Sesión Práctica de Git y GitHub

Módulo 2, Sesión 2: Gestión Práctica con Git y GitHub

El objetivo de hoy es que domines el flujo de trabajo completo de un Ingeniero DevOps en GitHub.

Taller 1: Flujo de Trabajo Fundamental: Branches y Merges

Objetivo: Aprender a trabajar de forma segura y organizada en un proyecto y utilizando ramas de funcionalidad para cada nueva tarea. Este es un flujo recomendado para proyectos personales.(No participan personas adicionales)

Paso 1: Crear y Clonar tu Repositorio

1. Crea el repositorio en github con el nombre: `dmc-devops` con las siguientes características:

- El `Owner` debe ser el nombre del Usuario
- El repositorio debe ser tener visibilidad público: `Visibility: public`
- Se debe habilitar la creación de README (`Add README: on`)

IMPORTANTE: A ESTE REPOSITORIO NO DEBE SUBIRSE DATA SENSIBLE: API keys, passwords, etc.

Paso 2: Realizar un nuevo Commit

1. Agregamos la siguiente linea dentro del archivo README.md

```
## Repositorio de Práctica DevOps
```

2. Prepara el archivo para ser versionado (lo añades al "Staging Area"):

```
git add README.md
```

3. Verificar estado de git del repositorio

```
git status
```

4. Confirma los cambios en tu historial local. (Se agrega el comando -m que hace referencia al mensaje que se escribirá a continuación)

```
git commit -m "Modificacion de archivo README.md"
```

5. Verificar estado de git del repositorio

```
git status
```

6. Sube tus cambios a GitHub:

```
git push origin main
```

Nota: `origin` es el nombre por defecto del repositorio remoto. `main` es la rama que estás empujando.

Paso 3: Usar una Rama de Funcionalidad (Feature Branch)

1. Crea una nueva rama para tu trabajo y cámbiate a ella:

```
git checkout -b feature/add-description
```

`checkout -b` es un atajo que crea y se mueve a la nueva rama. El nombre es descriptivo: es una *feature* y su propósito es *add-description*.

2. Añade contenido al `README.md`:

```
"Este repositorio contiene ejercicios para el curso de DevOps."
```

3. Haz commit de tus cambios en esta rama:

```
git add README.md
git commit -m "Agregando descripción de proyecto"
```

Paso 4: Fusionar tus Cambios en main

Una vez que tu trabajo en la rama de funcionalidad está completo, lo integras en la rama principal.

1. Vuelve a la rama `main`:

```
git checkout main
```

Observa que el `README.md` ha vuelto a su estado original. Tu cambio está seguro y aislado en la otra rama.

2. Fusiona los cambios desde tu rama de funcionalidad:

```
git merge feature/add-description
```

3. Sube la rama `main` actualizada a GitHub:

```
git push origin main
```

Taller 2: Flujo de Colaboración: Forks y Pull Requests

Objetivo: Aprender el estándar de la industria para contribuir a proyectos en los que no tienes permiso de escritura directo gestionandolos con Pull Requests (PRs).

Paso 1: Configura Ruleset en Github Actions

1. Ve a la página de del Repositorio en GitHub
2. Ve a `Settings` > `Rules` > `Rulesets`
3. Click en `New ruleset` > `New branch ruleset`
4. Completar con los siguientes datos
   - Ruleset Name: `master`
   - Enforcement status: `Active`
   - En la sessión `Target Branches` > `Add Target` > `Include by pattern` > coloca `main` > `Add Inclusion patter`
   - Marca la casilla: `Require a pull request before merging` (Esta es la configuración necesaria para que cualquier cambio a la branch main deba ser realizado a través de un pull request)
   - Navega hasta la parte inferior del formulario y haz click en `Create` para crear la Ruleset.

Paso 2: Crea una nueva branch a partir de branch main

1. Crea una nueva rama para tu contribución:

```
git status
git checkout -b feature/add-new-contribution
```

2. Edita el `README.md` para añadir tu:

```
- Contribución realizada a través de un pull request
```

3. Haz commit de tus cambios y súbelos a tu repositorio:

```
git add README.md
git commit -m "Agregar nueva contribución en repositorio"
```

```
git push origin feature/add-new-contribution
```

Paso 3: Crear el Pull Request

1. Ve a la página de del Repositorio en GitHub. Verás un banner sugiriendo crear un Pull Request. Haz clic en "**Compare & Pull Request**"
2. Revisa que la rama base sea `main` y la rama de comparación sea `feature/add-new-contribution`.
3. Escribe un título claro y una descripción para tu PR. Haz clic en "**Create pull request**".

Paso 4: Fusión

1. Revisa que no hayan conflictos y valida los apartados del Pull Request:
    - **"Conversation"**: Se pueden agregar comentarios y etiquetar a usuarios
    - **"Commits"**: Revisar lista de commits que se agregaron en el Pull Request
    - **"Checks"**: Revisión de Pipelines ejecutados, en este punto no tenemos pipelines configurados.
    - **"Files changed"**: Modificaciones realizadas

Taller 3: Automatización Básica con GitHub Actions (CI)

Objetivo: Crear una pipeline de Integración Continua (CI) que se ejecute automáticamente cuando se crea un Pull Request y se aprueba el Pull Request.

Paso 1: Crear workflow predefinido de GitHub

1. Ve a la página de del Repositorio en GitHub. Haz clic en "**Actions**", busca "**Simple workflow**" en la lista, y dale click en "**Configure**"
2. Se abrirá una nueva pantalla, cambiar nombre de archivo yaml a: "**simple-workflow.yml**", dar click en "**Commit changes...**" (botón verde)
3. Agregar un mensaje de commit, y marcar la opción de: "**Create a new branch for this commit and start a pull request**", cambiar nombre de branch a: "**feature/create-workflow**", click en "**Propose changes**"
4. Se creará un Pull Request con los cambios. Haz click en "**Create Pull Request**"
5. Verificar "**Checks**" del Pull Request, un wokflow de github actions de ha ejecutado con las definiciones del archivo yaml.
6. Haz click en "**Merge Pull Request**" y crear un mensaje o aceptar el mensaje predeterminado. Revisar logs de ejecución.

Taller 4: Estrategia de Branching con GitHub Flow

Objetivo: Gestionar un escenario realista combinando el desarrollo de una nueva funcionalidad con la resolución de un hotfix urgente, usando la estrategia de GitHub Flow.

Escenario: Estás trabajando en una nueva funcionalidad (`feature/user-profile`) cuando surge un bug crítico en producción que debes solucionar de inmediato.

Paso 1: Iniciar el Trabajo en la Nueva Funcionalidad

1. Asegúrate de estar en `main` y tener la última versión:

```
git checkout main
git pull origin main
```

2. Crea la rama para tu nueva funcionalidad:

```
git checkout -b feature/user-profile
```

3. Haz un commit inicial para simular trabajo:

```
echo "user_id,username" > profile.csv
git add profile.csv
git commit -m "Agregar estructura inicial para gestion de usuario"
```

Paso 2: Gestionar el Hotfix Urgente

1. **Pausa el trabajo actual:** Deja tu rama de feature tal como está y vuelve a la rama principal:

```
git checkout main
```

2. Crea una rama específica para el hotfix:

```
git checkout -b hotfix/critical-readme-update
```

3. Aplica la corrección urgente (en este caso, añade una línea al README):

```
echo "ACTUALIZACION URGENTE: Todos los sistemas operativos." >>
README.md
git add README.md
git commit -m "Agregar status de actualizacion urgente en README"
```

Paso 3: Desplegar el Hotfix

1. Sube la rama del hotfix a GitHub:

```
git push -u origin hotfix/critical-readme-update
```

2. Ahora ve a GitHub y crea una Pull Request

Paso 4: Sincronizar tu Rama de Funcionalidad

1. Vuelve a tu trabajo original:

```
git checkout feature/user-profile
```

2. Tu rama ahora está "detrás" de `main`. Debes traer los cambios del hotfix a tu rama para evitar futuros conflictos.

```
git fetch
git pull origin main
```

3. ¡Listo! Tu rama `feature/user-profile` ahora contiene tanto tu trabajo inicial como la actualización crítica de `main`. Puedes continuar desarrollando de forma segura.

4. Te aparecerá un error, ya que es necesario configurar la estrategia para hacer la mezcla de datos:

```
git config pull.ff true
```

5. Ahora reintenta la mezcla:

```
git pull origin main
```

6. Continua trabajando en la funcionalidad.