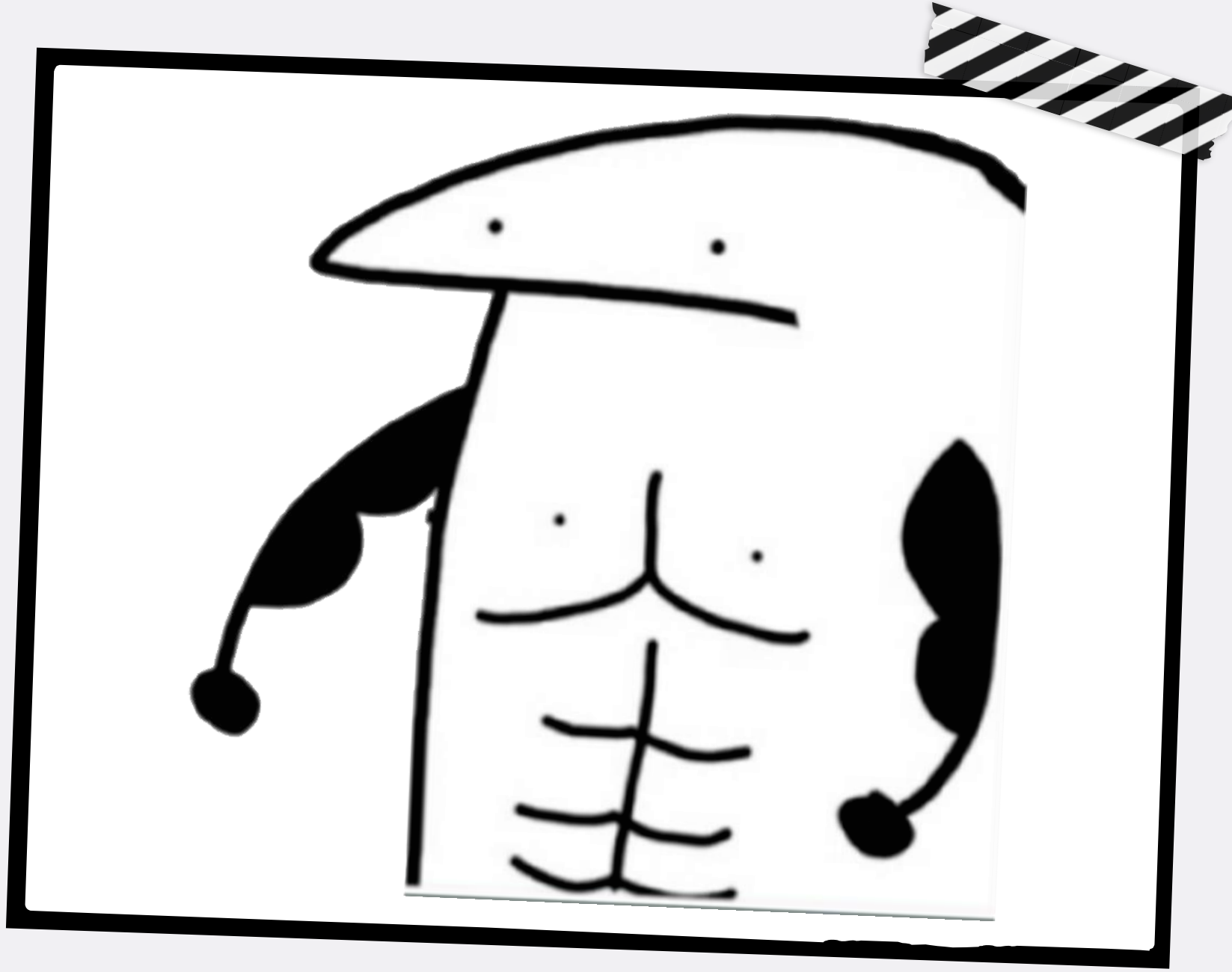

Clase 2

Agenda

- Manejo de Versiones
- Git
- GitHub
- GitLab
- GitHub Desktop
- GitKraken
- Comandos Básicos Git
- Releases (versiones)

Manejo de Versiones



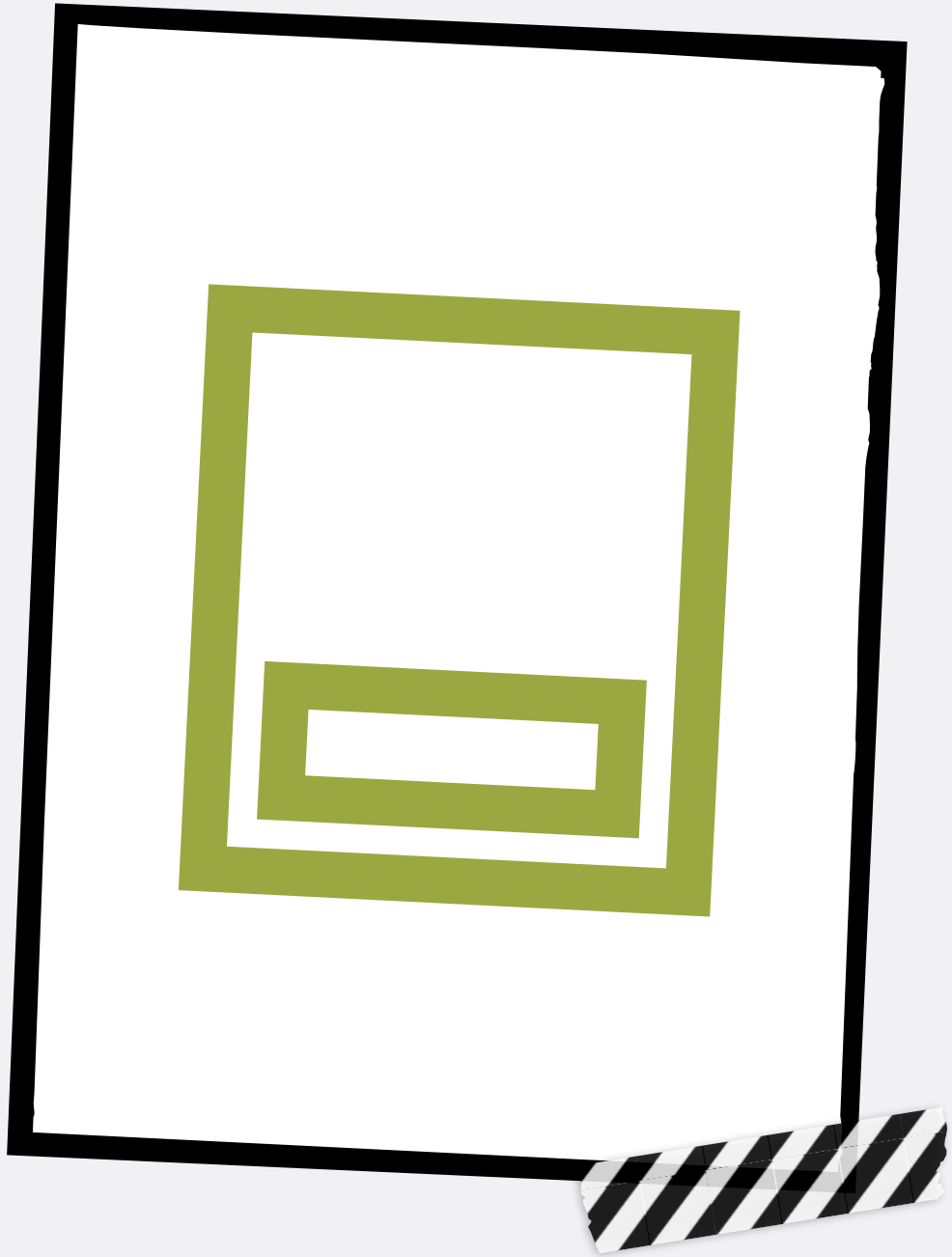
Los sistemas de control de versiones comienzan con una versión básica del documento y luego van guardando sólo los cambios que se hicieron en cada paso del proceso. Se puede pensar en ello como en una cinta: si se rebobina la cinta y se inicia de nuevo en el documento base, se puede reproducir cada cambio y terminar con la versión más reciente.

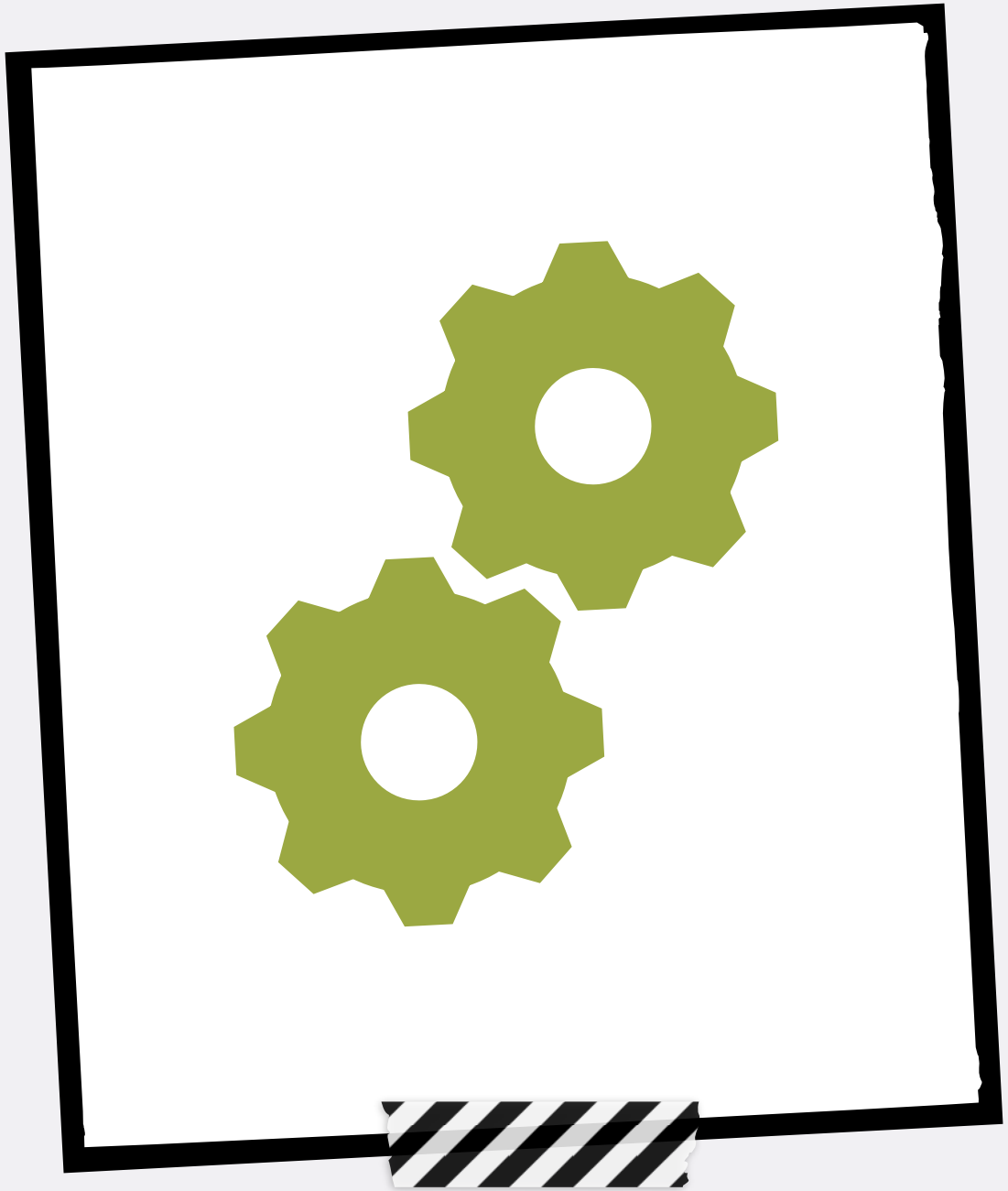
Sistema de Control de Versiones

Sistema de Control de Versiones

Una vez que piensas en los cambios como separados del documento en sí, entonces se puede pensar en “deshacer” diferentes conjuntos de cambios en el documento base y obtener así diferentes versiones del documento. Por ejemplo, dos usuarios pueden hacer conjuntos independientes de cambios basados en el mismo documento.

A menos que haya conflictos, se puede incluso tener dos conjuntos de cambios en el mismo documento base.





Sistema de Control de Versiones

Un sistema de control de versiones es una herramienta que realiza un seguimiento de estos cambios para nosotros y nos ayuda a controlar la versión y fusionar nuestros archivos. Nos permite decidir qué cambios conforman la siguiente versión, a lo que llamamos hacer un commit, y mantiene metadatos útiles sobre dichos cambios. El historial completo de commits para un proyecto en particular y sus metadatos forman un repositorio. Los repositorios pueden mantenerse sincronizados en diferentes computadoras, facilitando así la colaboración entre diferentes personas.

git

Git

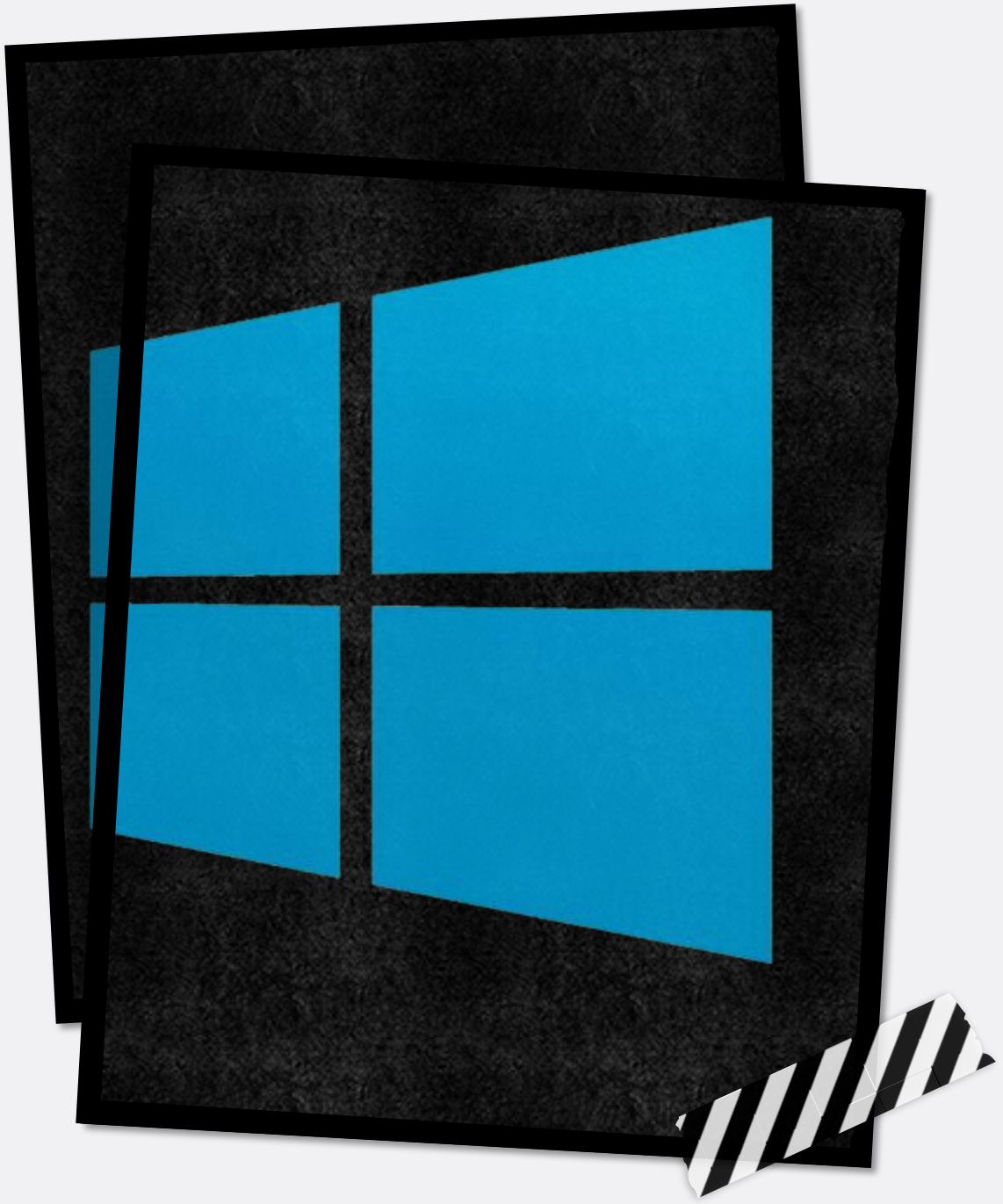
Git

Git es un proyecto de código abierto maduro y con un mantenimiento activo que desarrolló originalmente Linus Torvalds. Un asombroso número de proyectos de software dependen de Git para el control de versiones, incluidos proyectos comerciales y de código abierto. Los desarrolladores que han trabajado con Git cuentan con una buena representación en la base de talentos disponibles para el desarrollo de software, y este sistema funciona a la perfección en una amplia variedad de sistemas operativos e IDE.

Instalar Git (Windows)

Para instalar git es necesario ingresar el siguiente link y seleccionar el sistema operativo con el cual vas a descargar git:

https://git-scm.com/



- **Ubuntu/Debian**
  + `sudo apt update`
  + `sudo apt upgrade -y`
  + `sudo apt install git`
- **Arch Linux**
  + `sudo pacman -Syyu`
  + `sudo pacman -S git`

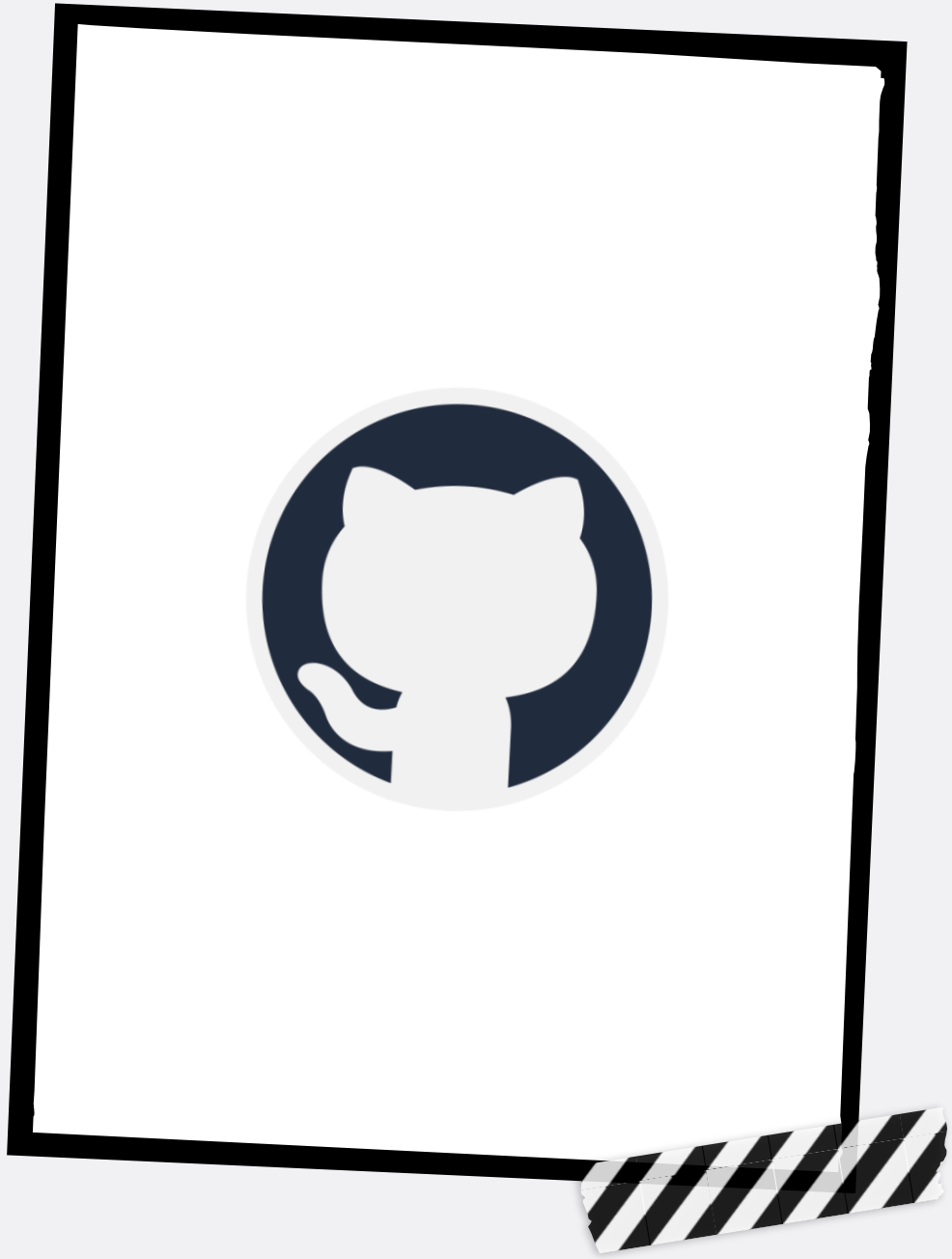
Instalar Git (Linux)

Alojamiento de Repositorios en la nube

GitHub

GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador. El software que opera GitHub fue escrito en Ruby on Rails.

A un alto nivel, GitHub es un sitio web y un servicio basado en la nube que ayuda a los desarrolladores a almacenar y administrar su código, así como a rastrear y controlar los cambios en su código.



Github Student Pack

Github ofrece beneficios con distintas herramientas de desarrollo para los estudiantes. Nosotros como estudiantes de la Facultad de Ingeniería si logramos contar con estos beneficios. Para ello es necesario ingresar al siguiente link, y registrarse con su correo institucional, junto con adjuntar su constancia de Incripción:

https://education.github.com/pack

Link directo al registro:

https://education.github.com/discount_requests/application

GitLab

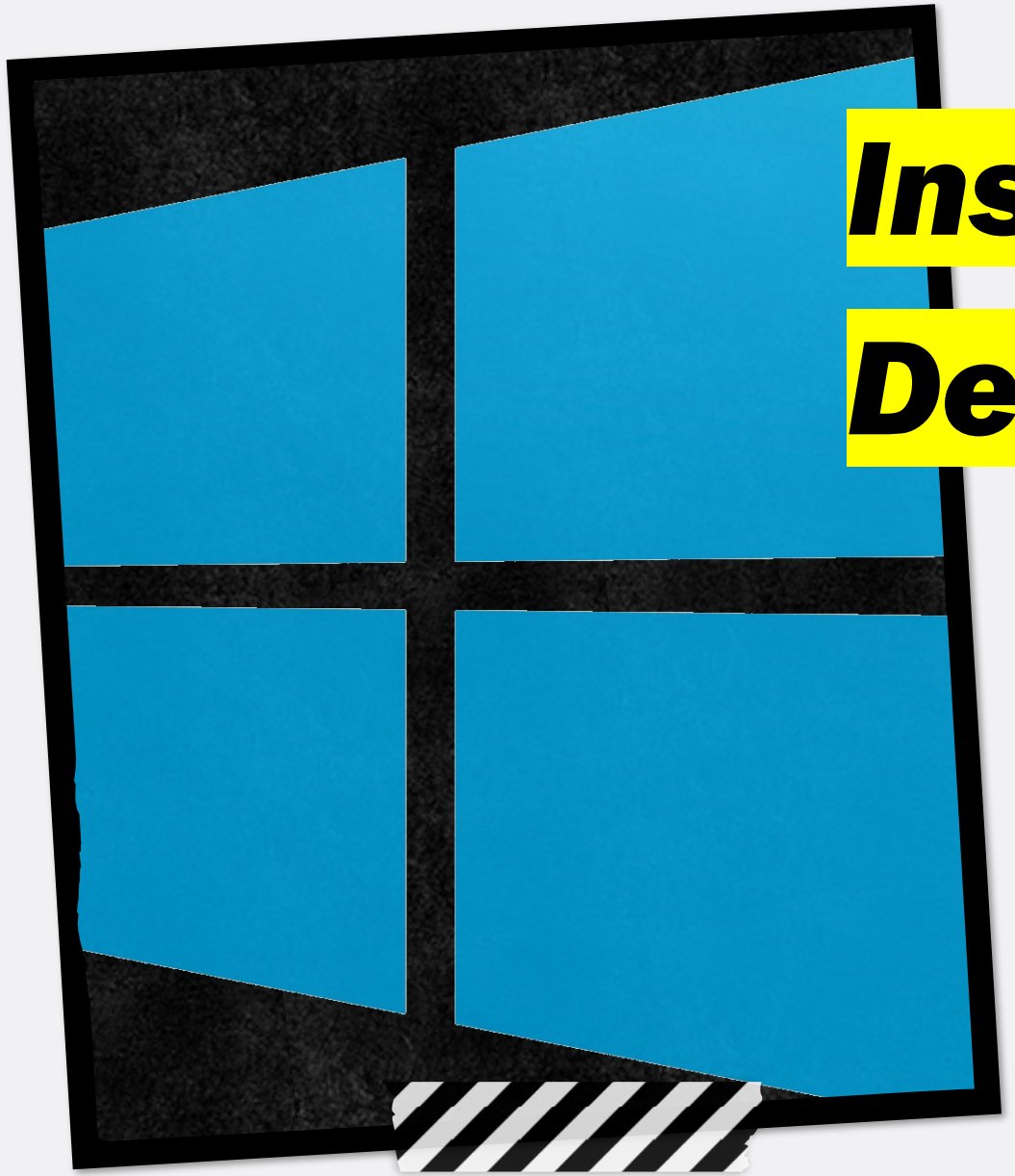
GitLab es una plataforma de desarrollo de software basada en la web que proporciona control de versiones mediante Git e integra herramientas para la gestión del ciclo de vida del desarrollo, como CI/CD (Integración y Despliegue Continuos), gestión de proyectos, control de versiones, revisión de código, y más.

Es una alternativa completa a otras plataformas como GitHub y Bitbucket, y se destaca por ofrecer tanto una versión autoalojada como una en la nube.

Herramientas

GitHub Desktop

GitHub Desktop es una aplicación gráfica gratuita que permite interactuar con los repositorios de GitHub sin necesidad de usar la línea de comandos. Está diseñada para simplificar el flujo de trabajo con Git, proporcionando una interfaz intuitiva para realizar tareas como clonar repositorios, gestionar ramas, realizar commits, resolver conflictos de fusión y enviar cambios al repositorio remoto.



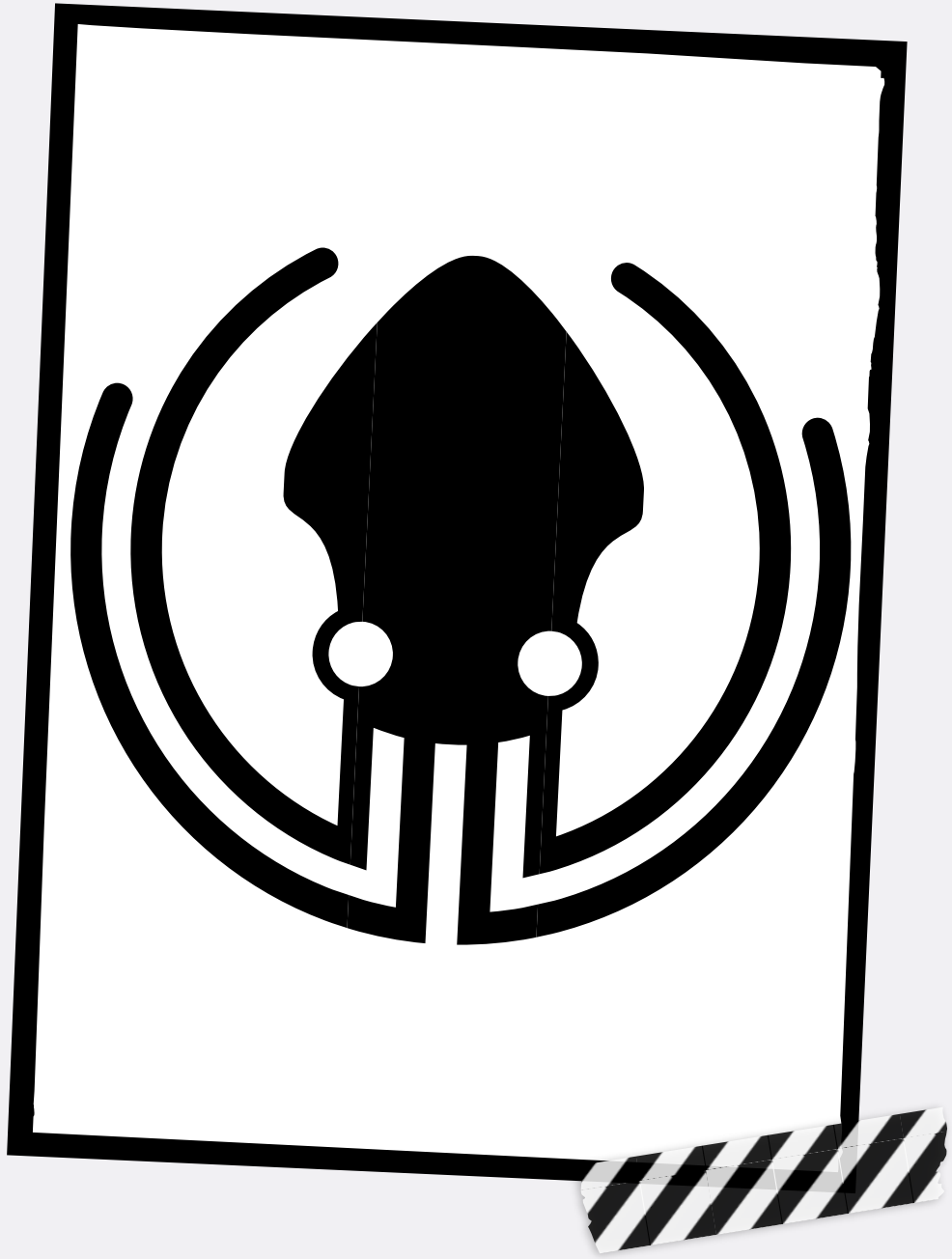
Instalación GitHub Desktop (Windows)

Descargar desde la página oficial

https://desktop.github.com/download/

GitKraken

A grandes rasgos, esta es una empresa de software global que ofrece un conjunto de herramientas para ayudar a los desarrolladores en la productividad. Además, permite que los equipos de trabajo puedan colaborar de forma más fácil sin importar el lugar donde estén. Esta plataforma te permite trabajar tanto con Windows como con Mac o Linux y está integrado con GitHub y GitLab. Al hablar de qué es GitKraken es importante mencionar que es una herramienta de uso gratuito cuando trabajas solo y quieres usar repositorios locales y públicos que se encuentran alojados en la nube



Instalación GitKraken (Windows)

Descargar desde la página oficial

https://www.gitkraken.com/

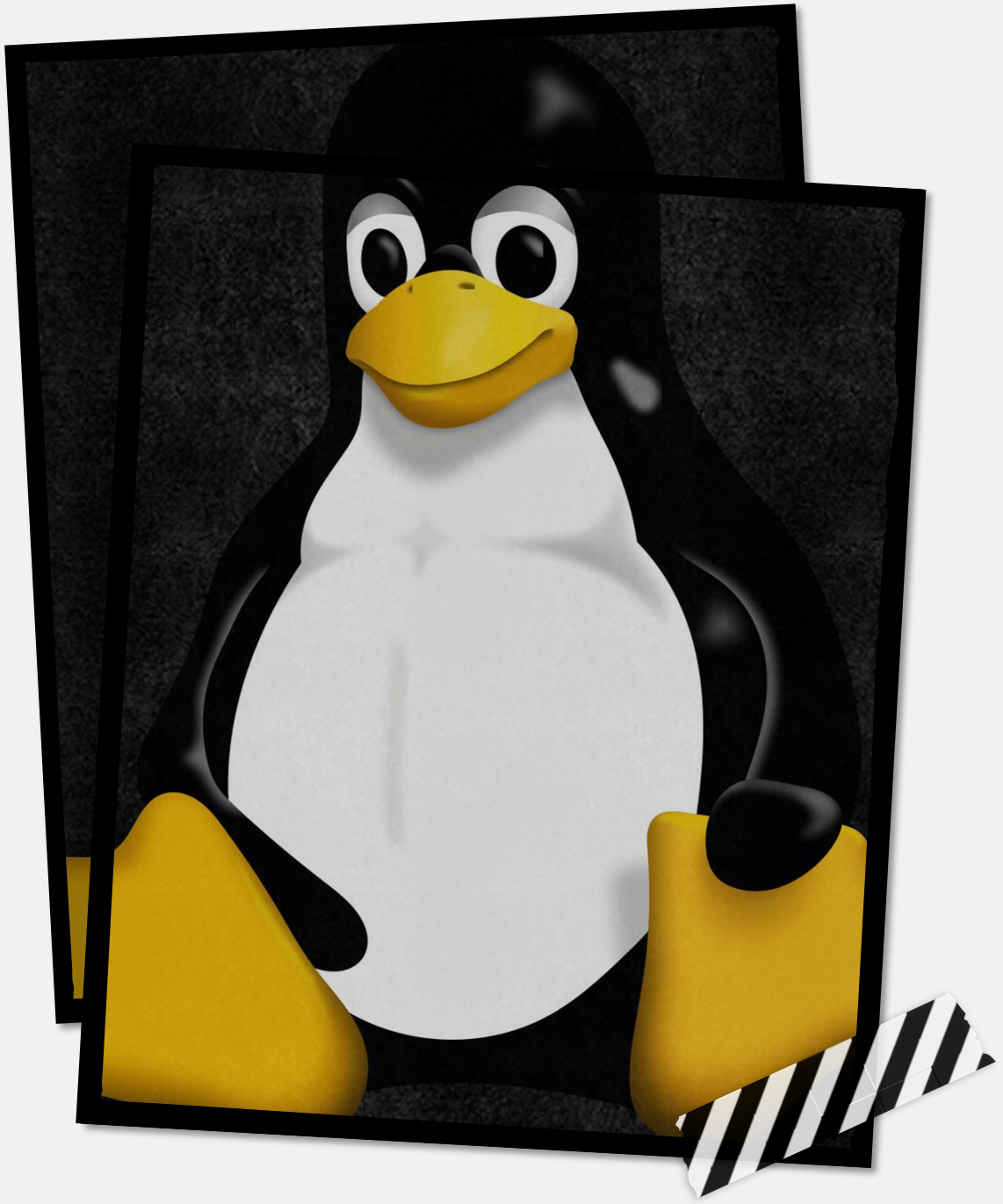
Instalación GitKraken (Linux)

Ubuntu/Debian

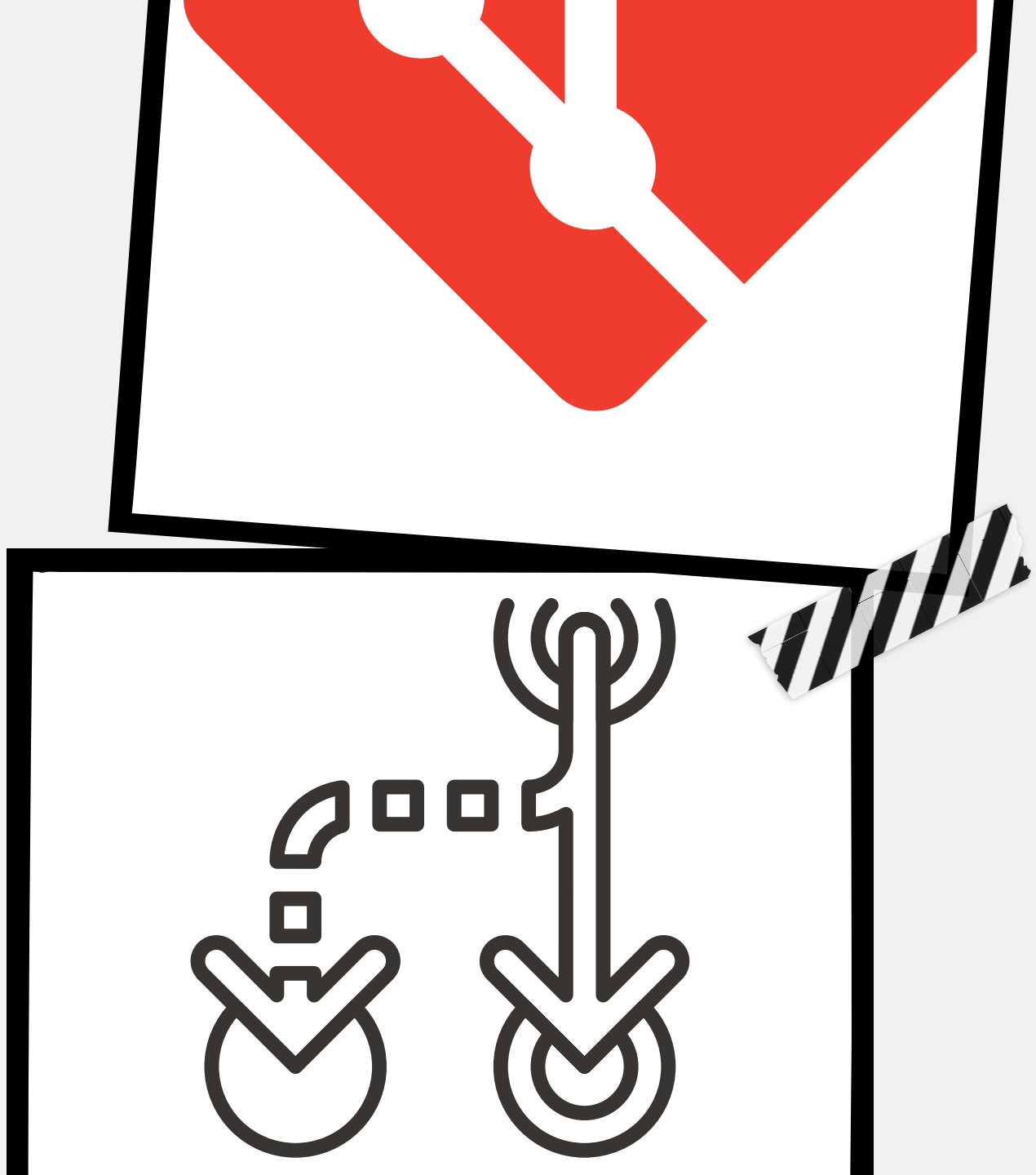
- Descargar de la página oficial: https://www.gitkraken.com/
- En la Terminal hacer lo siguiente:
  + cd ~/Descargas
  + sudo dpkg -i gitkraken-amd64.deb
    - En caso de problemas de dependencias
      + sudo apt-get install –f

Arch Linux

- paru -S gitkraken



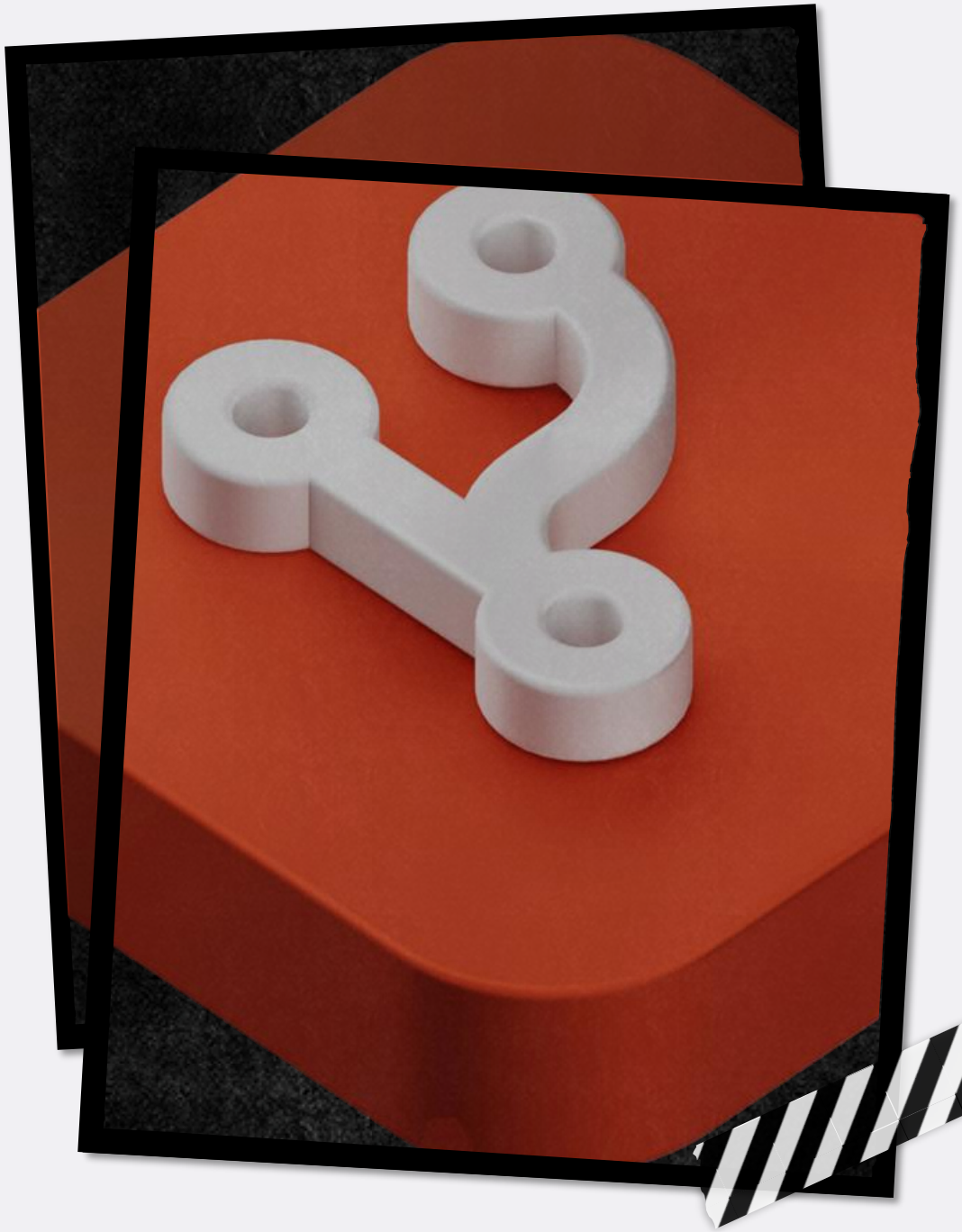
Comandos Básicos Git



git clone

Crea una copia de un repositorio Git existente. La clonación es la forma más común para que los desarrolladores obtengan una copia de trabajo de un repositorio central.

```
git clone https://github.com/MaxDuran08/IPC1-G_2025
```



git add .

Mueve los cambios del directorio de trabajo al área de puesta en escena. Esto le da la oportunidad de preparar una instantánea antes de comprometerla con la historia oficial.

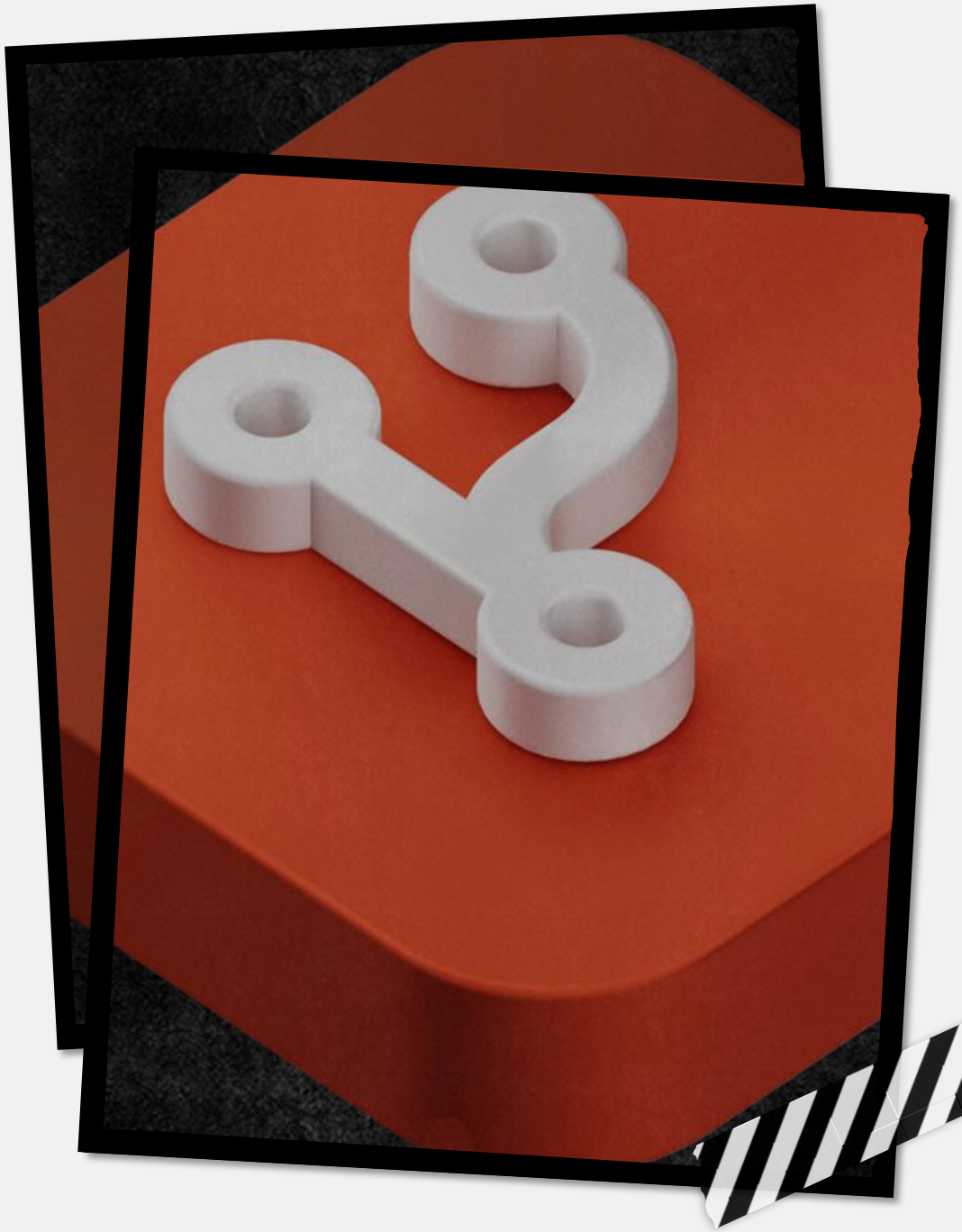
git status

Verifica que fueron agregados los cambios que se añadieron con git add.

git commit

Pasar la bandera de modificación a git commit le permite modificar la confirmación más reciente. Esto es muy útil cuando se olvida de organizar un archivo u omitir información importante del mensaje de confirmación.

```
git commit -m “[ADD]: Se agregaron …"
```



git push

Carga el ultimo commit al repositorio remoto

Creación Ramas

Crear rama:

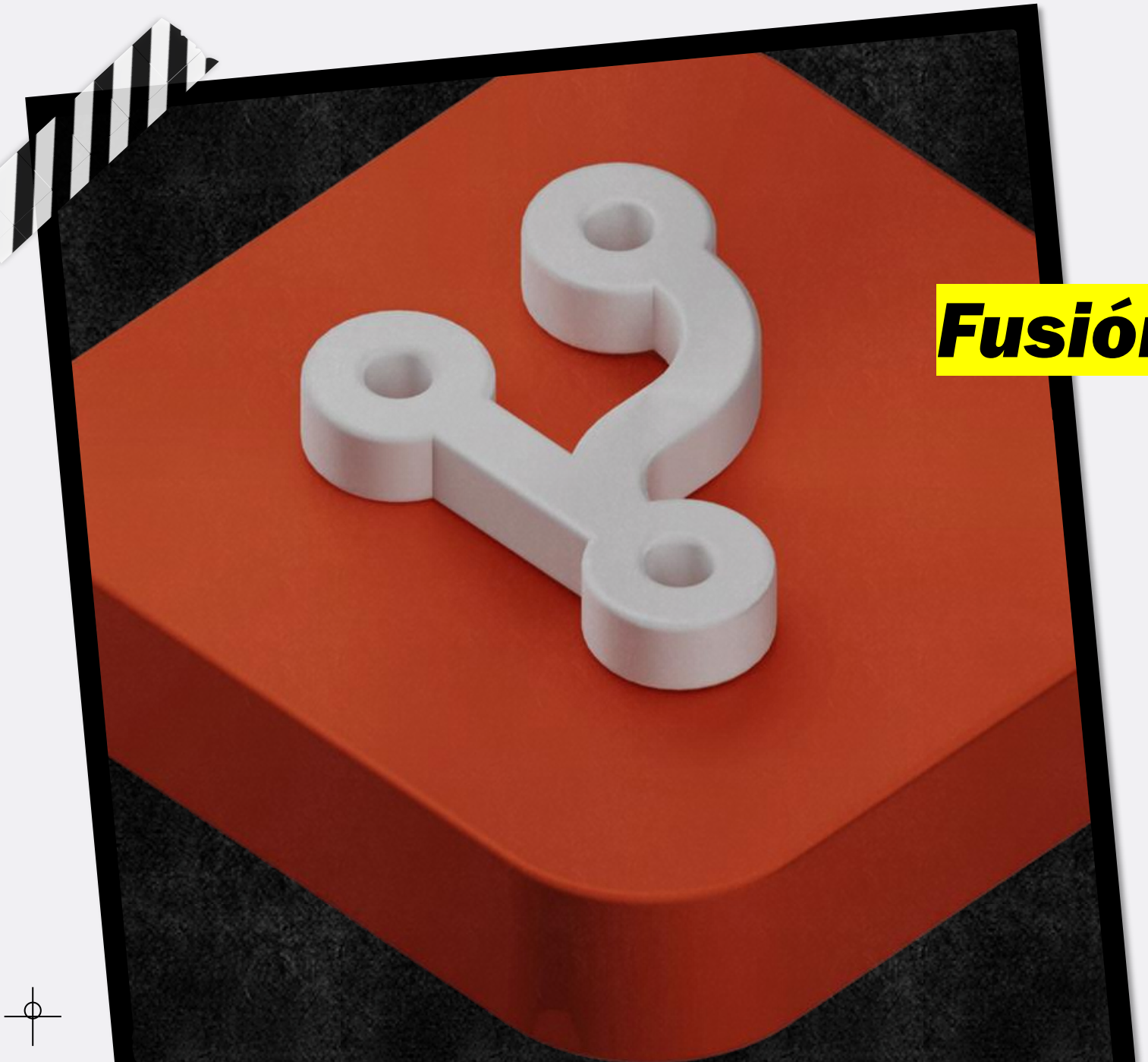
```
git branch rama-nueva
```

Crear la rama y ubicarse en la nueva rama:

```
git checkout -b rama-nueva
```

Listar ramas:

```
git branch
```



Fusión de Ramas

Si estoy en la rama main y quiero unir los cambios de la “rama1” a la rama “develop” se usa el siguiente comando:

```
git checkout develop
git merge --no-ff rama1
```

Eliminar Registro de Commit

Si cometes un error en el último commit que ya has enviado al repositorio remoto, se puede solucionar de la siguiente manera:

```
git reset –hard HEAD~1
git push origin HEAD –force
```

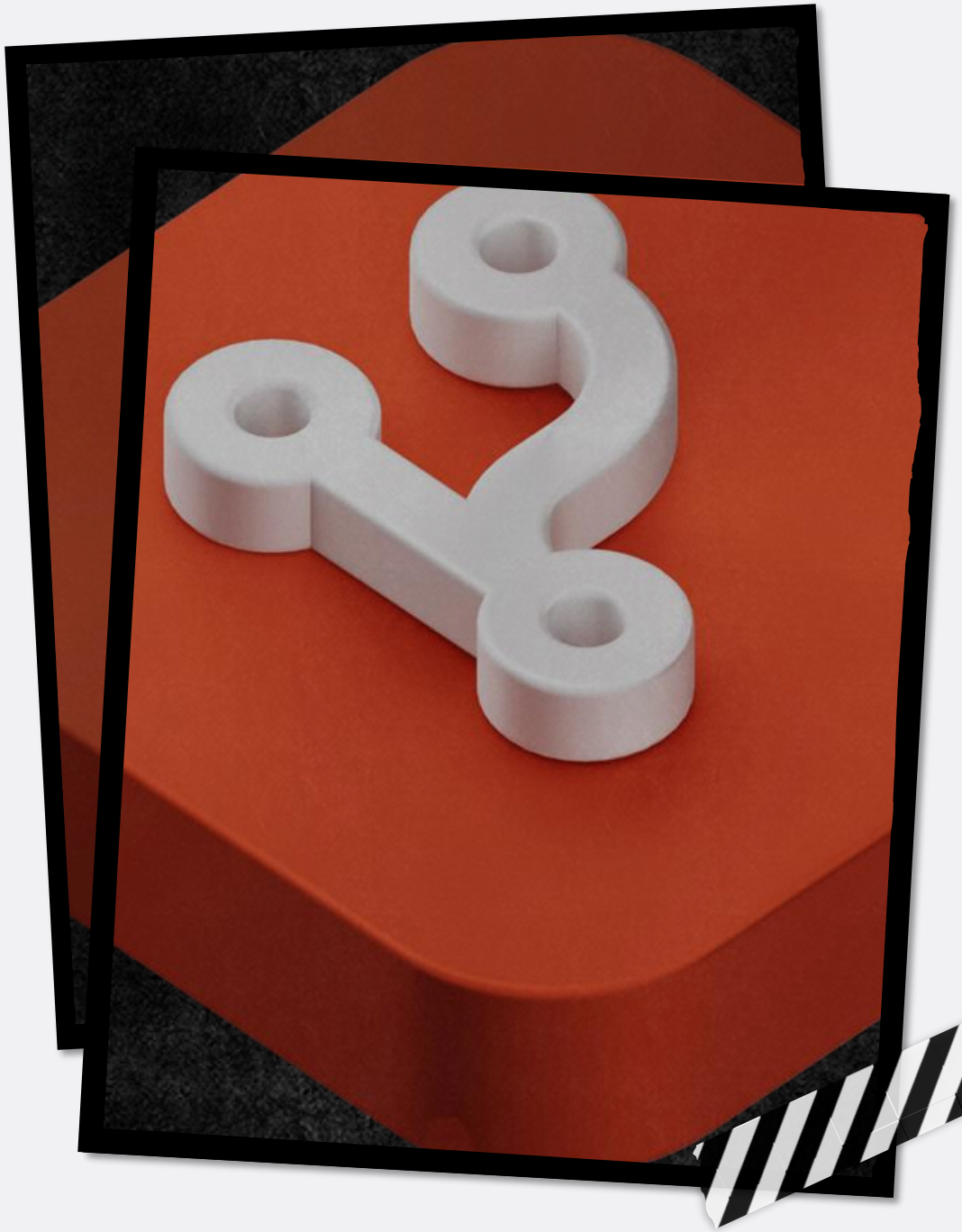
Más comandos

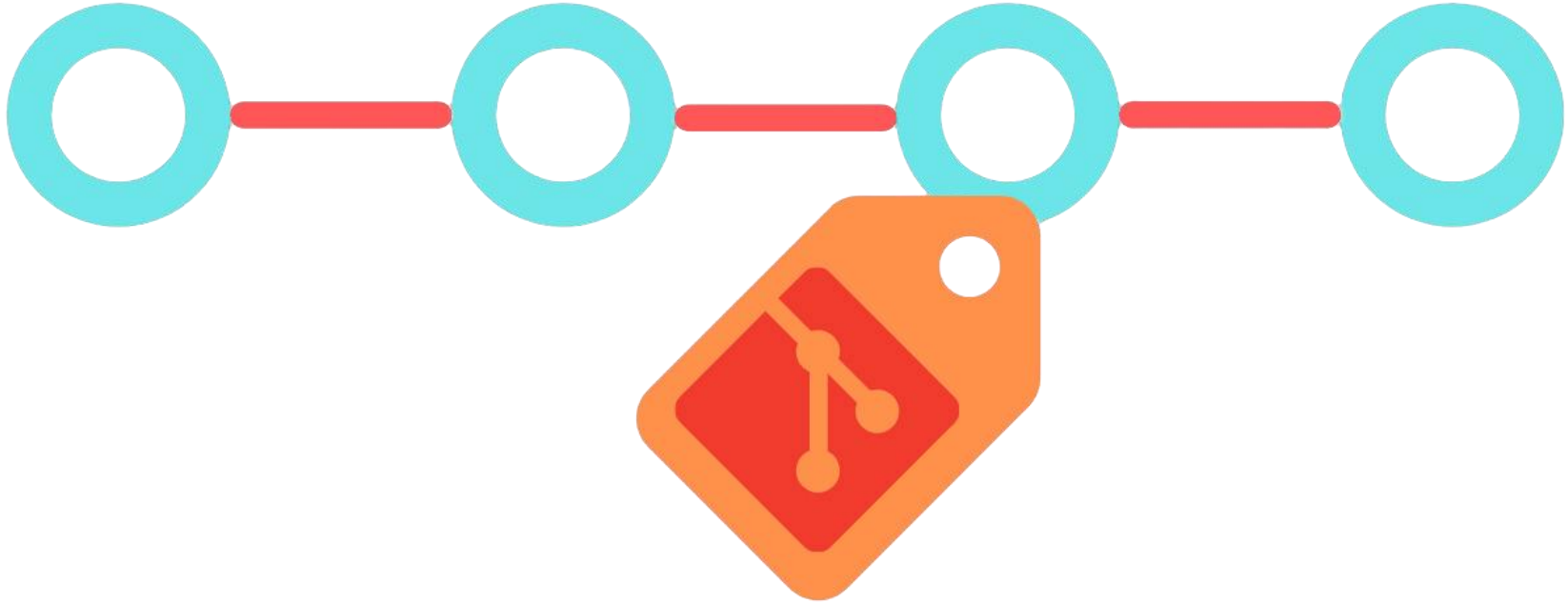
Si quieres ver más comandos de git, te recomiendo ver el siguiente MD:

https://github.com/MaxDuran08/IPC1-G_2025\Ejemplos\Ejemplo-29-01-2025\Git.md

Para aprender practicando:

https://learngitbranching.js.org/





Releases

Release

Son iteraciones de software implementables que puede empaquetar y poner a disposición de un público más amplio para descargar y utilizar.

Release

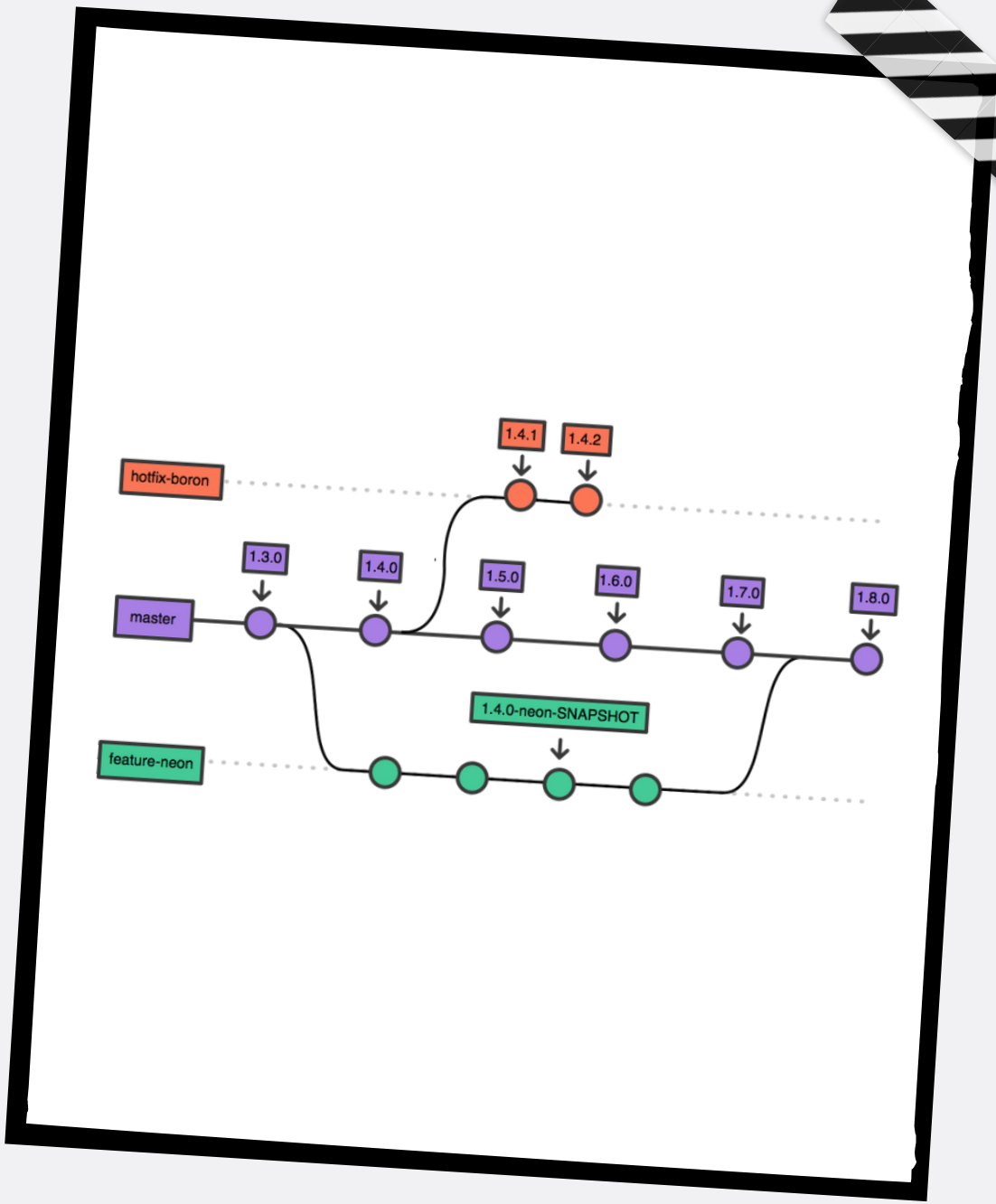
Algo común es realizar el manejo de versiones mediante 3 números: X.Y.Z y cada uno indica una cosa diferente:

- El primero (X) se le conoce como versión mayor y nos indica la versión principal del software. Ejemplo: 1.0.0, 3.0.0
- El segundo (Y) se le conoce como versión menor y nos indica nuevas funcionalidades. Ejemplo: 1.2.0, 3.3.0
- El tercero (Z) se le conoce como revisión y nos indica que se hizo una revisión del código por algun fallo. Ejemplo: 1.2.2, 3.3.4

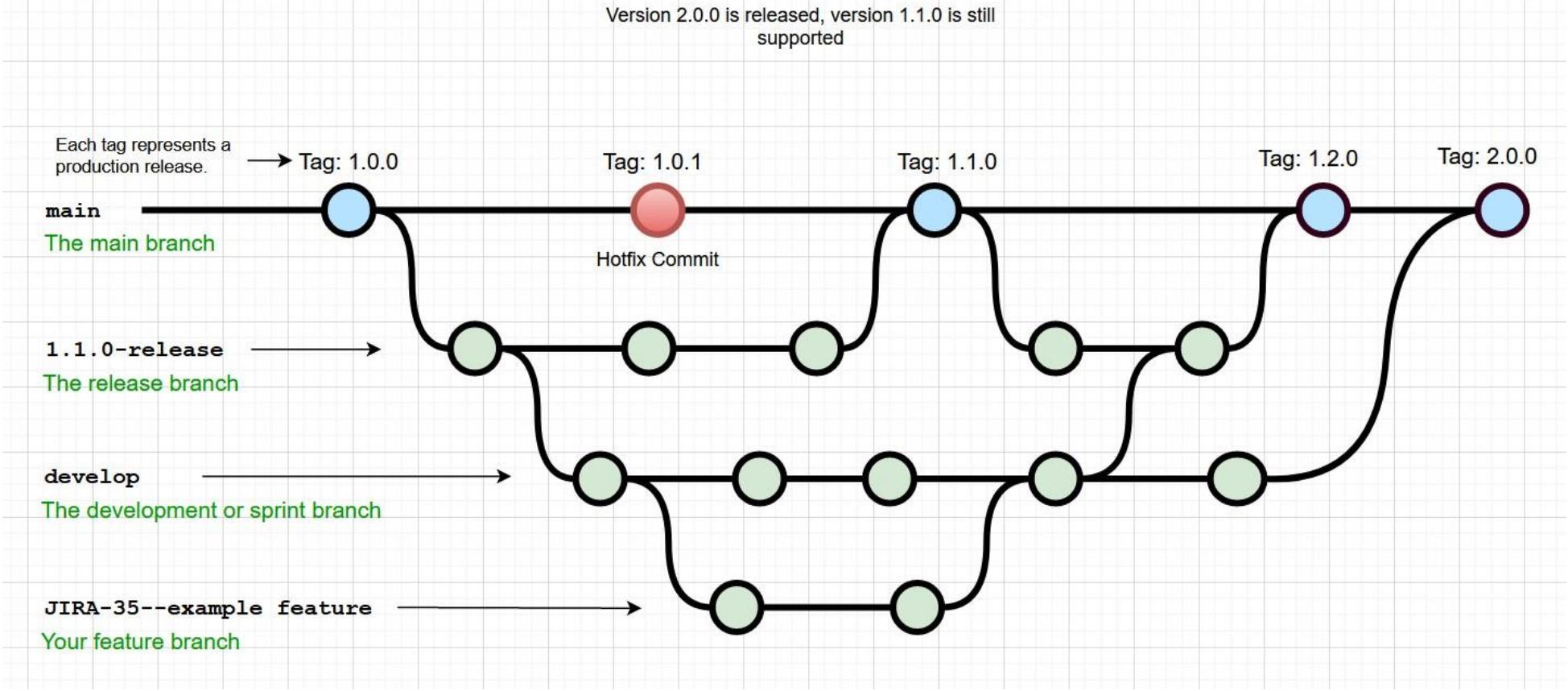
Ahora que conocemos el significado de cada número, viene una pregunta importante: ¿cómo sabemos cuándo cambiarlos y cuál cambiar?

- Versión mayor o X, cuando agreguemos nuevas funcionalidades importantes, puede ser como un nuevo módulo o característica clave para la funcionalidad.
- Versión menor o Y, cuando hacemos correcciones menores, cuando arreglamos un error y se agregan funcionalidades que no son cruciales para el proyecto.
- Revisión o Z, cada vez que entregamos el proyecto.

Release



GitFlow


Version 2.0.0 is released, version 1.1.0 is still supported
Each tag represents a production release.
Tag: 1.0.0
Tag: 1.0.1
Tag: 1.1.0
Tag: 1.2.0
Tag: 2.0.0
main
The main branch
Hotfix Commit
1.1.0-release
The release branch
develop
The development or sprint branch
JIRA-35--example feature
Your feature branch

Dudas o Comentarios