Taller de  git

Agenda

1. Git
   - Qué es Git?
   - Terminología importante
   - Flujo de trabajo
2. Instalación
3. Configuración
4. Empezando con Git
   - Crear contenido
   - Crear repositorio y hacer commit
   - Ver diferencias con diff
   - Corrección de mensajes
5. Repositorios externos
6. Revertir cambios
7. Branches y merging
   - Branches
   - Merging
   - Borrar un branch
8. Resolver conflictos
9. Rebase
10. Proveedores de hosting Git
    - Github
    - Bitbucket

Git

1.1. ¿Qué es Git?

Git es un sistema de control de versiones distribuido escrito en C.

Un sistema de control de versiones permite la creación de una historia para una colección de archivos e incluye la funcionalidad para revertir la colección de archivos a otro estado.

Permite cambiar una colección de archivos (código fuente), por ejemplo, a un estado de 2 días atrás, o permite cambiar entre los estados para características experimentales y problemas de producción.

Si haces cambios al código fuente, haces esos cambios relevantes para el control de versiones (los agregas al staging index) y después los agregas al repositorio (commit).

1.1. ¿Qué es Git? (cont.)

Git realiza los commits a tu repositorio local y es posible sincronizar ese repositorio con otros (tal vez remotos) repositorios.

Git te permite clonar los repositoriso, por ejemplo, crear una copia exacta de un repositorio incluyendo la historia completa del código fuente.

Los dueños de los repositorios pueden sincronizar los cambios via push (transfiere los cambios al repositorio remoto) o pull (obtiene los cambios desde un repositorio remoto).

Git soporta el concepto de branching, es decir, podés tener varias versiones diferentes de tu código fuente.

Si querés desarrollar una nueva característica, podés abrir un branch en tu código fuente y hacer los cambios en este branchsin afectar el principal desarrollo de tu código.

1.2. Terminología importante

Repositorio: contiene la historia, las diferentes versiones en el tiempo y todas los distintos branch y etiquetas.

Branch: Son utilizadas para desarrollar funcionalidades aisladas unas de otras. El branch *master* es el branch "por defecto" cuando creas un repositorio. Crea nuevos branches durante el desarrollo y fusiónalos al branch principal cuando termines.

Tags: una etiqueta apunta a un cierto punto en el tiempo en un branch específico. Con un tag, es posible tener un punto con un tiempo al cual siempre sea posible revertir el código.

Commit: Contiene los cambios a un repositorio, el autor y el que realizó el commit (comiteador), siendo así posible identificar el origen del cambio.

URL: Una URL en Git determina la ubicación de un repositorio.

Revisión: Representa una versión del código fuente. Git identifica revisiones con un id SHA1. Los id son de 160 bits de largo y son representados en hexadecimal

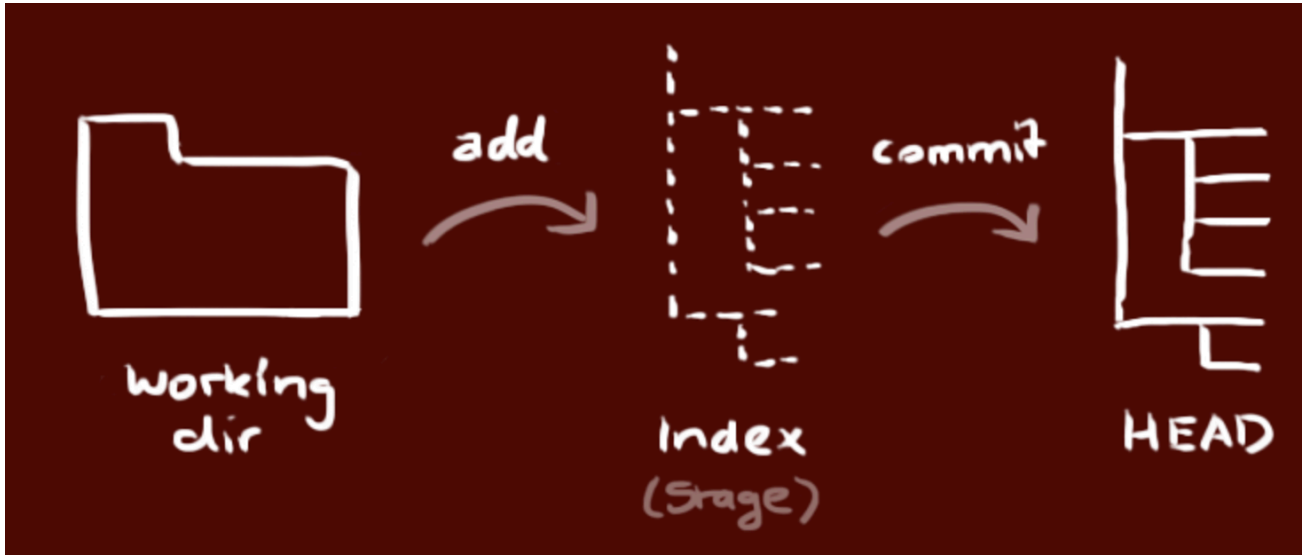
1.3. Flujo de trabajo

Tu repositorio local esta compuesto por tres "árboles" administrados por git.

El primero es tu **Directorio de trabajo** que contiene los archivos.

El segundo es el **Index** que actua como una zona intermedia.

El último es el **HEAD** que apunta al último commit realizado.



2. Instalación

Para usar en terminal:

- **Linux (Debian)**
  - sudo apt-get install git
- **Linux (Fedora)**
  - sudo yum install git
- **Mac**
  - http://git-scm.com/download/mac
- **Windows**
  - http://git-scm.com/download/win

Github Desktop:

https://desktop.github.com/

3. Configuración

3.1. Configuración del usuario

- git config --global user.name ”Raul Madrigal”
- git config --global user.email raul.madrigal@taller.de.progra.com

3.2. Resaltado de colores

- git config --global color.status auto
- git config --global color.branch auto

3.3. Ignorar ciertos archivos

Git puede ser configurado para que ignore ciertos archivos y directorios. Esta configuración es realizada mediante el archivo .gitignore. Este archivo puede estar en cualquier directorio y puede contener patrones para archivos.

4. Empezando con Git

Navegar al home

- cd ~/

Crea un directorio

- mkdir taller

Ingresar a ese directorio

- cd taller

Crea algunos archivos

- touch test01.txt
- touch test02.py
- touch test03.py

Agregar algo de texto adentro del primer archivo

- ls > test01.txt

4.2. Crear el repositorio y hacer un commit

Cada repositorio Git es almacenado en la carpeta .git del directorio en el cual el repositorio ha sido creado.

El archivo .git/config contiene la configuración local del repositorio.

Inicializamos el repositorio local Git

- git init

Agregamos todo (archivos y directorios) al repositorio

- git add .

Hacemos un commit al repositorio

- git commit -m "Initial commit”

Muestra el log (un historial)

- git log

4.3. Ver las diferencias via diff y commitear los cambios

El comando **diff** de Git permite al usuario ver los cambios hechos.

Agregamos cambios a los archivos

- echo "Este es un cambio" > test01.txt
- echo ”print(’Soy un archivo de python’)" > test02.py

Observar los cambios

- git diff

Agregar archivos con cambios

- git add test01.txt test02.py

Comitear los cambios

- git commit -m ”Soi hun comentario con herrores ortográficos"

4.4. Corrección de mensajes de commit - git amend

El comando **amend** hace posible cambiar el último mensaje de commit.

- git commit --amend -m ”Soy el comentario del commit arreglado”

También permite agregar más archivos al commit anterior, en caso de haber olvidado agregar algún archivo.

5. Trabajando con repositorios externos

Si se está creando el repositorio externo

- git init
- git remote add origin <url-servidor>

Si utilizas un servidor remoto, ejecuta

- git clone username@host:/path/to/repository

Enviar cambios al repositorio remoto

- git add archivos
- git commit –m ”comentario”
- git push origin <nombre_branch>

6. Revertir cambios

Deshacer los cambios de un archivo:

Agregar cambios

- echo "Este es otro cambio que vamos a deshacer" > test01.txt

Ver archivos modificados

- git status

Deshacer cambios

- git checkout test01.txt

7. Branches

Crea una nuevo branch llamada ”progra_1" y cámbiate a ella usando

- git checkout -b progra_1

vuelve a la rama principal

- git checkout master

y borra la rama

- git branch -d progra_1

Una rama nueva no estará disponible para los demás a menos que subas (push) la rama a tu repositorio remoto

- git push origin <branch>

7.1 Merging

Para actualizar tu repositorio local al commit más nuevo, en tu directorio de trabajo para *bajar* y *fusionar* los cambios remotos.

- git pull

Para fusionar otra rama a tu rama activa (por ejemplo master), utiliza

- git merge <branch>

8. Resolver conflictos

Un conflicto ocurre si dos personas han modificado el mismo contenido y Git no puede determinar automaticamente como deberían ser aplicados ambos cambios.

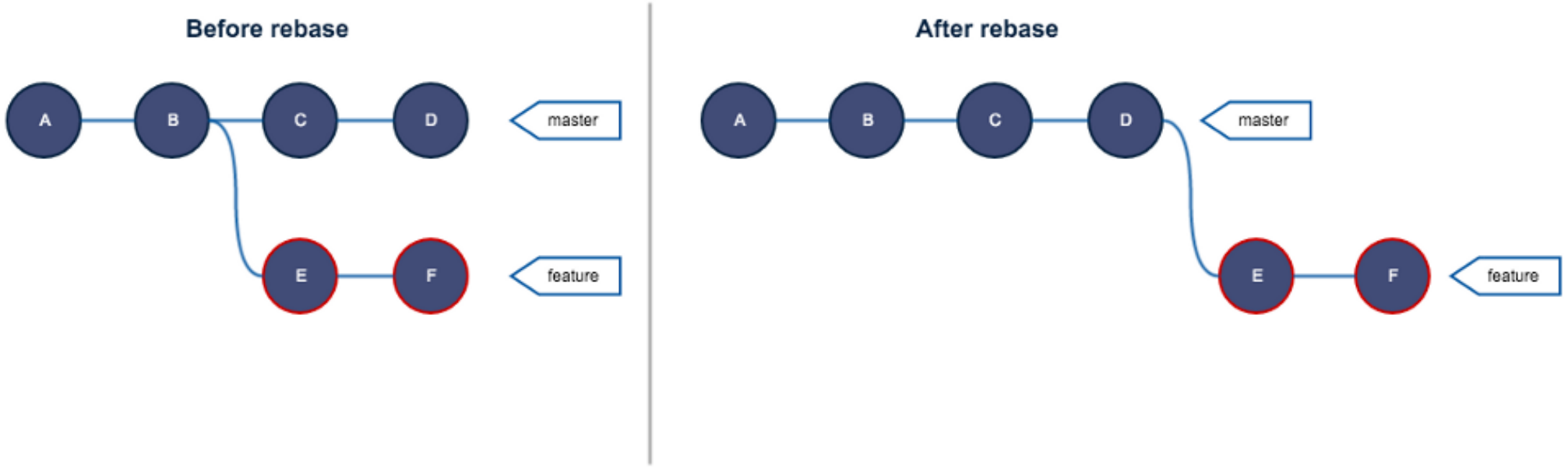
Git marca el conflicto en el archivo afectado. El archivo conflictivo contiene algo parecido a:

```
1  <<<<<<< HEAD
2  Cambio en el primer repo
3  =======
4  Cambio en el segundo repo
5  >>>>>>> b29196692f5ebfd10d8a9ca1911c8b08127c85f8
6
7
```

La parte superior (sobre la linea ====) contiene el contenido de tu repositorio, y la parte inferior es la que tiene el contenido del repositorio remoto. Ahora podés editar el archivo manualmente y después comitear los cambios.

9. Rebase

- git rebase <nombre-branch>



10. Proveedores de git

- Bitbucket
- Github