UNIVERSIDAD TECNOLOGICA NACIONAL

TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

PROGRAMACIÓN I

Trabajo Práctico 2: Git y GitHub

Estudiante: Pérez Campos, Joaquín

Comisión 8

UNIVERSIDAD TECNOLOGICA NACIONAL

Actividades

1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :

- ¿Qué es GitHub?
- ¿Cómo crear un repositorio en GitHub?
- ¿Cómo crear una rama en Git?
- ¿Cómo cambiar a una rama en Git?
- ¿Cómo fusionar ramas en Git?
- ¿Cómo crear un commit en Git?
- ¿Cómo enviar un commit a GitHub?
- ¿Qué es un repositorio remoto?
- ¿Cómo agregar un repositorio remoto a Git?
- ¿Cómo empujar cambios a un repositorio remoto?
- ¿Cómo tirar de cambios de un repositorio remoto?
- ¿Qué es un fork de repositorio?
- ¿Cómo crear un fork de un repositorio?
- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
- ¿Cómo aceptar una solicitud de extracción?
- ¿Qué es una etiqueta en Git?
- ¿Cómo crear una etiqueta en Git?
- ¿Cómo enviar una etiqueta a GitHub?
- ¿Qué es un historial de Git?
- ¿Cómo ver el historial de Git?
- ¿Cómo buscar en el historial de Git?
- ¿Cómo borrar el historial de Git?
- ¿Qué es un repositorio privado en GitHub?
- ¿Cómo crear un repositorio privado en GitHub?
- ¿Cómo invitar a alguien a un repositorio privado en GitHub?
- ¿Qué es un repositorio público en GitHub?
- ¿Cómo crear un repositorio público en GitHub?
- ¿Cómo compartir un repositorio público en GitHub?

2) Realizar la siguiente actividad:

- Crear un repositorio.
    - Dale un nombre al repositorio.
    - Elije el repositorio sea público.
    - Inicializa el repositorio con un archivo.
- Agregando un Archivo
    - Crea un archivo simple, por ejemplo, "mi-archivo.txt".

o Realiza los comandos git add . y git commit -m "Agregando mi-archivo.txt" en la línea de comandos.

o Sube los cambios al repositorio en GitHub con git push origin main (o el nombre de la rama correspondiente).

• Creando Branchs

o Crear una Branch

o Realizar cambios o agregar un archivo

o Subir la Branch

3) Realizar la siguiente actividad:

Paso 1: Crear un repositorio en GitHub

• Ve a GitHub e inicia sesión en tu cuenta.

• Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.

• Asigna un nombre al repositorio, por ejemplo, conflict-exercise.

• Opcionalmente, añade una descripción.

• Marca la opción "Initialize this repository with a README".

• Haz clic en "Create repository".

Paso 2: Clonar el repositorio a tu máquina local

• Copia la URL del repositorio (usualmente algo como https://github.com/tuusuario/conflict-exercise.git).

• Abre la terminal o línea de comandos en tu máquina.

• Clona el repositorio usando el comando: git clone https://github.com/tuusuario/conflict-exercise.git

• Entra en el directorio del repositorio: cd conflict-exercise

Paso 3: Crear una nueva rama y editar un archivo

• Crea una nueva rama llamada feature-branch: git checkout -b feature-branch

• Abre el archivo README.md en un editor de texto y añade una línea nueva, por ejemplo: Este es un cambio en la feature branch.

• Guarda los cambios y haz un commit: git add README.md git commit -m "Added a line in feature-branch"

Paso 4: Volver a la rama principal y editar el mismo archivo

• Cambia de vuelta a la rama principal (main): git checkout main

• Edita el archivo README.md de nuevo, añadiendo una línea diferente: Este es un cambio en la main branch.

• Guarda los cambios y haz un commit: git add README.md git commit -m "Added a line in main branch"

Paso 5: Hacer un merge y generar un conflicto

• Intenta hacer un merge de la feature-branch en la rama main: git merge feature-branch

• Se generará un conflicto porque ambos cambios afectan la misma línea del archivo README.md.

Paso 6: Resolver el conflicto

• Abre el archivo README.md en tu editor de texto. Verás algo similar a esto: <<<<<<< HEAD Este es un cambio en la main branch. ======= Este es un cambio en la feature branch. >>>>>>> feature-branch

• Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.

• Edita el archivo para resolver el conflicto y guarda los cambios(Se debe borrar lo marcado en verde en el archivo donde estés solucionando el conflicto. Y se debe borrar la parte del texto que no se quiera dejar).

• Añade el archivo resuelto y completa el merge: git add README.md git commit -m "Resolved merge conflict"

Paso 7: Subir los cambios a GitHub

• Sube los cambios de la rama main al repositorio remoto en GitHub: git push origin main

• También sube la feature-branch si deseas: git push origin feature-branch

Paso 8: Verificar en GitHub

• Ve a tu repositorio en GitHub y revisa el archivo README.md para confirmar que los cambios se han subido correctamente.

• Puedes revisar el historial de commits para ver el conflicto y su resolución.

UNIVERSIDAD TECNOLOGICA NACIONAL

Respuestas

Actividad 1

- GitHub es una plataforma en línea que permite almacenar proyectos, que usan el sistema de control de versiones Git, en la nube, lo que permite el trabajo colaborativo entre desarrolladores ubicados en cualquier parte del mundo. Es el servicio más popular para compartir y colaborar en proyectos de software, de código abierto o privado. También ofrece herramientas para organizar, discutir, revisar y aprobar cambios en el repositorio, lo que lo convierte en una solución integral para el desarrollo de software moderno.
- Para crear un repositorio nuevo en GitHub, podemos hacerlo desde un navegador o desde la línea de comandos de GitHub CLI. En ambos casos necesitamos tener una cuenta en la plataforma GitHub.
    - Desde un navegador: Luego de acceder con nuestro usuario y contraseña, seleccionar el signo + en la esquina superior derecha y hacer clic en *"New repository"* . Asignar un nombre al repositorio (se recomienda un nombre corto y fácil de recordar), opcionalmente podemos agregar una descripción al repositorio. Elegir la visibilidad del repositorio (público o privado) según corresponda al proyecto. Se recomienda crear un archivo README (Léame) como presentación del proyecto, con su descripción, instrucciones de uso, guía para la contribución, información de la licencia y presentación personal del desarrollador. Finalmente, hacer clic en *"Create repository"*
    - Desde GitHub CLI: Primeramente, navegar mediante comandos al directorio en donde queremos crear un clon local del repositorio. Usar el subcomando `gh repo create` seleccionar *Create a new repository on GitHub from scratch* (Crear un nuevo repositorio en GitHub desde cero) y escribir el nombre para el repositorio. Seguir los mensajes interactivos para clonar el repositorio localmente, configurar el tipo de visibilidad, agregar un archivo README, etc.
- Para crear una nueva rama con Git y asignarle un nombre, usamos el comando `git branch nombre_de_rama`
- Para cambiar de rama utilizamos el comando `git checkout` seguido del nombre de la rama a la que queremos movernos, por ejemplo: `git checkout nueva_funcion`
- Para fusionar ramas en Git necesitamos estar posicionado en la rama donde se quiere integrar los cambios, por ejemplo si queremos integrar la rama "nueva_uncion" a la rama principal "main", debemos estar posicionados en la rama "main". Para eso utilizamos el comando `git checkout main`, luego podemos verificar el estado del repositorio con `git status`, y finalmente hacer la fusión con el comando `git merge nueva_funcion`
- Para crear un *commit* y enviarlo a GitHub, por ejemplo, uno en el que agreguemos el archivo README al repositorio, primeramente debemos chequear el estado para verificar si hay archivos nuevos o modificados con el comando `git status`, luego

agregar el o los archivos al staging con el comando `git add README.md` y finalmente hacer el *commit* con el comando `git commit -m "Agrega README.md"`, agregando un mensaje descriptivo con la etiqueta `-m`.

- Un repositorio remoto es una versión de un proyecto, hospedada en internet o en cualquier otra red. Puede compartirse con otras personas para trabajar colaborativamente, enviar o traer datos de ellos cada vez que se necesite, crear ramas, fusionarlas, crear copias completas del repositorio (*fork*), etc.
- Teniendo un repositorio creado en GitHub, debemos copiar la URL del repositorio, como primer paso para poder vincularlo. Luego, desde la terminal de Git y posicionados dentro de la carpeta raíz del proyecto, ejecutar el comando `git remote add nombre_remoto url_remoto` , donde reemplazamos `nombre_remoto` con el nombre que queramos darle al repositorio remoto y `url_remoto` con la URL del repositorio en GitHub, en formato HTML o SSH, por ejemplo: `git remote add origin https://github.com/user/repo.git`
- Para empujar o subir los cambios a un repositorio remoto, debemos asegurarnos de tener los cambios localmente en un *commit* y luego usar el comando `git push` con el nombre que asignamos al repositorio remoto y el nombre de la rama, por ejemplo `git push origin main`.
- Para tirar o descargar los cambios en un repositorio remoto con Git, se utiliza el comando git pull, que combina los comandos git fetch (para obtener las actualizaciones sin fusionarlas) y git merge (para fusionar los cambios). Desde la terminal y posicionados en el directorio del repositorio local, ejecutamos el comando `git pull origin main`, donde `origin` es el nombre del repositorio remoto y `main` el nombre de la rama que se desea actualizar. En algunos casos, este comando puede generar conflictos si hay cambios en la rama local y remota que afectan a los mismos archivos. Estos conflictos se deberán resolver manualmente editando los archivos involucrados, luego agregarlos al staging con `git add nombre_de_archivo` y confirmar los cambios con `git commit`.
- Un *fork* es una copia completa de un repositorio de GitHub que se crea dentro de nuestra cuenta. Permite probar cambios sin afectar el proyecto original, proponer mejoras enviando los cambios mediante un *pull request* y trabajar con un repositorio a los que no tengamos permisos directos.
- Para crear un *fork* de un repositorio público en GitHub, debemos ingresar al mismo, hacer clic en el botón *Fork* y elegir nuestra cuenta en GitHub donde se creará la copia. Una vez creado, podemos copiar la URL del *fork* para clonarlo en nuestro repositorio local con el comando `git clone url_fork` y comenzar a trabajar sobre la copia.
- Para poder realizar un *pull request* debemos tener actualizados en el repositorio remoto del *fork,* todos los cambios que realizamos (`git add`, `git commit` y `git push`). Luego, debemos navegar en GitHub hasta el repositorio remoto del *fork* que

creamos. En esa página vamos a ver un botón verde que dice *"Compare & pull request"*, al hacer clic nos va a llevar a la página donde vamos a acceder a la información para realizar la *pull request*. Aquí debemos seleccionar la rama "target u objetivo" a la que queremos solicitar que se fusionen los cambios que realizamos (en el menú desplegable que dice *base: nombre_de_rama*) y la rama "source o fuente" en la que realizamos los cambios en nuestro *fork* (en el menú desplegable que dice *compare: nombre_de_rama*). Luego de asegurarnos de haber definido las ramas "target" y "source", debemos añadir un título y una descripción que explique los cambios que hicimos. Podemos guardar un borrador de la solicitud o enviarla directamente, con el botón *"Create pull request"*, para que los cambios sean revisados y luego de ser aprobados pueden fusionarse con la rama "target" del repositorio original.

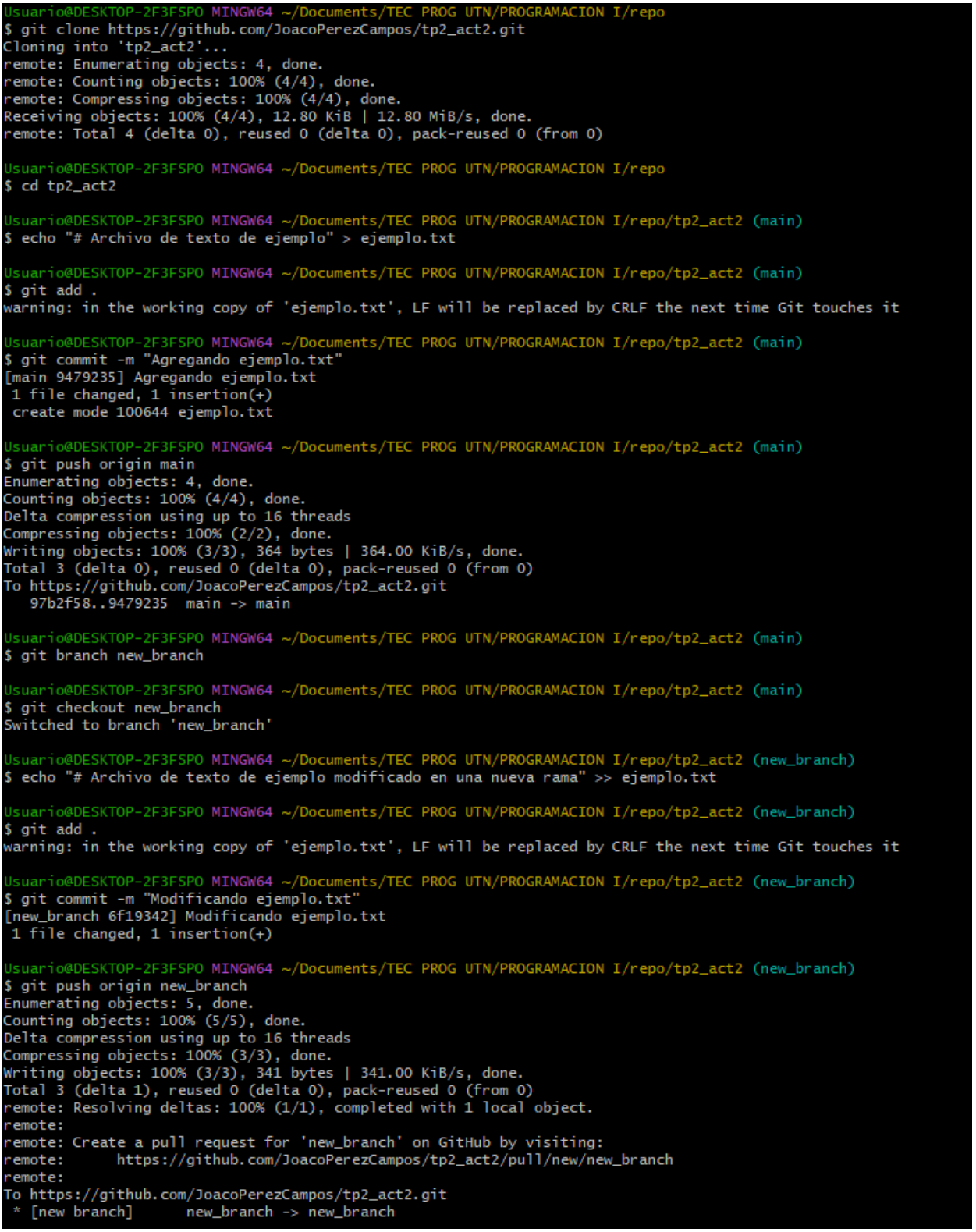
- Para aceptar una *pull request* debemos ir hasta la página del repositorio en GitHub, hacer clic en el menu desplegable "*Pull requests"*, seleccionar la que quisieramos revisar y dentro de ella ir a la sección *"Files changed"* (archivos modificados). En esta sección vamos a poder revisar y comentar los cambios propuestos haciendo clic en el botón *"Review changes".* Finalmente, si corresponde, podemos aprobar la combinación de los cambios propuestos haciendo clic en el botón "Aprove" (aprobar)
- Una etiqueta (tag) en Git, es un marcador que apunta a un *commit* específico en el historial del repositorio, sirve para señalar puntos importantes en el desarrollo, como lanzamientos de versiones o hitos clave. Es como una rama estática que no se mueve ni se actualiza con nuevos *commits*.
- Para crear una etiqueta en Git, se utiliza el comando git tag. Pueden ser etiquetas anotadas, que contienen información adicional (autor, correo, fecha, mensaje, etc.), o etiquetas ligeras, o simples, que sólo apunta a un *commit*.
  - `git tag -a nombre_etiqueta -m mensaje` crea una etiqueta anotada con un mensaje
  - `git tag nombre_etiqueta` crea una etiqueta ligera.
- Para enviar las etiquetas creadas localmente al repositorio remoto en GitHub, se utiliza el comando `git push --tags` (para enviar todas las etiquetas) o `git push nombre_repositorio_remoto nombre_de_etiqueta` para enviar una etiqueta específica.
- El historial de Git es un registro de todos los cambios realizados en un proyecto, guardado como una serie de *commits* que muestran la evolución del código a través del tiempo. Este historial permite ver las modificaciones realizadas, y también, quién las hizo, cuándo y con qué propósito.
- Para ver el historial, se utiliza el comando `git log`, que mostrará una lista de todos los *commits* en orden cronológico inverso, con información como el hash, el autor, la fecha y el mensaje de cada *commit*.
- Para buscar dentro del historial podemos usar varias opciones dentro del comando git log

  - `git log --author="nombre_del_autor"`; para buscar por autor del commit
  - `git log --grep="palabra_clave"`; para buscar una palabra clave en el mensaje de los commits.
  - `git log --since="fecha_inicial" --until="fecha_final"`; para buscar dentro de un rango de fechas. Se utiliza el formato de fecha "AAAA-MM-DD"
- Para borrar localmente el historial de Git debemos borrar la carpeta oculta *.git* del repositorio local (y volver a crearla si queremos volver a usar Git en el proyecto). Para borrar el historial en el repositorio remoto, debemos utilizar el comando `git reset` y luego `git push -f` para forzar la sobreescritura del historial.
- Un repositorio privado es aquel en el que sólo el propietario y las personas con permiso explícito pueden acceder y ver el contenido del mismo. Están diseñados para trabajar con información sensible sin exponer el código a otros. El propietario del repositorio puede invitar a otras personas como colaboradores, controlar los permisos que otorga a cada colaborador (roles), o cambiar la configuración de privacidad para convertirlo en un repositorio público.
- Para crear un repositorio remoto privado, podemos seleccionar la opción "Privado" en la sección "Visibilidad" al momento de crearlo, o cambiarlo de público a privado si ya lo habíamos creado.
- Para invitar a otros desarrolladores a ser colaboradores en nuestro proyecto debemos estar ubicados en la página de nuestro repositorio en GitHub, hacer clic en el botón "Configuración", ir a la sección "Acceso", y hacer clic en "Agregar personas". Dentro de esta sección debemos ingresar el correo electrónico o el nombre de usuario en GitHub de la persona a la que queremos invitar, y finalmente hacer clic en el botón "Agregar NOMBRE al REPOSITORIO". El usuario recibirá un correo electrónico y una vez que acepte la invitación, tendrá acceso de colaborador al repositorio.
- La principal característica de un repositorio público es que es accesible a cualquier persona con conexión a internet, a diferencia de los repositorios privados, que solo pueden ser vistos por los usuarios que tienen acceso explícito. Los repositorios públicos suelen utilizarse para compartir software de código abierto, donde otros pueden usar, modificar y distribuir el código sin restricciones. Se recomienda generar una licencia para los repositorios públicos de código abierto, lo que define los términos de uso del software.
- Para crear un repositorio remoto público, podemos seleccionar la opción "Público" en la sección "Visibilidad" al momento de crearlo, o cambiarlo de privado a público en cualquier momento.

UNIVERSIDAD TECNOLOGICA NACIONAL

Actividad 2

Para esta actividad creé un repositorio en GitHub y lo cloné localmente para poder trabajarlo a través de la terminal de Git Bash. El repositorio se encuentra alojado en el link https://github.com/JoacoPerezCampos/tp2_act2.git y a continuación adjunto la imagen de los comandos realizados en Bash.



UNIVERSIDAD TECNOLOGICA NACIONAL

Actividad 3

Para esta actividad creé un repositorio en GitHub y lo cloné localmente para poder trabajarlo a través de la terminal de Git Bash. El repositorio se encuentra alojado en el link https://github.com/JoacoPerezCampos/tp2_act3.git y a continuación adjunto la imagen de los comandos realizados en Bash.