**ANÁLISIS DEL USO DE REPOSITORIO GIT**

Primeros pasos en GitHub

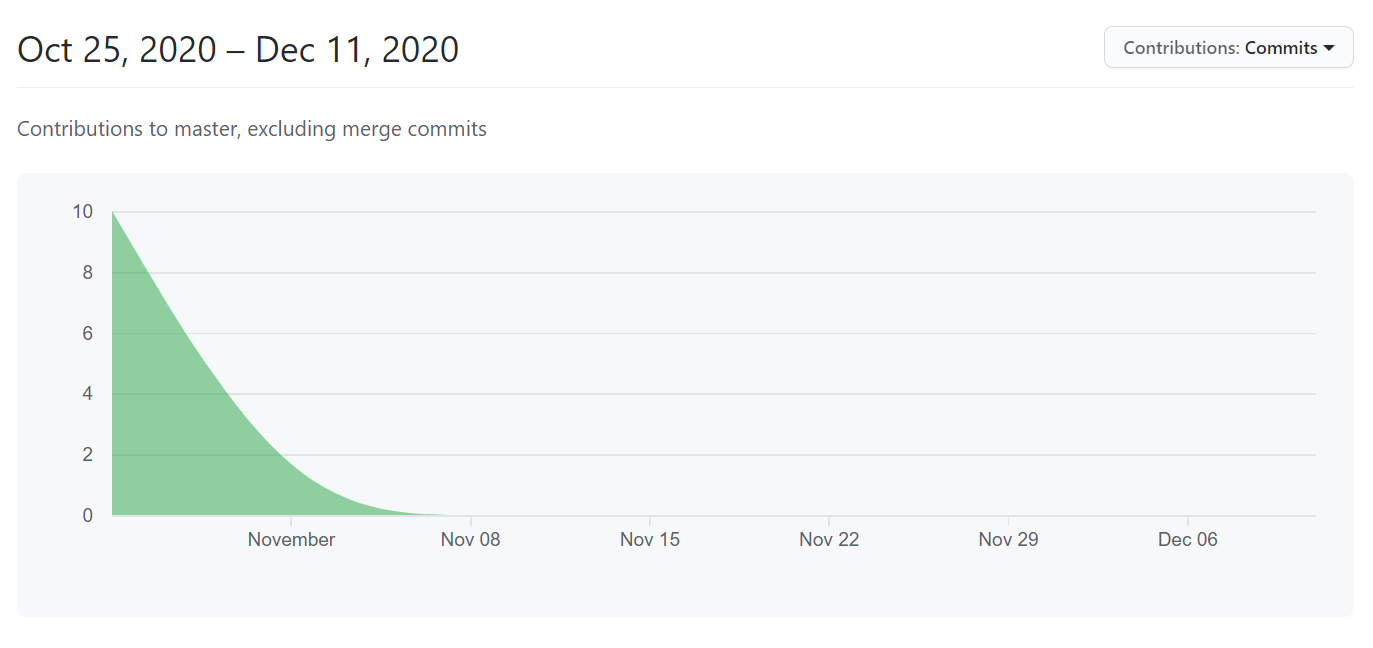
La función de los comandos básico es la siguiente:

* **git status:** Muestra los cambios que ha habido en el repositorio desde la última versión realizada.
* **git add:** Permite añadir archivos que están fuera de seguimiento o que han sido modificados.
* **git commit -m:** Confirmamos las modificaciones, crea una nueva versión con los archivos que ahora están en seguimiento: Siempre se indica con un mensaje descriptivo “Descripción del commit”
* **git push**: Añade las modificaciones de nuestro repositorio local con uno remoto, con el fin de tener los dos repositorios sincronizados. Tenemos que indicar la dirección de nuestro repositorio remoto con su url.
* **git pull:** Permite actualizar nuestro repositorio local, con los cambios mas recientes que tiene uno remoto. Se tiene que indicar la url del repositorio que debemos descargar. Cuando hacemos ***git pull*** se descargan esos nuevos ficheros y se intentan unir con los ficheros de nuestro repositorio local. Desde el terminal de Git se abre el editor de texto de linux .Para salir de él pulsamos las teclas (:wq!)
* **git clone:**  Usaré este comando cuando en mi ordenador no tengo una copia de mi repositorio. Uso la url de mi repositorio remoto para traerme una copia en local, y poder seguir trabajando. Esto genera un repositorio local idéntico al remoto.
* **git rm:** Borrar archivos ya confirmados. Si tengo actualmente ficheros en mis versiones que no quiero que sigan en mi proyecto, debo borrarlos de **git** y posteriormente volver a confirmar. Debo tener en cuenta que dichos archivos también desaparecerán de mi repositorio local.

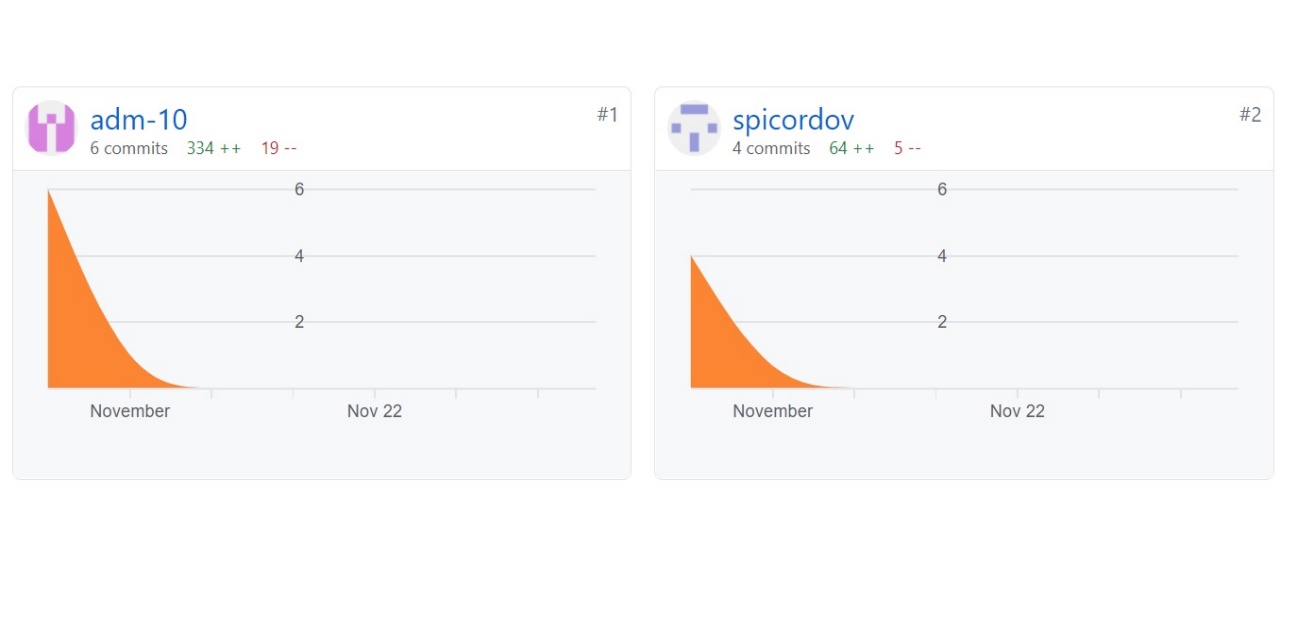
Trabajando con colaboradores

En nuestro repositorio en línea creado vamos a Settings, ahí seleccionamos la opción Manage acces, nos pide contraseña, una vez dentro seleccionamos la opción Invite a collaborator, para ello introducimos su nombre de usuario, y le llagara la invitación al correo con el que está vinculada la cuenta.

Esto nos permite trabajar varias personas en un mismo proyecto.



En este grafico vemos el periodo que se han hecho colaboraciones al proyecto.



Aquí vemos los colaboradores, además de mostrarnos el número de commits y modificaciones.

Solución de conflictos que podemos encontrarnos.

Uno de los conflictos más comunes que podemos encontrarnos es a la hora de modificar un archivo local e intentar sincronizarlo los cambios realizados con el repositorio remoto, si el local no ha sido actualizado antes con un git pulll, para sincronizar con el remoto nos indicara un error, y nos recomienda hacer un **git pull.**

Otro conflicto es cuando modificamos una línea de nuestro fichero y la subimos al repositorio,

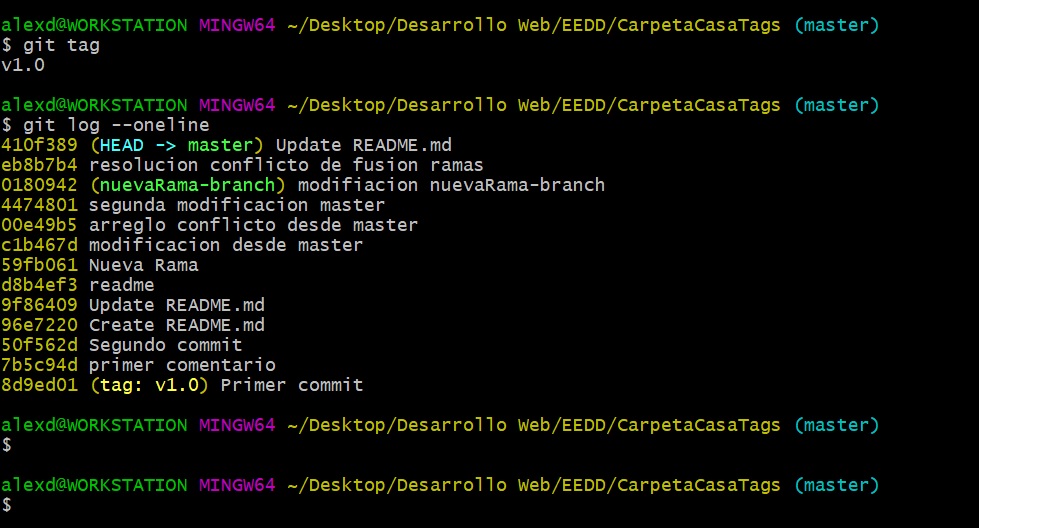
Intentamos modificar la misma linea, y al intentar subirlo nos dice que las pistas se están pisando, la solución es muy simple. El propio terminal git nos dice que resolvamos el conflicto modificando el fichero, por lo tanto, entramos al fichero y este nos muestra con unas pestañas las líneas que se están pisando.

Creación de Tags

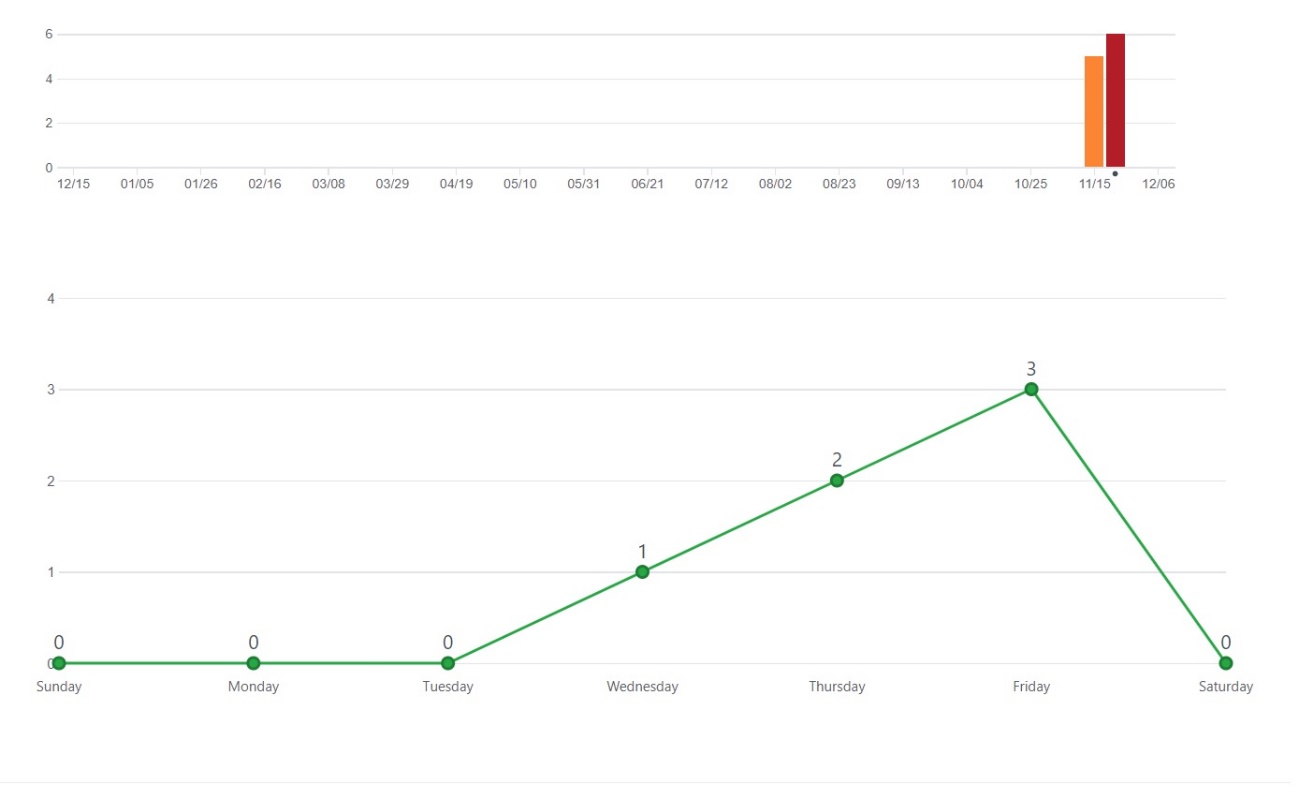
Un **tag** es un nombre que puedes usar para marcar un punto específico en la historia de un repositorio, este punto siempre se mantiene inalterable.

Los comandos usados son:

* **git tag -a [versión] [nombre commit] -m:** Nos permite realizar tags de commit antiguos, esto nos ayudara a tener localizada la versión en el momento que deseamos. También hay que hcer una breve descripción del tag: **“descripción tag”**
* **git tag:** Nos permite visualizar los tags que contiene el archivo.
* **git push [origin] [versión tag]:** Este comando nos es útil para subir nuestro tag a nuestro repositorio remoto, es importante indicarle que versión del tag queremos subir.
* **git pull -tags [origni]:** Nos permite traer los tags que se encuentran en nuestro repositorio remoto a nuestro repositorio local.
* **git log –oneline:**  Nos permite consultar el log de nuestro repositorio local.

****

En la imagen se muestra un ejemplo de **git tag** y **git log –online.**



Nuestro repositorio GIitHub también nos muestra un grafico en que se puede ver la semana y el número de commits que se han realizado, y el día en que se realizaron esos commits.

Creación de Ramas

La creación de ramas es una bifurcación del estado del código que crea un nuevo camino para la evolución del mismo.

Los comandos más usados son los siguientes:

* **git branch [nueva\_rama]:** nos permite crear una nueva rama.
* **git branch :** Visualizamos las ramas de las que se compone nuestro proyecto
* **git checkout [nombre nueva\_rama]:** Me posiciono en la nueva rama creada, y poder hacer las modificaciones que yo quiera. Para volver a mi rama master cambiare el nombre de nueva rama por master.
* **git push [origin] [nombre\_rama master/nueva\_rama]:** Puedo subir la nueva rama creada al repositorio, o mi rama master.

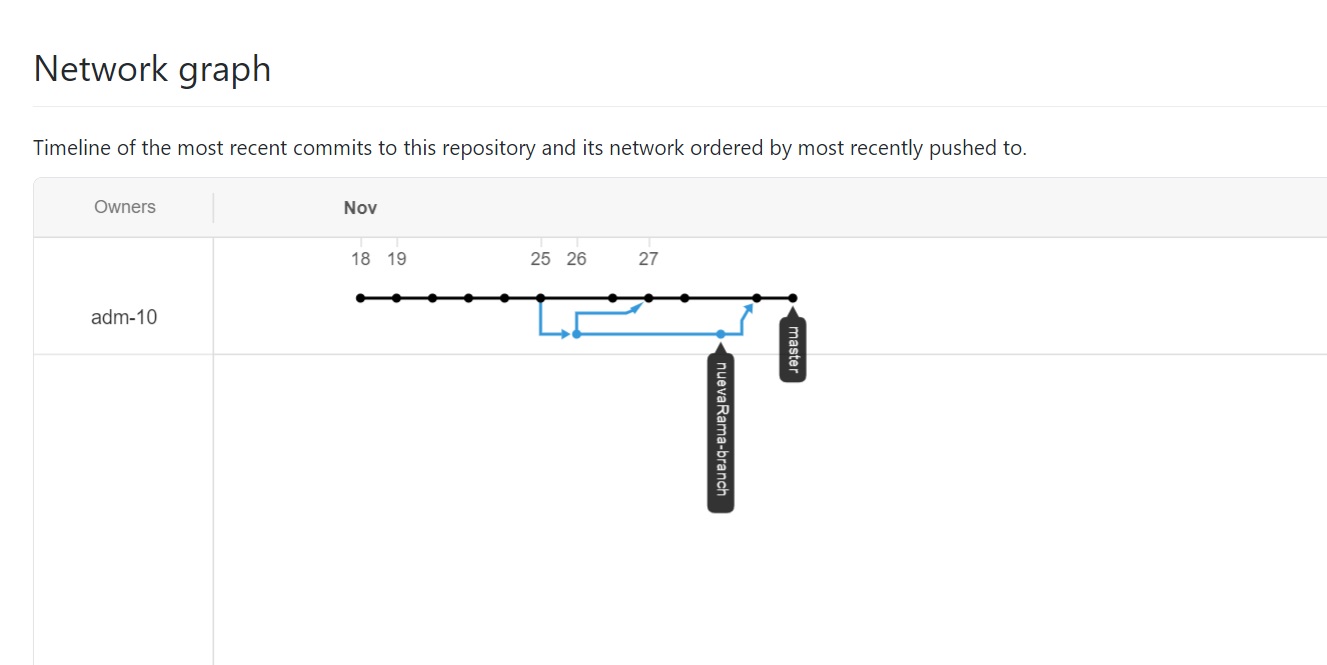
Clonar Ramas del repositorio

Los pasos para clonar una rama del repositorio son:

* **git clone:** clono mi repositorio online en local.
* **git branch:** solo aparece la rama master.
* **git branch – a :** Se hace dentro de la carpeta que se ha clonado, y ahora si me aparecen las ramas que tiene ese repositorio, en caso que tenga ramas.
* **git checkout [nombre de rama]:** Trae la rama del repositorio remoto al local, ya tengo la rama de mi repositorio remoto en el local, me cambio a la rama creada y puedo hacer las modificaciones que necesite.
* **git merge [nombre nueva\_rama]:** Se usa para fusionar ambos proyectos, es importante ubicarse en la rama master para realizar la fusión. Al fusionar los proyectos puede salir el error de pisando la misma línea, la solución está en el apartado de conflictos.

Graficas de nuestro repositorio

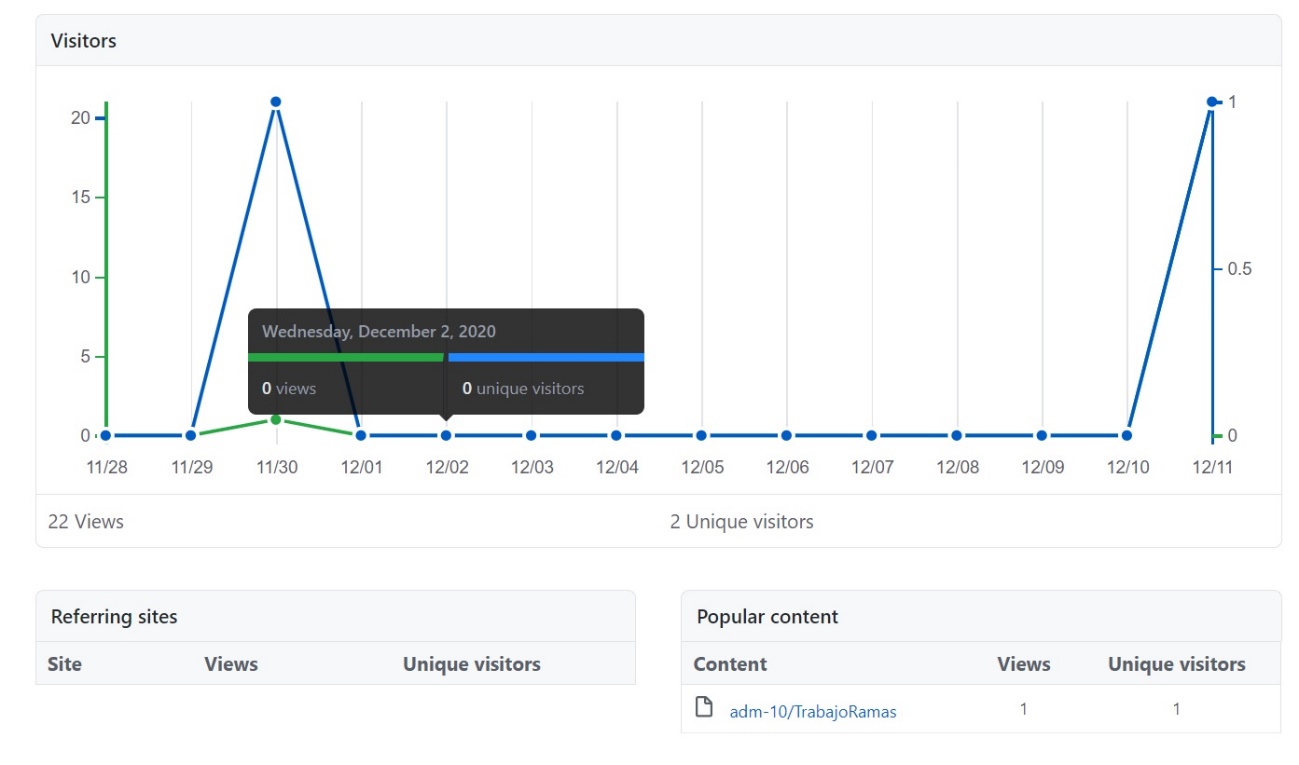
Desde nuestro repositorio remoto, podemos ver la gráfica de nuestras ramas.



En el apartado de trafico podemos ver las veces que se ha clonado nuestro proyecto.



Y los visitantes que ha tenido.



Repositorio usado:

https://github.com/adm-10