Introducción a git

¿Qué es?

- Sistema de control de versiones (SCM).

¿Qué es?

- Sistema de control de versiones (SCM).
- Esencialmente, rastrea los cambios de nuestros archivos (generalmente código fuente), y permite ir hacia atrás si hace falta.

¿Qué es?

- Sistema de control de versiones (SCM).
- Esencialmente, rastrea los cambios de nuestros archivos (generalmente código fuente), y permite ir hacia atrás si hace falta.
- Creado por Linus Torvalds en 2005...

¿Qué es?

- Sistema de control de versiones (SCM).
- Esencialmente, rastrea los cambios de nuestros archivos (generalmente código fuente), y permite ir hacia atrás si hace falta.
- Creado por Linus Torvalds en 2005... en semanas.

¿Qué es?

- Sistema de control de versiones (SCM).
- Esencialmente, rastrea los cambios de nuestros archivos (generalmente código fuente), y permite ir hacia atrás si hace falta.
- Creado por Linus Torvalds en 2005... en semanas.

git

/gɪt/

noun **DEROGATORY • INFORMAL**

an unpleasant or contemptible person.
"that mean old git"

¡Ah! ¿Como SVN?

- **No.**

¡Ah! ¿Como SVN?

- **No.** Si conocen SVN/CVS, les conviene olvidar todo lo que saben.

¡Ah! ¿Como SVN?

- **No.** Si conocen SVN/CVS, les conviene olvidar todo lo que saben.
- Principal diferencia: *distribuido*.

> Cada desarrollador tiene *todo* el código y *toda* la historia.

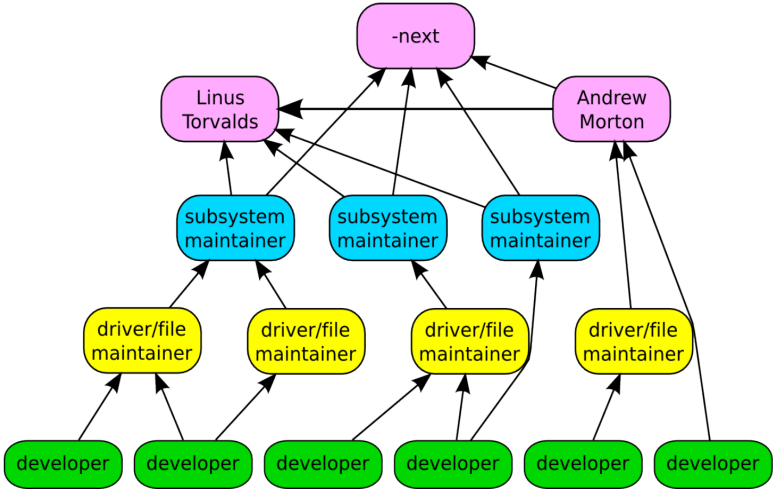
¡Ah! ¿Como SVN?

- **No.** Si conocen SVN/CVS, les conviene olvidar todo lo que saben.
- Principal diferencia: *distribuido*.

> Cada desarrollador tiene *todo* el código y *toda* la historia.

- Enviar/recibir cambios de otro host es *infrecuente*. Incluso sin ningún otro colaborador, `git` ayuda para organizarse.

Flexible, jerárquico, distribuido



Génesis

¿Cómo arranca un repositorio?

- 1, desde cero:

```
~$ mkdir proyecto/
~$ cd proyecto/
~/proyecto$ git init
Initialized empty Git repository in /home/guido/proyecto/.git/
```

Génesis

¿Cómo arranca un repositorio?

- 2, clonando:

```
~$ git clone https://github.com/torvalds/linux
Cloning into 'linux'
remote: Enumerating objects: 8716629, done.
remote: Total 8716629 (delta 0), [...]
Receiving objects: 100% [...]
Resolving deltas: 100% [...]
Updating files: 100% [...], done.
~$
```

Mirando el estado...

```
~/proyecto$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

Mirando el estado...

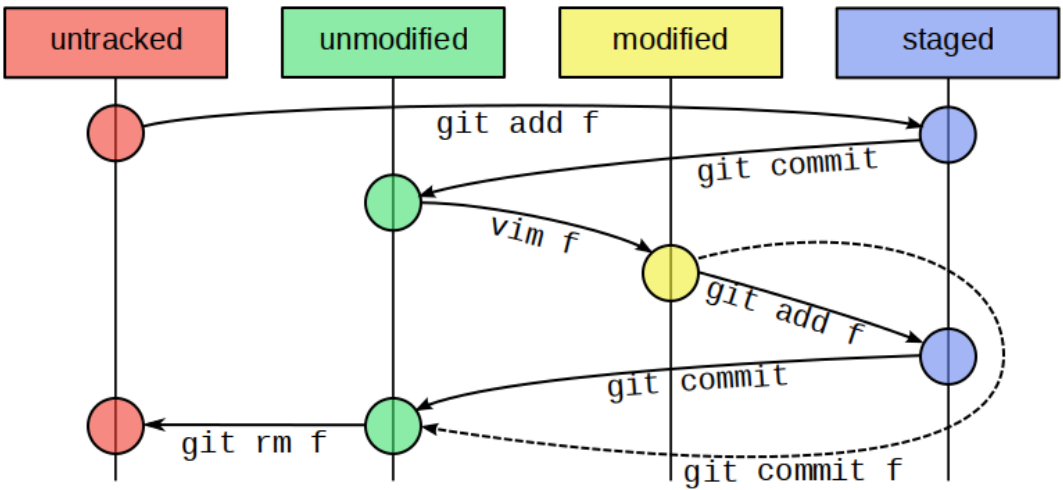
```
~/proyecto$ touch blah.c
~/proyecto$ echo 'int main(){return 0;}' > main.c
~/proyecto$ git status
On branch master

No commits yet

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

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

Estados


untracked
unmodified
modified
staged
git add f
git commit
vim f
git add f
git commit
git rm f
git commit f

Staging area

```
~/proyecto$ git add main.c
~/proyecto$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   main.c

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

Commit

```
~/proyecto$ git commit -m "Inicio proyecto"
[master (root-commit) 5a2e447] Inicio proyecto
 1 file changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 main.c
```

Commit

```
~/proyecto$ git commit -m "Inicio proyecto"
[master (root-commit) 5a2e447] Inicio proyecto
 1 file changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 main.c

~/proyecto$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        blah.c

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

¿Qué es un commit?

- Snapshot **completa** del árbol (con optimizaciones de espacio)
- Es un hash **criptográfico** de:
  - Todos los archivos
  - Mensaje de commit
  - Autor, fecha, etc
  - Commit padre

¿Qué es un commit?

- Snapshot **completa** del árbol (con optimizaciones de espacio)
- Es un hash **criptográfico** de:
  - Todos los archivos
  - Mensaje de commit
  - Autor, fecha, etc
  - Commit padre
- Criptográfico = no se puede invertir, ni encontrar colisiones (eficientemente)

¿Qué es un commit?

- Snapshot **completa** del árbol (con optimizaciones de espacio)
- Es un hash **criptográfico** de:
  - Todos los archivos
  - Mensaje de commit
  - Autor, fecha, etc
  - Commit padre
- Criptográfico = no se puede invertir, ni encontrar colisiones (eficientemente)
- Obviamente... con optimizaciones para no recomputar el hash desde cero cada vez (ver *Merkle trees*)

Log

```
~/proyecto$ git log
commit 4e80d6ce4a584e0a1d0708489f29dd60f6d67758
Author: Guido Martínez <mtzguido@gmail.com>
Date:   Tue Apr 19 10:01:43 2022 -0300

    Inicio proyecto
```

Log

```
~/proyecto$ git log --stat
commit 4e80d6ce4a584e0a1d0708489f29dd60f6d67758
Author: Guido Martínez <mtzguido@gmail.com>
Date:   Tue Apr 19 10:01:43 2022 -0300

    Inicio proyecto

 main.c | 1 +
 1 file changed, 1 insertion(+)
```

Log

```
~/proyecto$ git log -p
commit 4e80d6ce4a584e0a1d0708489f29dd60f6d67758
Author: Guido Martínez <mtzguido@gmail.com>
Date:   Tue Apr 19 10:01:43 2022 -0300

    Inicio proyecto

diff --git a/main.c b/main.c
new file mode 100644
index 0000000..2c99a52
--- /dev/null
+++ b/main.c
@@ -0,0 +1 @@
+int main(){return 0;}
```

Diff Unificado

```
~/proyecto$ git show
commit 61af2d85a1c4d2279c0dc6bc514accb1d96484df
Author: Guido Martínez <mtzguido@gmail.com>
Date:   Tue Apr 19 10:20:24 2022 -0300

    Corregir typo

diff --git a/main.c b/main.c
index 7c0913d..00dd468 100644
--- a/main.c
+++ b/main.c
@@ -1,5 +1,5 @@
 int main()
 {
-       printf("Holamundo\n");
+       printf("Hola mundo\n");
        return 0;
 }
```

Digresión: buenas prácticas

- Commits lo más pequeños posibles ("atómicos"): permite revertir fácilmente
- Mensajes descriptivos: "cambios" vs "Corregir algoritmo de Peterson con mfence"
- Nunca romper el build: permite bisectar (`git bisect`)

Borrar archivos

- `git rm` borra un archivo (y anota el cambio en la staging area). Es lo mismo que hacer `rm` y `git add`.

```
~/proyecto$ git rm main.c
rm 'main.c'
~/proyecto$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    main.c
$ git commit -m "Borrar main"
[master f068e5f] Borrar main
 1 file changed, 5 deletions(-)
 delete mode 100644 main.c
```

Config

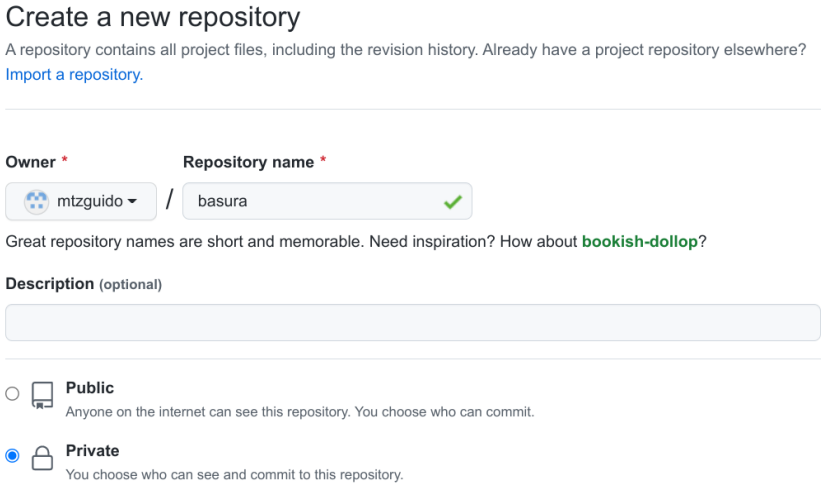
```
$ git config --global user.name "Alan Turing"
$ git config --global user.email "aturing@princeton.edu"
```

Remotes

Hasta ahora, **nada** requirió de la red.

Remotes

Hasta ahora, **nada** requirió de la red.



Remotes

Hasta ahora, **nada** requirió de la red.

```
~/proyecto$ git remote add origin git@github.com:mtzguido/basura
~/proyecto$ git push -u origin master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (9/9), 789 bytes | 789.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:mtzguido/basura
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
~/proyecto$
```

Push/pull

- **Push:** actualiza el repo remoto desde el local (sólo cambios commiteados)
- **Pull:** actualiza el repo local desde el remote

La base está...

Revirtiendo cambios

- `git checkout <file>`: revierte cambios locales.
- `git reset`: vacía el staging area.
- `git reset <commit>`: vuelve al commit, sin modificar archivos.
- `git reset --hard <commit>`: vuelve al commit, descartando **todo**.

Branches

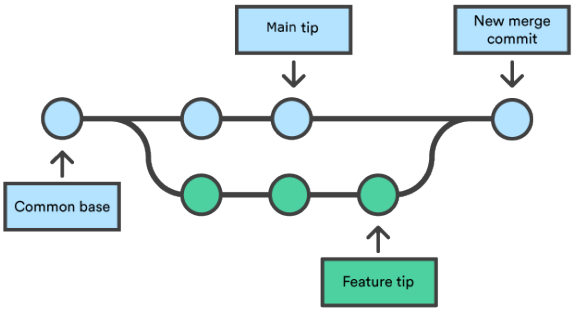
- Un branch es un nombre que “apunta” o “sigue” a un commit hash
- `git branch featureX`: crear y moverse a un branch
- `git checkout <b>`: cambiar de branch
- `master` suele ser la rama principal

Merges

- Toma dos commits y *une* sus cambios en uno.
- En branch *source* `git merge <commit>`. Pull tiene merge implícito, push sólo permite "fast-forwards".

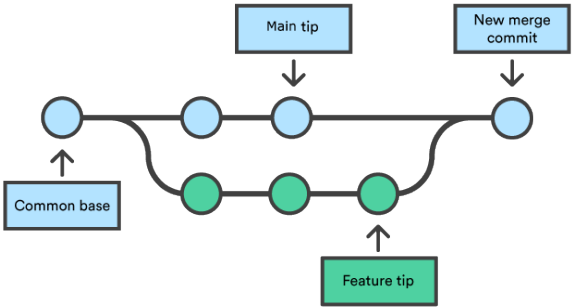
Merges

- Toma dos commits y *une* sus cambios en uno.
- En branch *source* `git merge <commit>`. Pull tiene merge implícito, push sólo permite "fast-forwards".



Merges

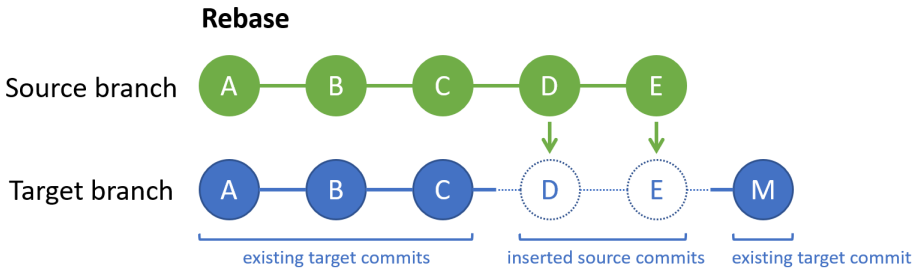
- Toma dos commits y *une* sus cambios en uno.
- En branch *source* `git merge <commit>`. Pull tiene merge implícito, push sólo permite "fast-forwards".



- Posiblemente haya que **corregir conflictos**
- Interfaz gráfica: `gitg` (o `git log --graph`)

Rebase

- Similar a un merge... pero “reescribe la historia”



- Generalmente sólo se hace en ramas privadas

Clean

Usar con cuidado...

- `git clean -dfx`: borra todo lo que no esté trackeado/staged
- `git clean -x`: borra sólo archivos ignorados (suele ser seguro)
- Flag `-n`: no hacer nada, imprimir lo que haría

Otras features

- Aliases: abreviar comandos (`git st` == `git status`, etc).
- Hooks: el remoto toma alguna acción al recibir un push.
- "Plomería" vs "Porcelana", permite scripting.
- `git bisect`: encontrar el commit que introdujo un bug.
- `git worktree`: mismo repositorio, *N* árboles de trabajo.
- `git bisect` automático.
- `git blame`: ver quién escribió cada línea.

Referencias

- Libro:
  `https://git-scm.com/book`
- TryGit:
  `https://try.github.io/`
- A Visual Git Reference:
  `https://marklodato.github.io/visual-git-guide/index-en.html`
- Git From the Bottom Up:
  `http://ftp.newartisans.com/pub/git.from.bottom.up.pdf`
- Why Git is Better than X:
  `https://bryankaraffa.github.io/whygitisbetter/`
- Learn Git Branching:
  `https://learngitbranching.js.org/`
- Charla Torvalds:
  `https://www.youtube.com/watch?v=4XpnKHJAok8`