**Práctica: Introducción a Git - 40 Ejercicios Básicos**

**Ejercicio 1: Verificar la Instalación de Git**

Antes de comenzar, asegúrate de que Git está instalado en tu sistema.

**Instrucciones Detalladas:**

1. Abre una terminal o línea de comandos en tu computadora.
2. Escribe el comando que te permitirá verificar si Git está instalado y cuál es su versión actual.
3. Si Git no está instalado, investiga cómo instalarlo en tu sistema operativo (por ejemplo, Windows, macOS o Linux).

**Preguntas a responder:**

* ¿Qué comando debes usar para verificar la versión de Git?
* Git --version
* Si no tienes Git instalado, ¿qué opciones tienes para instalarlo?

En terminal escribimos :

Git init

Luego:

FORMA DE INSTALARLO POR CONSOLA:

$ yum install curl-devel expat-devel gettext-devel \

openssl-devel zlib-devel

$ apt-get install libcurl4-gnutls-dev libexpat1-dev gettext \

libz-dev libssl-dev

LUEGO:

$ tar -zxf git-2.0.0.tar.gz

$ cd git-2.0.0

$ make configure

$ ./configure --prefix=/usr

$ make all doc info

$ sudo make install install-doc install-html install-info

FINALMENTE:

Una vez hecho esto, también puedes obtener Git, a través del propio Git, para futuras actualizaciones:

* $ git clone git://git.kernel.org/pub/scm/git/git.git

**Ejercicio 2: Crear un Directorio de Trabajo**

Crea un directorio donde trabajarás en este proyecto.

**Instrucciones Detalladas:**

1. En tu terminal, navega hasta la ubicación donde deseas crear el directorio. Por ejemplo, podrías crearlo en tu carpeta "Documentos".
2. Usa un comando para crear un nuevo directorio llamado mi-proyecto-git.
3. Cambia al directorio recién creado para comenzar a trabajar en él.

**Preguntas a responder:**

* ¿Qué comando usas para crear un nuevo directorio?
* Echo “texto random” > nombredearchivo.txt
* O
* Git add .\nombredearchivo.txt
* ¿Cómo puedes cambiar a un directorio específico desde la terminal?
* Cd ./nombre del directorio

**Ejercicio 3: Inicializar un Repositorio Git**

Convierte tu directorio de trabajo en un repositorio Git.

**Instrucciones Detalladas:**

1. Asegúrate de estar dentro del directorio mi-proyecto-git que creaste en el ejercicio anterior.
2. Escribe el comando necesario para inicializar Git en este directorio. Esto creará un directorio oculto llamado .git donde Git almacenará toda la información del repositorio.

**Preguntas a responder:**

* ¿Qué comando usas para inicializar Git en un directorio?
* Git init
* ¿Cómo puedes confirmar que Git ha sido inicializado correctamente?
* Git --version

**Ejercicio 4: Comprobar el Estado del Repositorio**

Verifica el estado actual de tu repositorio.

**Instrucciones Detalladas:**

1. Dentro del directorio mi-proyecto-git, escribe el comando que te mostrará el estado actual del repositorio.
2. Observa la salida del comando y asegúrate de que no haya archivos pendientes de seguimiento ni cambios no guardados.

**Preguntas a responder:**

* ¿Qué comando usas para comprobar el estado del repositorio?
* Git status
* ¿Qué significa cuando Git dice que no hay archivos rastreados?
* Luego de un ls hará una lista de lo que contenga, los rastreados son los que se incluyen

**Ejercicio 5: Crear un Archivo y Verificar el Estado**

Crea un archivo en tu repositorio y verifica cómo cambia el estado del mismo.

**Instrucciones Detalladas:**

1. Crea un archivo llamado README.md dentro de tu directorio mi-proyecto-git. Puedes hacerlo directamente desde la terminal o usando un editor de texto.
2. Vuelve a comprobar el estado del repositorio para ver cómo Git reconoce este nuevo archivo.

**Preguntas a responder:**

* ¿Qué comando usas para crear un archivo nuevo desde la terminal?
* echo > archivo.txt
* ¿Qué diferencia observas en el estado del repositorio después de crear el archivo?
* Que demora un milisegundo y que me permite luego listar para comprobar que lo creó

**Ejercicio 6: Añadir un Archivo al Área de Preparación**

Prepara el archivo README.md para ser guardado en el próximo commit.

**Instrucciones Detalladas:**

1. Usa el comando adecuado para añadir el archivo README.md al área de preparación (staging area).
2. Vuelve a comprobar el estado del repositorio para confirmar que el archivo ha sido añadido correctamente.

**Preguntas a responder:**

* ¿Qué comando usas para añadir un archivo al área de preparación?
* Git add README.md
* Luego lo confirmo con
* Git status
* ¿Qué significa añadir un archivo al área de preparación?
* Que nos guarda ese archivo en el repositorio

**Ejercicio 7: Realizar un Commit**

Guarda los cambios realizados en el repositorio mediante un commit.

**Instrucciones Detalladas:**

1. Escribe el comando necesario para realizar un commit. Al hacerlo, incluye un mensaje claro que describa lo que has hecho. Por ejemplo: "Añadido archivo README.md".
2. Después del commit, verifica nuevamente el estado del repositorio para asegurarte de que no quedan cambios pendientes.

**Preguntas a responder:**

* ¿Qué comando usas para realizar un commit?
* Uso: Git commit "Añadido archivo"
* ¿Por qué es importante incluir un mensaje claro en cada commit?
* Para confirmar que he realizado todo correctamente

**Ejercicio 8: Ver el Historial de Commits**

Consulta el historial de commits realizados en tu repositorio.

**Instrucciones Detalladas:**

1. Escribe el comando que te permitirá ver el historial de commits realizados en tu repositorio.
2. Observa la salida del comando y asegúrate de que puedas identificar el commit que acabas de realizar.

**Preguntas a responder:**

* ¿Qué comando usas para ver el historial de commits?
* Usamos Git log
* ¿Qué información muestra el historial de commits?
* En donde está ubicado, el autor y la fecha completa de realización

**Ejercicio 9: Crear una Nueva Rama**

Crea una nueva rama para trabajar en una característica específica.

**Instrucciones Detalladas:**

1. Usa el comando adecuado para crear una nueva rama llamada nueva-caracteristica.
2. No te preocupes por moverte a esta rama todavía; solo crea la rama por ahora.

**Preguntas a responder:**

* ¿Qué comando usas para crear una nueva rama en Git?
* Git Branch nueva-caracteristica
* ¿Qué diferencia hay entre crear una rama y moverte a ella?
* Que se usa otro comando
* Git checkout nueva-caracteristica

**Ejercicio 10: Listar Todas las Ramas**

Enumera todas las ramas existentes en tu repositorio.

**Instrucciones Detalladas:**

1. Escribe el comando que te permitirá listar todas las ramas disponibles en tu repositorio.
2. Observa la salida del comando y asegúrate de que puedes identificar tanto la rama principal (main o master) como la rama que acabas de crear.

**Preguntas a responder:**

* ¿Qué comando usas para listar todas las ramas en Git?
* Git Branch -a
* ¿Cómo puedes identificar en qué rama estás trabajando actualmente?
* Solo con escribir: git Branch, marcará con un \* la que estéis usando

**Ejercicio 11: Cambiar a una Rama Existente**

Mueve tu trabajo a la rama nueva-caracteristica que creaste en el ejercicio anterior.

**Instrucciones Detalladas:**

1. Usa el comando adecuado para cambiar a la rama nueva-caracteristica.
2. Verifica que estás trabajando en la rama correcta usando un comando que te muestre en qué rama estás actualmente.

**Preguntas a responder:**

* ¿Qué comando usas para cambiar de rama?
* Git checkout “nombre de la rama” y git branch
* ¿Cómo puedes confirmar en qué rama estás trabajando?
* Git Branch -a, en la que esté trabajando estará seleccionada con un \* y será la primera en la lista y también el comando: git branch

**Ejercicio 12: Crear y Cambiar a una Nueva Rama en un Solo Paso**

Crea una nueva rama llamada mejoras-ui y cambia a ella inmediatamente.

**Instrucciones Detalladas:**

1. Usa un comando que permita crear y cambiar a una nueva rama en un solo paso.
2. Verifica que ahora estás trabajando en la rama mejoras-ui.

**Preguntas a responder:**

* ¿Qué comando permite crear y cambiar a una nueva rama en un solo paso?
* Git switch -c mejoras-ui
* ¿Cuál es la ventaja de usar este comando?
* Creas y cambias de rama en una sola línea

**Ejercicio 13: Añadir Archivos al Área de Preparación con un Comando Abreviado**

Prepara todos los archivos nuevos o modificados en tu directorio para ser guardados.

**Instrucciones Detalladas:**

1. Modifica o crea un archivo nuevo en tu directorio.
2. Usa un comando abreviado para añadir todos los cambios (nuevos, modificados o eliminados) al área de preparación.

**Preguntas a responder:**

* ¿Qué comando abreviado puedes usar para añadir todos los cambios al área de preparación?
* Git add
* ¿Qué ventajas tiene usar este comando en lugar de añadir archivos individualmente?
* Es más práctico y directamente crea el file necesario

**Ejercicio 14: Realizar un Commit con un Mensaje Descriptivo**

Guarda los cambios realizados en la rama mejoras-ui mediante un commit.

**Instrucciones Detalladas:**

1. Escribe un mensaje claro que describa los cambios realizados. Por ejemplo: "Actualización inicial de la interfaz de usuario".
2. Verifica que el commit fue realizado correctamente comprobando el historial de commits.

**Preguntas a responder:**

* ¿Qué comando usas para realizar un commit?
* Git commit "texto"
* ¿Por qué es importante escribir mensajes descriptivos en los commits?
* Para confirmar que los archivos fueron creados correctamente

**Ejercicio 15: Fusionar Ramas**

Fusiona los cambios realizados en la rama mejoras-ui con la rama principal (main o master).

**Instrucciones Detalladas:**

1. Cambia a la rama principal (main o master).
2. Usa el comando adecuado para fusionar los cambios de la rama mejoras-ui en la rama principal.

**Preguntas a responder:**

* ¿Qué comando usas para fusionar ramas en Git?
* git merge mejoras-ui
* ¿Qué sucede si hay conflictos durante la fusión?
* No fusionan

**Ejercicio 16: Resolver Conflictos entre Ramas**

Introduce un conflicto intencional al modificar el mismo archivo en dos ramas diferentes y resuélvelo.

**Instrucciones Detalladas:**

1. Crea una nueva rama llamada conflicto-test.
2. Modifica un archivo en ambas ramas (main y conflicto-test) de manera que introduzcas un conflicto.
3. Intenta fusionar la rama conflicto-test con la rama principal y resuelve el conflicto manualmente.

**Preguntas a responder:**

* ¿Qué ocurre cuando Git detecta un conflicto durante la fusión?
* ¿Cómo puedes resolver un conflicto en Git?

**Ejercicio 17: Crear un Archivo .gitignore**

Crea un archivo .gitignore para excluir ciertos tipos de archivos de ser rastreados por Git.

**Instrucciones Detalladas:**

1. Crea un archivo llamado .gitignore en la raíz de tu repositorio.
2. Agrega entradas para ignorar archivos como \*.log, \*.tmp y cualquier otro tipo de archivo temporal o irrelevante.

**Preguntas a responder:**

* ¿Qué es un archivo .gitignore y para qué sirve?
* ¿Cómo puedes especificar qué tipos de archivos deben ser ignorados?

**Ejercicio 18: Configurar Git para Ignorar Cambios en un Archivo Específico**

Ignora los cambios en un archivo específico sin eliminarlo del repositorio.

**Instrucciones Detalladas:**

1. Selecciona un archivo en tu repositorio que no quieras que Git rastree sus cambios.
2. Usa un comando especial para indicarle a Git que ignore los cambios en ese archivo.

**Preguntas a responder:**

* ¿Cómo puedes hacer que Git ignore los cambios en un archivo específico sin eliminarlo del repositorio?
* ¿En qué situaciones sería útil esta técnica?

**Ejercicio 19: Clonar un Repositorio Remoto**

Clona un repositorio remoto existente en tu computadora local.

**Instrucciones Detalladas:**

1. Busca un repositorio público en GitHub o GitLab.
2. Usa el comando adecuado para clonar el repositorio en tu computadora.

**Preguntas a responder:**

* ¿Qué comando usas para clonar un repositorio remoto?
* ¿Qué diferencia hay entre clonar un repositorio y crear uno nuevo?

**Ejercicio 20: Conectarte a GitHub con SSH**

Configura la autenticación SSH para conectarte a GitHub de forma segura.

**Instrucciones Detalladas:**

1. Genera una clave SSH en tu computadora si aún no tienes una.
2. Copia la clave pública generada y agrégala a tu cuenta de GitHub.

**Preguntas a responder:**

* ¿Qué es SSH y por qué es importante para trabajar con GitHub?
* ¿Cómo puedes generar una clave SSH en tu computadora?

**Ejercicio 21: Subir Cambios Locales a GitHub**

Sube tus cambios locales al repositorio remoto en GitHub.

**Instrucciones Detalladas:**

1. Crea un repositorio vacío en GitHub.
2. Asocia tu repositorio local con el remoto usando el comando adecuado.
3. Sube tus cambios locales al repositorio remoto.

**Preguntas a responder:**

* ¿Qué comando usas para asociar un repositorio local con uno remoto?
* ¿Qué comando usas para subir cambios locales a un repositorio remoto?

**Ejercicio 22: Actualizar tu Repositorio Local desde GitHub**

Descarga los últimos cambios del repositorio remoto en GitHub.

**Instrucciones Detalladas:**

1. Supón que alguien más ha realizado cambios en el repositorio remoto.
2. Usa el comando adecuado para actualizar tu repositorio local con los cambios remotos.

**Preguntas a responder:**

* ¿Qué comando usas para descargar cambios del repositorio remoto?
* ¿Qué pasa si hay conflictos al sincronizar los cambios?

**Ejercicio 23: Crear una Rama en GitHub**

Crea una nueva rama en tu repositorio remoto en GitHub.

**Instrucciones Detalladas:**

1. Crea una nueva rama local llamada feature-nueva.
2. Sube esta rama al repositorio remoto en GitHub.

**Preguntas a responder:**

* ¿Cómo puedes subir una rama local a un repositorio remoto?
* ¿Cómo puedes verificar que la rama ha sido creada correctamente en GitHub?

**Ejercicio 24: Eliminar una Rama Local**

Elimina una rama local que ya no necesitas.

**Instrucciones Detalladas:**

1. Selecciona una rama local que ya no uses.
2. Usa el comando adecuado para eliminarla.

**Preguntas a responder:**

* ¿Qué comando usas para eliminar una rama local?
* ¿Es posible recuperar una rama eliminada?

**Ejercicio 25: Eliminar una Rama Remota**

Elimina una rama remota que ya no necesitas.

**Instrucciones Detalladas:**

1. Selecciona una rama remota que ya no uses.
2. Usa el comando adecuado para eliminarla desde el repositorio remoto.

**Preguntas a responder:**

* ¿Cómo puedes eliminar una rama remota?
* ¿Qué precauciones debes tomar antes de eliminar una rama remota?

**Ejercicio 26: Crear una Pull Request en GitHub**

Solicita que tus cambios en una rama sean revisados y fusionados en la rama principal.

**Instrucciones Detalladas:**

1. Trabaja en una rama nueva y realiza algunos cambios.
2. Sube tus cambios al repositorio remoto.
3. Crea una pull request en GitHub para solicitar la fusión de tu rama con la rama principal.

**Preguntas a responder:**

* ¿Qué es una pull request y para qué sirve?
* ¿Qué información debes incluir en una pull request?

**Ejercicio 27: Revisar y Comentar una Pull Request**

Revisa una pull request existente y deja comentarios sobre los cambios propuestos.

**Instrucciones Detalladas:**

1. Encuentra una pull request en tu repositorio de GitHub.
2. Lee los cambios propuestos y deja comentarios relevantes.

**Preguntas a responder:**

* ¿Cómo puedes revisar una pull request en GitHub?
* ¿Qué tipo de comentarios son útiles en una revisión?

**Ejercicio 28: Fusión Automática de una Pull Request**

Fusiona automáticamente una pull request que no tiene conflictos.

**Instrucciones Detalladas:**

1. Encuentra una pull request sin conflictos en tu repositorio.
2. Usa la interfaz de GitHub para fusionarla automáticamente con la rama principal.

**Preguntas a responder:**

* ¿Cuándo puedes fusionar una pull request automáticamente?
* ¿Qué sucede después de fusionar una pull request?

**Ejercicio 29: Descartar Cambios Locales**

Descarta accidentalmente cambios locales que no quieres guardar.

**Instrucciones Detalladas:**

1. Modifica un archivo local sin añadirlo al área de preparación.
2. Usa el comando adecuado para descartar estos cambios y restaurar el archivo a su estado anterior.

**Preguntas a responder:**

* ¿Qué comando usas para descartar cambios locales?
* ¿Cuándo es útil descartar cambios locales?

**Ejercicio 30: Restaurar un Commit Anterior**

Restaura tu repositorio a un commit anterior.

**Instrucciones Detalladas:**

1. Identifica un commit anterior en el historial de commits.
2. Usa el comando adecuado para restaurar tu repositorio a ese commit específico.

**Preguntas a responder:**

* ¿Qué comando usas para restaurar un commit anterior?
* ¿Qué precauciones debes tomar al restaurar un commit anterior?

**Ejercicio 31: Crear un Tag para Marcar una Versión**

Crea un tag para marcar una versión específica de tu proyecto.

**Instrucciones Detalladas:**

1. Elige un commit en el historial de tu repositorio que quieras marcar como una versión importante (por ejemplo, v1.0).
2. Usa el comando adecuado para crear un tag asociado a ese commit.
3. Verifica que el tag ha sido creado correctamente.

**Preguntas a responder:**

* ¿Qué es un tag en Git y para qué sirve?
* ¿Cómo puedes crear un tag asociado a un commit específico?

**Ejercicio 32: Listar Todos los Tags en tu Repositorio**

Enumera todos los tags existentes en tu repositorio.

**Instrucciones Detalladas:**

1. Escribe el comando que te permitirá listar todos los tags disponibles en tu repositorio.
2. Observa la salida del comando y asegúrate de que puedas identificar los tags que has creado.

**Preguntas a responder:**

* ¿Qué comando usas para listar todos los tags en Git?
* ¿Por qué es útil tener tags en un proyecto?

**Ejercicio 33: Subir Tags al Repositorio Remoto**

Sube los tags locales al repositorio remoto en GitHub.

**Instrucciones Detalladas:**

1. Selecciona uno o más tags locales que quieras compartir con el repositorio remoto.
2. Usa el comando adecuado para subir estos tags al repositorio remoto.

**Preguntas a responder:**

* ¿Cómo puedes subir tags locales a un repositorio remoto?
* ¿Qué pasa si intentas subir un tag que ya existe en el repositorio remoto?

**Ejercicio 34: Descargar Tags del Repositorio Remoto**

Descarga los tags existentes desde el repositorio remoto.

**Instrucciones Detalladas:**

1. Asegúrate de estar conectado al repositorio remoto en GitHub.
2. Usa el comando adecuado para descargar todos los tags disponibles en el repositorio remoto.

**Preguntas a responder:**

* ¿Qué comando usas para descargar tags desde un repositorio remoto?
* ¿Cuándo sería útil descargar tags desde un repositorio remoto?

**Ejercicio 35: Revertir un Commit**

Revierte un commit específico sin perder los commits posteriores.

**Instrucciones Detalladas:**

1. Identifica un commit en el historial que quieras revertir.
2. Usa el comando adecuado para revertir este commit, creando un nuevo commit que deshaga los cambios realizados.

**Preguntas a responder:**

* ¿Qué diferencia hay entre revertir un commit y eliminarlo del historial?
* ¿Por qué es útil usar revert en lugar de simplemente eliminar un commit?

**Ejercicio 36: Eliminar un Commit del Historial**

Elimina un commit específico del historial de tu repositorio.

**Instrucciones Detalladas:**

1. Identifica un commit en el historial que quieras eliminar permanentemente.
2. Usa un comando avanzado para modificar el historial y eliminar este commit.
3. Ten cuidado, ya que esta operación puede afectar a otros desarrolladores si el repositorio es compartido.

**Preguntas a responder:**

* ¿Qué comando puedes usar para eliminar un commit del historial?
* ¿Cuándo es seguro usar este comando en un repositorio compartido?

**Ejercicio 37: Comparar Ramas**

Compara los cambios entre dos ramas diferentes.

**Instrucciones Detalladas:**

1. Selecciona dos ramas en tu repositorio (por ejemplo, main y feature-nueva).
2. Usa el comando adecuado para comparar los cambios entre estas dos ramas.

**Preguntas a responder:**

* ¿Qué comando usas para comparar ramas en Git?
* ¿Qué información muestra esta comparación?

**Ejercicio 38: Deshacer Cambios en un Archivo Específico**

Deshaz accidentalmente los cambios realizados en un archivo específico sin afectar otros archivos.

**Instrucciones Detalladas:**

1. Modifica un archivo en tu repositorio sin añadirlo al área de preparación.
2. Usa el comando adecuado para restaurar este archivo a su estado anterior sin afectar otros archivos modificados.

**Preguntas a responder:**

* ¿Cómo puedes deshacer cambios en un archivo específico?
* ¿Qué ventajas tiene esta técnica sobre descartar todos los cambios locales?

**Ejercicio 39: Usar Alias para Simplificar Comandos**

Crea un alias para simplificar comandos frecuentes en Git.

**Instrucciones Detalladas:**

1. Selecciona un comando Git que uses con frecuencia (por ejemplo, git status).
2. Crea un alias personalizado para este comando usando el comando de configuración de Git.
3. Prueba tu nuevo alias para asegurarte de que funciona correctamente.

**Preguntas a responder:**

* ¿Qué es un alias en Git y para qué sirve?
* ¿Cómo puedes crear un alias personalizado para un comando Git?