HERRAMIENTAS DE DESARROLLO

Laboratorio: Configuración de Sistema de Control de Versiones en la Nube y Flujo Básico de Trabajo

Objetivo:

El objetivo de este laboratorio es enseñar cómo configurar un sistema de control de versiones en la nube utilizando una plataforma como GitHub o GitLab, y aplicar un flujo básico de trabajo para colaborar de manera efectiva en proyectos de desarrollo de software.

Procedimientos Básicos:

1. Crear una Cuenta en GitHub (o GitLab):

1. Dirígete a GitHub o GitLab y crea una cuenta si no tienes una.
2. Una vez registrada la cuenta, accede a tu panel de usuario.

2. Crear un Repositorio en la Nube:

1. En GitHub, haz clic en el botón de "New" para crear un nuevo repositorio.
2. Proporciona un nombre, descripción y selecciona si será público o privado.
3. No es necesario agregar un archivo README ni un `.gitignore` en este momento, ya que lo configuraremos más tarde.

3. Clonar el Repositorio Remoto en tu Máquina Local:

Para clonar el repositorio, primero copia la URL del repositorio en GitHub.

```
git clone https://github.com/<usuario>/<nombre_repositorio>.git
```

4. Crear y Configurar un Archivo `.gitignore`:

El archivo `.gitignore` especifica qué archivos y carpetas no se deben seguir (trackear) en el control de versiones.

1. Crea un archivo `.gitignore` en el directorio raíz de tu proyecto.
2. Añade configuraciones comunes para proyectos, por ejemplo, para Node.js:

```
node_modules/
.env
```

5. Agregar un Archivo README:

El archivo `README.md` describe el propósito del repositorio, cómo configurar y usar el proyecto.

1. Crea un archivo `README.md` en el directorio raíz.
2. Agrega información sobre el proyecto, como la descripción y las instrucciones de instalación.

6. Hacer un Commit Inicial:

Una vez que el archivo `.gitignore` y `README.md` estén configurados, puedes hacer un commit inicial.

```
git add .    # Añadir todos los archivos al Staging Area
git commit -m "Primer commit con README y .gitignore"
git push origin main  # Subir los cambios al repositorio remoto
```

7. Crear y Trabajar en una Nueva Rama:

Las ramas te permiten trabajar en nuevas funcionalidades sin afectar el código principal. Para crear y cambiar a una nueva rama:

```
git checkout -b feature/nueva-funcionalidad
```

8. Realizar Cambios y Hacer Commits:

Haz cambios en tu proyecto y luego haz commits frecuentemente. Por ejemplo:

```
git add <archivo_modificado>
git commit -m "Agregado nueva funcionalidad"
```

9. Subir los Cambios al Repositorio Remoto:

Para subir tus cambios en la nueva rama al repositorio remoto:

```
git push origin feature/nueva-funcionalidad
```

10. Crear una Solicitud de Extracción (Pull Request):

Para integrar tu rama `feature/nueva-funcionalidad` a la rama `main` en GitHub o GitLab, crea una Pull Request (PR).

1. Ve al repositorio en GitHub.
2. Haz clic en "Pull Request" y selecciona la rama `feature/nueva-funcionalidad` y la rama de destino `main`.
3. Completa los detalles y crea la PR.
4. El equipo revisa la PR y la fusiona si todo está correcto.

Preguntas:

1. **¿Qué es un sistema de control de versiones?**
    - **Respuesta**: Es una herramienta que permite gestionar y hacer seguimiento de los cambios realizados en los archivos de un proyecto a lo largo del tiempo. Permite trabajar de manera colaborativa y gestionar diferentes versiones del código.

2. **¿Qué es GitHub y qué beneficios ofrece en un flujo de trabajo de control de versiones?**
    - **Respuesta**: GitHub es una plataforma basada en la web que aloja repositorios de Git. Facilita la colaboración en proyectos de software, ofreciendo características como el control de versiones distribuido, revisión de código, gestión de ramas, Pull Requests y CI/CD.
3. **¿Qué es una rama (branch) en Git?**
    - **Respuesta**: Una rama en Git es una versión independiente del código que permite trabajar en una funcionalidad o corrección sin afectar el código principal (main). Permite desarrollar de manera aislada hasta que los cambios se integren.
4. **¿Qué significa hacer un "commit" en Git?**
    - **Respuesta**:
5. **¿Qué es una Pull Request (PR)?**
    - **Respuesta**:
6. **¿Qué es un archivo `.gitignore`?**
    - **Respuesta**:
7. **Crea un repositorio remoto en GitHub y clónalo en tu máquina local.**
    - **Respuesta**:
8. **Crea un archivo `.gitignore` que ignore los archivos de configuración y dependencias comunes de un proyecto Node.js.**
    - **Respuesta**:
9. **Haz un cambio en un archivo, agrégalo al Staging Area y haz un commit.**
    - **Respuesta**:
10. **Crea una nueva rama llamada `feature/actualizacion-ui` y cambia a ella.**
    - **Respuesta**:
11. **Realiza algunos cambios en la rama `feature/actualizacion-ui` y haz un commit.**
    - **Respuesta**:
12. **Sube tus cambios de la rama `feature/actualizacion-ui` al repositorio remoto.**
    - **Respuesta**:
13. **Cambia a la rama `main` y crea una Pull Request para fusionar la rama `feature/actualizacion-ui`.**
    - **Respuesta**: (Pasos dentro de GitHub)
        1. Cambia a la rama `main`.
        2. En GitHub, selecciona la opción "Pull Request" y elige `feature/actualizacion-ui` como la rama a fusionar.
14. **Fusiona la Pull Request de la rama `feature/actualizacion-ui` con la rama `main`.**
    - **Respuesta**: (Pasos dentro de GitHub)
        1. Haz clic en el botón de "Merge pull request" para fusionar la PR.
15. **Elimina la rama `feature/actualizacion-ui` después de fusionarla.**
    - **Respuesta**:
    - `git branch -d feature/actualizacion-ui`
16. **Realiza un `git pull` para obtener los últimos cambios de la rama `main`.**
    - **Respuesta**:
    - `git pull origin main`