1. Introducción

A través del desarrollo de este tutorial pondrá en práctica los diferentes conceptos vistos en esta novena sesión.

Se espera que al finalizar usted pueda:

- Crear un repositorio local con Git.
- Manipular los archivos de un repositorio local mediante los comandos básicos: `init`, `add`, `commit`, `status`, `log` y `checkout`.

El punto de partida es el ejercicio número 7, en donde creo una página web libre incluyendo hojas de estilo en cascada. Creará el repositorio y agregará nuevos estilos al sitio web para mejorar la tipografía.

Descargue los archivos que subió a la carpeta del drive como parte del ejercicio numero 7.

2. Prerrequisitos

Para realizar este tutorial usted debe tener claras las respuestas a las siguientes preguntas:

Pregunta 1. ¿Qué es un repositorio?

Pregunta 2. ¿Cuáles son las zonas de trabajo en Git?

Pregunta 3. ¿Cómo consultar el estado de un repositorio?

Pregunta 4. ¿Cómo se agregan archivos al área de preparación?

Pregunta 5. ¿Qué es un `commit`?

Pregunta 6. ¿Cómo consultar el historial del repositorio?

Pregunta 7. ¿Es posible devolverse a una versión anterior?

Si no es así, por favor revise la presentación de la sesión 9. De esta forma podrá sacarle más provecho.

3. Crear el repositorio

3.1. Cree la carpeta `/tutorial1` en el directorio de su preferencia.

```
$ mkdir tutorial1
```

3.2. Desde `Git Bash` o la `terminal` ubíquese en la carpeta.

```
$ cd tutorial1
```

3.3. Cree un repositorio mediante el comando `init`.

```
$ git init
```

Este comando crea un directorio oculto llamado `.git` en la raíz del proyecto. Posteriormente Git coloca toda la información de las versiones y su historia en dicho directorio.

Verifíquelo consultando los archivos y directorios ocultos en la carpeta.

```
$ ls -a
```

4. Agregar el código base al repositorio local

4.1. Copie los archivos descargados del drive en el directorio de su repositorio local.

4.2. Consulte el estado de su repositorio.

```
$ git status
```

Note que los archivos están en la copia de trabajo local, pero aún no han sido incluidos al área de preparación.

Este comando muestra los diferentes estados de los archivos en la copia de trabajo local y el área de preparación. Permite ver qué archivos están modificados y sin incluir en el área de preparación y cuáles están en el área de preparación, pero no han sido confirmados.

4.3. Añada los archivos .ccs y .html al área de preparación.

```
$ git add . //Al ejecutar el comando de esta forma, se están agregando todos los cambios al área de preparación.
```

4.4. Consulte el estado de su repositorio e identifique los cambios.

```
$ git status
```

Ahora los archivos han sido etiquetados como `new file`, es decir que se encuentran por primera vez en el área de preparación y están pendientes de ser enviados al repositorio.

4.5. Utilice el comando `commit` para añadir los archivos que están en el área de preparación al repositorio.

```
$ git commit -m "Se añade codigo base" //La opción -m implica que habrá un mensaje de confirmación asociado al commit
```

Note que Git especifica cuántos archivos han sido cambiados y cuántas líneas de código han sido insertadas o eliminadas.

El autor de un `commit` debe comentar lo que hizo mediante un breve mensaje. Esto ayuda a otras personas (y a sí mismo) a comprender más adelante el propósito de los cambios.

En caso de equivocarse en el `commit`, puede modificarlo usando la opción `amend.`

- Use el siguiente comando para modificar el mensaje de confirmación.

```
$ git commit --amend -m "nuevo mensaje"
```

//la opción `-m` permite escribir un nuevo mensaje desde la línea de comandos sin tener que abrir un editor.

- Si desea modificar archivos del `commit`, realice los ajustes correspondientes en la copia de trabajo local, agreguelos al área de preparación y ejecute el siguiente comando.

```
$ git commit --amend --no-edit
```

//la opción `--no-edit` permite hacer las correcciones en la confirmación sin cambiar el mensaje.

4.6. Consulte el historial de cambios.

```
$ git log
```

Note que además de la información básica del `commit`, también es posible identificar la rama donde se realizaron los cambios y el identificador único del mismo. Este último le permitirá buscarlo o devolverse de `commit` más adelante.

Este comando lista en orden cronológico el historial de los `commits` que se han realizado al repositorio.

Git también le permite listar el detalle de los cambios que ocurrieron en cada commit. Utilice la bandera `-p` con el comando `git log` para agregar este tipo de información.

En este momento, ya cuenta con el código base del proyecto en el repositorio y está listo para empezar a trabajar en él.

5. Modificar archivos del repositorio

Usted ha identificado que los estilos asociados con la tipografía merecen una mejora. Abra el archivo .css y modifique los estilos tipográficos.

Una vez realizado el ajuste realice los siguientes pasos:

5.1. Verifique el estado del repositorio.

```
$ git status
```

Ahora el archivo .css ha sido etiquetado como `modified` (en color rojo), es decir que Git identificó cambios en el archivo de la copia de trabajo local que aún no han sido movidos al área de preparación.

5.2. Agregue los cambios al área de preparación.

```
$ git add nombreDeSuArhivo.css //de esta forma se agrega archivo específico
```

5.3. Consulte el estado del repositorio para verificar los cambios.

```
$ git status
```

El archivo `/tutorial1/nombreDeSuArchivo.css` ha sido etiquetado como `modified` (en color verde), es decir que los cambios ya fueron movidos al área de preparación y están pendientes de guardarse en el repositorio.

5.4. Una vez esté seguro del cambio, confírmelo en el repositorio.

```
$ git commit -m "Se modifica los estilos tipograficos"
```

5.5. Consulte el historial de cambios.

```
$ git log
```

Verifique los 2 `commit` realizados hasta el momento: su identificador, fecha de creación, autor y mensajes de confirmación.

6. Consultar las diferencias entre un commit y otro

6.1. Utilice el comando `diff` para consultar las diferencias entre un commit y otro.

```
$ git diff <id_commit2>..<id_commit1>
```

Tenga en cuenta que este comando requiere tener conocimiento del id de los `commit`.

Una vez ejecutado le permite evidenciar el directorio del archivo modificado y las líneas específicas que fueron cambiadas, en este caso la línea roja representa lo que estaba en la versión anterior y la verde lo que está en la versión actual.

7. Agregar archivos al repositorio

Después de agregar el código base y modificar un archivo, es hora de agregar un archivo readme.txt al proyecto.

7.1. Cree un archivo readme.txt en el directorio `/tutorial1` de su copia de trabajo local.

7.2. Verifique el estado del repositorio.

```
$ git status
```

7.4. Agregue los cambios al área de preparación.

```
$ git add .
```

7.5. Consulte el estado del repositorio para verificar los cambios.

```
$ git status
```

7.6. Confirme los cambios en el repositorio.

```
$ git commit -m "Se incluye archivo readme"
```

7.7. Consulte el historial de cambios.

```
$ git log
```

Verifique los 3 `commit` realizados hasta el momento: su identificador, fecha de creación, autor y mensajes de confirmación para realizar la siguiente parte del tutorial.

8. Devolverse a una versión anterior

8.1. Identifique el ID del `commit` al cual desea regresar y utilice el comando `checkout`.

```
$ git checkout <id_commit>.
```

Observe que Git responde con la descripción del `commit` en el que se encontraba antes de ejecutar el comando y en el que se encuentra ahora.

8.2. Consulte el historial del repositorio nuevamente.

```
$ git log
```

Ahora se han revertido los cambios y su proyecto se encuentra como en el primer `commit`.