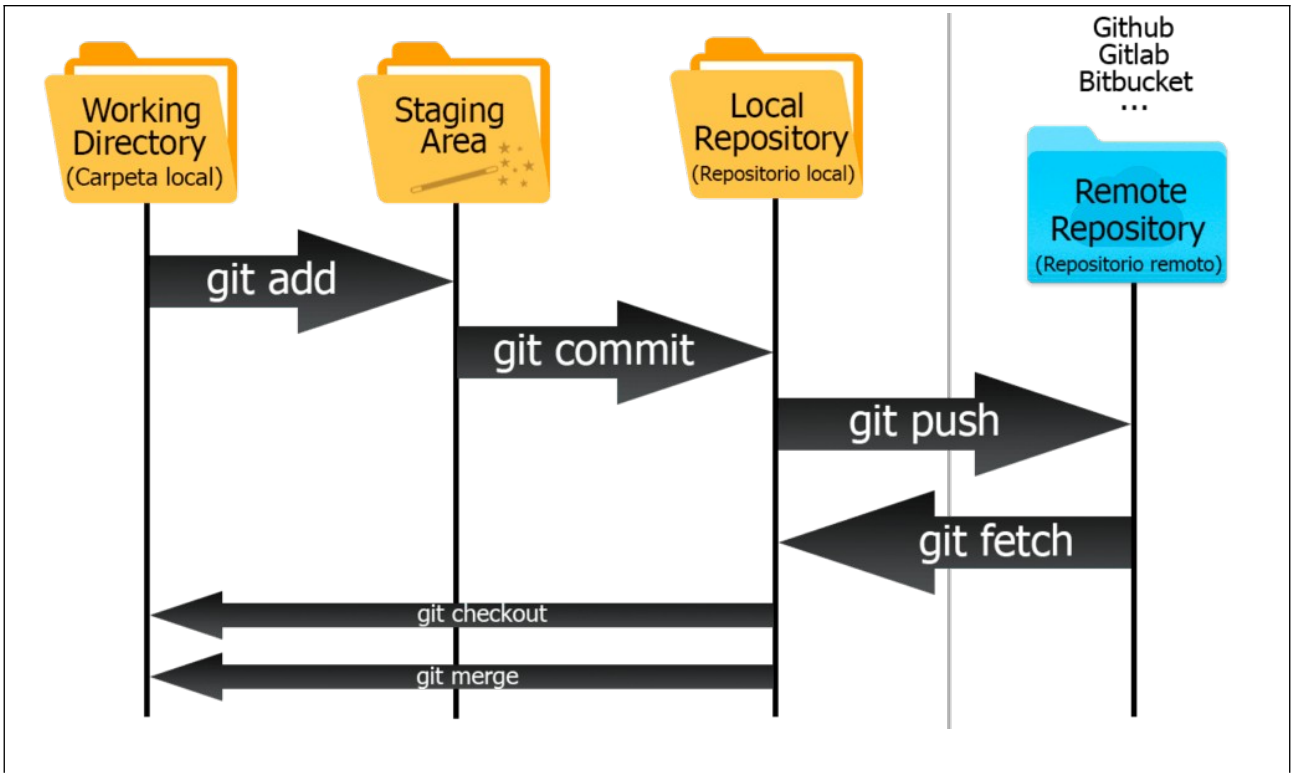
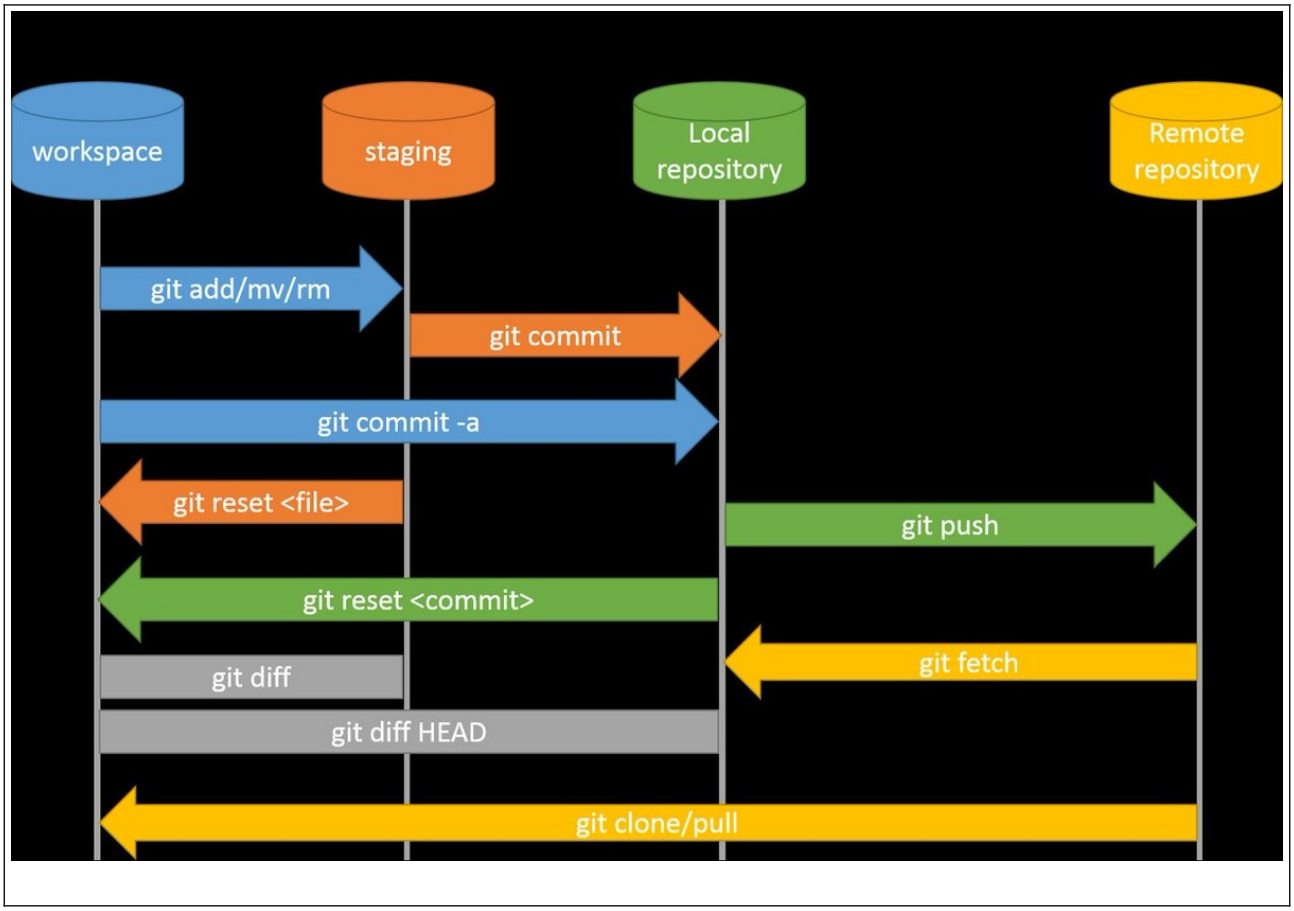
¿Qué es Git?

•Git es un sistema de control de versiones distribuido (DVCS por sus siglas en inglés) ampliamente utilizado en el desarrollo de software. Permite rastrear y gestionar los cambios realizados en el código fuente de un proyecto a lo largo del tiempo. A diferencia de los sistemas centralizados que dependen de un único repositorio, Git permite a los desarrolladores tener una copia completa del historial del proyecto en sus máquinas locales, lo que facilita el trabajo sin conexión y la colaboración. Fue creado por Linus Torvalds en 2005 para el desarrollo del kernel de Linux.

¿Cuáles son los conceptos clave en Git?

•Algunos conceptos clave en Git incluyen:
•**Repositorio:** El lugar donde se almacena el historial completo del proyecto, ya sea de forma local o remota.
•**Commit:** Una instantánea de los archivos del proyecto en un punto específico en el tiempo, con un identificador único y un mensaje descriptivo.
•**Branch (rama):** Una línea de desarrollo independiente que se deriva de la rama principal, permitiendo trabajar en paralelo sin afectar el código principal.
•**Merge (fusión):** El proceso de combinar los cambios de una rama en otra.
•**Directorio de trabajo (Working Directory):** La copia local de los archivos del proyecto donde se realizan las modificaciones.
•**Área de preparación (Staging Area):** Una zona intermedia donde se seleccionan los cambios del directorio de trabajo para ser incluidos en el próximo commit.





•

¿Cuáles son las principales ventajas de usar Git?

•Git ofrece numerosas ventajas, entre las que destacan:
•**Control de versiones eficiente:** Permite rastrear y revertir cambios fácilmente, facilitando la recuperación de versiones anteriores.
•**Trabajo en ramas y fusión:** Fomenta el desarrollo paralelo y la colaboración sin interferir en la rama principal.
•**Flexibilidad y trabajo sin conexión:** Al ser distribuido, cada desarrollador tiene una copia local del repositorio y puede trabajar independientemente.
•**Integridad de datos:** Utiliza algoritmos criptográficos para garantizar la autenticidad del historial y prevenir la corrupción de datos.
•**Colaboración y seguimiento:** Facilita el trabajo en equipo y el seguimiento de los cambios realizados por diferentes colaboradores.
•**Compatibilidad y comunidad activa:** Se integra con diversas herramientas y cuenta con una gran comunidad de soporte.

¿Cómo se compara Git con otros sistemas de control de versiones como SVN?

•Git es un sistema de control de versiones distribuido, mientras que SVN (Subversion) es un sistema centralizado. Esto significa que con Git, cada desarrollador tiene una copia completa del repositorio, lo que permite trabajar sin conexión y realizar operaciones como la creación de ramas localmente. Con SVN, se depende de un repositorio central. Las ramas y fusiones son generalmente más rápidas y eficientes en Git. Aunque SVN era un estándar antes de Git y aún se utiliza en algunos proyectos heredados o con necesidades específicas como monorepos grandes o gestión

explícita de archivos binarios grandes, Git se ha convertido en el estándar de la industria debido a su flexibilidad, rendimiento y características distribuidas.

¿Qué son GitHub y GitLab y cómo se relacionan con Git?

•GitHub y GitLab son plataformas de alojamiento de repositorios basadas en Git. Proporcionan servicios adicionales para facilitar la colaboración en proyectos de software. Si bien Git es el software de control de versiones subyacente, GitHub y GitLab son servicios web que permiten almacenar repositorios remotos, gestionar solicitudes de extracción (pull requests), realizar seguimiento de problemas, y mucho más. No son lo mismo que Git, pero son herramientas muy populares que extienden la funcionalidad de Git para el trabajo en equipo y la gestión de proyectos.

¿Cuáles son algunos comandos básicos de Git?

•Algunos comandos básicos de Git incluyen:
•git init: Inicializa un nuevo repositorio Git local.
•git clone [url]: Descarga un repositorio existente desde una URL.
•git status: Muestra el estado actual del directorio de trabajo y el área de preparación.
•git add [archivo]: Agrega cambios a un archivo al área de preparación.
•git commit -m "<mensaje>": Guarda los cambios en el área de preparación como un nuevo commit.
•git branch: Lista las ramas locales.
•git checkout [nombre_rama]: Cambia a la rama especificada.
•git merge [nombre_rama]: Fusiona los cambios de la rama especificada en la rama actual.
•git pull: Descarga los cambios de un repositorio remoto y los fusiona con la rama local actual.
•git push [alias] [rama]: Sube los commits locales a un repositorio remoto.

Comando	Descripción
`git init`	Inicializa un nuevo repositorio Git, creando un nuevo directorio `.git` en el proyecto.
`git clone [URL]`	Crea una copia local de un repositorio remoto. La URL puede ser tanto HTTPS como SSH.
`git status`	Muestra el estado actual de los cambios en el repositorio, indicando archivos modificados, agregados o no rastreados.
`git add [archivo]`	Añade archivos al área de preparación (staging) para ser incluidos en el próximo commit. Puedes usar `.` para agregar todos los archivos modificados.
`git commit -m "mensaje"`	Registra los cambios en el repositorio, creando un nuevo commit con un mensaje que describe las modificaciones realizadas.
`git pull`	Recupera cambios desde un repositorio remoto y los fusiona automáticamente con la rama local actual.
`git push [remoto] [rama]`	Sube los commits locales al repositorio remoto. Especifica el nombre del remoto y la rama. Ejemplo: `git push origin master`.
`git pull origin master --rebase`	Recupera cambios desde el repositorio remoto y aplica los commits locales encima de los commits remotos.

¿Se utiliza Git en empresas a gran escala?

•Sí, Git es ampliamente utilizado en empresas a gran escala. Aunque no siempre se utilizan plataformas públicas como GitHub para código propietario sensible, muchas empresas configuran sus propios servidores Git privados o utilizan ediciones empresariales de plataformas como GitHub, GitLab o Bitbucket. Empresas como Google, Microsoft, Facebook, Amazon Web Services (AWS) y muchas otras utilizan Git para gestionar su código. Git está diseñado para manejar proyectos grandes, como el kernel de Linux, que tiene millones de líneas de código y un historial extenso. Si bien puede haber desafíos con repositorios extremadamente grandes o con la gestión de archivos binarios masivos y que cambian con frecuencia, Git es capaz de manejar la gran mayoría de las necesidades de proyectos a escala empresarial.

¿Cómo se instala Git?

•Para instalar Git, generalmente se debe descargar el instalador desde el sitio web oficial de Git (git-scm.com) y seguir las instrucciones para el sistema operativo correspondiente (Linux, macOS o Windows). Es importante asegurarse de que Git se agregue al PATH del sistema para poder acceder a él desde la línea de comandos. Después de la instalación, se puede verificar escribiendo git --version en la terminal. También es recomendable configurar el nombre de usuario y la dirección de correo electrónico asociados a los commits utilizando los comandos git config --global user.name "Tu Nombre" y git config --global user.email "tu.correo@example.com".

Comando	Descripción	Ejemplo de Uso
`git init`	Inicializa un nuevo repositorio Git en el directorio actual.	`git init`
`git clone [url]`	Clona un repositorio remoto en tu máquina local.	`git clone https://github.com/ricardoinstructor/ejemplo.git`
`git status`	Muestra el estado actual del repositorio (archivos modificados, no rastreados, etc.).	`git status`
`git add [archivo]`	Añade un archivo al área de preparación (staging area) para el próximo commit.	`git add index.html`
`git add .`	Añade todos los archivos modificados o nuevos al área de preparación.	`git add .`
`git commit -m "[mensaje]"`	Guarda los cambios en el repositorio local con un mensaje descriptivo.	`git commit -m "Arreglar bug de inicio"`
`git log`	Muestra el historial de commits.	`git log`
`git push [remoto] [rama]`	Envía los commits locales a un repositorio remoto.	`git push origin main`
`git pull [remoto]`	Actualiza el repositorio local	`git pull origin main`

Comando	Descripción	Ejemplo de Uso
`[rama]`	con los últimos cambios del repositorio remoto.	
`git branch`	Lista todas las ramas del repositorio.	`git branch`
`git branch [nombre]`	Crea una nueva rama con el nombre especificado.	`git branch feature-login`
`git checkout [rama]`	Cambia a la rama especificada.	`git checkout feature-login`
`git checkout -b [rama]`	Crea y cambia a una nueva rama.	`git checkout -b feature-login`
`git merge [rama]`	Fusiona otra rama con la actual.	`git merge feature-login`
`git remote add [nombre] [url]`	Asocia un repositorio remoto con un nombre (por ejemplo, "origin").	`git remote add origin https://github.com/ricardoinstructor/ejemplo.git`
`git config --global user.name "[nombre]"`	Configura el nombre de usuario global para Git.	`git config --global user.name "Juan Pérez"`
`git config --global user.email "[email]"`	Configura el correo electrónico global para Git.	`git config --global user.email "juan@example.com"`
`git diff`	Muestra las diferencias entre los archivos del working directory y el staging area.	`git diff`
`git diff --staged`	Muestra las diferencias que están en el staging area listas para commit.	`git diff --staged`