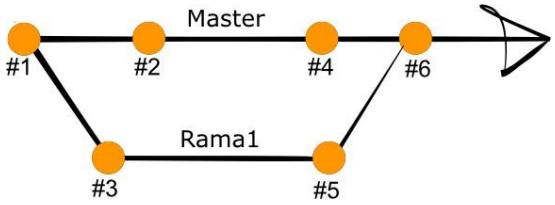


Trabajando con ramas en Git

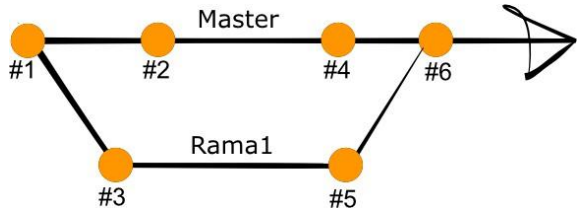
UD2: Desarrollo de aplicaciones basado en componentes con Angular

Objetivos de aprendizaje

- Aprender a usar los comandos de ramas
- Comprender las diferencias entre ramas locales y remotas
- Comenzar a usar ramas para el trabajo individual

Experimentando con ramas (*branches*)

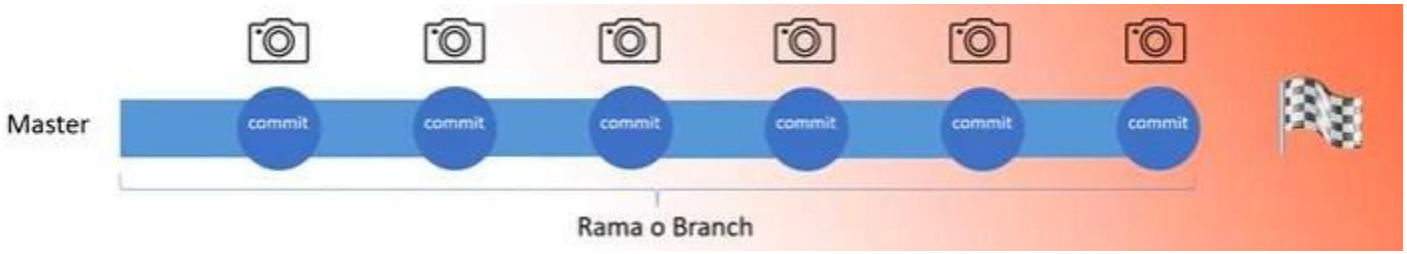
- Una rama permite trabajar de forma efectiva en un working directory completamiente nuevo
- El resultado es que un único repositorio Git puede tener múltiples versiones del código base entre las cuales podemos intercambiarnos sin movernos de directorio
- Cuando cambiamos de rama Git cambia los contenidos del Working Directory



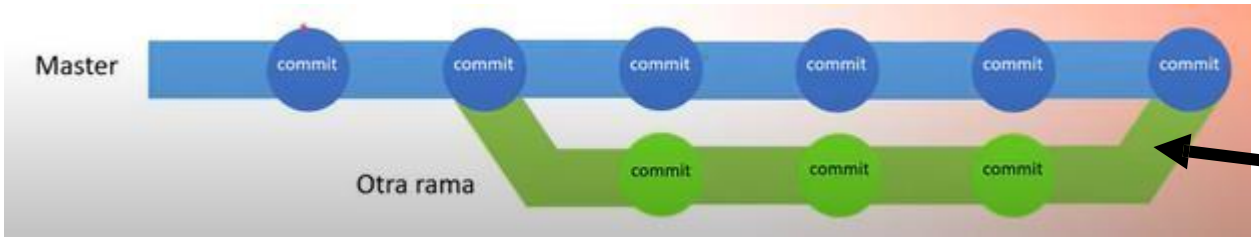
Ambas ramas son independientes y los cambios en una no afectan a la otra

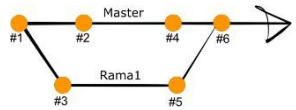
Concepto de rama (líneas de tiempo)

- Trabajo en una rama



- Trabajo en dos ramas (o más)





¿Para qué se usan las ramas?

- Hacer pruebas
- Otra persona trabaja en el mismo proyecto
- Creación de funcionalidades experimentales que queremos integrar sin que la rama principal se vea afectada

Una vez que el contenido de la rama es estable se puede fusionar para incorporar las modificaciones del proyecto en su rama principal

Conceptos básicos

- **`HEAD`**
  - Es el **commit** en el que está actualmente el repositorio.
  - Normalmente es el último commit de la rama actual
- **`origin`**
  - Nombre por defecto del repositorio remoto principal
- **`master (main`** en GitHub)
  - Nombre de la rama que se crea por defecto en la creación del repositorio
  - Normalmente es la rama principal.

Creación de una rama

`git branch nombre_rama rama_partida` crea una nueva rama llamada nombre_rama a través de una rama de partida

`git checkout nombre_rama` salta a una nueva rama

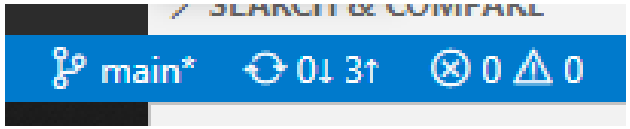
`git branch new_master master` → Creamos una rama llamada `new_master` a partir de la rama `master`

`git checkout nombre_rama` → Nos colocamos en la rama `new_master`

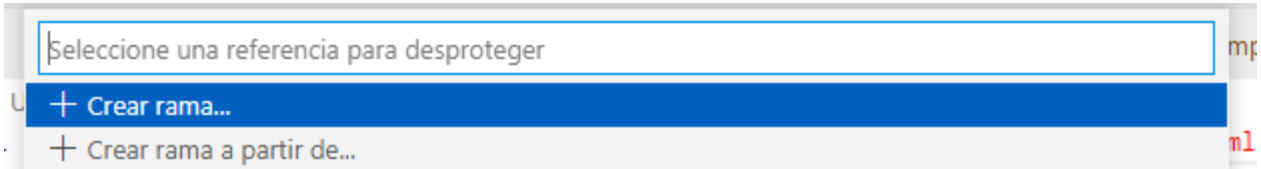
`git checkout -b nombre_rama` crea una nueva rama llamada nombre_rama y salta a ella

Creación de una rama en VSCode

- Desde la barra inferior tenemos capacidad de crear y cambiar ramas en VSCode



- Al pulsar en el nombre de la rama tendremos la opción de crear una nueva rama



Nombrado de ramas

1. Se consistente al nombrar las ramas
   - Usar mismos patrones de nombrado, mayus/minus, guionizado..
2. Usa el nombre de la acción que se realiza en la rama:
   - feature → Nuevas características
   - bug → Error de usuario
   - Experiment → Características experimentales
   - Hotfix → Error crítico

Consultando las ramas

`git branch` Permite consultar las ramas locales

`git branch -r` Permite consultar las ramas remotas

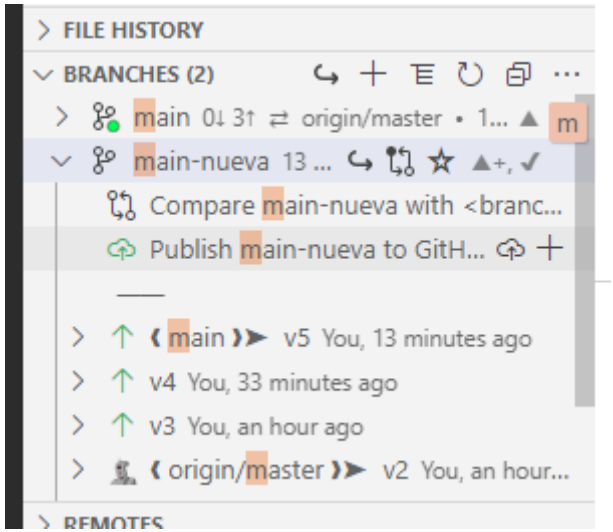
`git branch -a` Permite consultar las ramas locales y remotas

`git branch –v` Muestra el mensaje asociado al HEAD de la rama (último commit de la rama)

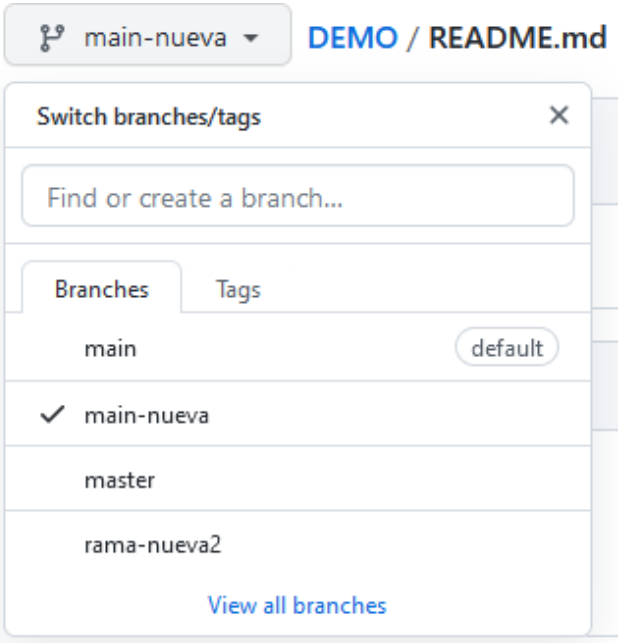
`git show-branch` Muestra las ramas y las versiones de las mismas

Consultando las ramas en VSCode

- Pestaña “Branches”
  - Cada rama me ofrece información sobre los commits realizados sobre ella



Ramas en GitHub



Fusionando ramas

```
git merge rama rama_principal
```

- Fusionamos colocados en la rama principal (`main`)

```
git checkout main
```

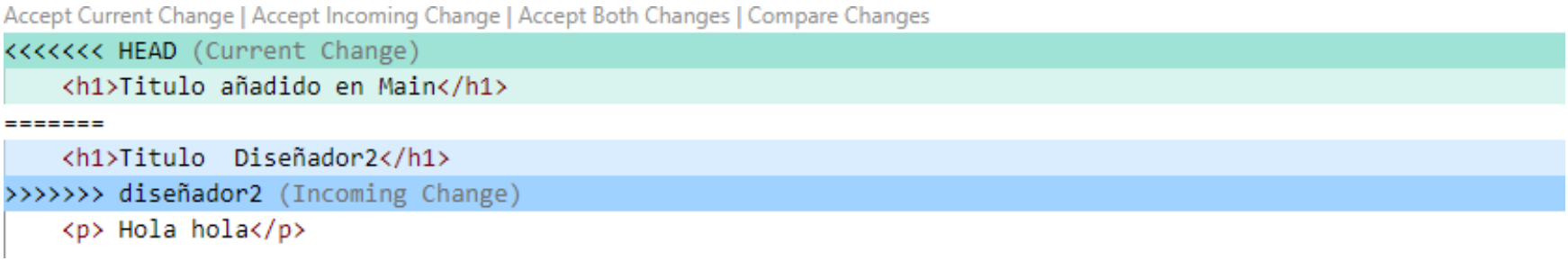
Nos colocamos en la rama `main`

```
git merge rama-secundaria
```

Fusionamos los cambios de la rama `rama-secundaria` en `main`

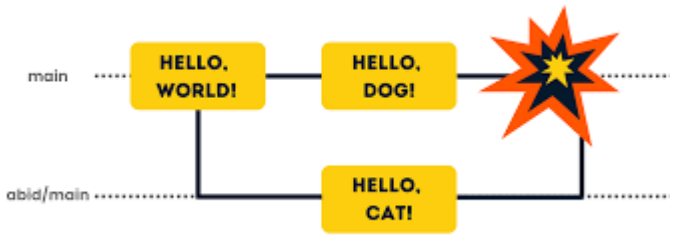
Solucionando conflictos de fusión

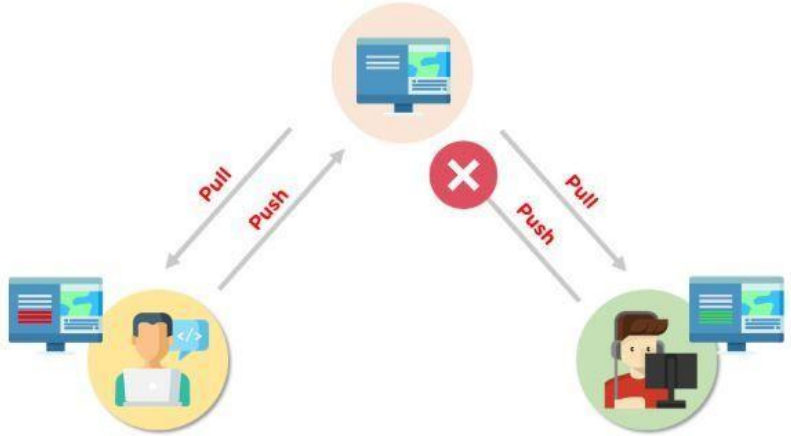
- Podría darse el caso de que cambie las mismas líneas del mismo archivo en dos ramas distintas
- En ese caso se nos da a elegir si queremos:
  - Dejar los cambios como estaban en main (current change)
  - Aceptar los cambios realizados en la rama (incoming change)
  - Aceptar ambos cambios (both changes)



Conflictos

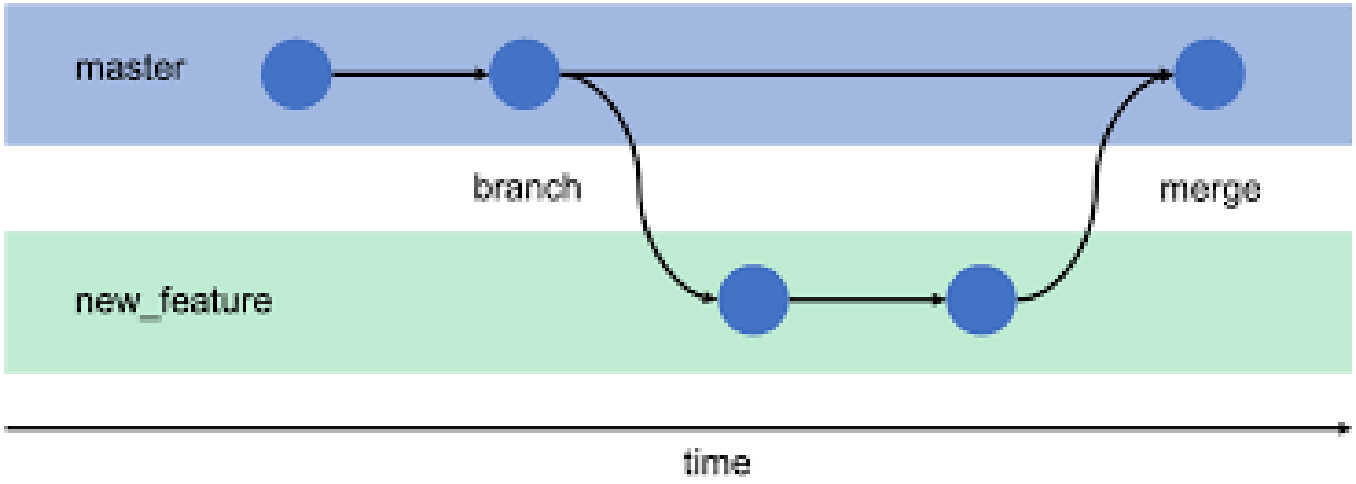
- Los conflictos pueden ocurrir:
  - Al fusionar dos ramas donde se han hecho commits
  - Cuando dos personas trabajan con el mismo repositorio remoto y hago pull de un código donde he hecho cambios





Conflictos

- Los conflictos NO ocurren:
  - Si fusiono una rama y en la rama original no he realizado commits



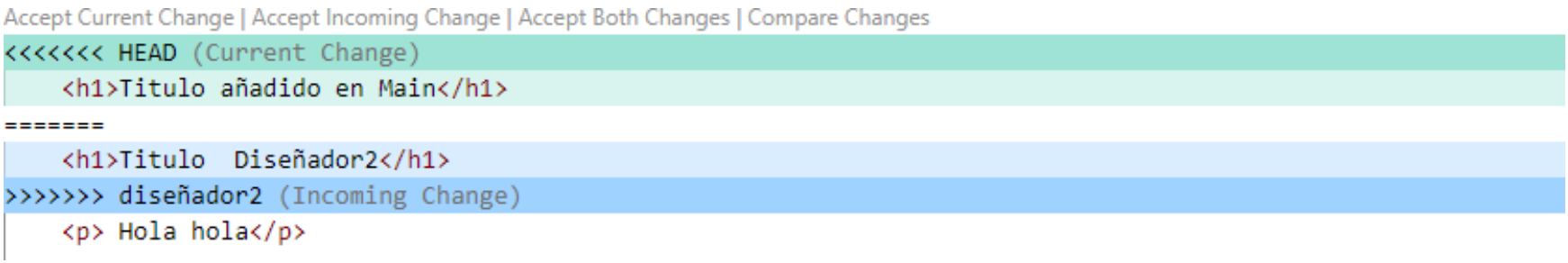
Solucionando conflictos de fusión (II)

- Lo que está entre `<<<<` HEAD y `=====` es el contenido que teníamos en la rama donde estamos haciendo el merge (Current Change)
- Lo que está entre `===` HEAD y `<<<` es el contenido que tenemos en la rama que queremos unir.

```
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< HEAD (Current Change)
    <h1>Titulo añadido en Main</h1>
=======
    <h1>Titulo  Diseñador2</h1>
>>>>>>> diseñador2 (Incoming Change)
    <p> Hola hola</p>
```

Solucionando conflictos de fusión

- **Podría darse el caso de que cambie las mismas líneas del mismo archivo en dos ramas distintas**
- En ese caso se nos da a elegir si queremos:
  - Dejar los cambios como estaban en main (current change)
  - Aceptar los cambios realizados en la rama (incoming change)
  - Aceptar ambos cambios (both changes)



Actualizando información remota

```
git fetch
```

- Descarga toda la información del repositorio remoto en el repositorio local
- Útil, por ejemplo, para hacer en casa cuando he estado trabajando en el instituto

```
git fetch --prune --verbose
```

Fuerzo a que se actualice la información del repositorio remoto

Eliminando ramas

```
git branch -d rama-borrar
```

```
git branch -d nombre-rama
```

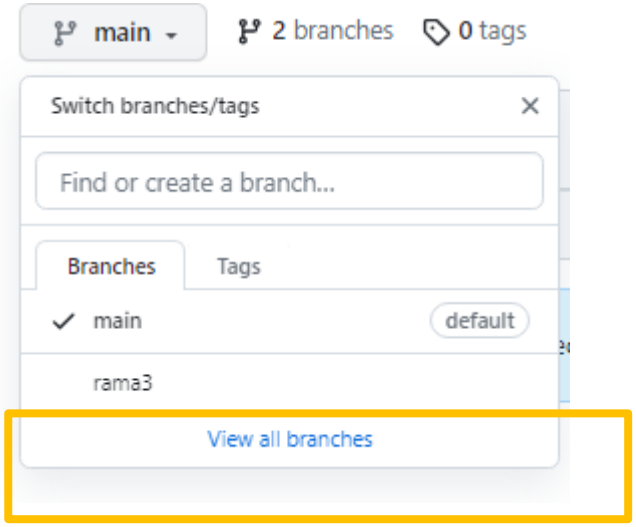
→ Elimina la rama **nombre-rama**

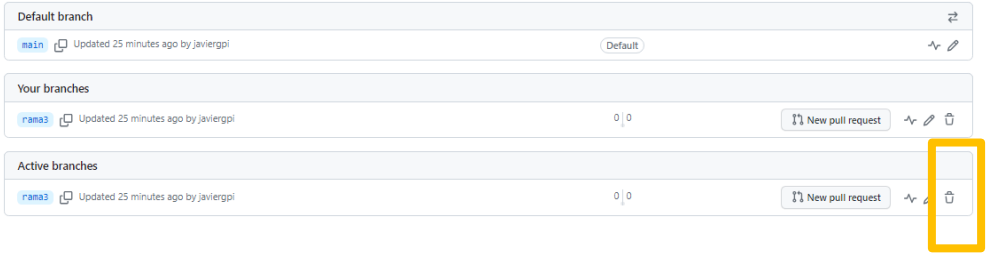
```
git branch -d -r rama-borrar
```
Borra las referencias a repositorios remotos

<u>El borrado de ramas debería hacerse una vez que se ha hecho el merge</u>

Eliminando ramas en GitHub

- Aunque borremos las ramas locales no se borran las ramas de GitHub
- Debemos hacerlo manualmente

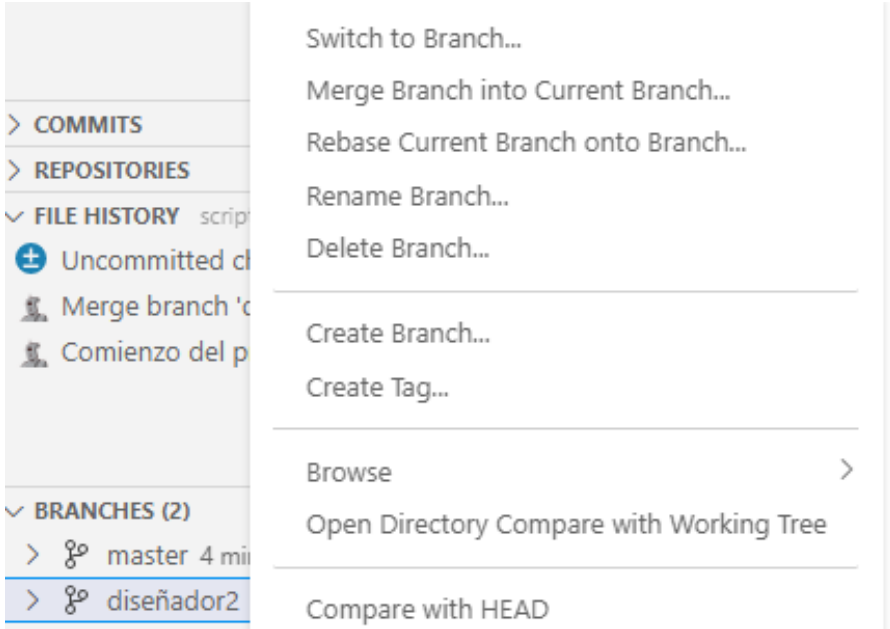





```
git push origin -d nombre-rama
```

Eliminando ramas en VSCode

- Pestaña Branches → Merge Branch into Current Branch…



Git y GitHub CheatSheet

Cheatsheet Git básico + Github

Completa - Autor: Sergi García Barea

Crear y compartir repositorios privados en Github

- Enlace visibilidad repositorio: https://docs.github.com/es/github/administering-a-repository/managing-repository-settings/setting-repository-visibility
- Enlace compartir repositorio: https://docs.github.com/es/github/setting-up-and-managing-your-github-user-account/managing-access-to-your-personal-repositories/inviting-collaborators-to-a-personal-repository

Token acceso para trabajar con repositorios

- Crear token de acceso Github (**Obligatorio: Actúa de contraseña para trabajar con repositorios**): https://docs.github.com/es/github/authenticating-to-github/keeping-your-account-and-data-secure/creating-a-personal-access-token

Establecer usuario y email Github (solo primera vez)

```
git config --global user.name "NOMBREUSUARIO GITHUB"
git config user.email "EMAILCUENTAGITHUB@SERVIDOR.COM"
```

- La primera vez que usamos git con Github, deberemos configurar estos parámetros. Esto indica las credenciales al conectarnos a cuentas Github para manipular repositorios.

Usando "git clone" para clonar un repositorio

```
git clone (dirección del repositorio)
```

- Esta orden clona un repositorio de Github en tu máquina local.
  - Si el repositorio es privado te pedirá tu cuenta de usuario y una contraseña. Esa contraseña **NO ES LA CONTRASEÑA DE TU CUENTA**, sino el token personal

GitHub

HOJA DE REFERENCIA PARA GITHUB GIT

Git es el sistema de control de versiones distribuido de fuente abierta que facilita las actividades de GitHub en su computadora portátil o de escritorio. Esta hoja de referencia rápida resume las instrucciones de las líneas de comando de Git más comúnmente usadas.

INSTALAR GIT

GitHub le ofrece a los clientes de computadoras de escritorio que incluye una interfaz gráfica de usuario para las acciones de repositorio más comunes y una edición de línea de comando de actualización automática de Git para escenarios avanzados.

GitHub para Windows
https://windows.github.com

GitHub para Mac
https://mac.github.com

EFECTUAR CAMBIOS

Revisa las ediciones y elabora una transacción de commit

```
$ git status
```

Enumera todos los archivos nuevos o modificados que se deben confirmar

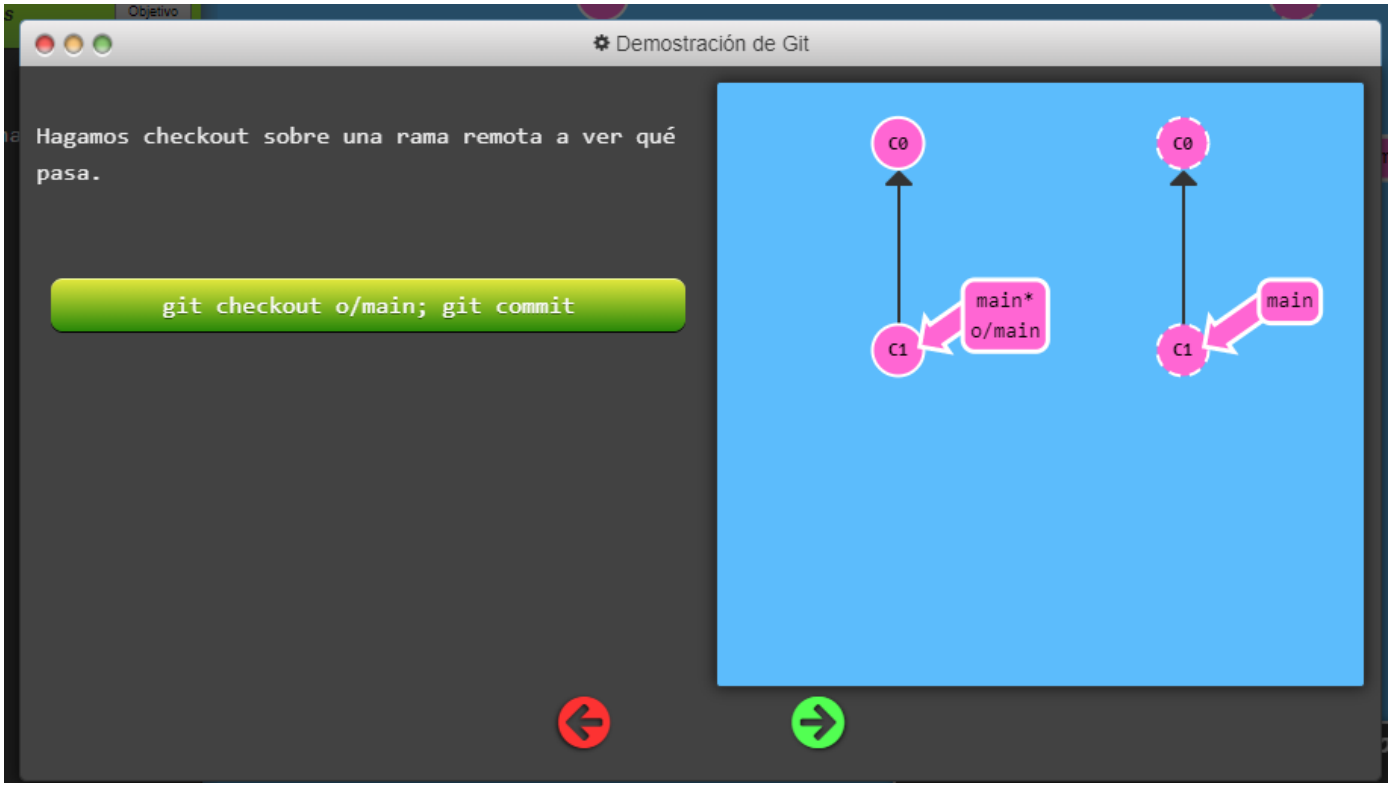
```
$ git diff
```

Muestra las diferencias de archivos que no se han enviado aún al área de espera

[CheatSheet #1]

[CheatSheet #2]

Aprendiendo Git



Otros enlaces

- Git- La guía sencilla
- Aprende Git en una hora
- Curso de Github con VSCode [Video]
- Documentación completa de Git