Comando de cmd para ver la contraseña guardada

Netsh wlan show profile name=Mza-internet key=clear

Importancia de la gestión de GIT

Evitar la perdida de código y datos valioso

Facilitar colaboración en equipo de desarrollo

Rastrar y documentar cambios en el código a lo largo del tiempo

Hacer posible la reversión a versiones en caso de problemas

git init

git add .

git commit -m "first commit"

git remote add origin https://github.com/NOMBRE\_USUARIO/NOMBRE\_PROYECTO.git

git push -u origin master

A continuación, se presenta una guía paso a paso:

1. **Paso 1: Inicializar un Repositorio Local (init):**
   * Para crear un nuevo repositorio Git en tu proyecto, ve a la carpeta raíz y ejecuta el siguiente comando:

git init

* + Esto establece un repositorio local en tu proyecto y crea una carpeta oculta llamada ".git" que almacena la información de control de versiones.

1. **Paso 2: Agregar Cambios al Área de Preparación (add):**
   * Antes de confirmar (commit) tus cambios, debes agregarlos al "área de preparación". Utiliza el siguiente comando para agregar archivos o cambios específicos:

git add nombre-del-archivo

* + También podés usar git add . para agregar todos los cambios en la carpeta actual al área de preparación.

1. **Paso 3: Confirmar Cambios (commit):**
   * Una vez que hayas agregado tus cambios al área de preparación, realiza un compromiso para registrarlos en el historial de versiones. Utiliza el siguiente comando:

git commit -m "Mensaje descriptivo"

* + Reemplaza "Mensaje descriptivo" con un mensaje que describa los cambios que has realizado. Este mensaje es importante para mantener un registro de los cambios realizados.

Estos son los comandos básicos que necesitas para comenzar a utilizar Git. 

* Con `init`, podés crear un nuevo repositorio local.
* Con `clone`, podés obtener una copia de un repositorio remoto.
* Luego, con `add` y `commit`, podés rastrear y confirmar tus cambios.

A medida que te familiarices con estos comandos, podrás gestionar eficazmente tus proyectos de desarrollo de software.

**Concepto de Rama:**

Una rama en Git es una línea independiente de tareas que permite trabajar en características, correcciones de errores u otras áreas de desarrollo y/o testing de forma aislada. Cada rama representa una bifurcación en el historial de versiones del proyecto, lo que facilita la gestión de cambios y la colaboración en equipo.

**Paso 1: Creación de una Nueva Rama:**

* Para crear una nueva rama en Git, utiliza el siguiente comando:

git branch nombre-de-la-rama

* Reemplaza "nombre-de-la-rama" con el nombre que deseas para tu nueva rama. Esto crea la rama, pero aún no te mueves a ella.

**Paso 2: Cambio a una Rama (Checkout):**

* Para cambiar a la nueva rama que has creado, utiliza el siguiente comando:

git checkout nombre-de-la-rama

* Esto te llevará a la rama recién creada, y podrás comenzar a trabajar en ella de forma aislada.

**Paso 3: Creación y Gestión de Cambios:**

* En la nueva rama, realiza los cambios necesarios en tu proyecto, como la implementación de nuevas características o correcciones de errores.
* Utiliza los comandos git add y git commit para agregar y confirmar los cambios, de la misma manera que lo harías en la rama principal (por ejemplo, "master").

**Paso 4: Eliminación de Ramas:**

* Si ya no necesitas una rama, podés eliminarla con el siguiente comando:

git branch -d nombre-de-la-rama

* El uso de "-d" garantiza que la rama se elimine solo si los cambios se han fusionado correctamente en la rama principal (Veremos en capítulos siguientes que es la fusión de ramas). Si deseas eliminar una rama sin realizar una fusión, utilizá "-D" en lugar de "-d".

**Rama "main" (o "master"):**

En Git, la rama "main" (anteriormente conocida como "master") es la rama principal de desarrollo. Es la rama predeterminada en un repositorio y generalmente contiene la versión estable y funcional de un proyecto. Es común que la rama "main" se utilice para despliegues en producción.

**Otras Ramas:**

Además de la rama "main", podés crear otras ramas para trabajar en características específicas, correcciones de errores o experimentos. Las ramas secundarias se utilizan para aislar el trabajo y evitar afectar la rama principal. Algunas de estas ramas pueden ser fusionadas nuevamente en "main" una vez que se hayan completado los cambios.

**Buenas Prácticas en el Uso de Ramas:**

Para una gestión eficiente de ramas en Git, considera las siguientes buenas prácticas:

* Nombrado de Ramas: Utiliza nombres descriptivos para tus ramas. Por ejemplo, nombrá una rama de desarrollo de una nueva característica como "feature/nombre-de-la-caracteristica".
* Separación de Trabajo: Cada rama debe enfocarse en una tarea o característica específica. Esto facilita el seguimiento y la colaboración.
* Fusión Regular: Fusiona las ramas secundarias en "main" regularmente para mantener un historial de versiones actualizado.
* Uso de Solicitudes de Extracción (Pull Requests): En entornos colaborativos, utiliza solicitudes de extracción para revisar y aprobar cambios antes de fusionarlos en "main".
* Eliminar Ramas Obsoletas: Después de fusionar cambios y verificar que no son necesarios más desarrollos en una rama, elimínala para mantener un repositorio limpio.
* Evitar Trabajar en "main": Evita realizar cambios directamente en la rama "main" para minimizar el riesgo de conflictos y errores.
* Principio del formulario
* ¿Cuál es el propósito del comando "git init" en Git?
* a.
* Para confirmar los cambios en el historial de versiones.
* b.
* Para crear un nuevo repositorio Git en la carpeta del proyecto.
* c.
* Para agregar cambios al área de preparación.
* **Retroalimentación**
* Respuesta correcta
* La respuesta correcta es:
* Para crear un nuevo repositorio Git en la carpeta del proyecto.
* **Pregunta 2**
* Incorrecta
* Se puntúa 0,00 sobre 2,00
* Marcar pregunta
* **Enunciado de la pregunta**
* ¿Cuál es el propósito del comando "git commit -m 'Mensaje descriptivo'"?
* a.
* Para registrar los cambios en el historial de versiones con un mensaje descriptivo.
* b.
* Para agregar cambios al área de preparación.
* c.
* Para verificar el estado del repositorio.
* d.
* Para crear una copia de un repositorio remoto.
* **Retroalimentación**
* Respuesta incorrecta.
* La respuesta correcta es:
* Para registrar los cambios en el historial de versiones con un mensaje descriptivo.
* **Pregunta 3**
* Correcta
* Se puntúa 2,00 sobre 2,00
* Marcar pregunta
* **Enunciado de la pregunta**
* ¿Qué comando se utiliza para agregar cambios al "área de preparación" antes de confirmarlos en el historial de versiones?
* a.
* git init
* b.
* git status
* c.
* git add
* d.
* git commit
* **Retroalimentación**
* Respuesta correcta
* La respuesta correcta es:
* git add
* **Pregunta 4**
* Incorrecta
* Se puntúa 0,00 sobre 2,00
* Marcar pregunta
* **Enunciado de la pregunta**
* ¿Qué deberías ver en la salida del comando "git status" si el repositorio se ha inicializado correctamente?
* a.
* Un mensaje de error.
* b.
* Una lista de todos los archivos en la carpeta.
* c.
* Un mensaje que te indica que no hay nada para confirmar y que estás en la rama principal.
* d.
* Un resumen de los cambios realizados en el repositorio.
* **Retroalimentación**
* Respuesta incorrecta.
* La respuesta correcta es:
* Un mensaje que te indica que no hay nada para confirmar y que estás en la rama principal.
* **Pregunta 5**
* Correcta
* Se puntúa 2,00 sobre 2,00
* Marcar pregunta
* **Enunciado de la pregunta**
* ¿Cuál de los siguientes comandos NO se menciona en el texto como parte de los comandos básicos de Git?
* a.
* git clone
* b.
* git commit
* c.
* git init
* d.
* git status
* **Retroalimentación**
* Respuesta correcta
* La respuesta correcta es:
* git clone
* Final del formulario

**Diferencias entre Git y GitHub**

Git y GitHub son dos conceptos clave en el desarrollo de software, pero son diferentes en su naturaleza y propósito. A continuación, se presentan las diferencias entre Git y GitHub:

1. **Paso 1: Git - Sistema de Control de Versiones:**
   * Git es un sistema de control de versiones distribuido. Permite el seguimiento de cambios en archivos y carpetas a lo largo del tiempo, lo que facilita la colaboración y la gestión de proyectos de desarrollo de software.
2. **Paso 2: GitHub - Plataforma de Hospedaje:**
   * GitHub, por otro lado, es una plataforma en línea que se utiliza para alojar repositorios de Git. Proporciona servicios adicionales, como seguimiento de problemas, administración de proyectos y colaboración en línea, además del control de versiones.
3. **Paso 3: Git - Local y Privado:**
   * Git se utiliza localmente en tu computadora para realizar el seguimiento de cambios en tus proyectos. Podés crear repositorios Git en tu máquina y mantenerlos privados sin necesidad de conectarte a Internet.
4. **Paso 4: GitHub - Alojamiento en la Nube:**
   * GitHub almacena repositorios en la nube. Podés alojar repositorios públicos de forma gratuita y colaborar con otros desarrolladores en proyectos de código abierto. También ofrece repositorios privados para proyectos comerciales, pero estos suelen tener un costo.
5. **Paso 5: Git - Comandos de Línea de Comandos:**
   * Para trabajar con Git, debes usar comandos en la línea de comandos de tu computadora. Git proporciona una interfaz de línea de comandos poderosa para gestionar repositorios.
6. **Paso 6: GitHub - Interfaz Web:**
   * GitHub ofrece una interfaz web que facilita la gestión de repositorios, seguimiento de problemas, revisión de código y colaboración con otros desarrolladores. No es necesario utilizar la línea de comandos para realizar todas estas tareas.

En resumen, Git es un sistema de control de versiones utilizado localmente en tu computadora para rastrear cambios en proyectos, mientras que GitHub es una plataforma en línea que aloja repositorios Git en la nube y proporciona herramientas adicionales para la colaboración y la administración de proyectos.

GitHub es una plataforma de desarrollo colaborativo que permite a los desarrolladores trabajar juntos en proyectos de código abierto y privados. A continuación, se presenta una guía paso a paso sobre cómo crear una cuenta en GitHub:

1. **Paso 1: Acceder al Sitio Web de GitHub:**
   * Abre tu navegador web y navega a la página de GitHub en [https://github.com](https://github.com/).
2. **Paso 2: Iniciar el Proceso de Registro:**
   * En la página de inicio de GitHub, haz clic en el botón "Sign up" o "Registrarse".
3. **Paso 3: Completar el Formulario de Registro:**
   * Rellena el formulario de registro con la siguiente información:
     + Nombre de usuario: Elige un nombre de usuario único para tu cuenta de GitHub.
     + Correo electrónico: Proporciona una dirección de correo electrónico válida.
     + Contraseña: Crea una contraseña segura.
4. **Paso 4: Verificar que no eres un robot:**
   * Completa cualquier proceso de verificación de seguridad que GitHub pueda requerir, como completar un captcha o resolver un rompecabezas.
5. **Paso 5: Revisar y Aceptar los Términos de Servicio:**
   * Lee los términos de servicio y la política de privacidad de GitHub. Asegúrate de entender y aceptar los términos antes de continuar.
6. **Paso 6: Completar el Registro:**
   * Haz clic en el botón "Create account" o "Crear cuenta" para finalizar el registro de tu cuenta.
7. **Paso 7: Verificar tu Correo Electrónico:**
   * GitHub te enviará un correo electrónico de verificación a la dirección proporcionada. Abre el correo electrónico y sigue las instrucciones para verificar tu dirección de correo.

Una vez completados estos pasos, habrás creado con éxito una cuenta en GitHub y estarás listo para comenzar a colaborar en proyectos, crear repositorios y contribuir al desarrollo de software.

Clonar un repositorio remoto en GitHub es el proceso de obtener una copia local de un proyecto alojado en GitHub en tu computadora. A continuación, se presenta una guía paso a paso:

1. **Paso 1: Acceder al Repositorio en GitHub:**
   * Navega a la página del repositorio que deseas clonar en GitHub. Por ejemplo, puedes acceder a un repositorio en la dirección:

<https://github.com/usuario/nombre-del-repositorio>

1. **Paso 2: Encontrar la URL del Repositorio:**
   * Haz clic en el botón verde "Code" en la parte superior derecha del repositorio. Esto desplegará una ventana emergente que muestra la URL del repositorio. Copia la URL haciendo clic en el icono de copiar o utilizando Ctrl+C (o Command+C en Mac).
2. **Paso 3: Abrir la Terminal o Línea de Comandos:**
   * Abre una terminal o línea de comandos en tu computadora. Asegúrate de que te encuentres en la ubicación donde deseas clonar el repositorio.
3. **Paso 4: Clonar el Repositorio:**
   * Ejecuta el siguiente comando en la terminal, reemplazando "URL-del-repositorio" con la URL que copiaste en el paso 2:

git clone URL-del-repositorio

* + Por ejemplo, si la URL es "<https://github.com/usuario/proyecto.git>", el comando sería:

git clone <https://github.com/usuario/proyecto.git>

* + Esto descargará una copia completa del repositorio en tu computadora y creará una carpeta con el nombre del repositorio en tu ubicación actual.

Una vez que hayas completado estos pasos, habrás clonado con éxito un repositorio remoto de GitHub en tu computadora. Ahora puedes trabajar en el proyecto localmente y realizar cambios que luego podrás enviar de vuelta al repositorio remoto si eres un colaborador o puedes hacer un fork y enviar solicitudes de extracción si deseas contribuir a proyectos de código abierto.

En GitHub, puedes colaborar en proyectos compartiendo y gestionando cambios. Aquí tienes una guía detallada sobre cómo realizar operaciones esenciales:

**Push (Subir Cambios)**

1. **Paso 1: Cambios Locales:**
   * Realiza cambios en los archivos de tu repositorio local usando Git. Asegúrate de confirmar (commit) tus cambios.
2. **Paso 2: Push a GitHub:**
   * Para subir tus cambios a GitHub, utiliza el comando:

git push origin nombre-de-la-rama

* + Reemplaza "nombre-de-la-rama" con el nombre de la rama en la que trabajas. Esto enviará tus cambios al repositorio remoto en GitHub.

**Pull (Bajar Cambios)**

1. **Paso 1: Cambios en GitHub:**
   * Otro colaborador realiza cambios en el repositorio remoto en GitHub.
2. **Paso 2: Pull de GitHub:**
   * Para obtener los cambios del repositorio remoto a tu repositorio local, utiliza el comando:

git pull origin nombre-de-la-rama

* + Reemplaza "nombre-de-la-rama" con el nombre de la rama en la que estás trabajando. Esto descargará los cambios desde GitHub y los fusionará con tus archivos locales.