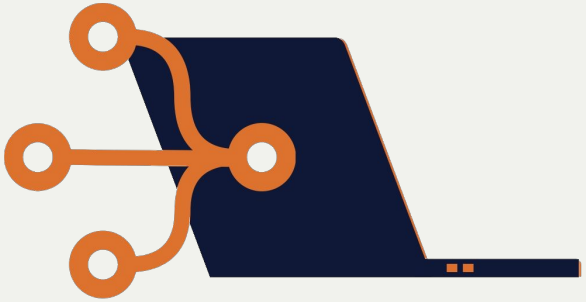


$ sudo

Git + GitHub 101

15 / Noviembre / 2024

01

¿Que es Git?

¿Cómo se configura? ¿Cómo se usa?

¿Qué es Git?

Git es un software de control de versiones. Fue creado por **Linus Torvalds** para llevar el control de versiones del kernel de Linux.

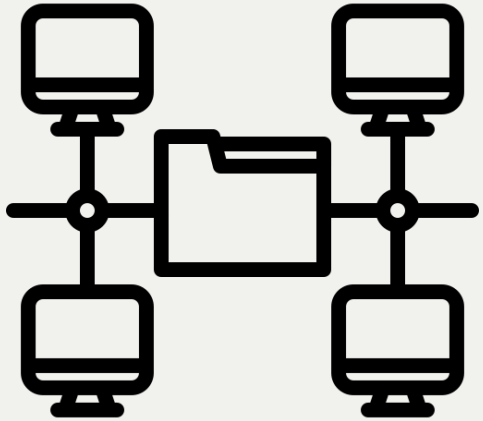
Git soporta desarrollo distribuido del código sobre el cual se haga el control, por lo que, sumando a su licencia open source, lo hace ser el control de versiones más usados de todos.

Segun Linus, Git significa “***General Information Tracker***”

¿Qué es un Repo?

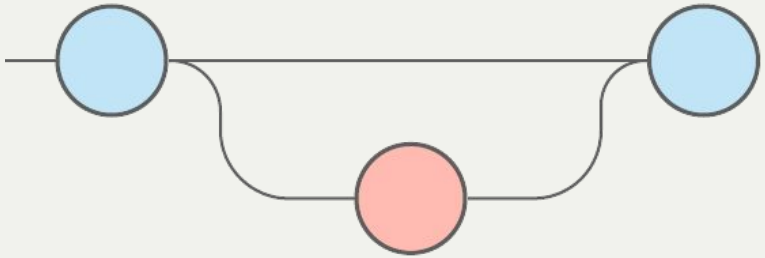
Un ***repo***, corto para ***repositorio***, es el directorio en el que se irán alojando y gestionando los cambios que hagas en tu proyecto.

La configuración e historia del repo se guarda en una carpeta llamada ***'.git'***, la cual permanece oculta para evitar modificaciones que ocurran por error.



¿Qué es un Commit?

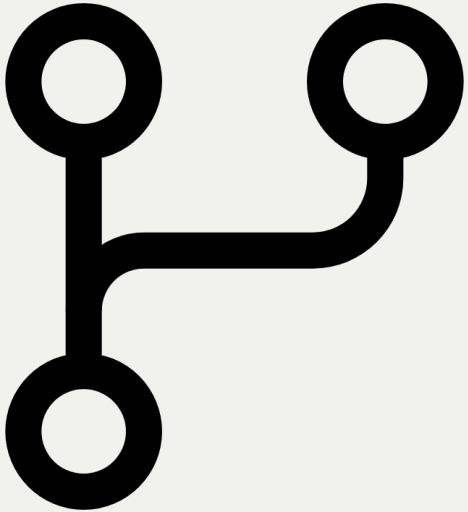
Un ***commit*** es una **instantánea** del directorio de trabajo de tu proyecto en un momento determinado, tomada por un autor específico. La idea de usar Git en un proyecto es poder hacer varias instantáneas como se necesiten para saber cómo han sido las modificaciones al mismo.



¿Qué es un Rama?

La estructura en la que Git trabaja es como la de un árbol, en la que ciertos cambios se pueden probar ‘en un espacio separado’ para luego reintegrarse al espacio principal.

Este espacio ‘separado’, en el contexto de un repo, le llamamos **rama o branch**. La rama “raíz” le llamamos “***master*” *o* “*main*”.**



¿Qué es un README?

Un ***README*** es archivo de texto, generalmente en formato *Markdown, .md*, que proporciona una descripción del proyecto, sus propósitos y cómo usarlo.

Suele incluir **instrucciones** de instalación, **uso**, contribución y cualquier otra información relevante

Es compatible con HTML en cuanto a títulos, tablas, espacios, etc, pero no al 100%.

Configurando Git

Para asegurarnos que siempre que hagas un commit este tu información.

01 **git config --global user.name "Tu Nombre"**

Aquí configuramos tu nombre de manera global en todos los commits.

02 **git config --global user.email "yo@sudo.com"**

Y aquí configuramos tu correo de manera global en todos los commits.

Iniciando un Repo

Si no no sería control de versiones con Git.

01 git init

Inicializa el repo en el directorio actual.

02 git clone <ruta/del/directorio>

Incluso podemos iniciar uno copiando el contenido de un directorio.

Añadiendo Cambios

¿Si no cual es el punto del control de versiones?

01 **git add [file] / git add .**

Se puede dar la instrucción de añadir archivo por archivo o todos a la vez.

02 **git commit / git commit -m “Hice Cambios!”**

Se usa para confirmar los cambios que estamos haciendo con un mensaje des

03 **git reset [file]**

Para esos casos en donde no había que añadir un archivo después de todo.

Los Mensajes de un Commit

Procura que el mensaje que le das a tu commit sea *descriptivo* de los cambios que hiciste. Esto ayuda a tu “yo futuro” a saber que hiciste y a los demás a entender que cambiaste.

X **git commit -m “Hice Cambios!”**

¿Pero qué cambios se hicieron?

✓ **git commit -m “Refactorización de BúsquedaX”**

Es más legible y fácil de referenciar a futuro.

Revisando Cambios

aka, el historial de Git

01 git status

Para conocer el estado actual, en rama estás, si hay cambios por subir…

02 git log / git log --stat

Nos deja ver el ID de todos los commits (y la lista de archivos modificados).

03 git diff [rama-b]...[rama-c]

Para revisar los cambios que existen entre ramas.

Fragmentos de Cambios

Para guardar esos cambios a medias.

01 **git stash -p -m “WIP de BusquedaZ”**

Guarda un cambio parcial con un mensaje para identificarlo posteriormente.

02 **git stash list**

Para ver todos los cambios parciales almacenados.

03 **git stash apply stash@{x}**

Aplica un cambio parcial guardado a los archivos correspondientes.

Deshaciendo Commits

Tal vez no tenías que cambiar eso después de todo…

04 git reset [commit]

Deshace todos los commits después de [commit], preservando cambios locales

05 git reset --hard [commit]

Desecha todo el historial y regresa al commit especificado

Trabajando con Ramas

Todo tiene un lugar, y un lugar para todo.

01 git branch

Para ver todas las ramas del repo.

02 git branch <rama-nueva>

Para crear una nueva rama dado un nombre dado.

03 git checkout (-d) <nombre-de-la-rama> (otra-rama)

Para ir cambiando de ramas tras hacer un commit.

Trabajando con Ramas

Todo tiene un lugar, y un lugar para todo.

04 **git merge <rama_a_fusionar>**

Los cambios de una rama se fusionan con los de otra.

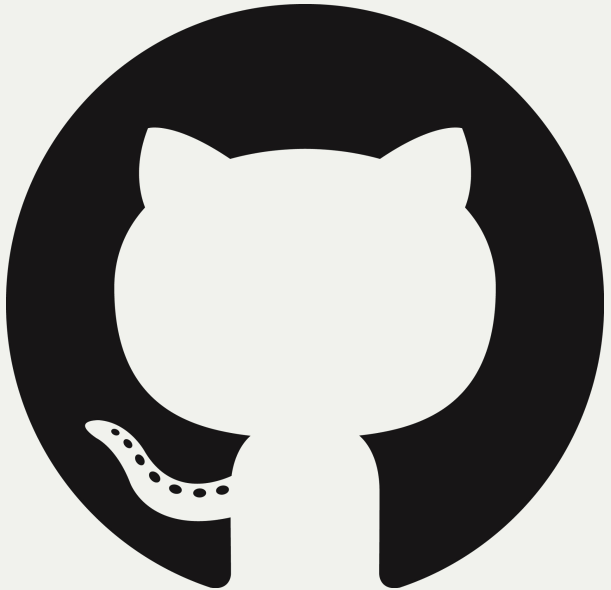
05 **git merge <rama_a_fusionar> -m <mensaje>**

Añade un mensaje a la fusión.

02

Entendiendo GitHub

¿Qué hace especial a la plataforma?
¿Qué ventajas tiene?



¿Qué es GitHub?

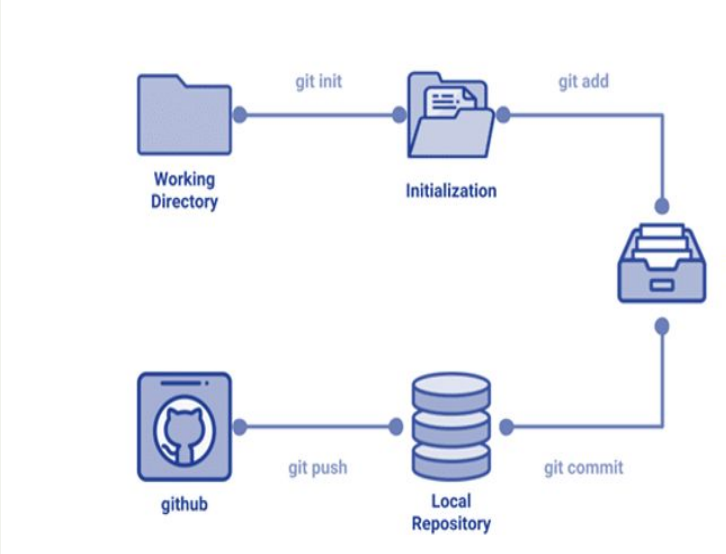
GitHub es una plataforma de desarrollo colaborativo para alojar proyectos que utilizan Git (en otras palabras, es un *remote)*.

En si, es un VCS, o *Version Control Service*. Aunque en un inicio estaba pensado sólo como un servicio de alojamiento, la plataforma se ha expandido a otros servicios como *Codespaces*, *Gists*, *Wikis*, etc.

Fun Fact: GitHub originalmente fue escrito en Ruby!

¿Qué es un Remote?

Un ***remote*** es, a resumidas cuentas, un “repo remoto”. Más que un repositorio alojado en otro lado directamente, es una referencia a cuál es ese *otro lado* hacia y desde el cual se pueden subir y/o bajar cambios a nuestro repo local.



Algunas Ventajas de GitHub

GitHub Repo

No solo sirve como un *remote* de tu repo de Git local, si no que es un repositorio con funciones extras como ***tags, markdown a nivel raiz, forks, stars...***

Pull Requests

Es una "*solicitud de cambios*". A lo que esto se refiere es que tu puedes hacer tus modificaciones en tu rama y después pedir una revisión para hacer un *merge* con otra rama.

Algunas Ventajas de GitHub

GitHub Profile

Tu perfil de GitHub puede tener su propio repo para presentarte ante al mundo!. Ya llegaremos a eso pronto…

Tags e Issues

GitHub permite crear etiquetas para los errores/bugs/problemas de código desarrollando colaborativamente. Útil para saber qué cosa incluye cada commit o cada PR.

Algunas Ventajas de GitHub

Markdown Extendido

GitHub tiene sus propios ajustes al hacer READMEs en Markdown. Estos van desde poner emojis en el archivo hasta poner fórmulas matemáticas en formato LaTeX.

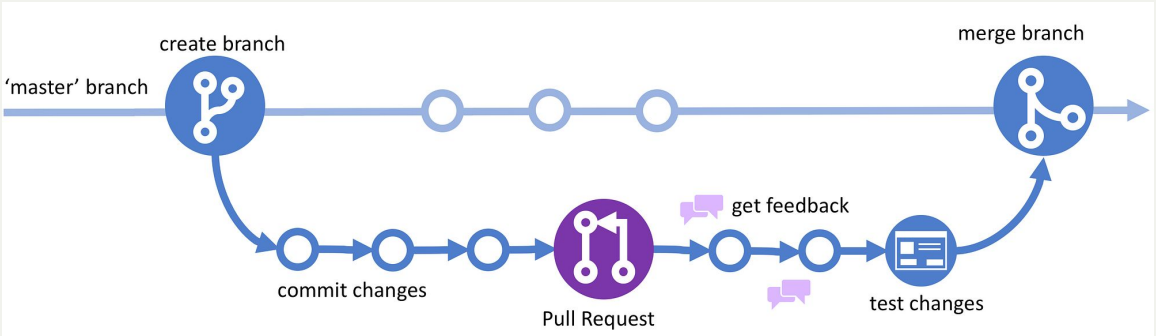
github.dev + Codespaces

github.dev es un IDE para edición express de archivos. Si se necesita poder de cómputo, es cuando uno necesita usar *Codespaces*.

GitHub Flow

Es la forma de trabajo recomendada por GitHub al trabajar con un repo. Se basa en 6 pasos.

1. Crea una rama para tus cambios.
2. Crea tus commits en la rama nueva
3. Haz una PR con tus cambios.
4. Revisa y Comenta con los demás tus cambios
5. Haz merge de tu rama
6. Borra la rama



Trabajando con Remotes

El equivalente a hacer “upload” y “download”

01 git pull

Descarga todos los cambios del remote y los aplica al repo local.

02 git push (-u origin <branch>)

Sube todos los cambios locales al remote. Esto después de hacer commit.

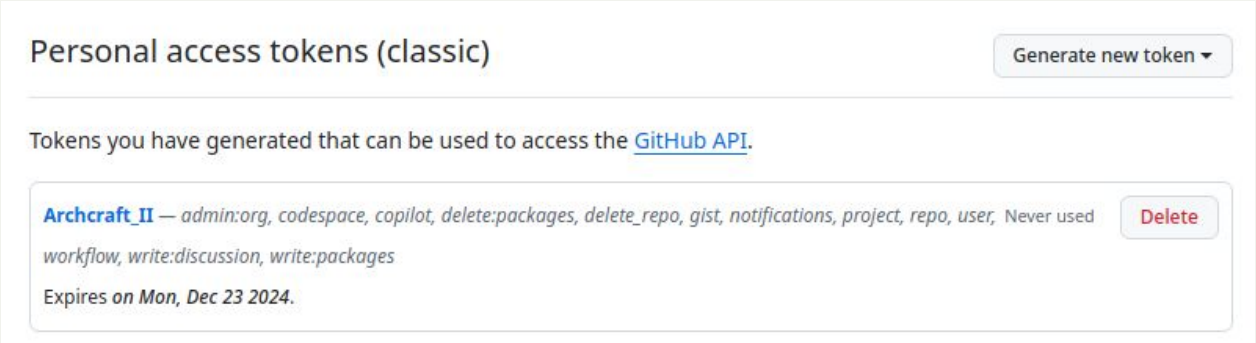
03 git fetch

Baja los cambios del remote pero no los aplica automáticamente.

¿Qué es un Token (En GitHub)?

Es como la contraseña para subir commits a tus repos. Sin el token no vas a poder subir tus cambios, hacer PRs, etc, desde la terminal.

También es una forma de “seguridad”
¿Cómo sabemos que si eres tú quien está haciendo el commit?



GitHub Student Pack

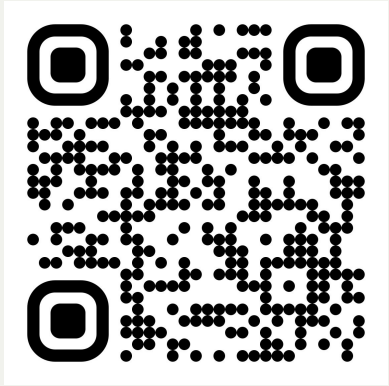
El Student Pack es el conjunto de *experiencias* gratuitas que GitHub ofrece a estudiantes de escuelas registradas para mejorar su aprendizaje y desarrollo de software (entre otras cosas).

Al ser alumnx de Ciencias, tú puedes acceder a este paquete creando una cuenta con tu correo institucional y validando tu credencial

¿Qué incluye el Student Pack?

Entre las cosas más notables que incluye el paquete están

- Copilot - IDE's de JetBrains - Notion + - Stream Yard
- Certificaciones Gratis!



Registrate Aqui!

03

Seguridad en GitHub

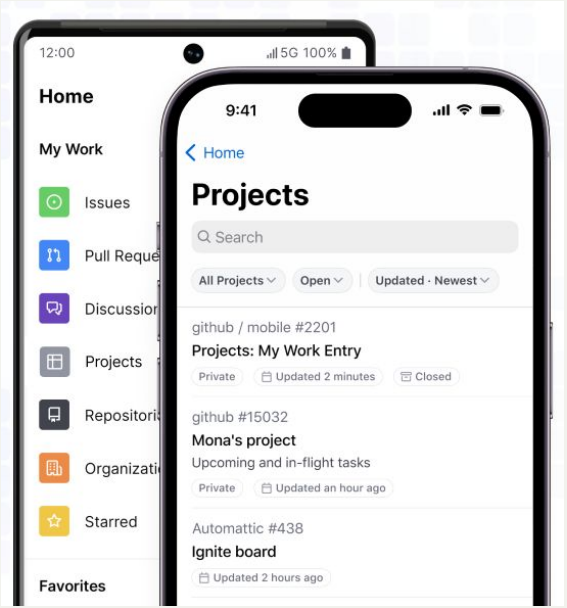
Más allá de tu contraseña, tanto para iniciar sesión como hacer un *push*.

¿Qué es GitHub Mobile?

GitHub Mobile es la aplicación movil de GitHub. No se puede hacer lo mismo que con la versión web, pero es una herramienta útil para poder revisar PRs, codigo, revisar READMEs, etc.

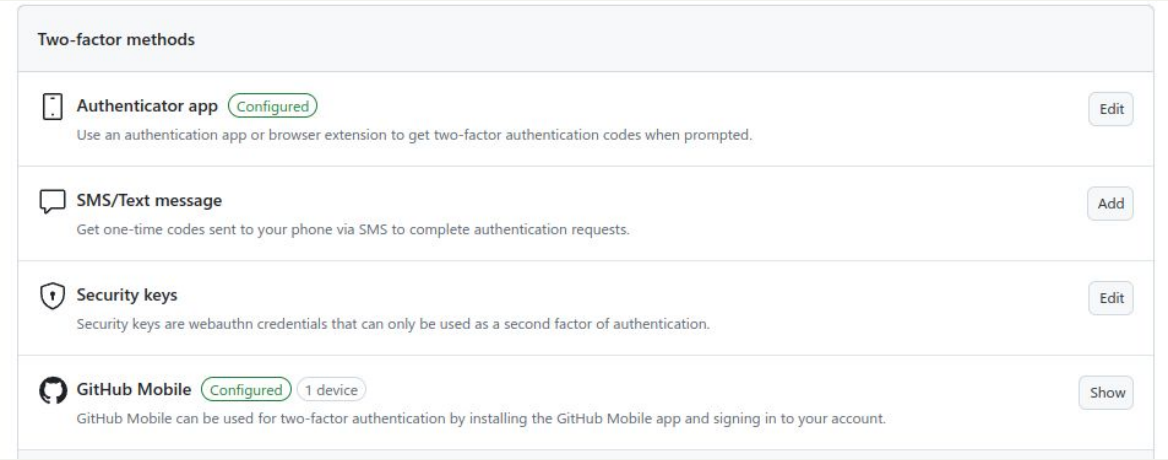
Android

IOS

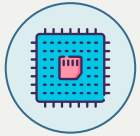


¡Pero tiene un uso principal y necesario!

La aplicación está también pensada como un 2FA (Segundo Factor de Autenticación). De está manera, siempre que necesites realizar algún ajuste de seguridad, como eliminar un repo, cambiar tu contraseña, etc, lo puedes usar para autenticarte!



Otras Formas de Seguridad



Otros 2FAs

Correos Extra, SMS, App de Autenticación Secundaria.

Commits SSH

Ayuda tener tus commits seguros con tu clave pública.

Llaves

Descarga y guarda tus llaves de recuperación. No las pierdas nunca!

04

Personalización 101

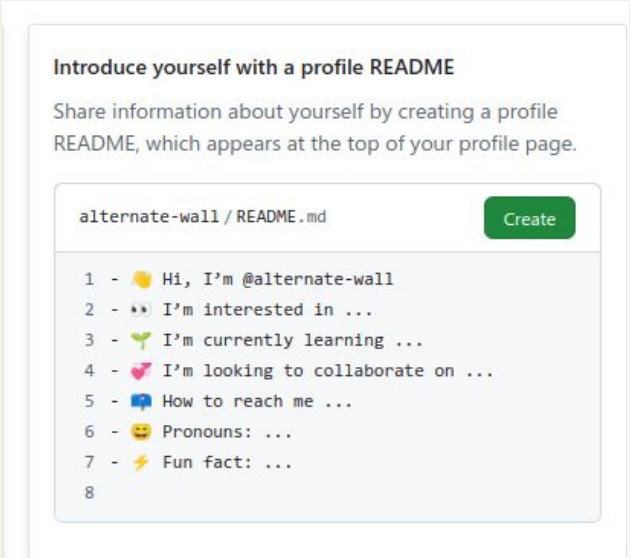
¡Haz destacar tu perfil frente a la comunidad!

El Readme de tu Perfil

GitHub tiene un repo especial con el cual puedes personalizar tu perfil de manera que siempre que uno entre a tu perfil pueda ver una descripción tuya. Esto mediante un README.

Para hacer esto, debes crear un repo con tu mismo nombre de usuario.

Por ejemplo, si tu perfil se llama “BatMaN”, el repo sería ‘*BatMaN/BatMaN*’



Personalizaciones Populares

Shields.io

Para hacer ‘medallas’ de los lenguajes (y otros) que sabes usar.

Readme Stats

Tu “puntuación” de Github vista como un resumen.

Arte ASCII

Como es un formato .md, fácilmente puedes pegar arte ASCII en el archivo.

Y un par de cosas más...

Pinear Repos

Puedes pinear hasta 6 repos en tu perfil para llevar la atención a ellos.

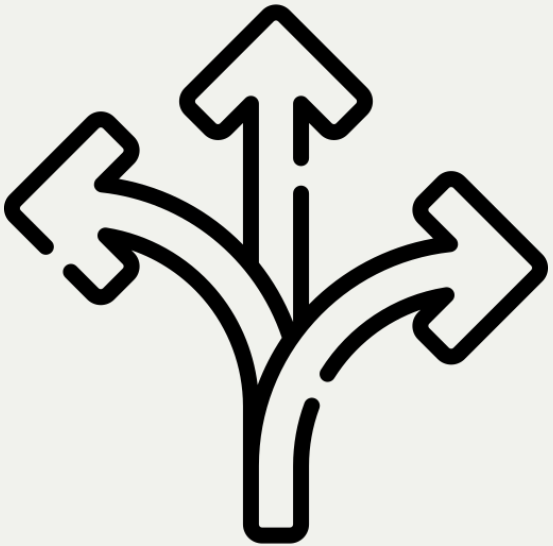
Muestra tus Logros y Organizaciones

GitHub llega a dar “logros” por ciertas acciones que realizas. Estos se pueden mostrar debajo de tu foto de perfil.

05

Alternativas A GitHub

¿Solo se puede ocupar GitHub como un *remote?*

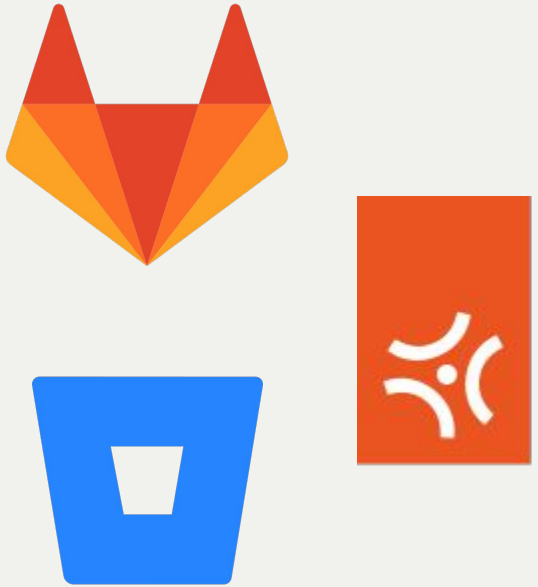


¿Solo existe GitHub?

GitHub no es el único servicio de *remotes* que existe (pero si es más popular).

Al ser Git un proyecto open source, crea la posibilidad de que existan alternativas, cada una con sus propias ventajas y desventajas.

Las más populares son GitLab, BitBucket, SourceForge, Launchpad y CodeCommit.

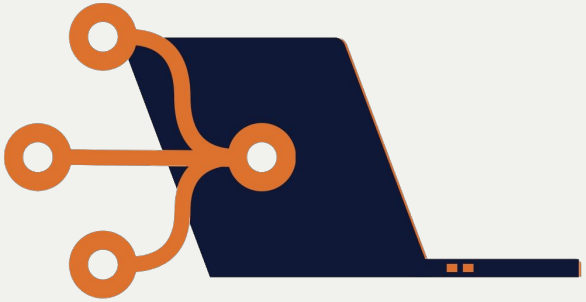


¡Unete a Sudo!

Nuestro LinkTree

¿Quieres ver que sigue? ¿Quieres ser parte de la comunidad?

No perdamos el contacto.
Escanea el QR para más información.



$ sudo

¡Gracias por Asistir!