Git

Nivel básico

Sistemas de control de versiones

Centralizado

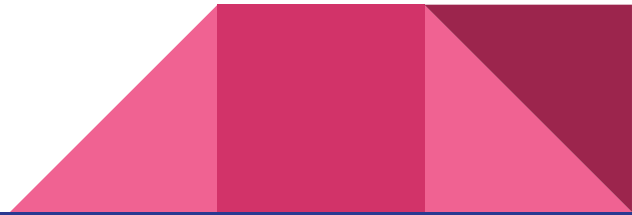
- Una única copia centralizada del proyecto.
- No dispone de ramas.
- No permite trabajar offline.
- SVN.

Distribuido

- No es necesario un repositorio central (clonación).
- Permite la creación de ramas.
- Permite trabajar offline, ya que cada desarrollador tiene una copia funcional del servidor.
- Git y Mercurial.

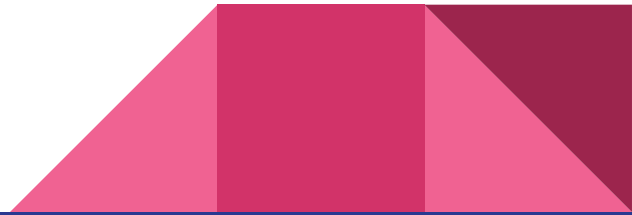
¿Por qué los usamos?

- Para tener la posibilidad de deshacer cambios puntuales que hayamos hecho o volver al código de un determinado momento.
- Para poder trabajar con múltiples personas al mismo tiempo.
- Automatización de ciertas tareas, como los tests, véase Jenkins o Travis.



Git

- Creado en 2005 por Linus Torvalds.
- Se creó debido a que Bitkeeper, propietaria del VCS que estaban usando hasta el momento les retiró la licencia gratuita (acusaron a uno de los ingenieros de hacer ingeniería inversa) así que, cuando buscaron opciones gratuitas no encontraron ninguna que rindiera lo suficiente con tantos trabajadores. Así que crearon la suya.

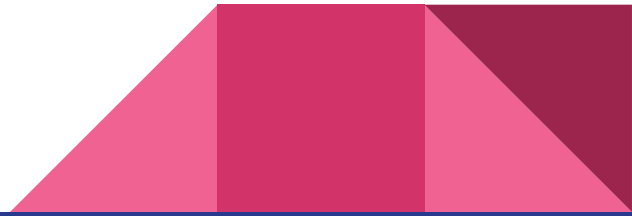


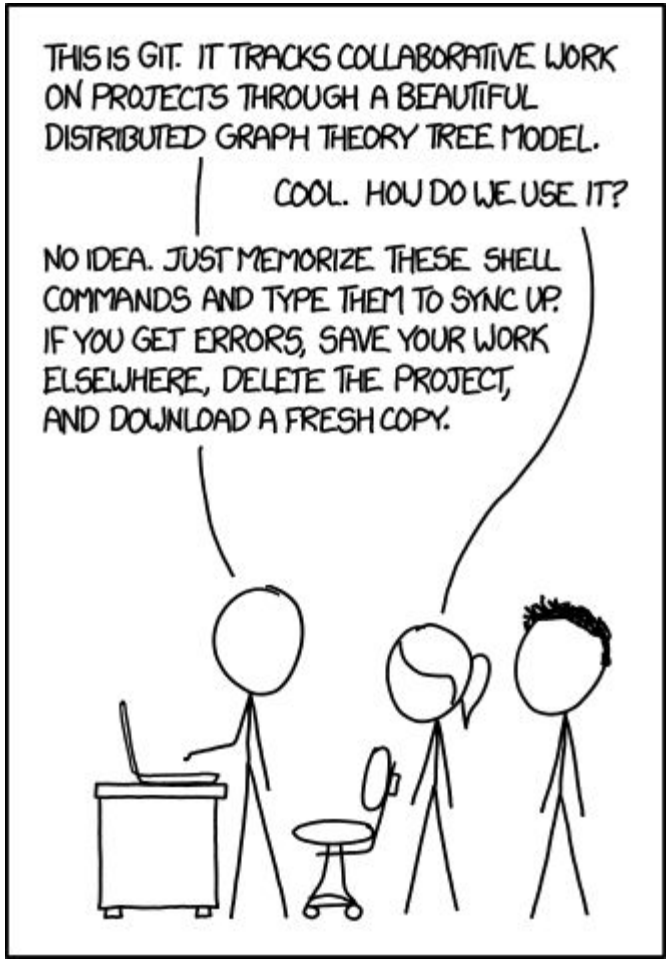
Comandos básicos - 1

- git clone <url> <path>: clona el repositorio a local.
- git pull <remote> <rama>: descarga los cambios del servidor remoto.
  - Si no pones la rama, se descargan todas.
- git add <path>: añade las modificaciones (recursivamente) al commit que vas a hacer.
- git commit: crea un commit (conjunto de cambios). Necesita un mensaje.
  - -m <mensaje>: añades el mensaje directamente en el comando.
- git push <remote> <rama>: sube los commits que has creado al servidor.

Workflow - 1

1. Clonar el repositorio.
2. Programar.
3. Hacer un commit cada vez que hagas algo importante / cada x tiempo.
4. Subirlo al servidor o seguir trabajando en local.
5. Volver a programar o, si vas a terminar, subir todos los commits al servidor.




THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

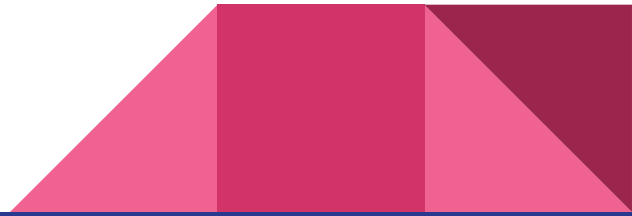
GUIs

Programas gráficos que puedes usar en vez de el terminal:

- GitHub app (Windows y MacOSX).
- SourceTree (Windows y MacOSX).
- gitk (Linux): muestra el historial de cambios únicamente.

GitHub

- Creado en 2008.
- Ofrece servidores de Git gratuitos para los usuarios. Cantidad ilimitada hasta 1 Gb cada uno.
  - https://education.github.com/pack
- Características de redes sociales: seguidores, timeline, …
- Permite crear una wiki para cada proyecto, así como una página web propia.

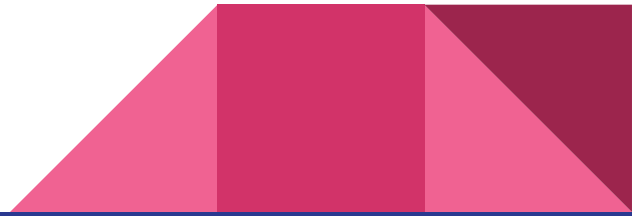


Funcionamiento de GitHub

- El funcionamiento básico de GitHub es el siguiente:
  - Si tienes acceso directo al repo:
    - Ya está, clonas el repositorio y puedes hacer push.
  - Si no tienes acceso directo pero quieres colaborar:
    - Tienes que hacer un *Fork* del repositorio. Así creas un copia de él en tu cuenta.
    - Clonas y editas lo que quieras.
    - Y luego haces un *Pull Request* en el repositorio principal. Con él solicitas que aprueben los cambios que tú has hecho en el proyecto y los añadan al repositorio principal.

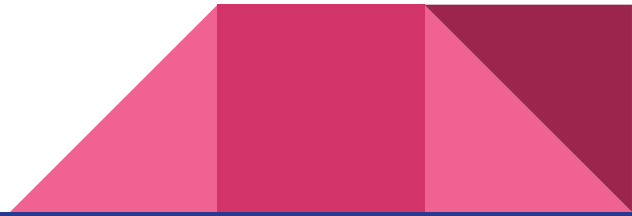
Ejercicios - 1

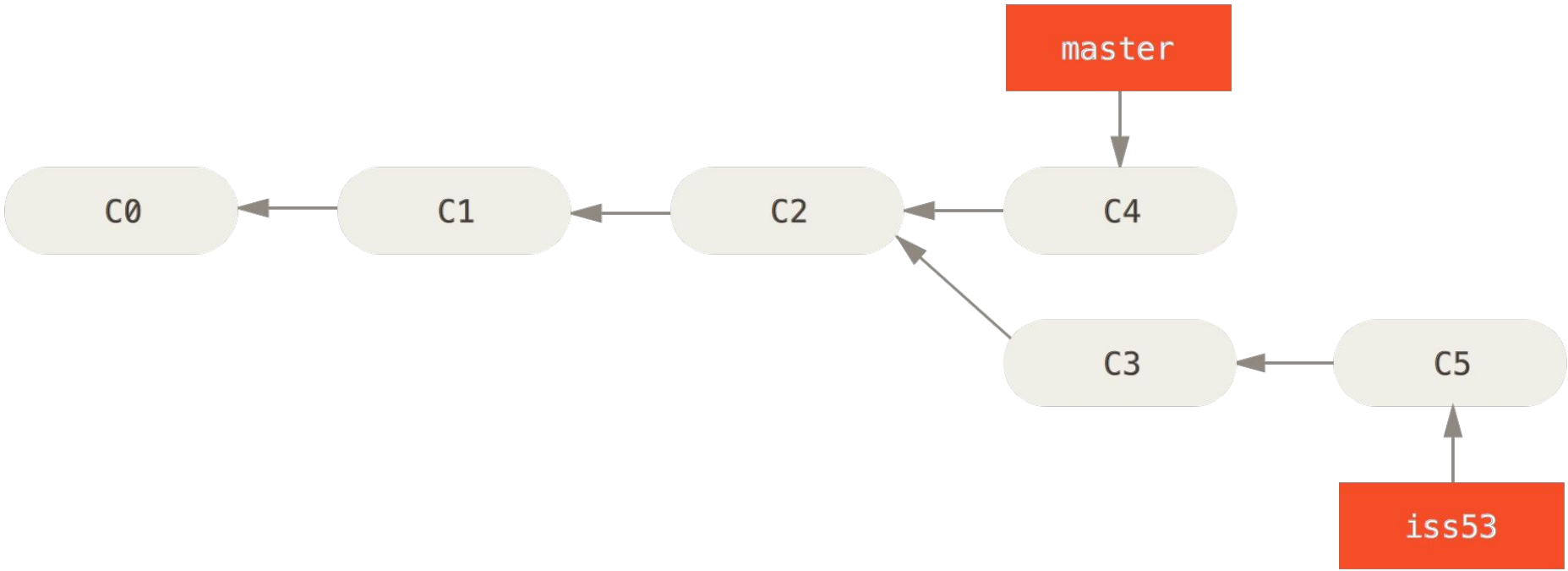
- Forkear el repositorio: https://github.com/StudentTechClubAsturias/TallerGitBasico
- Clonarlo al ordenador.
- Arreglar el bug del script de python.
- Subirlo de nuevo.



Ramas

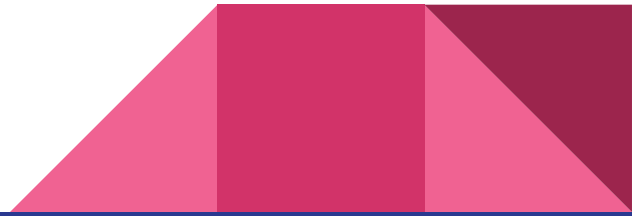
- Los commits se agrupan en ramas, siendo la rama principal *master*.
- Cada programador puede crear una rama individual, trabajar sobre ella y luego volver a unirla con la rama principal.
- Permite trabajar a varios programadores al mismo tiempo sobre el mismo código.
- Cada rama tiene una etiqueta que apunta al último commit realizado y cada commit apunta al anterior. De esta manera poder reconstruir la “historia”.

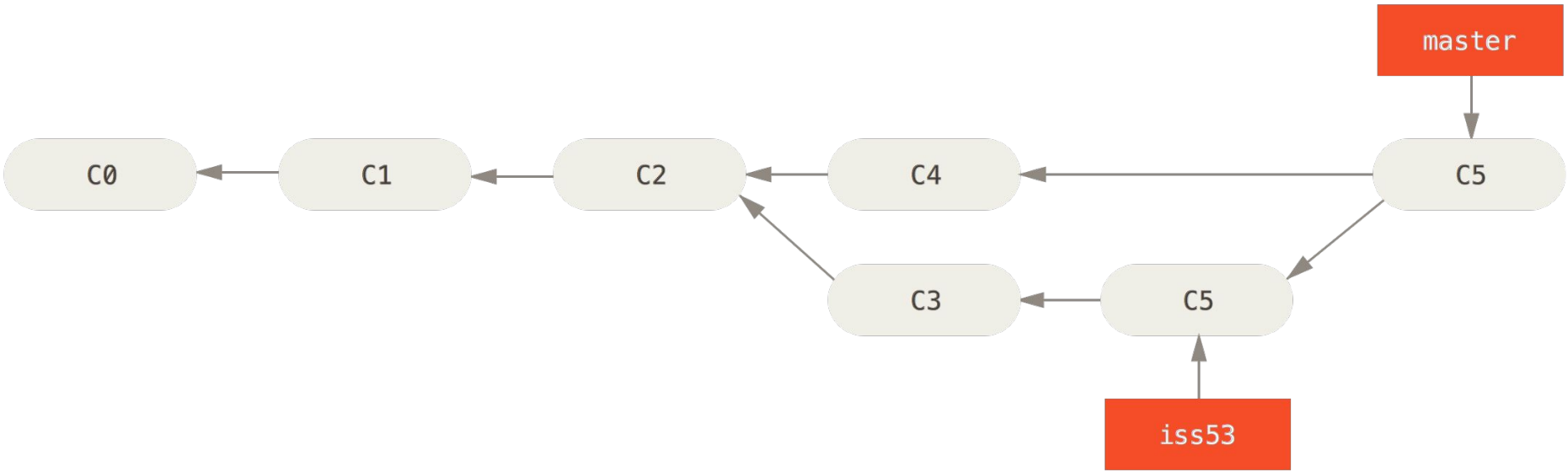



master
C0
C1
C2
C4
C3
C5
iss53

Conflictos

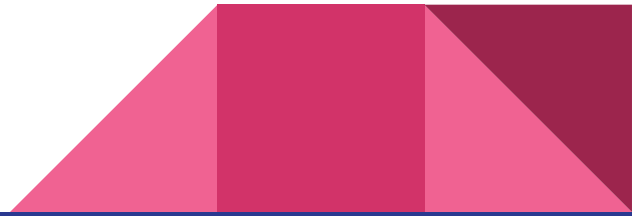
- Pero, ¿qué pasaría si, cuando quiero unir la rama que creé, la otra rama ha cambiado con respecto a cuando yo creé la mía? ---> Conflicto.




master
C0
C1
C2
C4
C5
C3
C5
iss53

Conflictos - 2

- Hay que resolver los conflictos antes de poder hacer cerrar con un commit.
- Cada archivo en conflicto tiene que ser editado. Git automáticamente marca que partes del código han cambiado.
- Algunas GUIs ofrecen editores integrados para resolverlos.
- En Linux, puedes utilizar por ejemplo “meld”. Para ejecutarlo hay que instalarlo y luego escribir: “git mergetool”.



Comandos básicos - 2

- git checkout <opciones> <rama>: cambias a la rama que desees.
  - Poniendo *-b* creas una rama nueva a partir de aquella en la que te encuentras.
  - En vez del nombre de una rama, puedes poner el número de referencia de un commit.
- git merge <rama>: fusiona la rama actual con la rama indicada.
- git branch: lista toda las ramas locales.
  - -d <rama>: destruye la rama escogida.
  - -a : muestra todas las ramas (locales y remotas).
- git remote: muestra la lista de repositorios remotos.
  - -v: muestra información extra sobre cada uno.
  - add <name> <url>: añade uno nuevo.
  - set-url <name> <url>: cambia la url del remote.
  - remove <name>: elimina el remote.

Ejercicios - 2

- Cambiar a la rama contador divisores.
- Añadir un nuevo método que cuente la cantidad de divisores que tiene un número.
- Mergear con la rama master y subir al servidor.