Laboratorio - Control de versiones de software con Git
Objetivos

Parte 1: Iniciar la consola Git Bash

Parte 2: Inicializar Git

Parte 3: Preparar y confirmar un archivo en el repositorio de Git

Parte 4: Gestión del archivo y seguimiento de los cambios

Parte 5: Sucursales y fusión

Parte 6: Gestión de conflictos de combinación

Parte 7: Integración de Git con GitHub

Aspectos básicos/Situación.

En este laboratorio, explorará los fundamentos del sistema de control de versiones distribuido Git, incluyendo la mayoría de las características que necesita conocer para colaborar en un proyecto de software. También integrará su repositorio Git local con el repositorio GitHub basado en la nube.

Recursos necesarios

- Una computadora con el sistema operativo de su elección.
- Consola Git Bash

Instrucciones

Parte 1: Iniciar la Consola Git Bash

Si no ha completado la Guía 6 **- Instalando Git y abriendo cuanta en GitHub**, hágalo ahora. Si se ha completado ya, inicie la consola Git Bash.

Parte 2: Inicializando Git

En esta parte, inicializará un repositorio de Git.

Paso 1: Inicializar un repositorio de Git.

a. Utilizar el comando **ls** para mostrar el directorio actual. Recuerde que comandos distinguen entre mayúsculas y minúsculas.

```
MINGW64 ~$ ls
Descargar plantillas de música de públicas
Documentos de laboratorio, Imágenes instantáneas y videos.
MINGW64 ~$$

Nota: se despliega la información de los archivos y carpetas de la unidad actual
```

b. Haga un directorio **git-intro** y cambie el directorio en él:

```
MINGW64 ~$ mkdir git-intro
MINGW64 ~$ cd git-intro
```

```
MINGW64 ~/git-intro$
```

c. Utilice el comando **git init** para inicializar el directorio actual (git-intro) como repositorio de Git. El mensaje que se muestra indica que ha creado un repositorio local dentro del proyecto contenido en el directorio oculto **.git**. Aquí es donde se encuentra todo el historial de cambios. Puedes verlo con el comando **ls -a**.

```
MINGW64 ~/git-intro$ git init
Initialized empty Git repository in C:/Users/Orlando/git-intro/.git/
MINGW64 ~/git-intro (main)$ ls -a
./  ../  .git/
MINGW64 ~/git-intro (main)$
```

d. A continuación, configure la información de usuario que se utilizará para este repositorio local. Esto asociará su información con el trabajo que contribuye a un repositorio local. Utilice su nombre en lugar de "Usuario de prueba" para el nombre entre comillas. Utilice @example.com para su dirección de correo electrónico.

Nota: Esta configuración puede ser cualquier cosa que desee en este momento. Sin embargo, cuando restablezca estos valores globales en la Parte 7, usará el nombre de usuario de su cuenta de GitHub. Si lo desea, puede usar su nombre de usuario de GitHub ahora.

```
MINGW64 ~/git-intro (main)$ git config --global user.name "SampleUser"
MINGW64 ~/git-intro (main)$ git config --global user.email sample@example.com
```

e. En cualquier momento, puede revisar esta configuración con el comando **git config --list**.

```
MINGW64 ~/git-intro (main)$ git config --list
User.name=SampleUser
user.email=sample@example.com
MINGW64 ~/git-intro (main)$
```

f. A medida que trabaje en su proyecto, querrá comprobar qué archivos han cambiado. Esto es útil cuando está enviando archivos al repositorio, y no desea comprometerlos todos. El comando **git status** muestra los archivos modificados en el directorio de trabajo que se almacenan en etapas para su siguiente confirmación.

Este mensaje le dice:

- Estas en una rama maestra. (Los niveles se discuten más adelante en este laboratorio)
- El mensaje de confirmación es la confirmación inicial.
- No hay cambios que comprometer

Verá que el estado de su repositorio cambiará una vez que se agreguen archivos y se comience a realizar cambios.

```
MINGW64 ~/git-intro (main)$ git status
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)
MINGW64 ~/git-intro (main)$$
```

Parte 3: Preparando y confirmando un archivo en el repositorio

En esta parte creará un archivo, pondrá en escena ese archivo y confirmará ese archivo en el repositorio de Git.

Paso 1: Crear un archivo.

a. El repositorio **git-intro** se creó pero está vacío. Usando el comando **echo**, crea el archivo **gitversion.txt** con la información contenida entre comillas.

```
MINGW64 ~/git-intro (main)$ echo "Estoy en camino de aprender a utilizar GIT"
> gitversion.txt
MINGW64 ~/git-intro (main)$
```

b. Utilzar el comando **ls -la** para verificar el archivo, así como el directorio.git, que están en el directorio **git intro**. A continuación, utilice **cat** para mostrar el contenido de **gitversion.txt**.

```
MINGW64 ~/git-intro (main)$ ls -la
total 33
drwxr-xr-x 1 Orlando 197121  0 Jul 23 12:19 ./
drwxr-xr-x 1 Orlando 197121  0 Jul 23 12:12 ../
drwxr-xr-x 1 Orlando 197121  0 Jul 23 12:14 .git/
-rw-r--r-- 1 Orlando 197121 43 Jul 23 12:19 gitversion.txt
MINGW64 ~/git-intro (main)$ cat gitversion.txt
Estoy en camino de aprender a utilizar GIT
MINGW64 ~/git-intro (main)$
```

Paso 2: Examinar el estado del repositorio.

Examinar el estado del repositorio usando el **estado de git**. Observar que Git encontró el nuevo archivo en el directorio y sabe que no se rastreó.

```
MINGW64 ~/git-intro (main)$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        gitversion.txt

nothing added to commit but untracked files present (use "git add" to track)
MINGW64 ~/git-intro (main)$
```

Paso 3: Ponga en ejecución el archivo.

a. A continuación, use el comando **git add** para "ejecutar" el archivo **gitversion.txt**. La ejecución es una fase intermedia previa a enviar un archivo al repositorio con el comando **git commit**. Este comando crea el contenido del archivo inmediatamente al mismo tiempo de que el comando se haya introducido. Cualquier cambio en el archivo requiere otro comando **git add** antes de confirmar el archivo.

```
MINGW64 ~/git-intro (main)$ git add gitversion.txt
```

```
warning: LF will be replaced by CRLF in gitversion.txt.
The file will have its original line endings in your working directory
MINGW64 ~/git-intro (main)$
```

b. Usando el comando **git status** nuevamente, observe los cambios por etapas que se muestran como "new.file: gitversion.txt".

```
MINGW64 ~/git-intro (main)$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   gitversion.txt
MINGW64 ~/git-intro (main)$
```

Paso 4: Confirmando un archivo.

Ahora que ha puesto sus cambios en ejecución, tiene que confirmarlos para que Git sepa que quiere comenzar a rastrear esos cambios. Confirmar su contenido por etapas como una nueva confirmación instantánea mediante el comando **git commit**. El mensaje **-m** permitirá agregar un mensaje explicando los cambios realizados. Observe la combinación de número y letra resaltada en la salida. Este es el ID de confirmación. Cada confirmación se identifica mediante un hash SHA1 único. El ID de confirmación es los primeros 7 caracteres del hash de confirmación completo. Su ID de confirmación será diferente al mostrado.

```
MINGW64 ~/git-intro (main)$ git commit -m "Confirmando gitversion.txt para comenzar el seguimiento de cambios"
[main (root-commit) 770208a] Confirmando gitversion.txt para comenzar el seguimiento de cambios
 1 file changed, 1 insertion(+)
 create mode 100644 gitversion.txt
MINGW64 ~/git-intro (main)$
```

Paso 5: Ver el historial de confirmación.

Utilice el comando **git log** para mostrar todas las confirmaciones en el historial de la rama actual. De forma predeterminada, todas las confirmaciones son realizadas en la rama principal. (Las ramas se discutirán más adelante). La primera línea es el hash de confirmación con el ID de confirmación como los primeros 7 caracteres. El archivo está comprometido con la rama maestra. A continuación se indica su nombre y dirección de correo electrónico, la fecha de la confirmación y el mensaje que ha incluido con la confirmación.

```
MINGW64 ~/git-intro (main)$ git log
commit 770208ae683dfc85e7d2e067f9c0ce68189588d7 (HEAD -> main)
Author: Sena-Jocastron <jocastron@gmail.com>
Date:   Fri Jul 23 12:29:50 2021 -0500

    Confirmando gitversion.txt para comenzar el seguimiento de cambios
MINGW64 ~/git-intro (main)$
```

Parte 4: Modificación del archivo y seguimiento de los cambios

En esta parte, modificará un archivo, pondrá en ejecución el archivo, confirmará el archivo y verificará los cambios en el repositorio.

Paso 1: Modificar el archivo.

a. Realice un cambio en gitversion.txt mediante el comando **echo**. Asegúrese de usar ">>" para agregar el archivo existente. El ">" sobrescribirá el archivo existente. Utilice el comando **cat** para ver el archivo modificado.

```
MINGW64 ~/git-intro (main)$ echo "Estoy empezando a entender Git!" >>
gitversion.txt
MINGW64 ~/git-intro (main)$
```

b. Utilice el comando **cat** para ver el archivo modificado.

```
MINGW64 ~/git-intro (main)$ cat gitversion.txt
Estoy en camino de aprender a utilizar GIT
Estoy empezando a entender Git!
MINGW64 ~/git-intro (main)$
```

Paso 2: Verificar el cambio en el repositorio.

Verificar el cambio en el repositorio usando el comando **git status**.

```
MINGW64 ~/git-intro (main)$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   gitversion.txt

no changes added to commit (use "git add" and/or "git commit -a")
MINGW64 ~/git-intro (main)$
```

Paso 3: Ejecución del archivo modificado.

El archivo modificado tendrá que ser puesto en escena de nuevo antes de que pueda ser confirmado usando el comando **git add** nuevamente.

```
MINGW64 ~/git-intro (main)$ git add gitversion.txt
```

Paso 4: Confirmar el archivo por ejecución.

Confirmar el archivo por ejecución usando el comando **git commit**. Observe el nuevo ID de confirmación.

```
MINGW64 ~/git-intro (main)$ git commit -m "Agregada línea adicional al
archivo gitversion.txt"
[main bd1e471] Agregada linea adicional al archivo gitversion.txt
 1 file changed, 1 insertion(+)
MINGW64 ~/git-intro (main)$
```

Paso 5: Compruebe los cambios en el repositorio.

a. Utilice el comando **git log** de nuevo para mostrar todas las confirmaciones. Observe que el registro contiene la entrada de confirmación original junto con la entrada para la confirmación que acaba de realizar. La última confirmación se muestra primero. La salida resalta el ID de confirmación (los primeros 7 caracteres del hash SHA1), la fecha/hora de la confirmación y el mensaje de la confirmación para cada entrada.

```
MINGW64 ~/git-intro (main)$ git log
commit bd1e47153497584e61a70916843b5b5c80f66870 (HEAD -> main)
Author: Sena-Jocastron <jocastron@gmail.com>
Date:   Fri Jul 23 12:42:37 2021 -0500

    Agregada linea adicional al archivo gitversion.txt

commit 770208ae683dfc85e7d2e067f9c0ce68189588d7
Author: Sena-Jocastron <jocastron@gmail.com>
Date:   Fri Jul 23 12:29:50 2021 -0500

    Confirmando gitversion.txt para comenzar el seguimiento de cambios
MINGW64 ~/git-intro (main)$
```

b. Cuando tiene varias entradas en el registro, puede comparar las dos confirmaciones usando el comando **git diff** agregando el ID de confirmación original primero y el último ID después: **git diff** <commit ID original> <commit ID latest>. Deberá usar sus ID de confirmación. El signo "+" al final, seguido del texto, indica el contenido que se agregó al archivo.

```
MINGW64 ~/git-intro (main)$ git diff bd1e471 770208a
diff --git a/gitversion.txt b/gitversion.txt
index 420c014..d320133 100644
--- a/gitversion.txt
+++ b/gitversion.txt
@@ -1,2 +1 @@
 Estoy en camino de aprender a utilizar GIT
-Estoy empezando a entender Git!
MINGW64 ~/git-intro (main)$
```

Parte 5: Ramas y fusiones.

Cuando se crea un repositorio, los archivos se colocan automáticamente en una rama llamada **master**. Siempre que sea posible, se recomienda utilizar ramas en lugar de actualizar directamente la rama maestra. La derivación se utiliza para que pueda realizar cambios en otra área sin afectar a la rama maestra. Esto se hace para ayudar a evitar actualizaciones accidentales que podrían sobrescribir el código existente.

En esta parte, creará una nueva rama, retirará la rama, hará cambios en la rama, ejecutará y confirmará la rama, fusionará los cambios de esa rama en la rama principal y luego eliminará la rama.

Paso 1: Crear una nueva rama

Cree una nueva rama llamada **feature** usando el comando **git branch**<branch-name>

```
MINGW64 ~/git-intro (main)$ git branch feature
```

```
MINGW64 ~/git-intro (main)$
```

Paso 2: Verificar la rama actual

Utilice el comando **git branch** sin nombre de rama para mostrar todas las ramas de este repositorio. El "*" junto a la rama principal indica que se trata de la rama actual, la rama que está actualmente "desprotegida".

```
MINGW64 ~/git-intro (main)$ git branch
  feature
* main
MINGW64 ~/git-intro (main)$
```

Paso 3: Verifique la nueva rama.

Utilizar el comando **git checkout** <branch-name> para cambiar las *características* de la rama.

```
MINGW64 ~/git-intro (main)$ git checkout feature
Switched to branch 'feature'
MINGW64 ~/git-intro (feature)$
```

Paso 4: Verificar la rama actual.

a. Verificar que haya cambiado la rama *características* usando el comando **git branch**. Tenga en cuenta el "*" junto a la rama *características*. Esta es ahora la *rama de trabajo*.

```
MINGW64 ~/git-intro (feature)$ git branch
* feature
  main
MINGW64 ~/git-intro (feature)$
```

b. Añadir una nueva línea de texto al archivo gitversion.txt, utilizando nuevamente el comando **echo** con los signos ">>".

```
MINGW64 ~/git-intro (feature)$ echo "Este texto se agregó originalmente en la rama de features" >> gitversion.txt
```

c. Comprobar que la línea se ha añadido al archivo mediante el comando **cat**.

```
MINGW64 ~/git-intro (feature)$ cat gitversion.txt
Estoy en camino de aprender a utilizar GIT
Estoy empezando a entender Git!
Este texto se agregó originalmente en la rama de features
MINGW64 ~/git-intro (feature)$
```

Paso 5: Presente el archivo modificado en la rama de feature.

a. Presentar el archivo actualizado a la rama de *feature* actual.

```
MINGW64 ~/git-intro (feature)$ git add gitversion.txt
warning: LF will be replaced by CRLF in gitversion.txt.
The file will have its original line endings in your working directory
MINGW64 ~/git-intro (feature)$
```

b. Utilice el comando **git status** y observe que el archivo modificado **gitversion.txt** está en la rama *características*.

```
MINGW64 ~/git-intro (feature)$ git status
```

```
On branch feature
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   gitversion.txt
MINGW64 ~/git-intro (feature)$
```

Paso 6: Confirmar el archivo por ejecución en la rama de características

a. Confirmar el archivo por ejecución usando el comando **git commit**. Observe el nuevo ID de confirmación y su mensaje.

```
MINGW64 ~/git-intro (feature)$ git commit -m "Agregado una tercera línea en la rama de feature"
[feature 47d1452] Agregado una tercera línea en la rama de feature
 1 file changed, 1 insertion(+)
MINGW64 ~/git-intro (feature)$
```

b. Use el comando **git log** para mostrar todas las confirmaciones, incluida la confirmación que acaba de hacer en la rama de *características*. La confirmación previa se realizó dentro de la rama *principal*.

```
MINGW64 ~/git-intro (feature)$ git log
commit 47d145264a14709fbdf63ea089b19d1a02abf2a5 (HEAD -> feature)
Author: Sena-Jocastron <jocastron@gmail.com>
Date:   Fri Jul 23 13:09:13 2021 -0500

    Agregado una tercera línea en la rama de feature

commit bd1e47153497584e61a70916843b5b5c80f66870 (main)
Author: Sena-Jocastron <jocastron@gmail.com>
Date:   Fri Jul 23 12:42:37 2021 -0500

    Agregada linea adicional al archivo gitversion.txt

commit 770208ae683dfc85e7d2e067f9c0ce68189588d7
Author: Sena-Jocastron <jocastron@gmail.com>
Date:   Fri Jul 23 12:29:50 2021 -0500

    Confirmando gitversion.txt para comenzar el seguimiento de cambios
MINGW64 ~/git-intro (feature)$
```

Paso 7: Verificación de la rama maestra.

Cambie a la rama maestra usando el comando **git checkout** *master* y verifique la rama de trabajo actual usando el comando **git branch**.

```
MINGW64 ~/git-intro (feature)$ git checkout main
Switched to branch 'main'
MINGW64 ~/git-intro (main)$ git branch
  feature
* main
MINGW64 ~/git-intro (main)$
```

Paso 8: Combinar el contenido del archivo de las características con la rama maestra.

a. Las ramas se utilizan a menudo al implementar nuevas características o correcciones. Los miembros del equipo pueden enviarlos para su revisión, y luego, una vez verificados, pueden ser arrastrados a la base de código principal: la rama maestra.

   Combine el contenido (conocido como el historial) de la rama de características en la rama maestra usando el comando <branch-name> X **git merge**. El nombre de la rama es la rama de la que se extraen los historiales a la rama actual. La salida muestra que se ha cambiado un archivo con una línea insertada.

```
MINGW64 ~/git-intro (main)$ git merge feature
Updating bd1e471..47d1452
Fast-forward
 gitversion.txt | 1 +
 1 file changed, 1 insertion(+)
MINGW64 ~/git-intro (main)$
```

b. Compruebe el contenido anexado al archivo gitversion.txt en la rama maestra mediante el comando **cat**.

```
MINGW64 ~/git-intro (main)$ cat gitversion.txt
Estoy en camino de aprender a utilizar GIT
Estoy empezando a entender Git!
Este texto se agregó originalmente en la rama de features
MINGW64 ~/git-intro (main)$
```

Paso 9: Eliminar una rama.

a. Verifique que la rama de **características** aún esté disponible usando el comando **git branch**.

```
MINGW64 ~/git-intro (main)$ git branch
  feature
* master
MINGW64 ~/git-intro (main)$
```

b. Eliminar la rama de **características** usando el comando <branch-name> **git branch -d**. .

```
MINGW64 ~/git-intro (main)$ git branch -d feature
Deleted branch feature (was 47d1452).
MINGW64 ~/git-intro (main)$
```

c. Verifique que la rama de características ya no esté disponible usando el comando **git branch**.

```
MINGW64 ~/git-intro (main)$ git branch
* main
MINGW64 ~/git-intro (main)$
```

Parte 6: Manejo de conflictos de fusión.

A veces, puede experimentar un conflicto de fusión. Esto es cuando es posible que haya realizado cambios superpuestos en un archivo, y Git no puede combinar automáticamente los cambios.

En esta parte, creará una rama de prueba, modificará su contenido, pondrá en ejecución y confirmará la rama de prueba, cambiará a la rama maestra, modificará el contenido de nuevo, pondrá en ejecución y

comprometerá la rama maestra, intentará fusionar ramas, localizar y resolver el conflicto, poner en escena y confirmar la rama maestra de nuevo, y verificará su confirmación.

Paso 1: Crear una nueva rama test.

Crear una nueva **rama de prueba**

```
MINGW64 ~/git-intro (main)$ git branch test
MINGW64 ~/git-intro (main)$
```

Paso 2: Revisar la rama de prueba

a. Cambiar a la rama de **prueba**

```
MINGW64 ~/git-intro (main)$ git checkout test
Switched to branch 'test'
MINGW64 ~/git-intro (test)$
```

b. Verificar que la rama de trabajo sea la rama **de prueba**.

```
MINGW64 ~/git-intro (test)$ git branch
  main
* test
MINGW64 ~/git-intro (test)$
```

Paso 3: Comprobar el contenido actual de gitversion.txt.

Comprobar el contenido actual del archivo **gitversion.txt**. Observar que la primera línea incluya la palabra "Cisco".

```
MINGW64 ~/git-intro (test)$ cat gitversion.txt
Estoy en camino de aprender a utilizar GIT
Estoy empezando a entender Git!
Este texto se agregó originalmente en la rama de features
MINGW64 ~/git-intro (test)$
```

Paso 4: Modificar el contenido de gitversion.txt en la rama test.

Utilizar el comando **sed** para cambiar la palabra "GIT" a "Git y GitHub" en el archivo **gitversion.txt**.

```
MINGW64 ~/git-intro (test)$ sed -i 's/GIT/Git y GitHub/' gitversion.txt
MINGW64 ~/git-intro (test)$
```

Paso 5: Comprobar el contenido del gitversion.txt modificado en la rama de prueba.

Comprobar el cambio en el archivo **gitversion.txt**.

```
MINGW64 ~/git-intro (test)$ cat gitversion.txt
Estoy en camino de aprender a utilizar Git y GitHub
Estoy empezando a entender Git!
Este texto se agregó originalmente en la rama de features
MINGW64 ~/git-intro (test)$
```

Paso 6: Ponga en escena y confirme la rama de prueba

Ponga en escena y confirme el archivo con un solo comando **git commit -a**. La opción **-a** sólo afecta a los archivos modificados y eliminados. No afecta a los archivos nuevos.

```
MINGW64 ~/git-intro (test)$ git commit -a -m " Cambiar GIT a Git y GitHub "
warning: LF will be replaced by CRLF in gitversion.txt.
The file will have its original line endings in your working directory
[test ac05c54] Cambiar GIT a Git y GitHub
 1 file changed, 1 insertion(+), 1 deletion(-)
MINGW64 ~/git-intro (test)$
```

Paso 7: Revisar la rama principal.

a. Cambiar a la rama **principal**

```
MINGW64 ~/git-intro (test)$ git checkout main
Switched to branch 'main'
MINGW64 ~/git-intro (main)$
```

b. Compruebe que la rama principal es la rama de trabajo actual.

```
MINGW64 ~/git-intro (main)$ git branch
* main
  test
MINGW64 ~/git-intro (main)$
```

Paso 8: Modificar el contenido de gitversion.txt en la rama principal.

Verifique el contenido del archivo gitversion.txt
MINGW64 ~/git-intro (main)$ **cat gitversion.txt**

Utilice el comando **sed** para cambiar la palabra "GIT" a "Git" en el archivo gitversion.txt.

```
MINGW64 ~/git-intro (main)$ sed -i 's/GIT/Git/' gitversion.txt
```

Paso 9: Compruebe el contenido del archivo gitversion.txt modificado en la rama principal.

Verificar el cambio en el archivo.

```
MINGW64 ~/git-intro (main)$ cat gitversion.txt
Estoy en camino de aprender a utilizar Git
Estoy empezando a entender Git!
Este texto se agregó originalmente en la rama de features
MINGW64 ~/git-intro (main)$
```

Paso 10: Ponga en ejecución y confirme la rama de maestra.

Ponga en ejecución y confirme el archivo con un solo comando **git commit -a.**

```
MINGW64 ~/git-intro (main)$ git commit -a -m "Cambiado GIT a Git"
warning: LF will be replaced by CRLF in gitversion.txt.
The file will have its original line endings in your working directory
[main 5bd9abd] Cambiado GIT a Git
 1 file changed, 1 insertion(+), 1 deletion(-)
```

```
MINGW64 ~/git-intro (main)$
```

Paso 11: Intentar fusionar la rama de prueba en la rama maestra.

Intentar combinar el historial de rama de prueba en la rama maestra.

```
MINGW64 ~/git-intro (main)$ git merge test
Auto-merging gitversion.txt
CONFLICT (content): Merge conflict in gitversion.txt
Automatic merge failed; fix conflicts and then commit the result.
MINGW64 ~/git-intro (main)$
```

Paso 12: Encuentra el conflicto.

a. Utilice el comando **git log** para ver las confirmaciones. Observe que la versión HEAD es la rama maestra. Esto será útil en el siguiente paso.

```
MINGW64 ~/git-intro (main)$ git log
commit 5bd9abd66c17fc2d508fcf3e9accf779ef72a048 (HEAD -> main)
Author: Sena-Jocastron <jocastron@gmail.com>
Date:   Fri Jul 23 14:03:31 2021 -0500

    Cambiado GIT a Git

commit 47d145264a14709fbdf63ea089b19d1a02abf2a5
Author: Sena-Jocastron <jocastron@gmail.com>
Date:   Fri Jul 23 13:09:13 2021 -0500

    Agregado una tercera línea en la rama de feature

commit bd1e47153497584e61a70916843b5b5c80f66870
Author: Sena-Jocastron <jocastron@gmail.com>
Date:   Fri Jul 23 12:42:37 2021 -0500

    Agregada linea adicional al archivo gitversion.txt

commit 770208ae683dfc85e7d2e067f9c0ce68189588d7
Author: Sena-Jocastron <jocastron@gmail.com>
Date:   Fri Jul 23 12:29:50 2021 -0500

    Confirmando gitversion.txt para comenzar el seguimiento de cambios
MINGW64 ~/git-intro (main)$
```

b. Usar el comando **cat** para ver los contenidos del archivo gitversion.txt. El archivo ahora contiene información para ayudarle a encontrar el conflicto. La versión HEAD (rama main) que contiene la palabra "Git y GitHub" está en conflicto con la versión de la rama test y la palabra "Git".

```
MINGW64 ~/git-intro (main)$ cat gitversion.txt
<<<<<<< HEAD
Estoy en camino de aprender a utilizar Git
=======
```

```
Estoy en camino de aprender a utilizar Git y GitHub
>>>>>>> test
Estoy empezando a entender Git!
Este texto se agregó originalmente en la rama de features
MINGW64 ~/git-intro (main)$
```

Paso 13: Editar manualmente el archivo gitversion.txt para eliminar el texto en conflicto.

a. Utilice el comando **vim** para editar el archivo.

```
MINGW64 ~/git-intro (main)$ vim gitversion.txt
```

b. Utilice las flechas arriba y abajo para seleccionar la línea de texto adecuada. Presione **dd** (delete) en las siguientes líneas que se resaltan. **dd** eliminará la línea en la que está el cursor.

```
<<<<<<< HEAD
Estoy en camino de aprender a utilizar Git
=======
Estoy en camino de aprender a utilizar Git y GitHub
>>>>>>> test
Estoy empezando a entender Git!
Este texto se agregó originalmente en la rama de features
```

c. Guarde sus cambios en vim presionando **ESC** (la tecla de escape) y luego escribiendo **:** (dos puntos) seguido de **wq** y presione enter.

```
ESC
:
wq
<Enter or Return>
```

Paso 14: Comprobar las ediciones de gitversion.txt en la rama principal.

Comprobar los cambios mediante el comando **cat**.

```
MINGW64 ~/git-intro (main)$ cat gitversion.txt
Estoy en camino de aprender a utilizar Git
Estoy empezando a entender Git!
Este texto se agregó originalmente en la rama de features
MINGW64 ~/git-intro (main)$
```

Paso 15: Ponga en ejecución y confirme la rama de maestra

Ponga en ejecución y confirme el archivo con un solo comando **git commit -a.**

```
MINGW64 ~/git-intro (main)$ git add gitversion.txt
MINGW64 ~/git-intro (main)$ git commit -a -m "Fusionada manualmente desde la rama main"
[main 9cd1d81] Fusionada manualmente desde la rama main
MINGW64 ~/git-intro (main)$
```

Paso 16: Verificar la confirmación.

Usar el comando **git log** para mirar las confirmaciones. Si es necesario, puede usar **q** para salir de la pantalla del registro de git.

```
MINGW64 ~/git-intro (main)$ git log
commit 9cd1d81deaef6fbae66c28a4ab01a54a3d9b5628 (HEAD -> main)
Merge: 5bd9abd ac05c54
Author: Sena-Jocastron <jocastron@gmail.com>
Date:   Fri Jul 23 14:52:57 2021 -0500

    Fusionada manualmente desde la rama main

commit 5bd9abd66c17fc2d508fcf3e9accf779ef72a048
Author: Sena-Jocastron <jocastron@gmail.com>
Date:   Fri Jul 23 14:03:31 2021 -0500

    Cambiado GIT a Git

commit ac05c54970a0929ffbc5fd0c260599d117e83abb (test)
Author: Sena-Jocastron <jocastron@gmail.com>
Date:   Fri Jul 23 13:54:13 2021 -0500

    Cambiar GIT a Git y GitHub

commit 47d145264a14709fbdf63ea089b19d1a02abf2a5
Author: Sena-Jocastron <jocastron@gmail.com>
Date:   Fri Jul 23 13:09:13 2021 -0500

    Agregado una tercera línea en la rama de feature

commit bd1e47153497584e61a70916843b5b5c80f66870
Author: Sena-Jocastron <jocastron@gmail.com>
Date:   Fri Jul 23 12:42:37 2021 -0500

    Agregada linea adicional al archivo gitversion.txt

commit 770208ae683dfc85e7d2e067f9c0ce68189588d7
Author: Sena-Jocastron <jocastron@gmail.com>
Date:   Fri Jul 23 12:29:50 2021 -0500

    Confirmando gitversion.txt para comenzar el seguimiento de cambios
(END)
```

Parte 7: Integración de Git con GitHub

Hasta ahora, todos los cambios que ha realizado en su archivo se han almacenado en su máquina local. Git se ejecuta localmente y no requiere ningún servidor de archivos central ni servicio de alojamiento basado en la nube. Git permite a un usuario almacenar y administrar archivos localmente.

Aunque Git es útil para un solo usuario, integrar el repositorio local de Git con un servidor basado en la nube como GitHub es útil cuando se trabaja dentro de un equipo. Cada miembro del equipo mantiene una copia en el repositorio de su máquina local y actualiza el repositorio central basado en la nube para compartir cualquier cambio.

Hay bastantes servicios populares de Git, incluyendo GitHub, Stash de Atlassian y GitLab. Debido a que es fácilmente accesible, usará GitHub en estos ejemplos.

Paso 1: Crear una cuenta GitHub.

Si no lo ha hecho anteriormente, vaya a github.com y cree una cuenta de GitHub. Si tiene una cuenta de GitHub, vaya al paso 2.

Paso 2: Inicie sesión en su cuenta de GitHub - Crear un repositorio.

Inicie sesión en su cuenta de GitHub.

Paso 3: Crear un repositorio.

a. Seleccione el botón "**Nuevo repositorio**" o haga clic en el ícono "**+**" en la esquina superior derecha y seleccione "**Nuevo repositorio**".

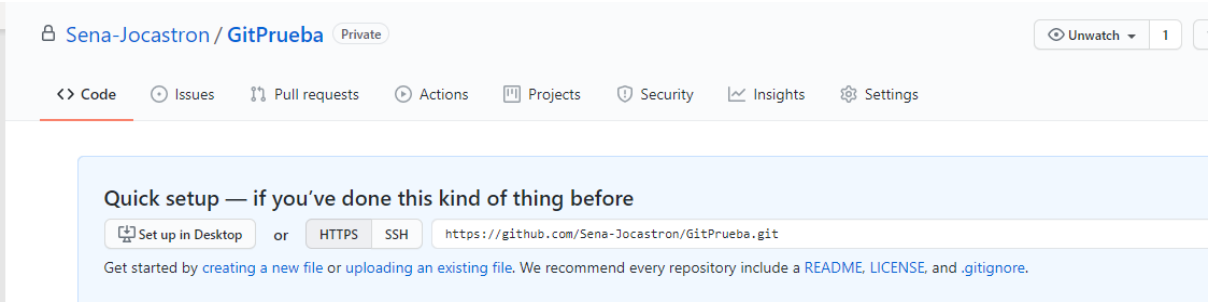
b. Cree un repositorio utilizando la siguiente información:

Nombre del repositorio: **GitPrueba**

Descripción: **Trabajando juntos para aprender a utilizar Git**

Público/Privado: **Privado**

c. Seleccionar: **Crear repositorio**



Paso 4: Crear un nuevo directorio GitPrueba.

a. Borrar el repositorio que está en la carpeta **git-intro**.

MINGW64 ~/git-intro (main)$ **rm -rf .git**

MINGW64 ~/git-intro $

b. Crear un nuevo directorio llamado GitPrueba. El directorio no tiene que coincidir con el nombre como repositorio.

```
MINGW64 ~/git-intro $ mkdir GitPrueba
```

Paso 5: Cambie el directorio a GitPrueba.

Utilice el comando **cd** para cambiar de directorio a GitPrueba.

```
MINGW64 ~/git-intro $ cd GitPrueba
MINGW64 ~/git-intro/GitPrueba $
```

Paso 6: Copie el archivo gitversion.

a. Utilice el comando **cp** para copiar el **gitversion.txt** del directorio principal **git-intro** al subdirectorio **GitPrueba** con el nombre gitprueba.txt. Los dos puntos y una barra diagonal anterior al nombre del archivo indican el directorio principal. El espacio y el punto que siguen al nombre del archivo indica que se debe copiar el archivo en el directorio actual con el mismo nombre de archivo.

```
MINGW64 ~/git-intro/GitPrueba $ cp ../gitversion.txt gitprueba.txt
```

b. Verifique que el archivo se haya copiado con el comando **ls** y el contenido del archivo con el comando **cat**.

```
MINGW64 ~/git-intro/GitPrueba $ ls
gitprueba.txt
MINGW64 ~/git-intro/GitPrueba $ cat gitprueba.txt
Estoy en camino de aprender a utilizar Git
Estoy empezando a entender Git!
Este texto se agregó originalmente en la rama de features
MINGW64 ~/git-intro/GitPrueba $
```

Paso 7: Inicializar un nuevo repositorio de Git.

a. Utilice el comando **git init** para inicializar el directorio actual (GitPrueba) como repositorio de Git. El mensaje que se muestra indica que ha creado un repositorio local dentro del proyecto contenido en el directorio oculto **.git**. Aquí es donde se encuentra todo el historial de cambios.

```
MINGW64 ~/git-intro/GitPrueba $ git init
Initialized empty Git repository in C:/Users/Orlando/git-intro/GitPrueba/.git/
MINGW64 ~/git-intro/GitPrueba (main)$
```

b. A continuación, compruebe sus variables globales de git con el comando **git config --list**.

```
MINGW64 ~/git-intro/GitPrueba (main)$ git config --list
User.name=SampleUser
user.email=sample@example.com
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
MINGW64 ~/git-intro/GitPrueba (main)$
```

c. Si las variables user.name y user.email no coinciden con sus credenciales de GitHub, cámbielas ahora.

```
MINGW64 ~/git-intro/GitPrueba (main)$ git config --global user.name "GitHub Username"
MINGW64 ~/git-intro/GitPrueba (main)$ git config --global user.email "GitHub-email-address"
```

Paso 8: Apunte el repositorio de Git al repositorio de GitHub.

a. Utilice el comando **git remote add** para agregar una URL de Git como un alias remoto. El valor "origen" apunta al repositorio recién creado en GitHub. Utilice su nombre de usuario de GitHub en la ruta URL para *github-username*.

Nota: Su nombre de usuario distingue entre mayúsculas y minúsculas.

```
MINGW64 ~/git-intro/GitPrueba (main)$ git remote add origin https://github.com/github-username/GitPrueba.git
MINGW64 ~/git-intro/GitPrueba (main)$
```

b. Verifique que el **remote** se esté ejecutando en github.com.

```
MINGW64 ~/git-intro/GitPrueba (main)$ git remote --verbose
origen https://github.com/username/GitPrueba.git (buscar)
```

```
origen https://github.com/username/GitPrueba.git (push)
MINGW64 ~/git-intro/GitPrueba (main)$
```

c. Ver el **registro de git**. El error indica que no hay confirmaciones.

```
MINGW64 ~/git-intro/GitPrueba (main)$ git log
fatal: your current branch 'main' does not have any commits yet
MINGW64 ~/git-intro/GitPrueba (main)$
```

Paso 9: Ponga en escena y confirme el archivo gitprueba.txt.

a. Utilice el comando **git add** para poner en escena el archivo gitprueba.txt.

```
MINGW64 ~/git-intro/GitPrueba (main)$ git add gitprueba.txt

MINGW64 ~/git-intro/GitPrueba (main)$
```

b. Utilice el comando **git commit** para confirmar el archivo gitprueba.txt.

```
MINGW64 ~/git-intro/GitPrueba (main)$ git commit -m "Adicione el archivo
gitprueba.txt a GitPrueba"
[main (root-commit) d38f51a] Adicione el archivo gitprueba.txt a GitPrueba
 1 file changed, 3 insertions(+)
 create mode 100644 gitprueba.txt
MINGW64 ~/git-intro/GitPrueba (main)$
```

Paso 10: Verificar la confirmación.

a. Usar el comando **git log** para mirar las confirmaciones.

```
MINGW64 ~/git-intro/GitPrueba (main)$ git log
commit d38f51a0d8b3c7c2880b6def088b27df199b26e3 (HEAD -> main)
Author: Sena-Jocastron <jocastron@gmail.com>
Date:   Fri Jul 23 16:26:53 2021 -0500

    Adicione el archivo gitprueba.txt a GitPrueba
MINGW64 ~/git-intro/GitPrueba (main)$
```

b. Utilice el comando **git status** para ver la información de estado. La frase "working tree clean" significa que Git ha comparado tu lista de archivos con lo que le has dicho a Git, y es una pizarra limpia sin nada nuevo que informar.

```
MINGW64 ~/git-intro/GitPrueba (main)$ git status
On branch main
nothing to commit, working tree clean
MINGW64 ~/git-intro/GitPrueba (main)$
```

Paso 11: Enviar (push) el archivo de Git a GitHub.

Utilice el comando **git push origin main** para enviar (push) el archivo a su repositorio de GitHub. Se le pedirá un nombre de usuario y una contraseña, que será la que utilizó para crear su cuenta de GitHub.

```
MINGW64 ~/git-intro/GitPrueba (main)$ git push origin main
Nombre de usuario para 'https://github.com': nombre de usuario
```

```
Contraseña para 'https://username@github.com':contraseña
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 341 bytes | 341.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Sena-Jocastron/GitPrueba.git
 * [new branch]      main -> main
MINGW64 ~/git-intro/GitPrueba (main)$
```

Nota: Si, después de introducir su nombre de usuario y contraseña, obtiene un error fatal que indica que no se encuentra el repositorio, lo más probable es que haya enviado una URL incorrecta. Deberá revertir su comando **git add** con el comando **git remote rm origin**.

Paso 12: Verifique el archivo en GitHub.

a. Vaya a su cuenta de GitHub y en "Repositorios" seleccione nombre de **usuario/GitPrueba**.

b. Debería ver que el archivo gitprueba.txt se ha agregado a este repositorio de GitHub. Haga clic en el archivo para ver el contenido.