



Colaboración Efectiva en Equipos Distribuidos con Control de Versiones



Estudiante:

Choque Chura Oliver Fran



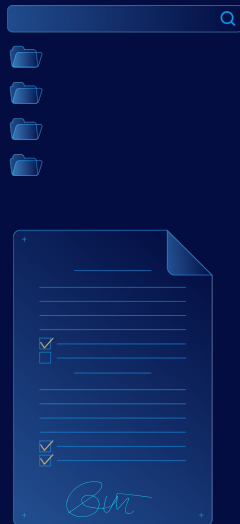


Paso 1: Introducción

1. Presentación:

· Importancia del control de versiones en proyectos de software colaborativos.

- Colaboración Eficiente
- Historial de Cambios
- Detección y Resolución de Conflictos
- Seguimiento de Responsabilidades



· Conceptos clave como repositorios, commits, ramas y fusiones.

- Repositorios
- Commits
- Ramificación (Branching)
- Fusiones (Merges)





Paso 2: Configuración del Entorno

2. Creación de Cuentas:

- Crear cuenta en GitHub.

<https://github.com/>

```
Welcome to GitHub!
Let's begin the adventure

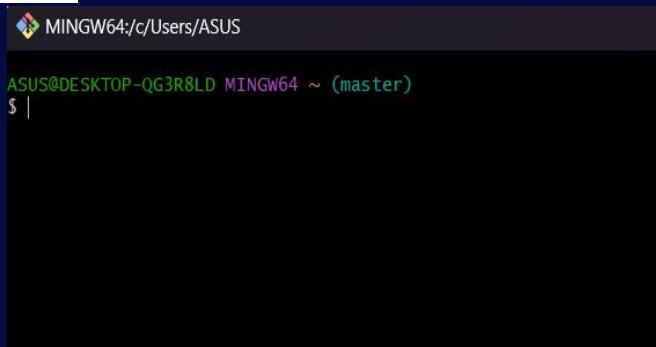
Enter your email
✓ espitiahumanez92@hotmail.com

Create a password
✓ .....

Enter a username
✓ humanez92
```

- Configurar Git en máquinas locales.

<https://git-scm.com/download/win>



Configuración de Usuario:

Configurar su nombre de usuario y dirección de correo electrónico en Git.

```
git config --global user.name "Nombre Apellido"
```

```
git config --global user.email "correo@example.com"
```

Verificación de la Configuración:

```
git config --global --list
```



3. Creación de Repositorio:



- Crear un repositorio en GitHub.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner *

Repository name *

OliverChoque

/ Control de Versiones

✔ Your new repository will be created as **Control-de-Versiones**.
The repository name can only contain ASCII letters, digits, and the characters `.`, `-`, and `_`.

Great repository names are short and memorable. Need inspiration? How about [glowing-octo-chainsaw](#)?

Description (optional)

Control de Versiones y Colaboración en Equipo

☒ Public

Anyone on the Internet can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

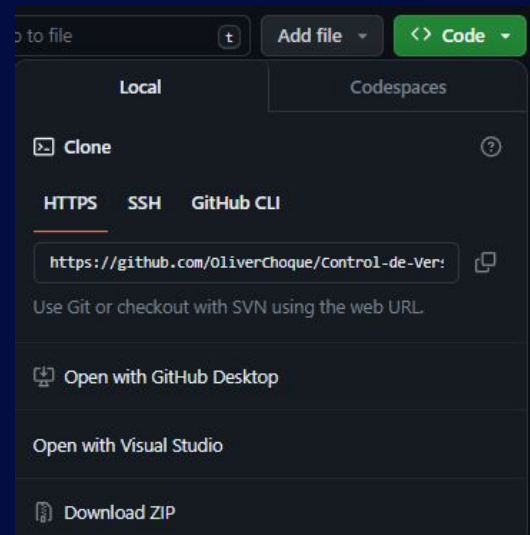
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

① You are creating a public repository in your personal account.

Create repository

- Compartir la URL del repositorio.



<https://github.com/OliverChoque/Control-de-Versiones.git>





Paso 3: Trabajo Individual

4. Clonación del Repositorio:

- Instrucciones para clonar el repositorio en la máquina local.

Obtener la URL del Repositorio:

→ <https://github.com/OliverChoque/Control-de-Versiones.git>

Abrir la Terminal o Consola:

En las máquinas locales, debemos abrir la terminal o consola.

Navegar al Directorio de Trabajo:

Utilizando el comando `cd`, navegar al directorio donde desean clonar el repositorio. Creamos una carpeta `PruebaCV` en el escritorio.

Clonar el Repositorio:

Ejecutar el siguiente comando en la terminal, reemplazando `<URL>` con la URL copiada del repositorio:

```
git clone <URL>
```

Esto descargará una copia del repositorio en la máquina.



```
MINGW64/c/Users/ASUS/desktop/PruebaCV
ASUS@DESKTOP-QG3R8LD MINGW64 ~ (master)
$ cd desktop
ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop (master)
$ cd PruebaCV
ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV (master)
$ git clone https://github.com/OliverChoque/Control-de-Versiones.git
bash: https://github.com/OliverChoque/Control-de-Versiones.git
Cloning into 'Control-de-Versiones'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
bash: bash:: command not found
ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV (master)
$ |
```

Control-de-Versiones

README.md





5. Realización de Cambios Locales:

Instrucciones para Realizar Cambios Locales:

Navegar al Directorio del Repositorio:

Utilizando el comando `cd`, navegar al directorio recién clonado:

`cd nombre-del-repositorio`

Crear una Rama (Opcional):

Opcionalmente, pueden crear una rama para trabajar en su tarea específica. Esto ayuda a mantener el código principal limpio.

`git checkout -b nombre-de-la-rama`

Realizar Cambios en Archivos:

Usar un editor de código para realizar cambios en los archivos específicos asignados.

Verificar el Estado de los Cambios:

Verificar qué archivos han cambiado y están listos para ser confirmados.

`git status`

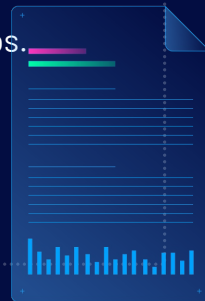
Agregar Cambios al Área de Staging:

Agregar los archivos modificados al área de staging antes de confirmar los cambios.

`git add nombre-del-archivo`

O agregar todos los archivos cambiados.

`git add .`





5. Realización de Cambios Locales:

Realizar un Commit Local:

Confirmar los cambios con un mensaje descriptivo.

`git commit -m "Descripción del cambio"`

Repetir si es necesario:

Repetir los pasos 3-6 según sea necesario para realizar más cambios.

Subir los Cambios al Repositorio Remoto (Push):

Si se creó una rama, subir los cambios a esa rama específica:

`git push origin nombre-de-la-rama`

Si se está trabajando en la rama principal:

`git push origin main`

Estos pasos permitirán que se realice cambios en su propia copia del repositorio, preparándonos para la colaboración en la siguiente etapa del proyecto.





MINGW64:/c/Users/ASUS/desktop/PruebaCV/Control-de-Versiones

```
ASUS@DESKTOP-QG3R8LD MINGW64 ~ (master)
$ cd desktop

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop (master)
$ cd PruebaCV

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV (master)
$ cd Control-de-Versiones

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV/Control-de-Versiones (main)
$ git checkout -b Ejemplo-CV1
Switched to a new branch 'Ejemplo-CV1'

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV/Control-de-Versiones (Ejemplo-CV1)
1)
$ git status
On branch Ejemplo-CV1
nothing to commit, working tree clean

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV/Control-de-Versiones (Ejemplo-CV1)
$ git add .

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV/Control-de-Versiones (Ejemplo-CV1)
$ git commit -m "Cambio 1"
On branch Ejemplo-CV1
nothing to commit, working tree clean

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV/Control-de-Versiones (Ejemplo-CV1)
$ git push origin Ejemplo-CV1
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'Ejemplo-CV1' on GitHub by visiting:
remote:   https://github.com/OliverChoque/Control-de-Versiones/pull/new/Ejemplo-CV1
remote:
To https://github.com/OliverChoque/Control-de-Versiones.git
 * [new branch]      Ejemplo-CV1 -> Ejemplo-CV1

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV/Control-de-Versiones (Ejemplo-CV1)
$ git push origin main
Everything up-to-date
```





Branches

[New branch](#)

Overview **Yours** Active Stale All

Q Search branches...

Default

Branch	Updated	Check status	Behind	Ahead	Pull request
<code>main</code>	4 hours ago				Default

Your branches

Branch	Updated	Check status	Behind	Ahead	Pull request
<code>Ejemplo-CV1</code>	8 minutes ago		0	0	

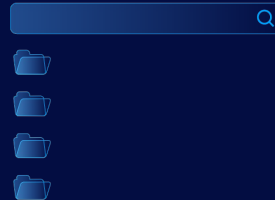
Active branches

Branch	Updated	Check status	Behind	Ahead	Pull request
<code>Ejemplo-CV1</code>	8 minutes ago		0	0	





Paso 4: Trabajo Colaborativo



6. Creación de Ramas:

- Instrucciones para crear ramas individuales.

Asegurarse de estar en la Rama Principal:

Antes de crear una nueva rama, es buena práctica asegurarse de estar en la rama principal (por ejemplo, main o master).

```
git checkout master
```

Crear una Rama Individual:

Crear su propia rama para trabajar en la tarea asignada.

```
git checkout -b nombre-de-la-rama
```





MINGW64:/c/Users/ASUS/desktop/PruebaCV



ASUS@DESKTOP-QG3R8LD MINGW64 ~ (master)

\$ cd desktop

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop (master)

\$ cd PruebaCV

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV (master)

\$ git checkout master

Already on 'master'

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV (master)

\$ git checkout -b Rama-CV1

Switched to a new branch 'Rama-CV1'

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV (Rama-CV1)

\$



MINGW64:/c/Users/ASUS/desktop/PruebaCV/Control-de-Versiones

```
ASUS@DESKTOP-QG3R8LD MINGW64 ~ (Rama-CV1)
$ cd desktop

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop (Rama-CV1)
$ cd PruebaCV

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV (Rama-CV1)
$ cd Control-de-Versiones

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV/Control-de-Versiones (Ejemplo-CV
1)
$ git init
Reinitialized existing Git repository in C:/Users/ASUS/Desktop/PruebaCV/Control-
de-Versiones/.git/

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV/Control-de-Versiones (Ejemplo-CV
1)
$ touch Archivo-Nuevo1

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV/Control-de-Versiones (Ejemplo-CV
1)
$ git add Archivo-Nuevo1

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV/Control-de-Versiones (Ejemplo-CV
1)
$ git commit -m "Archivo Nuevo 1"
[Ejemplo-CV1 fcbf181] Archivo Nuevo 1
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Archivo-Nuevo1

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV/Control-de-Versiones (Ejemplo-CV
1)
$ git push -u origin master
error: src refspec master does not match any
error: failed to push some refs to 'https://github.com/OliverChoque/Control-de-V
ersiones.git'

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV/Control-de-Versiones (Ejemplo-CV
1)
$ git push -u origin Ejemplo-CV1
git: 'push' -u is not a git command. See 'git --help'.

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV/Control-de-Versiones (Ejemplo-CV
1)
$ git push -u origin Ejemplo-CV1
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 317 bytes | 317.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/OliverChoque/Control-de-Versiones.git
 2668bd7..fcbf181 Ejemplo-CV1 -> Ejemplo-CV1
branch 'Ejemplo-CV1' set up to track 'origin/Ejemplo-CV1'.

ASUS@DESKTOP-QG3R8LD MINGW64 ~/desktop/PruebaCV/Control-de-Versiones (Ejemplo-CV
```





7. Realización de Cambios Colaborativos:

- Trabajar en equipo para realizar cambios en diferentes ramas y resolverán posibles conflictos.

Colaborar en el Repositorio Principal:

Realizar cambios en sus propias ramas para evitar conflictos directos con otros miembros del equipo.

Agregar Cambios al Área de Staging:

Antes de confirmar los cambios, agregar los archivos modificados al área de staging.

```
git add nombre-del-archivo
```

O agregar todos los archivos cambiados:

```
git add .
```

Realizar un Commit Local:

Confirmar los cambios con un mensaje descriptivo.

```
git commit -m "Descripción del cambio"
```

Subir los Cambios a la Rama Remota:

Subir los cambios a la rama específica creada por el estudiante.

```
git push origin nombre-de-la-rama
```





Manejo de Conflictos (si es necesario):

Si dos o más han realizado cambios en la misma parte del código, puede ocurrir un conflicto.

Ejecutar `git pull origin nombre-de-la-rama` para obtener los cambios más recientes antes de intentar empujar los propios.

Resolver los conflictos en los archivos marcados y luego realizar el commit y push.

Solicitar Fusiones (Pull Requests):

Una vez que ha completado su tarea en su rama, puede crear una solicitud de fusión (pull request) en GitHub para fusionar sus cambios en la rama principal.

Revisar y Fusionar:

Otro miembro del equipo (o el responsable del proyecto) debe revisar los cambios propuestos y fusionarlos en la rama principal si todo está correcto.

Siguiendo estos pasos, los integrantes podrán colaborar de manera efectiva en el proyecto, trabajando en ramas individuales y resolviendo cualquier conflicto que pueda surgir durante el proceso. La utilización de ramas individuales facilita la gestión de cambios y contribuye a un flujo de trabajo más organizado.



Paso 5: Fusionando Cambios



8. Pull Requests:

· Cada integrante creará un Pull Request para fusionar sus cambios en la rama principal.

Creación de Pull Requests:

Ingresar a GitHub:

Cada integrante debe ingresar a GitHub y navegar al repositorio del proyecto.

Ir a la Rama Correcta:

Asegurarse de estar en la rama que contiene los cambios que desean fusionar (la rama creada en el paso anterior).

Hacer clic en "New Pull Request":

En la interfaz del repositorio, hacer clic en el botón "New Pull Request" o "Crear solicitud de fusión".

Seleccionar Ramas:

Asegurarse de que la rama base sea la rama principal (main o master) y la rama de comparación sea la rama que contiene los cambios.

Crear la Solicitud de Fusión:

Hacer clic en "Create Pull Request".

Proporcionar Información:

Proporcionar un título y una descripción clara y concisa que explique los cambios realizados.

Crear el Pull Request:

Hacer clic en "Create Pull Request" para finalizar la creación del Pull Request.






Label issues and pull requests for new contributors


[Dismiss](#)

Now, GitHub will help potential first-time contributors [discover issues](#) labeled with [good first issue](#)

Filters ▾

Q is:pr is:open

 Labels **9**

 Milestones **0**

New pull request



Welcome to pull requests!

Pull requests help you collaborate on code with other people. As pull requests are created, they'll appear here in a searchable and filterable list. To get started, you should [create a pull request](#).

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#) or [learn more about diff comparisons](#).



base: main ▾




compare: Ejemplo-CV1 ▾



There isn't anything to compare.

main and Ejemplo-CV1 are identical.

 Showing **0** changed files with **0** additions and **0** deletions.

Split

Unified

- Revisión y discusión de los Pull Requests por parte de los demás integrantes.



Revisión y Discusión:

Revisar Cambios:

Otros miembros del equipo deben revisar los cambios propuestos en el Pull Request.

Discusión y Comentarios:

Utilizar la sección de comentarios del Pull Request para discutir los cambios, hacer preguntas y proporcionar retroalimentación.

Solicitar Cambios (si es necesario):

Si se identifican problemas o se sugieren mejoras, los revisores pueden solicitar cambios y agregar comentarios directamente en el Pull Request.

Aprobar el Pull Request:

Si los cambios son satisfactorios, un miembro del equipo puede aprobar el Pull Request.

Fusionar el Pull Request:

Una vez que el Pull Request ha sido aprobado y no hay conflictos, puede fusionarse con la rama principal.

Eliminar la Rama (opcional):

Después de la fusión, se puede optar por eliminar la rama utilizada para el desarrollo si ya no es necesaria.

Repetir para Otros Integrantes:

Repetir este proceso para cada integrante que haya trabajado en una rama diferente.

Este paso permite una revisión colaborativa de los cambios propuestos por cada persona antes de la integración en la rama principal. Facilita la detección de posibles problemas y asegura que todos los miembros del equipo estén al tanto de las modificaciones realizadas en el proyecto.





Paso 6: Resolución de Conflictos

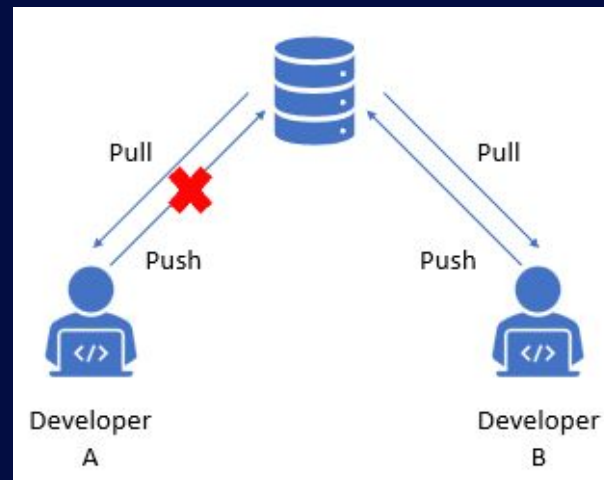
9. Simulación de Conflictos:

- Introducción a conflictos de fusión.

Ocurren cuando dos o más ramas contienen cambios en las mismas líneas de código y Git no puede determinar automáticamente cuál versión debe prevalecer.

Causas Comunes:

- Cambios simultáneos en la misma línea de código.
- Fusiones paralelas de ramas sin sincronización adecuada.





- Los integrantes provocarán conflictos y aprenderán a resolverlos.

Consejos para Evitar Conflictos:

Comunicación:

Mantener una comunicación efectiva sobre las áreas en las que se está trabajando.

Fusiones Frecuentes:

Realizar fusiones frecuentes con la rama principal para mantenerse actualizado.

Pequeñas Actualizaciones:

Dividir tareas en unidades más pequeñas para reducir la probabilidad de conflictos.

Revisión Previa:

Revisar los cambios realizados por otros miembros del equipo antes de fusionar.

Este paso ayuda a los integrantes a comprender cómo manejar conflictos de fusión y promueve la importancia de la comunicación y la coordinación en un entorno de desarrollo colaborativo.





Paso 7: Informe y Entrega

10. Documentación:

- Documentará del proceso seguido, los desafíos encontrados y las soluciones implementadas en un informe.
- Subir el informe al repositorio y aula virtual utilizando un formato proporcionado.

Formato para el informe que incluya secciones como:

- Introducción
- Configuración del Entorno
- Trabajo Individual
- Trabajo Colaborativo
- Fusionando Cambios
- Resolución de Conflictos
- Conclusiones y Lecciones Aprendidas

- Inclusión de Capturas de Pantalla
- Desafíos y Soluciones
- Instrucciones de Subida

Subir el Archivo al Repositorio:

Subir el archivo del informe al repositorio local.

```
git add informe.txt # Reemplazar 'informe.txt' con el nombre del archivo
```

```
git commit -m "Agregar informe sobre el proceso seguido"
```

```
git push origin nombre-de-la-rama
```

Crear un Pull Request:

Si se ha trabajado en una rama diferente para el informe, crear un Pull Request para fusionar los cambios en la rama principal.





Gracias

