simplificando git

piérdele el miedo a la línea de comandos

de antemano

- La charla se grabará y se compartirá su acceso
- Las diapositivas se compartirán al final de la charla
- Cualquier duda no dudes en comentarla (chat o voz, lo que prefieras :D)

¿para quién está pensada la charla?

- Aquellas personas que le tienen reparo a los comandos
- Quienes no hayan usado, o apenas, git
- Quienes tengan algo de experiencia pero les gustase más soltura
- De principiante a intermedio
  - Avanzado es responder en stackoverflow, y eso me queda lejos
- Personas con afán de revisitar conceptos

estructura de las charlas

- I – Introducción y modelo mental, los fundamentos
- II – Manejo de *armas*, aprende a defenderte
- III – Profundización y salvavidas, entiende errores y aprende su *origin*

presentaciones

Soy **Pepe**, aunque Jose no me molesta, hago cosas de front, experto en tirar abajo producción, años de experiencia constatada rompiendo ramas en conflictos de merge.

Odiaba los comandos… y ahora… no tanto

introducción a git

Antes de nada, vamos a explicar algunos conceptos que ayudarán a entender/repasar cómo funciona Git

Git es un Sistema de Control de Versiones (VCS), esto quiere decir que nos permite **gestionar** un **historial de versiones** de lo que queramos

Vamos a empezar desde lo más básico e ir construyendo desde ahí

temas a tratar

- Origin
- Upstream
- stategy option
  - ours/theirs
- Merge
- Rebase
- Bisect
- Checkout
- Branches
- branch rename
- crear ramas con checkout
- Remote
- git stash
  - keys y pop
- git fetch vs git pull vs git push
  - git fetch especial (git pull && git push)
- git cherry pick
- Entornos
  - stating, local, remote, origin
- Add, commit, reset
- git config
  - repo, system, global, user
- Git patch
- Git submodules and subtrees
- Git log
- Git rebase interactive
- Reflog
- deltas

modelo mental

Antes de empezar explicaremos dos conceptos, por encima:

- Hashes
- Listas enlazadas (Linked Lists)

La idea de este “modelo mental” es tener ejemplos cercanos para poder entender la *magia* de git

hashes

Funciones matemáticas
Únicas, uid, un hash no da dos veces el mismo id
Inmutabilidad, un cambio, nuevo hash

listas enlazadas

Estructuras de datos
Nodos anteriores y posteriores

ESQUEMAS

- Un cambio altera la lista enlazada, y cada cambio en la lista enlaza es un nuevo hash, explicar entonces porque hace un rebase el cambio y aparecen commits a descargar
- Dibujo de una lista enlazada
- Dibujo simplificado de un hash

Ideas

- en git, para lo de git, si la rama es un hilo, un merge es hacerle un nudo y un rebase es deshilacharlo y volverlo a hilar desde determinado punto, según el contexto, una cosa interesa más que la otra

entornos

- **untracked**
  - git no tiene constancia de la existencia de este archivo, es decir, se acaba de “crear”, al menos para git
- **unstaged**
  - se ha modificado (su contenido ha cambiado, o se ha eliminado), pero no está en **staging**
- **staging**
  - hemos registrado cambios, y los subimos al área antes de un **commit**
- **committed**
  - un conjunto de cambios en **staging** a los que se les otorga un **commit** al que pertenecer
- **remote**
  - un **commit** que se encuentra en la fuente de verdad (**remote**) de la rama

add

Nos permite habilitar el seguimiento y pasar a **staging** los archivos o carpetas que queramos.

Archivos o carpetas, porque podemos añadir elementos y subelementos de una.

Git add . Añade todo el contenido apartir del punto de llamada.

Git add se ve sujeto al **.gitignore**, solamente aquellos archivos que NO se encuentren excluidos por el **.gitignore** se subirán

gitignore

Permite ignorar archivos, o crear reglas para **ignorar**, estas reglas pueden ser **exclusivas** o **inclusivas**, ignora todo esto, no ignores nada salvo que tenga esto.

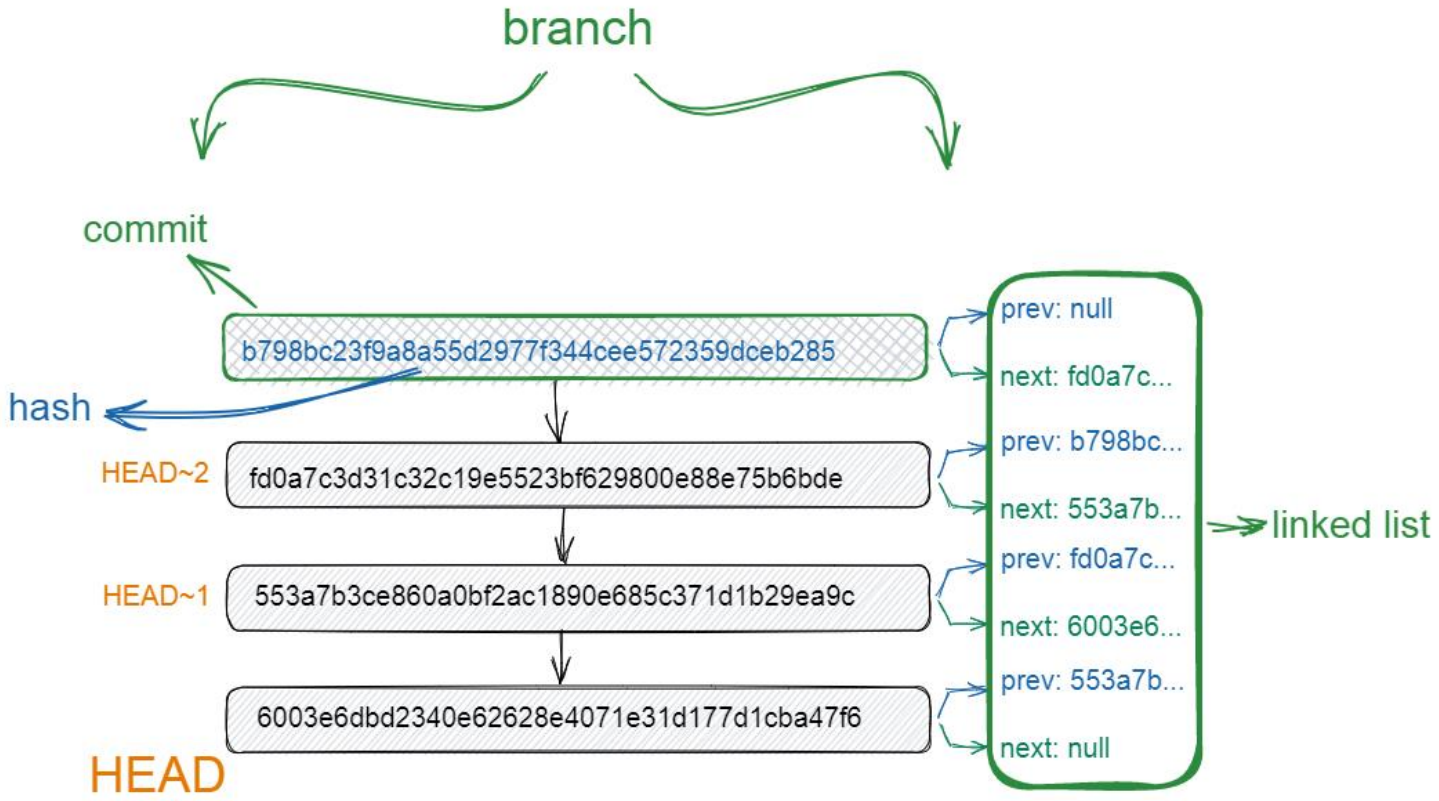
Añadir ejemplos de exclusivo e inclusivo, y de ficheros normales

Es importante destacar que si git ya tiene constancia de un archivo, añadirlo a un **.gitignore** podrá no tener el funcionamiento que deseas.

- Para arreglarlo podemos usar **git rm –cached […fichero(s)]**

recapitulando...

- git init
- git add
- git commit
- git branch
- git checkout
- git push
- git pull
- git fetch
- git merge
- git rebase



ganando confianza con los comandos

- simplifying-git-gaining-confidence-in-the-cli

- Git rebase interactive
  - Squash
  - Edit
  - Rename
  - Remove
  - Add
- Git log
- Git config

los salvavidas

- --no-verify
  - Commit and push
- --amend
- --allow-empty
- Cherry-pick
- --strategy-options
- Git bisect
- Rm --cached

créditos

- **uiGradients**, los degradados han sido en su mayoría de aquí
  - https://uigradients.com/
- **excalidraw**, diagramas y dibujitos
  - https://excalidraw.com/

expandiendo el conocimiento

- Aprendiendo Git – Miguel Angel Durán
  - https://leanpub.com/aprendiendo-git