no excluido por .gitignore
- git commit -m <"mensaje"> ejecuta el respaldo.

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git add -A
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git status
En la rama master
Cambios para hacer commit:
  (use «git reset HEAD <archivo>...» para sacar del stage)

    nuevo archivo: .gitignore

carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git commit -m "Segundo respaldo"
[master e48c129] Segundo respaldo
  1 file changed, 2 insertions(+)
   create mode 100644 .gitignore
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git status
En la rama master
nothing to commit, working tree clean
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ █
```



¿Qué respaldos se han hecho hasta hora?

- **git log** nos muestra los **respaldos** realizados.
 - ◊ Cada commit tiene su identificador único.

```
carlos@Lenovo-G50-80:~/Zinja/HackerRank$ git log
commit e48c1297707579ec752681080438a426b35764c3 (HEAD -> master)
Author: carlos.rodolfo.rodriguez <carlos.rodriguez@docentes.frm.utn.edu.ar>
Date:   Sat Apr 28 13:17:24 2018 -0300
```

Segundo respaldo

```
commit 36392ae2396b9419470b1701037c4436e41ca034
Author: carlos.rodolfo.rodriguez <carlos.rodriguez@docentes.frm.utn.edu.ar>
Date:   Thu Apr 26 12:44:41 2018 -0300
```

Primer respaldo



Para copiar un repositorio debemos obtener su <url>:

The screenshot shows a GitHub repository page for 'Seudocodigo'. A large red box highlights the 'Clone or download' button in the top right corner of the main content area. Below it, another red box highlights the 'Clone with HTTPS' section, which contains the URL <https://github.com/Carlos-Rodolfo-Rodriguez/Seudocodigo>. The page also displays basic repository statistics: 0 releases, 1 contributor, and various uploaded files like README.md and program1.h.



git clone nos permite clonar un repositorio en otro:

□ **git clone <origen> <destino>**

- ◊ Normalmente uno de los repositorios es remoto.
- ◊ <destino> debe ser una carpeta vacía.

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git clone https://github.com/Carlos-Rodolfo-Rodriguez/Seudocodigo.git .
fatal: destination path '.' already exists and is not an empty directory.
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ mkdir Seudocódigo
carlos@lenovo-G50-80:~/Zinjai/HackerRank$ cd Seudocódigo/
carlos@Lenovo-G50-80:~/Zinjai/HackerRank/Seudocódigo$ git clone https://github.com/Carlos-Rodolfo-Rodriguez/Seudocodigo.git .
Clonar en «.»...
remote: Counting objects: 6, done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 6
Unpacking objects: 100% (6/6), done.
carlos@Lenovo-G50-80:~/Zinjai/HackerRank/Seudocódigo$ ls -l
total 24
-rw-rw-r-- 1 carlos carlos 19538 abr 29 13:46 program1.h
-rw-rw-r-- 1 carlos carlos     14 abr 29 13:46 README.md
```



git diff nos muestra las diferencias:

- Creamos un enlace suave (softlink) a la última versión del archivo program1.h.

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank/Seudocódigos$ ln -Lrvs ../../program1.h .
ln: fallo al crear el enlace simbólico './program1.h': El archivo ya existe
carlos@Lenovo-G50-80:~/Zinjai/HackerRank/Seudocódigos$ rm program1.h -v
'program1.h' borrado
carlos@Lenovo-G50-80:~/Zinjai/HackerRank/Seudocódigos$ ln -Lrvs ../../program1.h .
'./program1.h' -> '../../program1.h'
```

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank/Seudocódigo$ git diff
diff --git a/program1.h b/program1.h
deleted file mode 100644
index 4883cef..0000000
--- a/program1.h
+++ /dev/null
@@ -1,443 +0,0 @@
// Version 2016
// Version 2016
#ifndef PROGRAM1_H
#define PROGRAM1_H
// Inclusiones (<DE>ndo)
#include <iostream>
#include <iomanip>
#include <fstream>
#include <cstdlib>
#include <cmath>
#include <chrono>
#include <ctime>
#include <cstring>
#include <list>
#include <vector>
#include <array>
#include <queue>
#include <algorithm>
#include <assert.h>
#include <stdio.h>
using namespace std;
// S&lt;DE>lo para Linux:
```

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank/Seudocódigo$ git diff
diff --git a/program1.h b/program1.h
deleted file mode 100644
index 4883cef..0000000
--- a/program1.h
+++ /dev/null
@@ -1,443 +0,0 @@
// Version 2016
// Version 2016
#ifndef PROGRAM1_H
#define PROGRAM1_H
// Inclusiones (<DE>ndo)
#include <iostream>
#include <iomanip>
#include <fstream>
#include <cstdlib>
#include <cmath>
#include <chrono>
#include <ctime>
#include <cstring>
#include <list>
#include <vector>
#include <array>
#include <queue>
#include <algorithm>
#include <assert.h>
#include <stdio.h>
using namespace std;
// S&lt;DE>lo para Linux:
```



git status refleja los cambios (y git add -A los marca):

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank/Seudocódigo$ git status
En la rama master
Su rama está actualizada con «origin/master».
```

Cambios no preparados para el commit:

```
(use «git add <archivo>...» para actualizar lo que se confirmará)
(use «git checkout -- <archivo>...» para descartar cambios en el di
    tipo cambiado: program1.h
```

no hay cambios agregados al commit (use «git add» o «git commit -a»)

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank/Seudocódigo$ git add -A
carlos@Lenovo-G50-80:~/Zinjai/HackerRank/Seudocódigo$ git status
En la rama master
Su rama está actualizada con «origin/master».
```

Cambios para hacer commit:

```
(use «git reset HEAD <archivo>...» para sacar del stage)
    tipo cambiado: program1.h
```



git commit actualiza el repositorio local.

git push origin <rama> actualiza la rama sobre el repositorio remoto:

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank/Seudocódigo$ git push origin master
Username for 'https://github.com': carlos-rodolfo-rodriguez
Password for 'https://carlos-rodolfo-rodriguez@github.com':
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 314 bytes | 314.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Carlos-Rodolfo-Rodriguez/Seudocodigo.git
 2137433..1c643a5  master -> master
```



git push origin <rama> tiene efecto inmediato:

The screenshot shows a GitHub repository page for 'Carlos-Rodolfo-Rodriguez / Seudocodigo'. A red arrow points from the text 'git push origin <rama> tiene efecto inmediato:' to the commit history. The commit 'program1.h' is highlighted with a red box and shows a timestamp of '33 minutes ago', indicating it was pushed immediately after the command was run.

Carlos-Rodolfo-Rodriguez/Seudocodigo

This repository

Pull requests Issues Marketplace Explore

Carlos-Rodolfo-Rodriguez / Seudocodigo

No description, website, or topics provided.

Add topics

3 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

Carlos-Rodolfo-Rodriguez Cabecera

README.md first commit Cabecera a year ago

program1.h 33 minutes ago

README.md

Seudocodigo



git pull origin <rama> verifica si hay modificaciones en el repositorio remoto:

- Primero hacemos una pequeña modificación...

The screenshot shows a GitHub repository page for 'Seudocodigo'. The URL in the address bar is <https://github.com/Carlos-Rodolfo-Rodriguez/Seudocodigo/blob/master/README.md>. The page displays the contents of the README.md file, which contains the text 'Seudocodigo'. At the bottom right of the file content area, there is a dark button labeled 'Edit this file' with a white pencil icon. A large red circle and a red arrow point to this button, indicating it as the target of the first step in the process.



git pull origin <rama> verifica si hay modificaciones en el repositorio remoto:

- Primero hacemos una pequeña *modificación...*

The screenshot shows a GitHub repository interface. At the top, there's a header with a user icon, 'Programación I', and a search bar. Below the header, the repository name 'Carlos-Rodolfo-Rodriguez / Seudocodigo' is visible, along with statistics for issues, pull requests, projects, and a star/fork button.

In the main area, a file named 'README.md' is open for editing. A red box highlights the second line of the file content: '# Seudocodigo' and 'Con las modificaciones 2018.' A red arrow points from this highlighted text to a 'Commit changes' dialog box.

The 'Commit changes' dialog has the following fields and options:

- A preview image of a person's face.
- A title field: 'Commit changes'.
- A text input field: 'Update README.md'.
- A description input field: 'Add an optional extended description...'. A red circle highlights this field.
- Two radio button options:
 - Commit directly to the `master` branch.
 - Create a **new branch** for this commit and start a pull request. [Learn more about pull requests.](#)
- Two buttons at the bottom: 'Commit changes' (green) and 'Cancel' (red).



git pull origin <rama> verifica si hay modificaciones en el repositorio remoto:

- El archivo **README.md** ha sido *modificado...*

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank/Seudocódigo$ git pull origin master
remote: Counting objects: 3, 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), done.
De https://github.com/Carlos-Rodolfo-Rodriguez/Seudocodigo
 * branch            master      -> FETCH_HEAD
   2c00e8d..df74fd1  master      -> origin/master
Updating 2c00e8d..df74fd1
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
```



git branch <rama> crea una ramificación del proyecto.

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRanks$ git branch prueba
carlos@Lenovo-G50-80:~/Zinjai/HackerRanks$ git branch
* master
  prueba
carlos@Lenovo-G50-80:~/Zinjai/HackerRanks$ git checkout prueba
Switched to branch 'prueba'
carlos@Lenovo-G50-80:~/Zinjai/HackerRanks$ git branch
  master
* prueba
```



git branch muestra las ramas, marcando (*) la que está actualmente **activa**.

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git branch prueba
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git branch
* master
  prueba
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git checkout prueba
Switched to branch 'prueba'
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git branch
  master
* prueba
```



git checkout <rama> cambia la rama **activa**.

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git branch prueba
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git branch
* master
  prueba
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git checkout prueba
Switched to branch 'prueba'
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git branch
  master
* prueba
```



git checkout <rama> cambia la rama **activa**.

```
1 // Archivo de traducción de seudocódigo a C++
2 #include "program1.h"
3 /**
4  * Enunciado: You are given a list of numbers, to be added to a database. Each of these numbers must be added in the same just-in-the-order in
5  * which they are given. If a number has already been added to the database, print "REDUNDANT" (without quotation marks).
6  * Otherwise, add it to the database and print "ADDED"(again, without quotation marks).
7  * The answer of each operation should be printed in a new line.
8 Input Format:
9 The first line contains a single integer , denoting the number of numbers to be added.
10 The second line contains the space-separated integers - .
11 Output Format:
12 lines, each having the answer to the subsequent operation.
13 */
14
15 función largo cuantasVeces(largo valABuscar,vectorDin(largo) v) {
16 largo resul = 0;
17 paraCada(ele,v)
18     si(ele ES valABuscar) resul++;
19     finParaCada
20 regresa(resul);
21 }
22
23 función logico estaRepetido(largo valABuscar,vectorDin(largo) v) {
24 logico resul = tamano(v) > 0 Y cuantasVeces(valABuscar,v) >= 1;
25 regresa(resul);
26 }
27
28 principal                                // Unidad de programa principal
29 largo N;
30 leer(N);
31 vectorDin(largo) v(N),v2(0);
32 paraCada(ele,v)
33     leer(ele);
34     si(estaRepetido(ele,v2)) entonces
35         mostrar << "REDUNDANT" << salto;
36         sino
37             mostrar << "ADDED" << salto;
38             agregaEleVdin(v2,ele);
39             finSi
40     finParaCada
41 finPrincipal                                // Fin de unidad de programa principal.
```

Guardar el archivo actual



En el repositorio remoto, vemos la rama prueba:

The screenshot shows a GitHub repository page for 'Carlos-Rodolfo-Rodriguez/Seudocodigo'. The 'Code' tab is selected. A dropdown menu indicates the current branch is 'prueba'. The commit history is displayed in three sections corresponding to commit dates: April 30, 2018; April 28, 2018; and April 26, 2018. Each section contains one commit by 'Carlos-Rodolfo-Rodriguez' with a timestamp and a copy icon.

- Commits on Apr 30, 2018:
 - Con submódulos (commit fd14b1a39ffab9511e9680921e275a7610ee124e)
- Commits on Apr 28, 2018:
 - Segundo respaldo (commit e48c129)
- Commits on Apr 26, 2018:
 - Primer respaldo (commit 36392ae)

At the bottom, there are 'Newer' and 'Older' buttons. The URL in the address bar is https://github.com/Carlos-Rodolfo-Rodriguez/Seudocodigo/commit/fd14b1a39ffab9511e9680921e275a7610ee124e.



En el repositorio remoto, vemos la rama **prueba**:

Carlos-Rodolfo-Rodriguez/Seudocodigo at prueba

No description, website, or topics provided.

Add topics

3 commits 2 branches 0 releases 1 contributor

Branch: prueba New pull request Create new file Upload files Find file Clone or download

This branch is 3 commits ahead, 5 commits behind master.

File	Commit Message	Time
Seudocódigo @ df74fd1	Con submódulos	10 hours ago
.gitignore	Segundo respaldo	2 days ago
.gitmodules	Con submódulos	10 hours ago
Ejercicio 01.cpp	Primer respaldo	4 days ago
Ejercicio 02.cpp	Primer respaldo	4 days ago
Ejercicio 03.cpp	Primer respaldo	4 days ago
Ejercicio 04.cpp	Primer respaldo	4 days ago
Ejercicio 05.cpp	Primer respaldo	4 days ago
Ejercicio 06.cpp	Primer respaldo	4 days ago
Ejercicio 07.cpp	Primer respaldo	4 days ago



git push -u <repRemoto> <rama> mantendrá sincronizada la rama entre los repositorios local y remoto:

- Pide usuario y clave de la cuenta en github.
- Da un mensaje final confirmando la *sincronización establecida*.

```
carlos@Lenovo-G50-80:~/Zinjai/HackerRank$ git push --set-upstream https://github.com/Carlos-Rodolfo-Rodriguez/Seudocodigo.git prueba
Username for 'https://github.com': carlos-rodolfo-rodriguez
Password for 'https://carlos-rodolfo-rodriguez@github.com':
Counting objects: 45, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (41/41), done.
Writing objects: 100% (45/45), 1.23 MiB | 70.00 KiB/s, done.
Total 45 (delta 8), reused 0 (delta 0)
remote: Resolving deltas: 100% (8/8), done.
To https://github.com/Carlos-Rodolfo-Rodriguez/Seudocodigo.git
 * [new branch] prueba -> prueba
Branch prueba set up to track remote branch prueba from https://github.com/Carlos-Rodolfo-Rodriguez/Seudocodigo.git
```



`git branch --merged`

- Muestra las ramas ya vueltas a mezclar.

`git checkout <rama>`

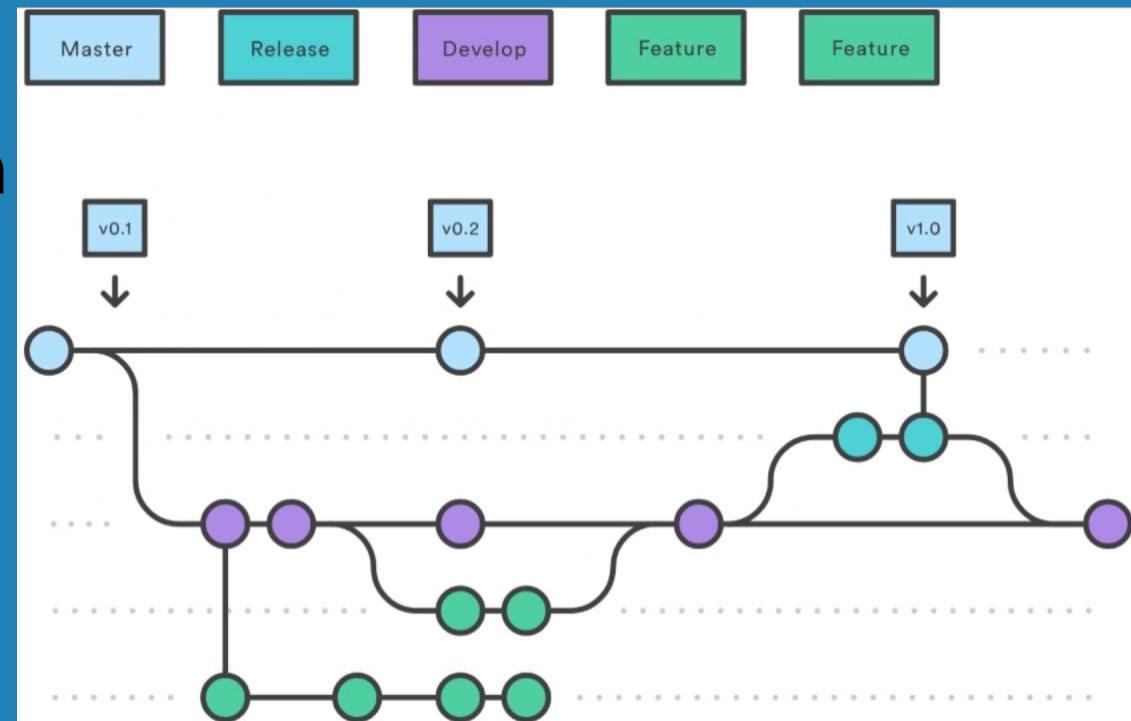
- Cambia la rama activa.

`git merge <rama>`

- Mezcla la rama con la rama activa.

`git branch -d <rama>`

- Elimina una rama.





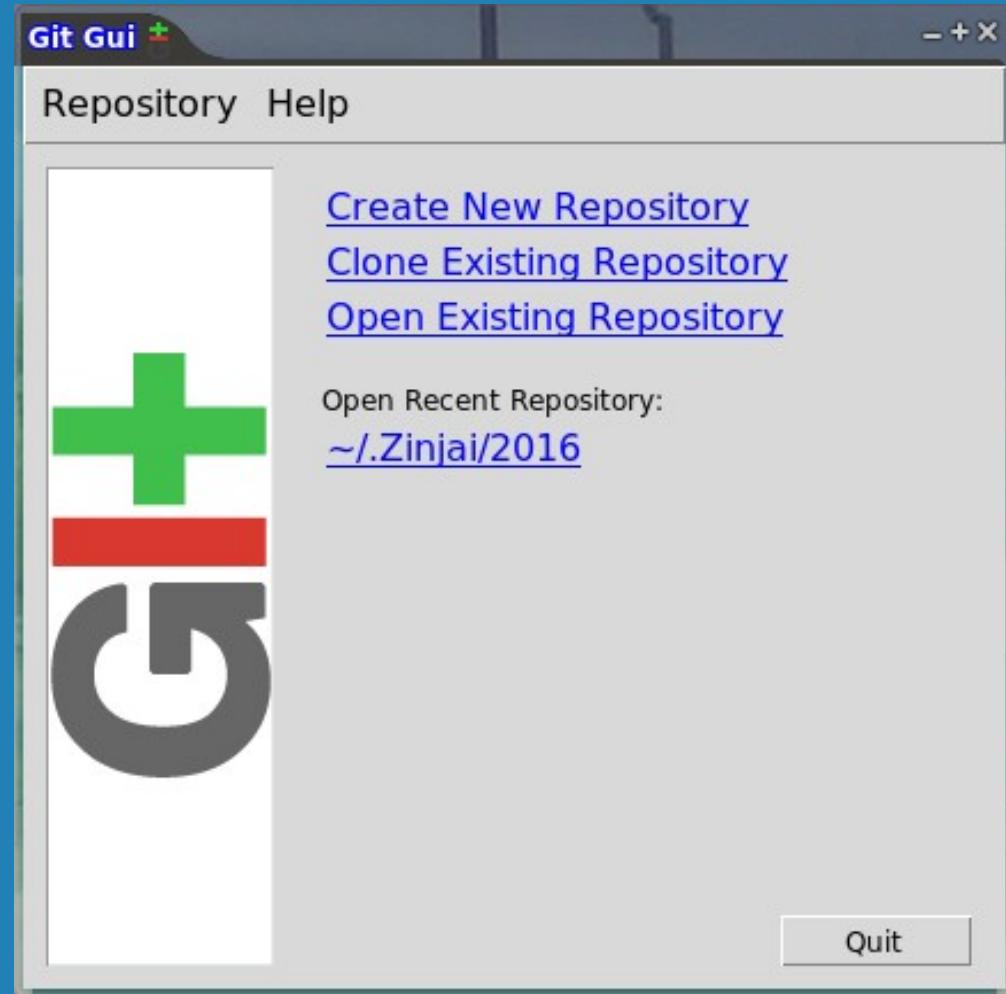
¿Cómo se usa GIT?

Desde la consola

- git init
- git clone
- Etc.

Desde el entorno gráfico

- git gui





Solucionar un problema de programación (01)

Enunciado

- En <https://adventofcode.com/2015/day/1>
- Está en inglés
 - ◊ Será necesario traducirlo
- Está redactado como un cuento
 - ◊ Habrá que compactarlo
- Solución(es) del problema
 - ◊ Una rama por solución
 - ◊ Habrá que mezclarlas y seleccionar la “mejor”



Solucionar un problema de programación (02)

Inicializar una carpeta como Adventofcode y posicionarse en ella

- mkdir ~/Adventofcode
- cd ~/Adventofcode

Inicializar el proyecto como “2015” y posicionarse

- git init 2015
- cd 2015

Mostrar:

- ls -la

```
carlos@Lenovo-G50-80:~/.Zinjai$ mkdir AdventOfCode
carlos@Lenovo-G50-80:~/.Zinjai$ cd AdventOfCode/
carlos@Lenovo-G50-80:~/.Zinjai/AdventOfCode$ git init 2015
Inicializado repositorio Git vacío en /home/carlos/.Zinjai/AdventOfCode/2015/.git/
carlos@Lenovo-G50-80:~/.Zinjai/AdventOfCode$ cd 2015/
carlos@Lenovo-G50-80:~/.Zinjai/AdventOfCode/2015$ ls -la
total 12
drwxrwxr-x 3 carlos carlos 4096 abr  2 12:15 .
drwxrwxr-x 3 carlos carlos 4096 abr  2 12:15 ..
drwxrwxr-x 7 carlos carlos 4096 abr  2 12:15 .git
carlos@Lenovo-G50-80:~/.Zinjai/AdventOfCode/2015$ █
```



Solucionar un problema de programación (03)

Crear un archivo llamado Enunciado.txt

touch Enunciado.txt

Traducir el enunciado y guardarlo en Enunciado.txt

Conectar con el Repositorio remoto en GitHub

Resolver el enunciado corto.

Subir la solución.

Comparar resultados.



GIT funciona localmente con
3 zonas:

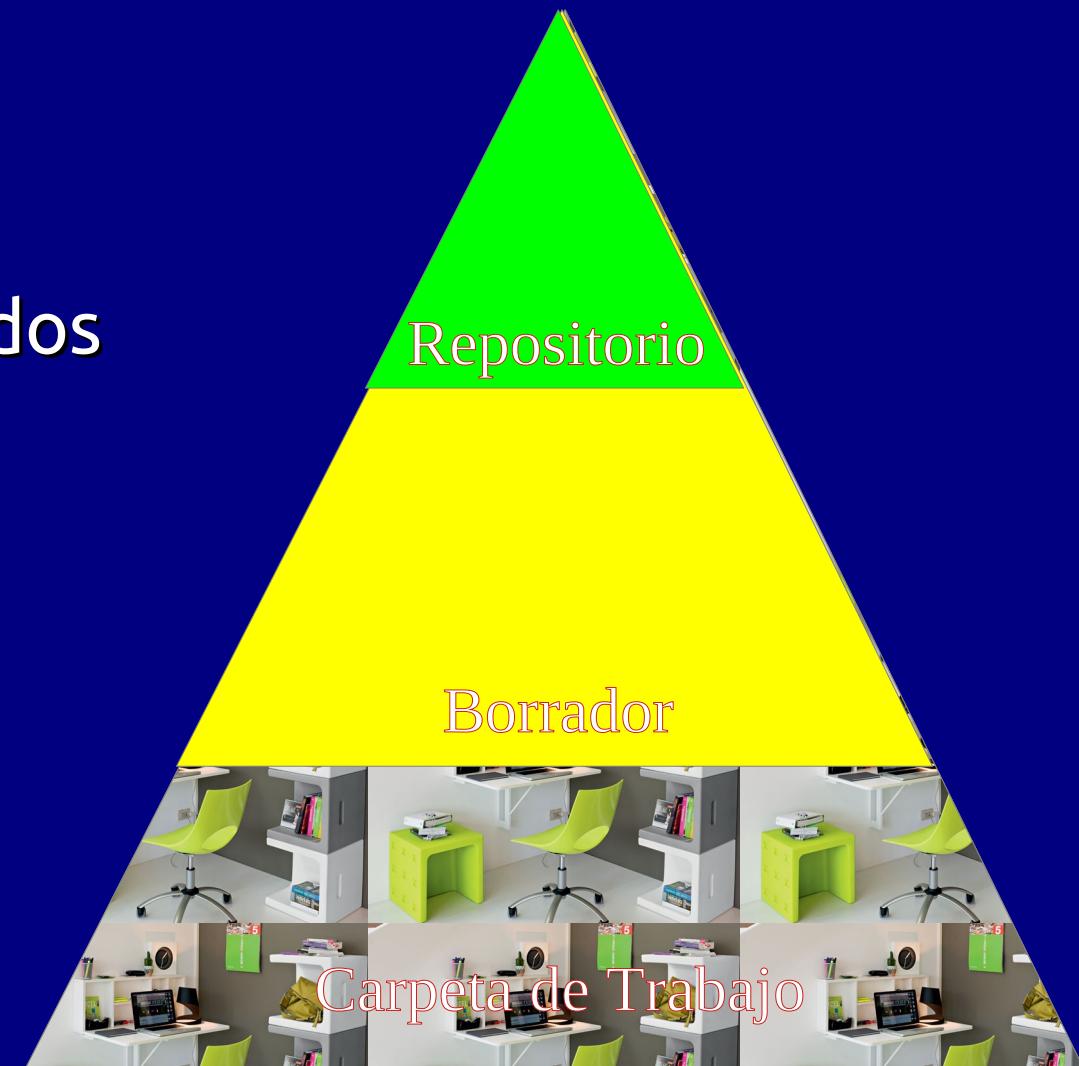
- ◊ Carpeta de Trabajo
- ◊ Borrador
- ◊ Repositorio





Carpeta de Trabajo

- Archivos a incluir
 - ◊ `git add`
- Archivos modificados
 - ◊ `git diff`
- Archivos a excluir
 - ◊ `.gitignore`





Borrador

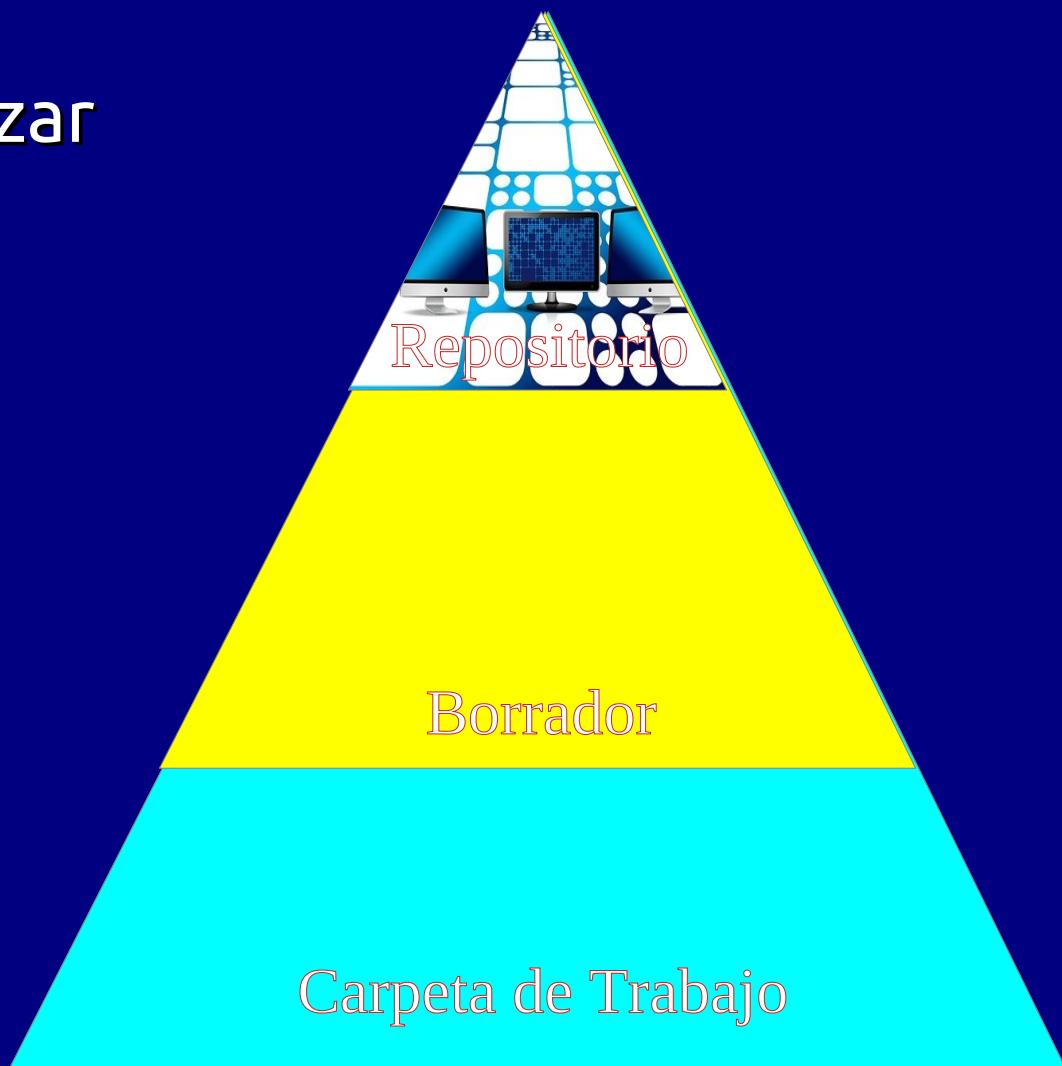
- Recopilación provisoria
 - ◊ Archivos modificados pero no actualizados al repositorio
 - ▷ `git diff --staged`
 - ◊ A grabar en el repositorio
 - ▷ `git commit`





Repositorio

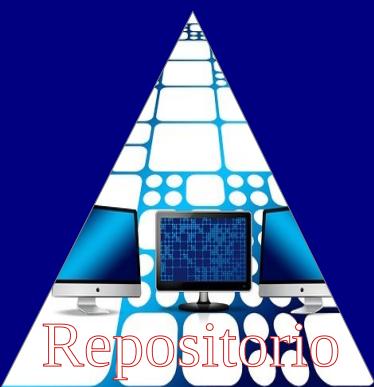
- Archivos a sincronizar
 - ◊ git remote





Repositorio remoto y repositorios locales

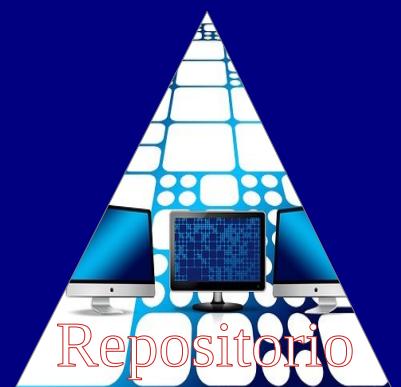
- `git remote add origin <dirección>`
- `git remote -v`



Repositorio



Repositorio



Repositorio



Repositorio remoto y repositorios locales

- `git push -u origin master`
- `git fetch`

