

MANUAL DE USO GITLAB

VERSIONAMIENTO

V1.4

16/01/2019

Ante cualquier consulta contactarse a:

Soporte.ControldeVersiones.cl@telefonica.com / +56 226913121 ; +56 323819046

Guardia Servicio (Horario inhábil): +569 94497049

TABLA DE CONTENIDOS

1. REPOSITORIO GITLAB	3
1.1. ¿Qué es GitLab?.....	3
1.2. ¿Con qué cuenta de usuario debo acceder al repositorio GitLab?	3
1.3. ¿Cómo creo un repositorio para mi proyecto?, ¿Con quién me contacto?	3
1.4. Definiciones de estructura y nomenclaturas dentro del repositorio	3
1.5. Nomenclatura de Branches dentro de un repositorio.....	5
2. CONEXIÓN A LA INTERFAZ WEB GITLAB.....	5
3. VERSIONAMIENTO EN GITLAB.....	6
3.1. Creación de Directorios y Archivos	6
3.2. Subir archivos a GitLab.....	9
3.3. Nomenclatura de directorios dentro de una Branch.....	11
4. CREACION DE BRANCH FEATURE.....	12
5. GENERACION DE MERGE REQUEST	13
5.1. Creando Merge Request	13
5.2. Aprobación de un Merge Request	16
6. CREACIÓN DE TAG.....	17
7. UTILIZACION DE TORTOISE GIT (OPCIONAL)	19
7.1. ¿Cómo conectarnos a TortoiseGit?	19
7.2. ¿Cómo conectarnos a TortoiseGit?	19
7.3. Instalar Cliente GIT para Windows.....	21
7.4. Crear repositorio en directorio local.....	23
7.5. Versionando en GitLab con TortoiseGit.....	25
8. CONTROL DE VERSIONES DEL DOCUMENTO.....	32

1. REPOSITORIO GITLAB

1.1. ¿Qué es GitLab?

GitLab es un servicio web de control de versiones y desarrollo de *software* colaborativo basado en **Git**. Además de gestor de repositorios, el servicio ofrece también alojamiento de *wikis* y un sistema de seguimiento de errores, todo ello publicado bajo una licencia de código abierto.

1.2. ¿Con qué cuenta de usuario debo acceder al repositorio GitLab?

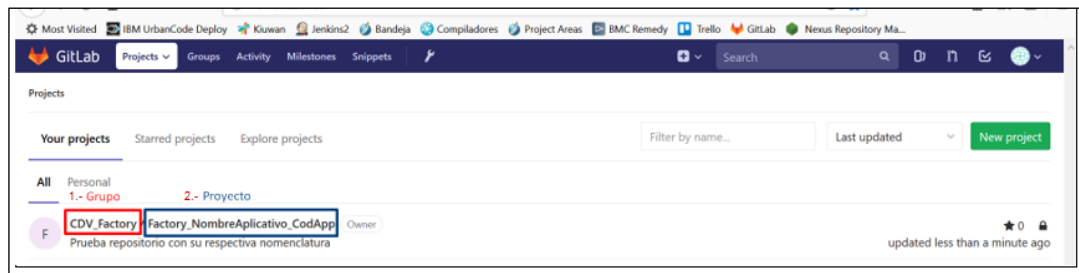
Actualmente se tiene configurado un LDAP para la autenticación de usuarios, por ende, para poder acceder a **GitLab** el usuario deberá contar con su respectivo usuario “**TCHILE**” y **enrolarse en el servicio**, en caso de no tener, deberá solicitarlo al JP Telefónica a cargo del proyecto.

1.3. ¿Cómo creo un repositorio para mi proyecto?, ¿Con quién me contacto?

Si es un proyecto o célula **NUEVO**, primero deben contactar a Marcelo Morales (marcelo.morales@telefonica.com) para gestionar el código del aplicativo, él enviará la solicitud al equipo control de versiones para la creación del repositorio para el proyecto.

1.4. Definiciones de estructura y nomenclaturas dentro del repositorio

Grupos – Proyecto (Repositorio)



Grupos

A cada grupo se le asocia un conjunto de usuarios, y cada usuario tiene un nivel de permisos sobre los proyectos. Los grupos, se dividirán en dos tipos, uno relacionado a la Factory y otro a una célula.

Tipos de grupos:

- Grupo asociado a una Factory:

CDV_{Nombre Factory}

- Grupo asociado a una Célula:

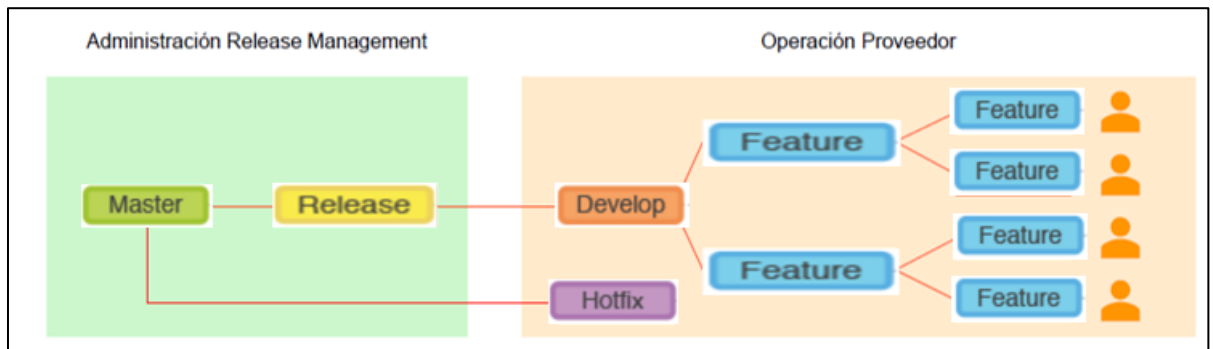
CDV_CELULA_{Nombre Célula}

Proyecto

El Repositorio deberá estar definido por “Nombre Factory”, “Nombre Aplicativo” y “Código del aplicativo”, tal como se muestra a continuación:

{Nombre Factory}_{Nombre Aplicativo}_{Código Aplicativo}

La estructura del proyecto (repositorio), está definida según la siguiente imagen:



- **MASTER:** Se encuentran las RELEASES estables del aplicativo. Esta es la rama que un usuario típico descargará para usar, por lo que todo lo que hay en esta rama debería ser funcional. Sin embargo, puede que las últimas mejoras introducidas en el software no estén disponibles todavía en esta rama.
- **RELEASE:** Se crean cuando se publicará la siguiente versión del software y surgen de la rama DEVELOP. Es la rama que se utilizará para realizar el paso a PRE-PRODUCCIÓN y finalmente a PRODUCCIÓN. Una vez instalado, se integra en máster y se etiqueta (TAG) con el número de versión correspondiente. En caso de realizar una modificación en RELEASE, deberán integrarse en la rama DEVELOP.
- **DEVELOP:** En ella se van integrando todas las nuevas características hasta la siguiente RELEASE.
- **FEATURE:** Cada nueva mejora o característica que vayan a introducir en el aplicativo tendrá una rama que contendrá su desarrollo. Nace de la rama DEVELOP y una vez completado el desarrollo de la mejora, se vuelven a integrar en DEVELOP y al mismo tiempo ser eliminadas.
- **HOTFIX:** Si el código contiene bugs críticos que es necesario corregir de manera inmediata, es posible crear una rama HOTFLIX a partir de la última versión contenida en la rama MASTER. Una vez corregido, se integrará en MASTER, con su etiqueta (TAG) de versión correspondiente, al igual que en DEVELOP y FEATURE sólo en caso de aplicar.

1.5. Nomenclatura de Branches dentro de un repositorio

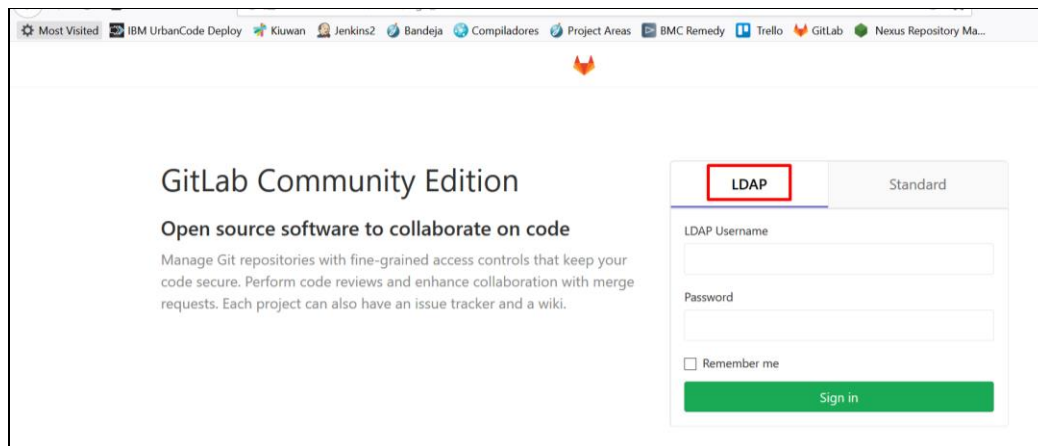
Las ramas o Branches estan definidas y deben ser creadas según lo detallado a continuación:

- **Master:** master
- **HotFix:** hotFix_{Ticket}_{DDMMYYYY}
- **Reléase:** release_{DDMMYYYY}
- **Develop:** develop
- **Feature:** {OSI2018-XXXX} (Puede ser cualquier nombre deseado que identifique las modificaciones, pero se recomienda el mencionado anteriormente)

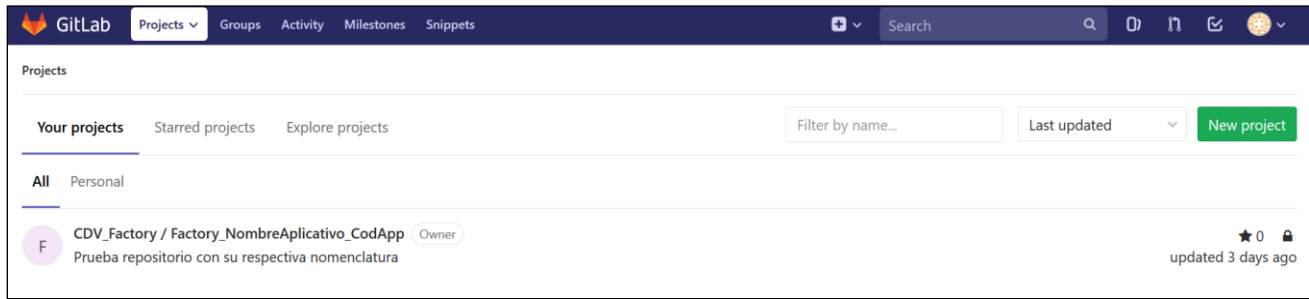
2. CONEXIÓN A LA INTERFAZ WEB GITLAB

Para conectarnos a través de la interfaz web realizar los siguientes pasos:

2.1. Ingresar a la ruta: <http://git.tchile.local/>



- 2.2. En la pestaña LDAP, ingresar credenciales de usuario TCHILE en los campos de “username” y “password”, luego clic en “Sign In”.
- 2.3. Luego en el HOME de GitLab, se mostrarán todos los proyectos en los cuáles el usuario tiene permisos, según grupo y aplicativos asociados.

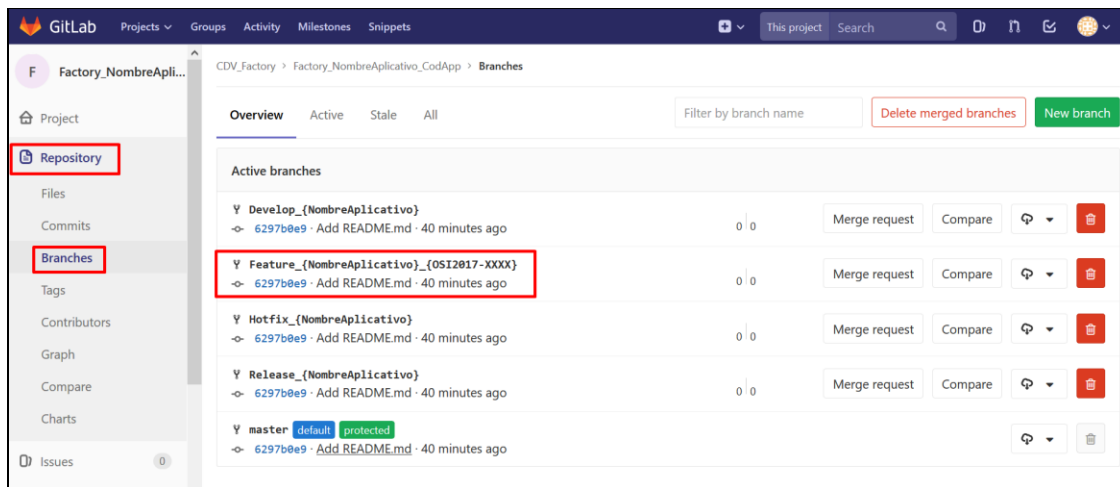


3. VERSIONAMIENTO EN GITLAB

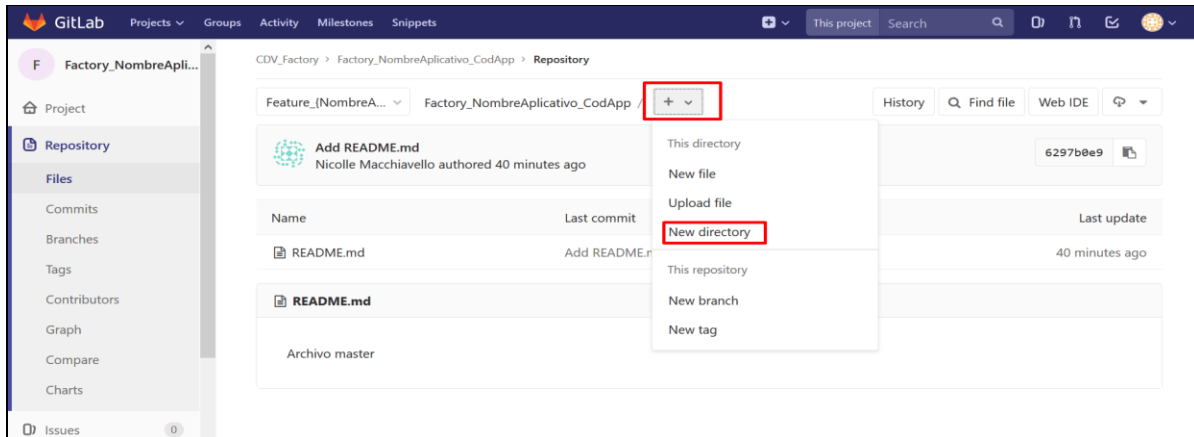
Cada equipo debe trabajar sobre las ramas FEATURE creada desde DEVELOP.

3.1. Creación de Directorios y Archivos

- 3.1.1. Para versionar dentro del repositorio del aplicativo debes ir al menú del lado izquierdo, acceder a *Repository > Branches*, acá encontraras todas las ramas que contiene el aplicativo y puedes crear nuevas en base a la rama DEVELOP.



- 3.1.2. Selecciona la rama FEATURE, se abrirá una nueva ventana, en ella haz clic en el botón “+” y “New Directory”, así agregaremos una nueva carpeta.



3.1.3. En “Directory Name” asignamos el nombre de la carpeta o directorio, en “Commit Message” indicamos la descripción del contenido de la carpeta y en “Target Branch” verificamos que la rama a cargar sea la FEATURE creamos con clic en botón verde “Create Directory”.

Create New Directory

Directory name

CarpetaPrueba

Commit message

Primera carpeta de la rama desarrollo

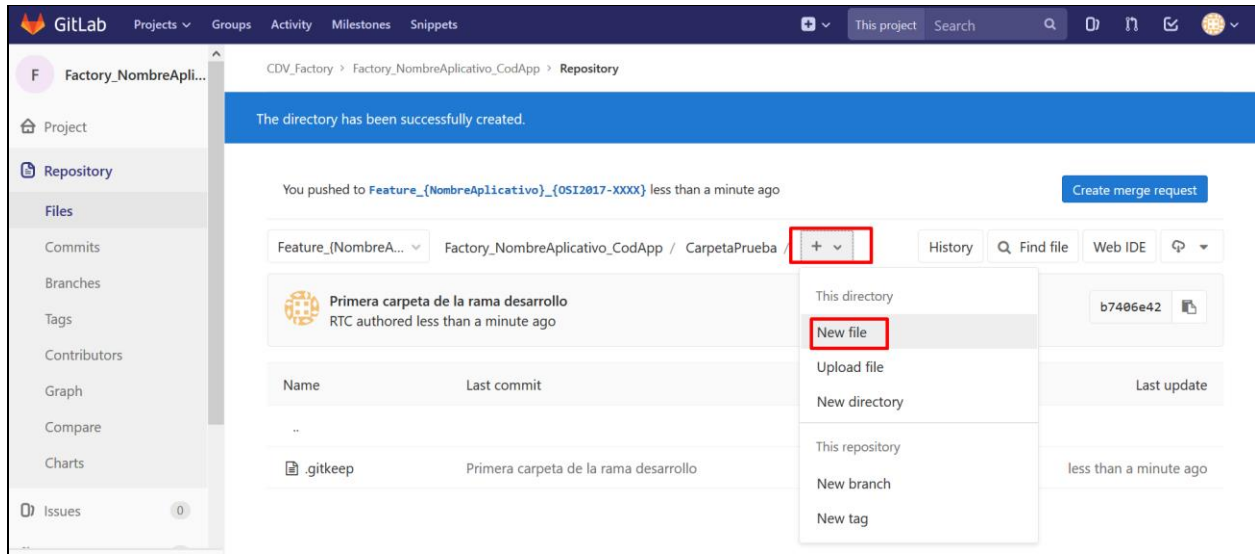
Target Branch

Feature_{NombreAplicativo}_{OSI2017-XXXX}

Create directory

Cancel

3.1.4. Ahora podemos crear archivos dentro de la carpeta creada anteriormente, de la misma forma haz clic en “+” luego en “New File”



- 3.1.5. En el primer campo debemos agregar el nombre del archivo y su contenido en el segundo campo, “commit message” corresponde a la descripción para el commit del archivo y al igual que antes verificamos que estemos cargando la rama FEATURE.

CDV_Factory > Factory_NombreAplicativo_CodApp > Repository

New file Template Choose type

Feature_{NombreAplicativo}_{OSI2017-XXXX} Archivo1 Nombre Del Archivo

1 Prueba Primer Archivo

Contenido Del Archivo

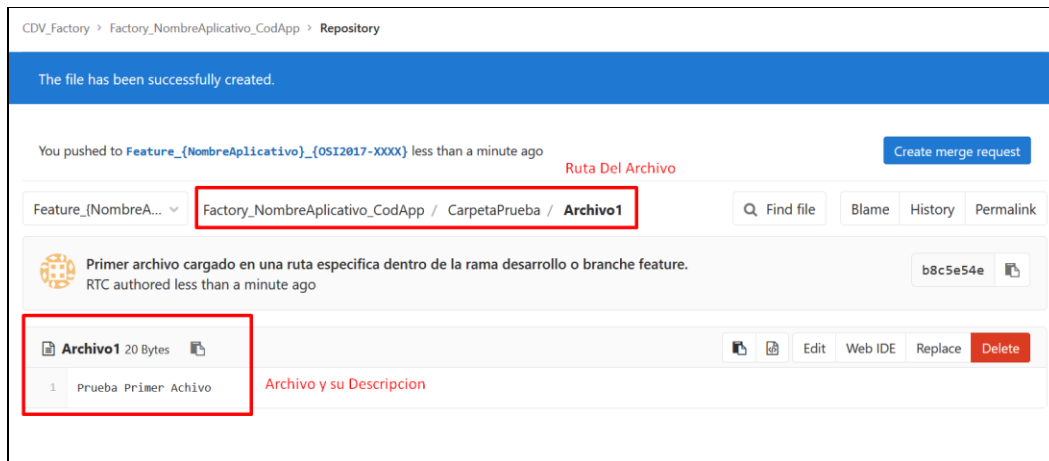
Commit message Primer archivo cargado en una ruta especifica dentro de la rama desarrollo o branche feature.

Descripcion Del Archivo

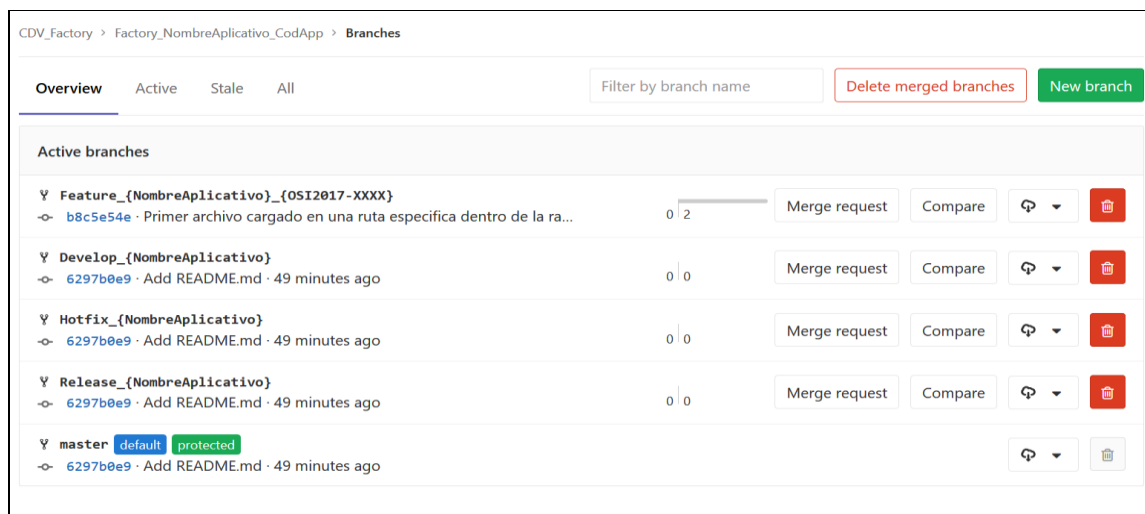
Target Branch Feature_{NombreAplicativo}_{OSI2017-XXXX} Verificar Rama a Cargar

Commit changes Cancel

Con esto tenemos creado el archivo en nuestra rama FEATURE.

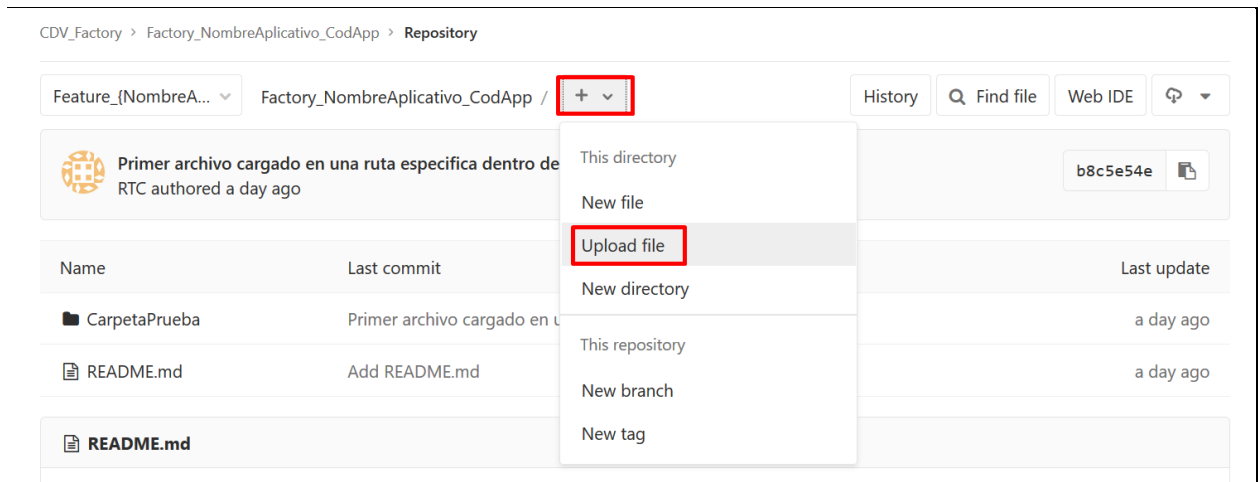


3.1.6. Si volvemos al menú principal del repositorio, “Repository > Branches” podemos ver que la rama fue actualizada.



3.2. Subir archivos a GitLab

3.2.1. Entramos a la rama FEATURE y podemos subir nuestros archivos ya creados de manera local, clic en “+” luego “Upload file”



- 3.2.2. Nos aparece la siguiente ventana para cargar nuestro archivo, en el primer campo “*click to upload*” para agregar el archivo de nuestro equipo local, “*Commit Message*” agregar un comentario sobre el archivo y “*Target Branch*” para verificar que el archivo se suba en la rama FEATURE.

Upload New File

Attach a file by drag & drop or [click to upload](#)

Commit message

Upload New File

Target Branch

Feature_{NombreAplicativo}_{OSI2017-XXXX}

Upload file

Cancel

Upload New File

0 b

Archivo2.txt

Remove file

Commit message

Archivo traído desde escritorio

Target Branch

Feature_{NombreAplicativo}_{OSI2017-XXXX}

Upload file

Cancel

Y listo tenemos nuestro “Archivo2” cargado de manera local.

CDV_Factory > Factory_NombreAplicativo_CodApp > Repository

You pushed to Feature_{NombreAplicativo}_{OSI2017-XXXX} 3 minutes ago

Create merge request

Feature_{NombreA...}

Factory_NombreAplicativo_CodApp / +

History

Find file

Web IDE

Archivo traído desde Escritorio

Niccolle Macchiavello authored 3 minutes ago

43a03775

Name	Last commit	Last update
CarpetaPrueba	Primer archivo cargado en una ruta específica dentro de la ra...	a day ago
Archivo2.txt	Archivo traído desde Escritorio	3 minutes ago
README.md	Add README.md	a day ago

3.3. Nomenclatura de directorios dentro de una Branch

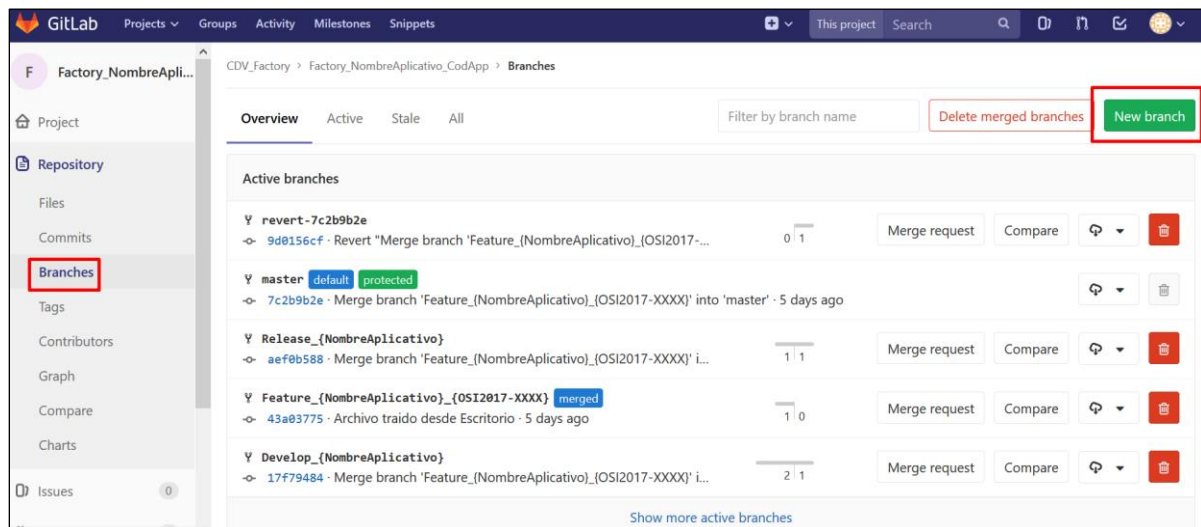
Dentro de cada Branch se debe versionar creando carpetas según la estructura definida a continuación:

- DOCS --> Documentos de apoyo al aplicativo y/o su funcionamiento
- PROPERTIES --> Archivos Properties según entorno CERT/BUAT/PRO
- SQL --> Archivos de Base de datos
- BATCH --> Archivos unix, de tipo ksh ejecutables
- WEB – Código fuente de aplicaciones Web

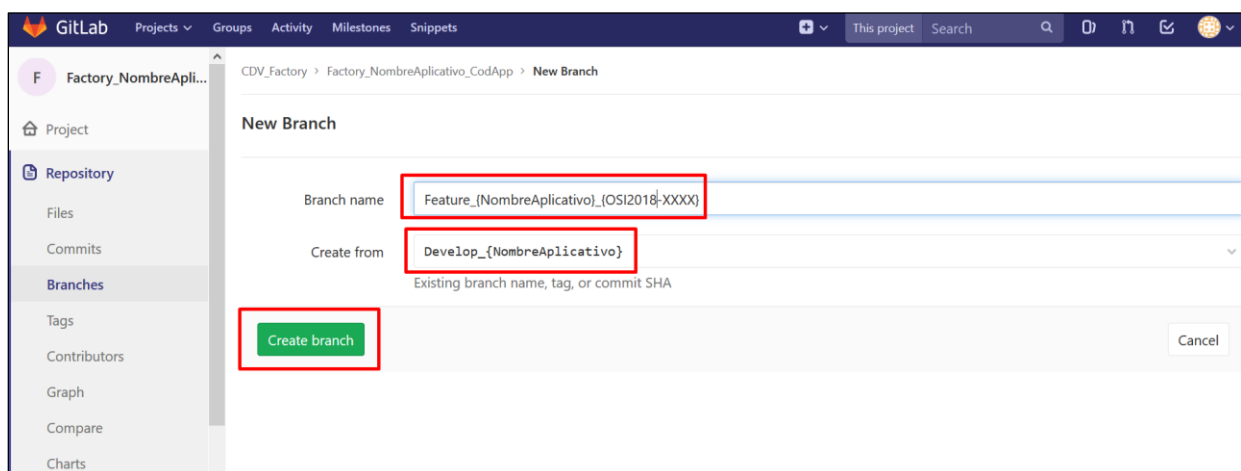
4. CREACION DE BRANCH FEATURE

Podemos crear cuantas ramas sean necesarias para desarrollar, estas deben ser creadas desde la Branch DEVELOP, debido a que esta tiene la última actualización del repositorio autorizada por el líder funcional.

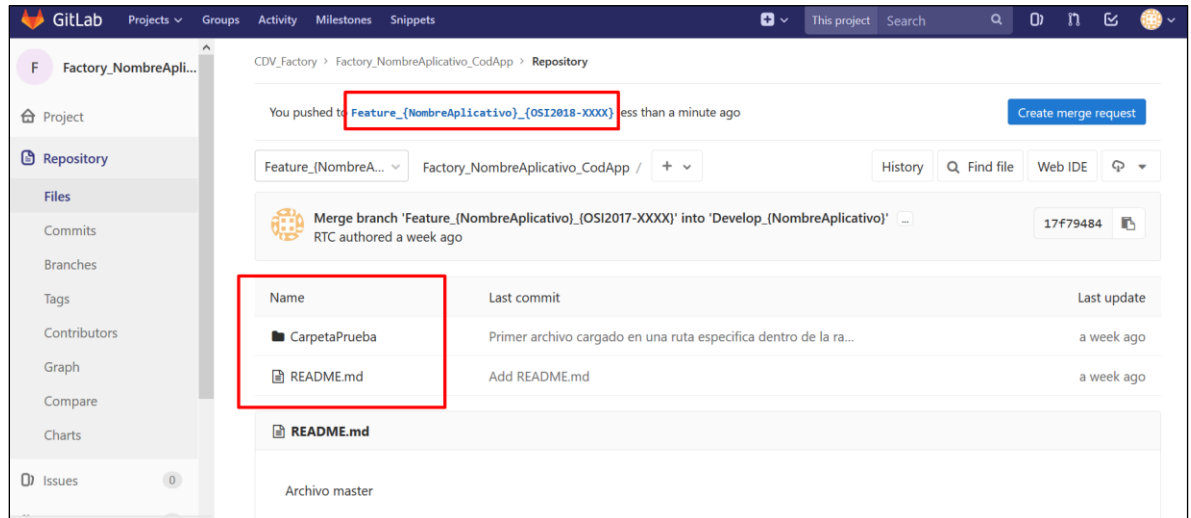
4.1. Para crear la rama vamos al menú izquierdo *Repository* > *Branches* y presionamos el botón verde en esquina superior derecha “*New Branch*” tal como lo muestra la imagen a continuación.



4.2. Asignamos un nuevo nombre acorde a las nomenclaturas definida anteriormente en “*Branch Name*” y creamos desde DEVELOP, click en “*Create Branch*”.



Y así es como creamos una branch FEATURE desde DEVELOP, por ende, la nueva rama contiene la última actualización del repositorio.

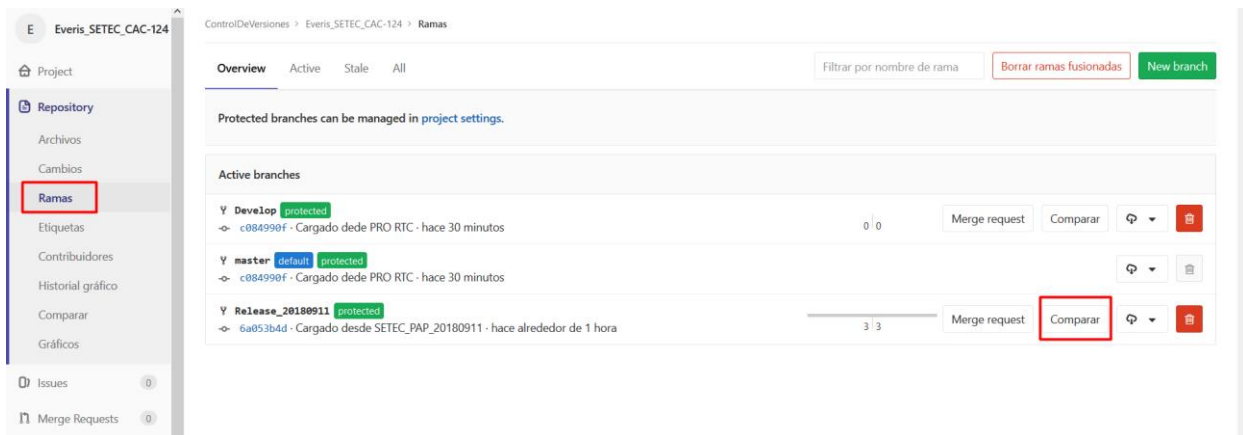


5. GENERACION DE MERGE REQUEST

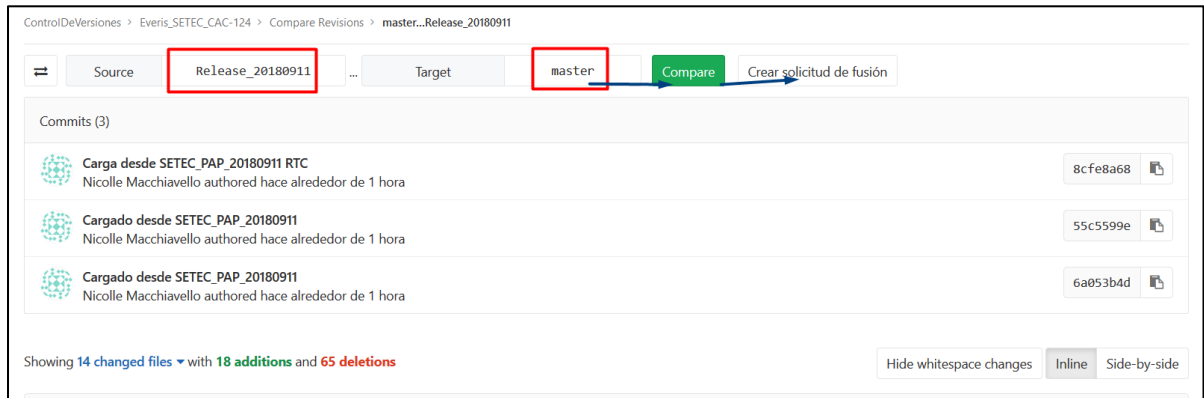
Antes de realizar el merge debemos tener claro de donde deseamos copiar nuestros archivos y hacia que branch se verá reflejado.

5.1. Creando Merge Request

5.1.1. Para crear vamos al menú izquierdo, “*Ramas*”, nos posicionamos en la rama que actualizamos y botón derecho “*Coaparar*”



- 5.1.2. En “Source” indicamos nuestra branch y en “Target” seleccionar branch que deseamos actualizar con este Merge, teniendo esto indicado, haz clic en “Compare”. Esto mostrara todos los cambios entre las ramas, después de validar presionas “Crear Solicitud de fusión”.



- 5.1.3. En “Title” debemos indicar el título de este Merge ejemplo “Merge de Reléase a Master”, en “Description” damos una breve descripción que sea significativa sobre el contenido del Merge.

ControlDeVersiones > Everis_SETEC_CAC-124 > Merge Requests

New Merge Request

From **Release_20180911** into **master**

Title

Release 20180911 a master

Start the title with **WIP:** to prevent a **Work In Progress** merge request from being merged before it's ready.

Add [description templates](#) to help your contributors communicate effectively!

Description

Write Preview

Actualziacion ticket CRQ000000123456

[Markdown](#) and [quick actions](#) are supported

- 5.1.4. En “Assignee” debemos seleccionar el usuario que aprobara este Merge, en el caso de Feature -> Develop debe aprobar el líder funcional y en caso de Reléase -> master quien aprueba es el equipo de control de

versiones, en “*Target Branch*” revisamos que indique la rama que deseamos actualizar , de no ser así haz clic en “*Change Branches*” para seleccionar el destino del Merge.

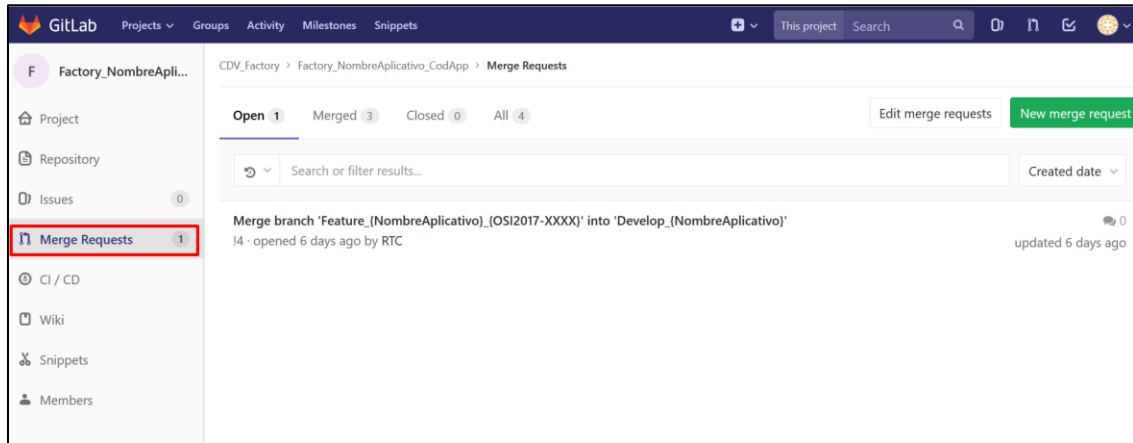
- 5.1.5. Si tuvimos que indicar de forma manual el “*Target Branch*” nos lleva a otra ventana donde debemos indicar el Merge deseado.

- 5.1.6. Una vez completado y validado lo anterior, presionamos botón “*Submit Merge Request*”. Ahora solo debemos avisar a nuestro **Líder Funcional** para su **aprobación del Merge**.

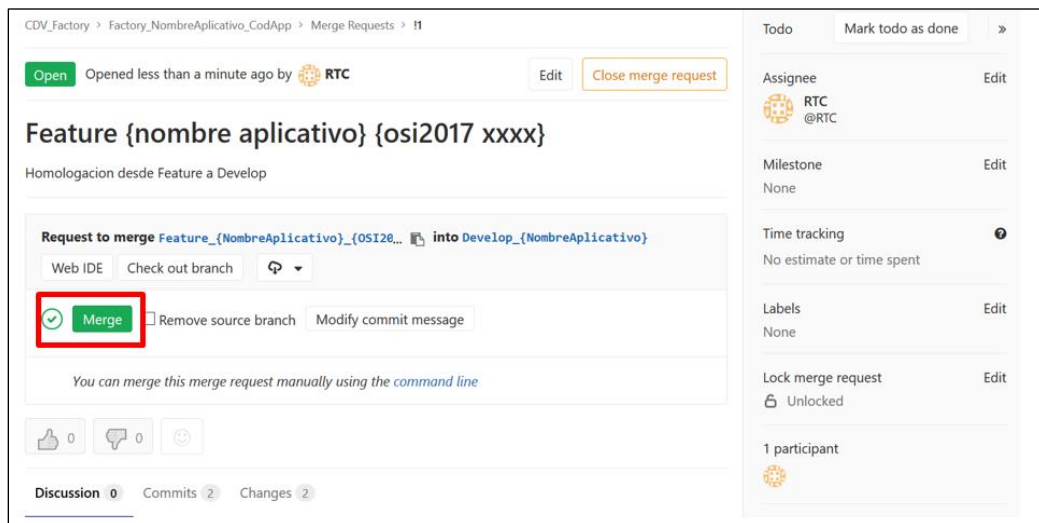
5.2. Aprobación de un Merge Request

Teniendo el Merge completado y validado por el **desarrollador**, éste debe dar aviso al **líder funcional** para aprobar dicho Merge.

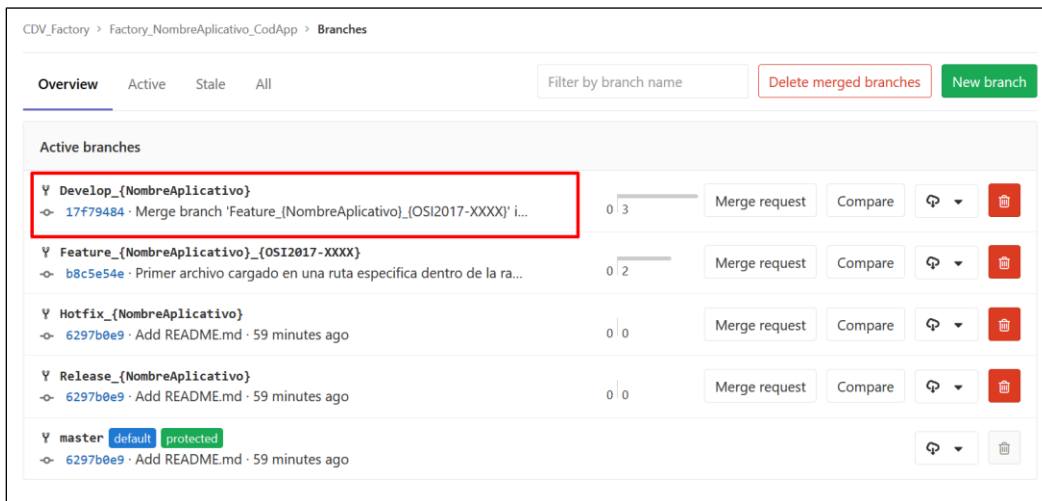
5.2.1. Líder funcional debe ingresar al Menú izquierdo “Merge Requests”.



5.2.2. Ingresa al Merge indicado por el equipo desarrollador, revisa, valida y aprueba haciendo clic en botón verde “Merge” enmarcado en rojo en imagen a continuación.



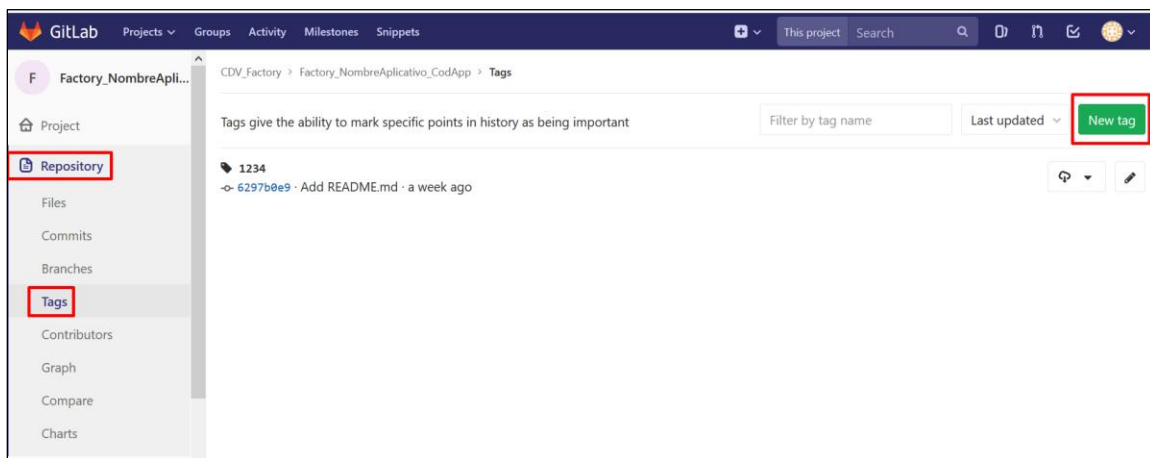
5.2.3. Una vez aprobado el Merge, la branch DEVELOP se encontrará actualizada.



6. CREACIÓN DE TAG

Los tag o etiquetas de GitLab sirven para marcar instancias o hitos en nuestros proyectos, ya sea la versión inicial, o última de nuestro desarrollo.

6.1. Para crearlas debemos ir al menú izquierdo *Repository > Tags*, clic botón verde “New tag”



6.2. Nos aparece una nueva ventana donde debemos ingresar nombre identificativo del tag “Tag Name”, en “Create From” indicamos nuestra rama desarrollo FEATURE y en “Message” una descripción del tag a crear y finalmente click en “Create tag”

CDV_Factory > Factory_NombreAplicativo_CodApp > New Tag

New Tag

Tag name: V_0.1

Create from: Feature_{NombreAplicativo}_{OSI2018-XXXX}

Existing branch name, tag, or commit SHA

Message: Carga inicial

Optionally, add a message to the tag.

Release notes: Write your release notes or drag files here...

Markdown is supported

Optionally, add release notes to the tag. They will be stored in the GitLab database and displayed on the tags page.

Create tag Cancel

6.3. Y listo tenemos nuestro tag creado para la branch FEATURE seleccionada. Ahora este tiene los cambios realizados en la branch hasta el momento de crear el tag.

CDV_Factory > Factory_NombreAplicativo_CodApp > Tags > V_0.1

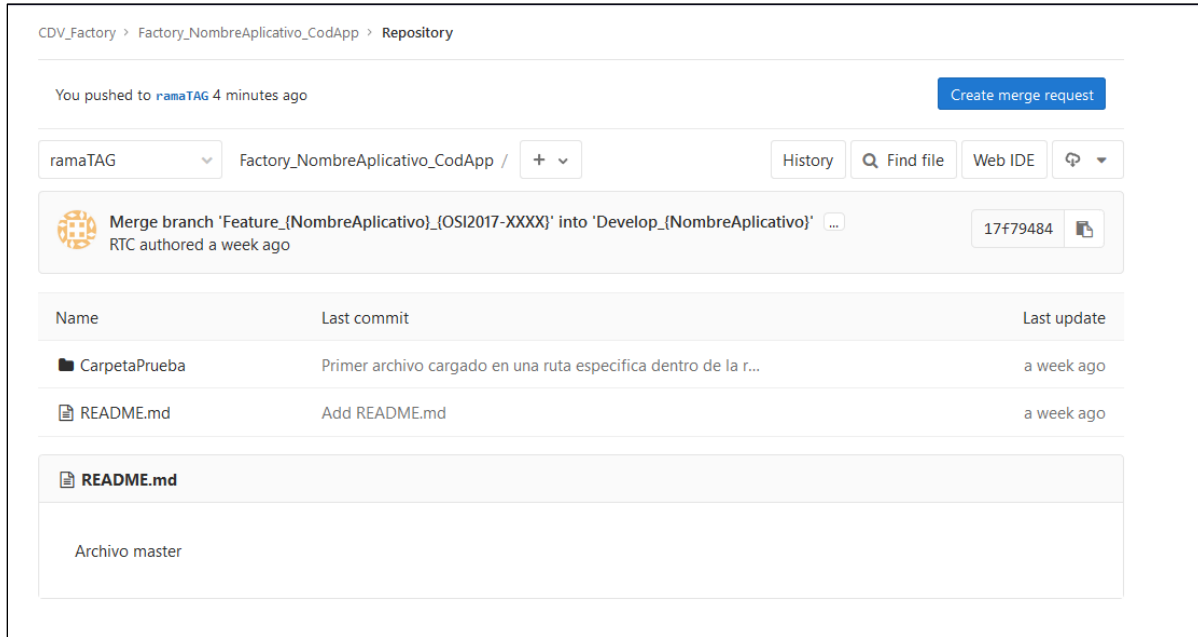
V_0.1

17f79484 · Merge branch 'Feature_{NombreAplicativo}_{OSI2017-XXXX}' into 'Develop_{NombreAplicativo}' · a week ago

Carga inicial

This tag has no release notes.

6.4. Los tags nos sirven para crear nuevas branch en base a ellos, para crearlas volvemos al paso 4 “**CREACIÓN DE BRANCH FEATURE**” siguiendo los pasos, pero seleccionando el tag creado y así tendremos nuestro Branch creado desde un TAG.



7. UTILIZACION DE TORTOISE GIT (OPCIONAL)

7.1. ¿Qué es TortoiseGit?

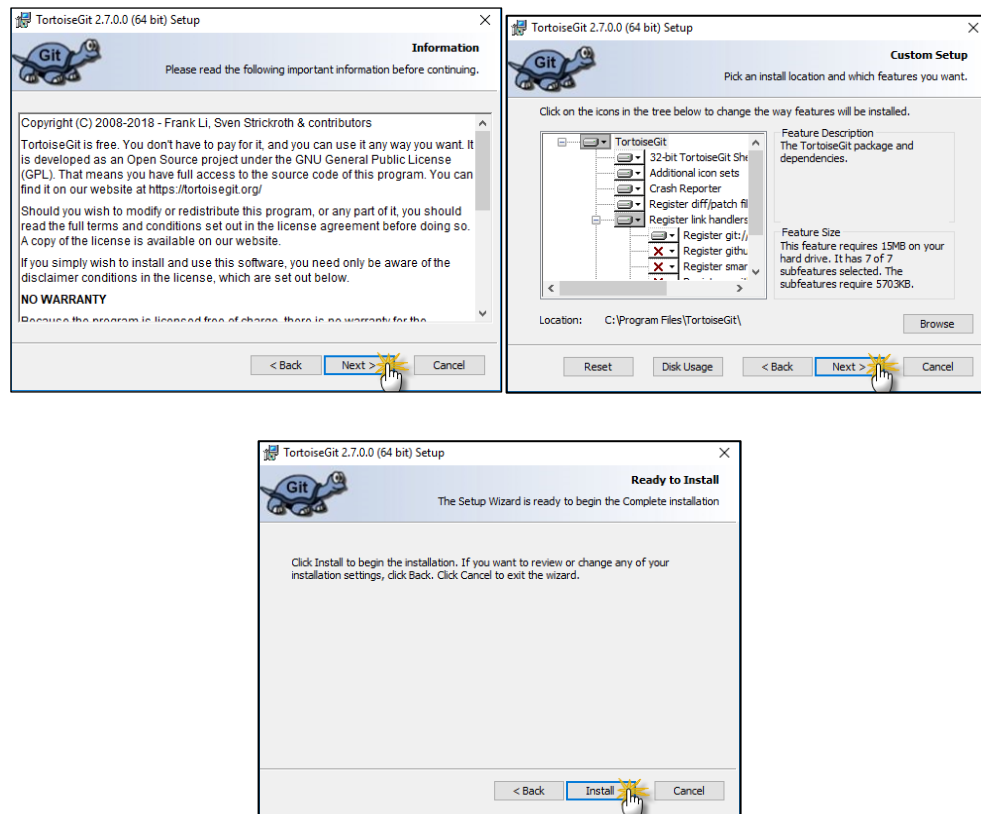
TortoiseGit es un cliente de control de versión de Git, implementado como una extensión de shell de Windows y basado en TortoiseSVN. Es un software libre lanzado bajo la Licencia Pública General de GNU.

En el Explorador de Windows, además de mostrar los elementos del menú contextual para los comandos de Git, **TortoiseGit** proporciona superposiciones de iconos que indican el estado de los árboles y archivos de trabajo de Git.

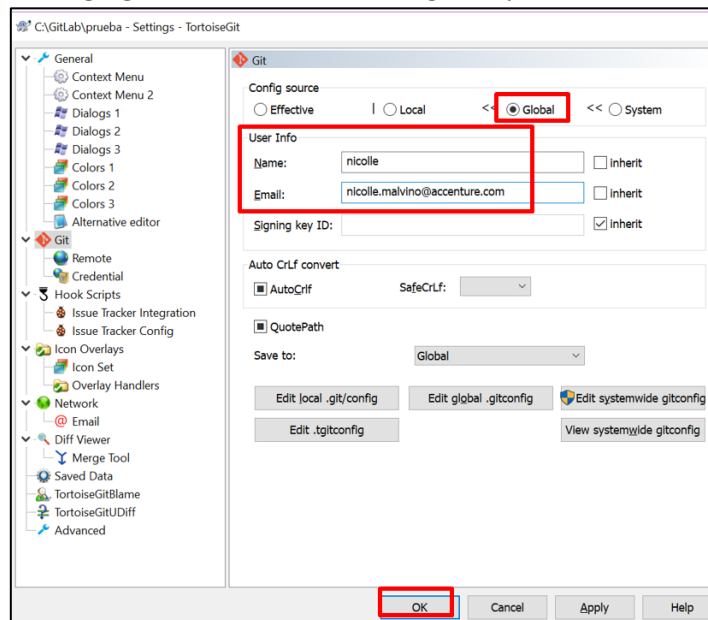
7.2. ¿Cómo conectarnos a TortoiseGit?

7.2.1. Descargar desde la siguiente URL: <https://tortoisegit.org/download/>

7.2.2. Instalar **TortoiseGit** en su equipo. Seguir los siguientes pasos:



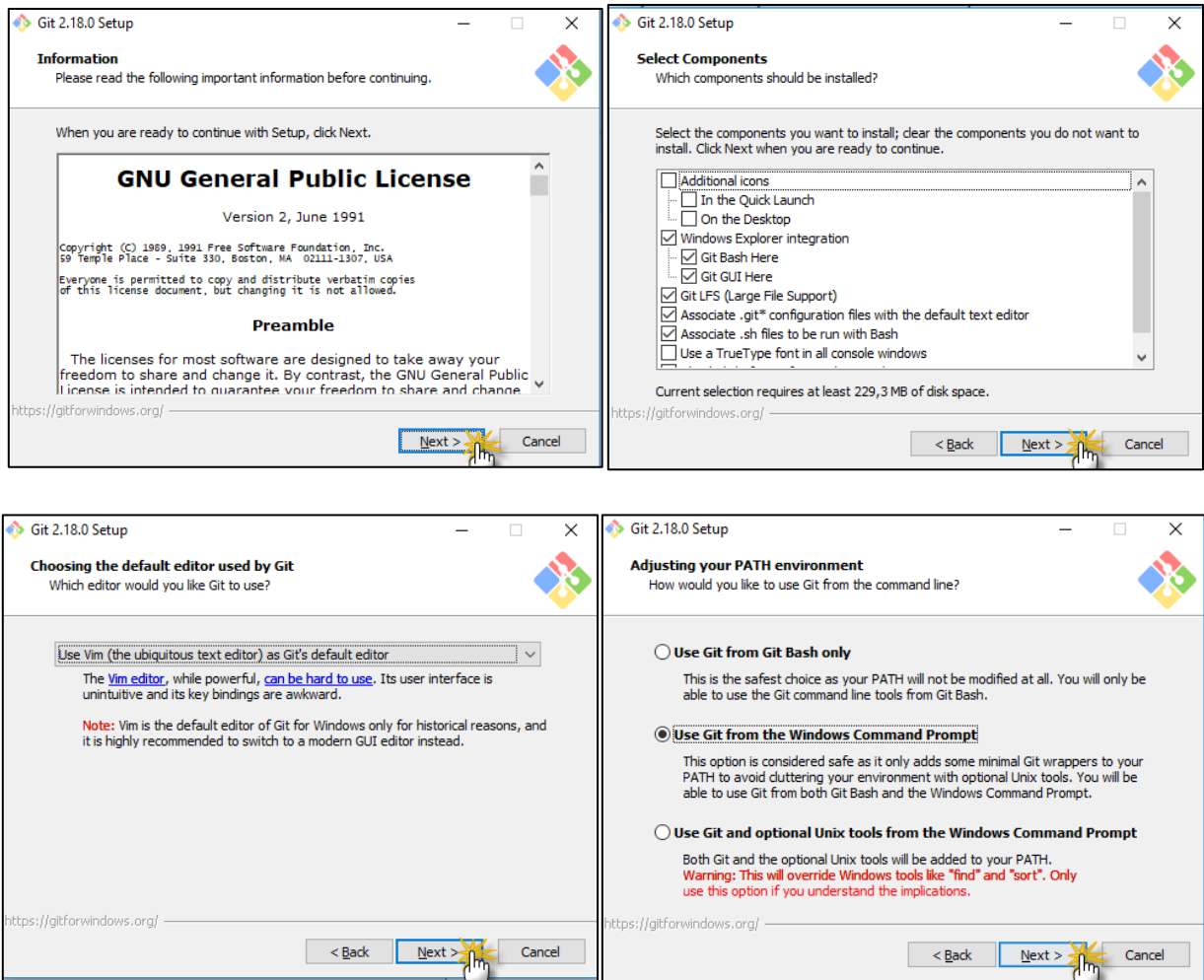
7.2.3. Configurar TortoiseGit, agregamos nuestros datos en global y "OK"



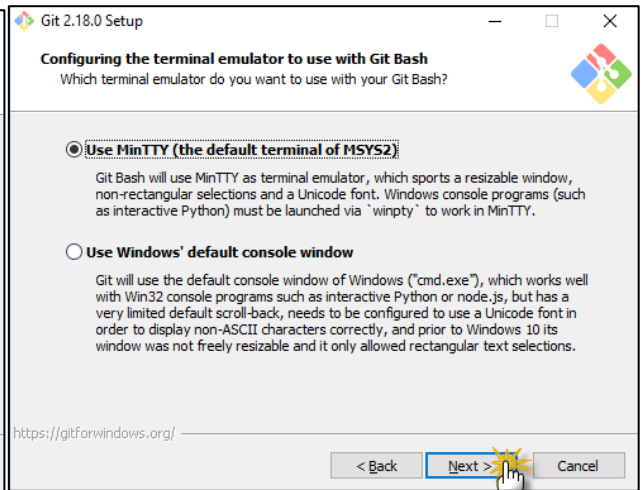
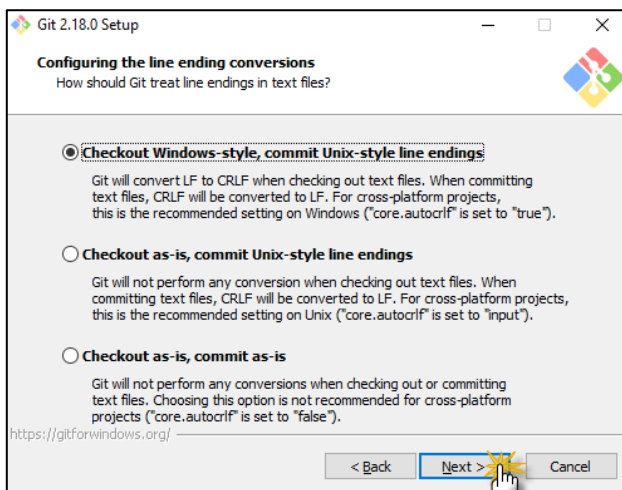
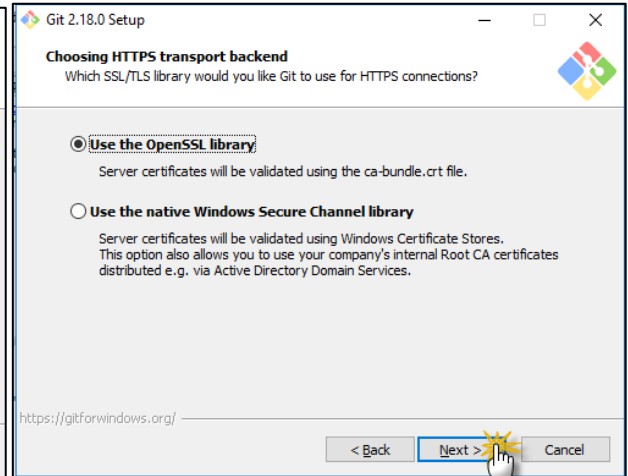
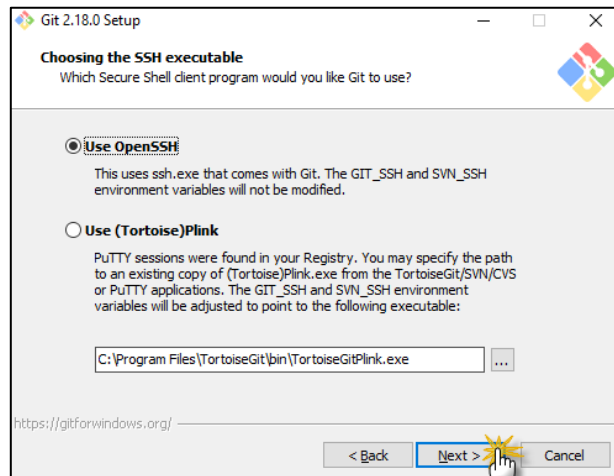
7.3. Instalar Cliente GIT para Windows

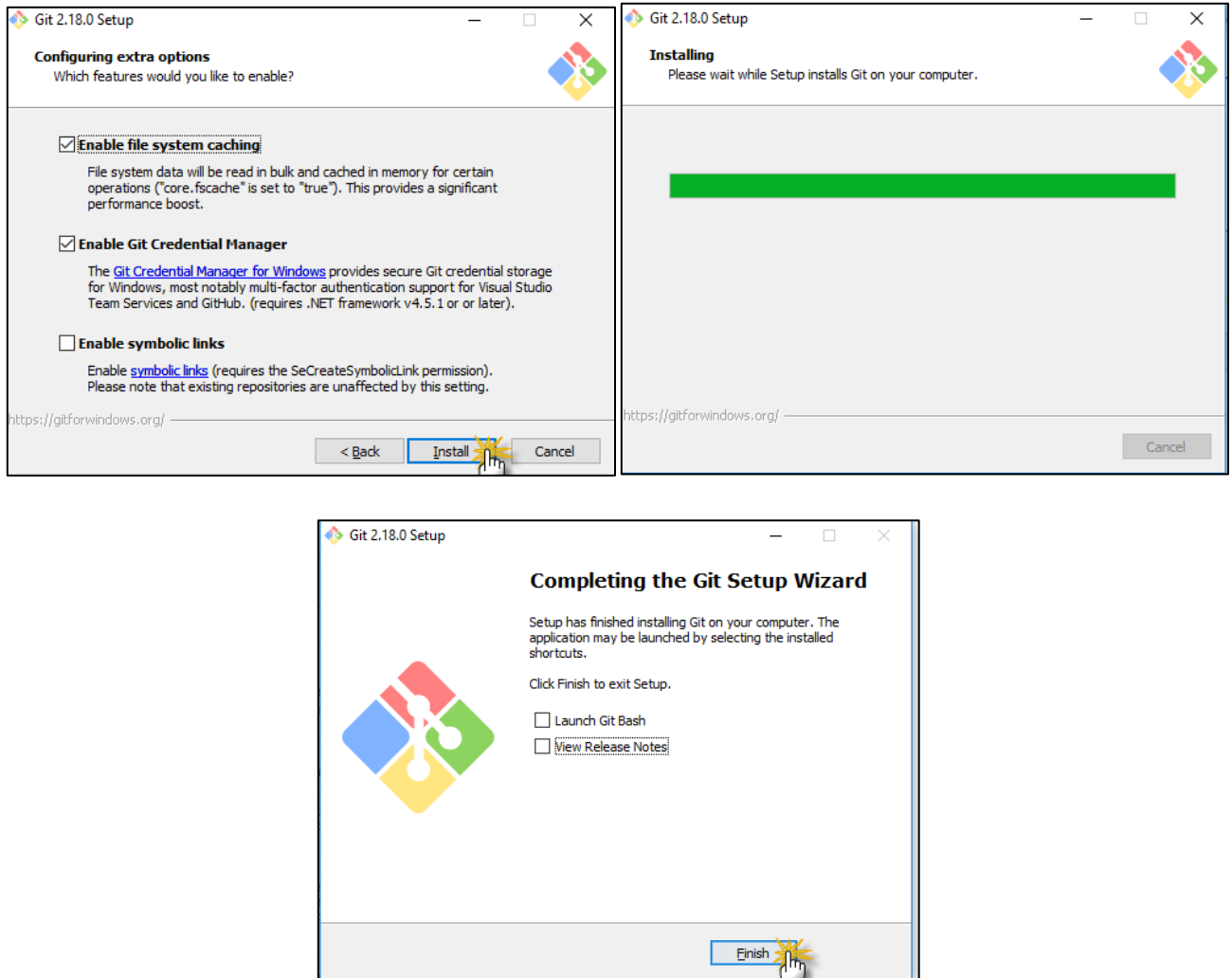
7.3.1. Descargar desde la siguiente URL: <https://github.com/git-for-windows/git/releases/download/v2.18.0.windows.1/Git-2.18.0-64-bit.exe>

7.3.2. Seguir los siguientes pasos:



Sigue las imágenes para instalar Git

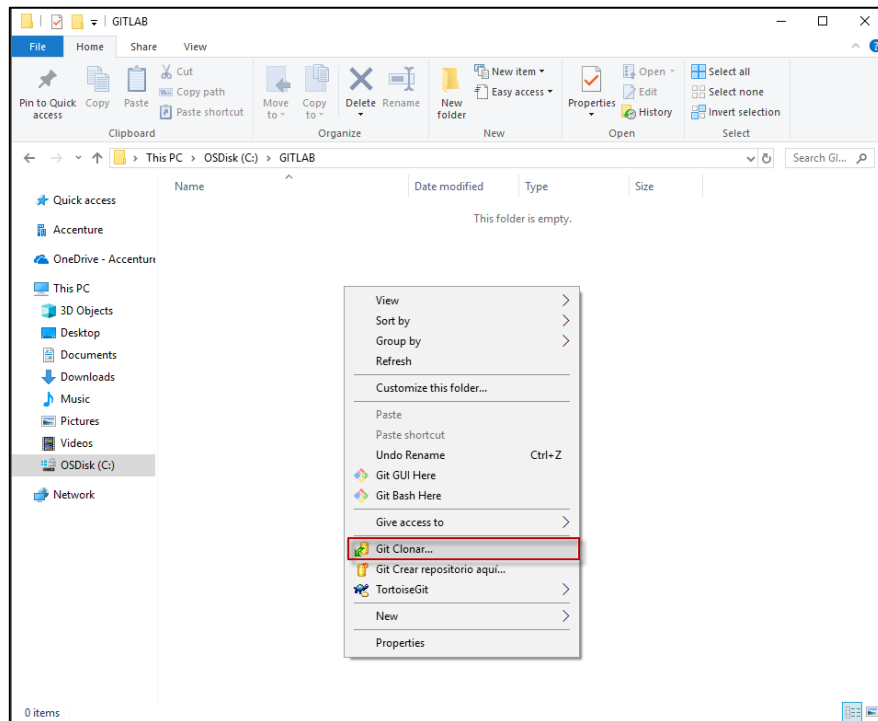




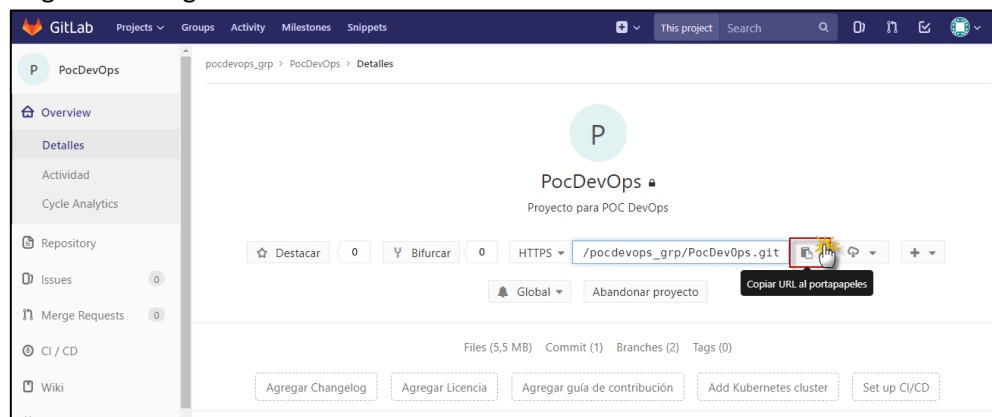
7.4. Crear repositorio en directorio local

7.4.1. Seguir los siguientes pasos:

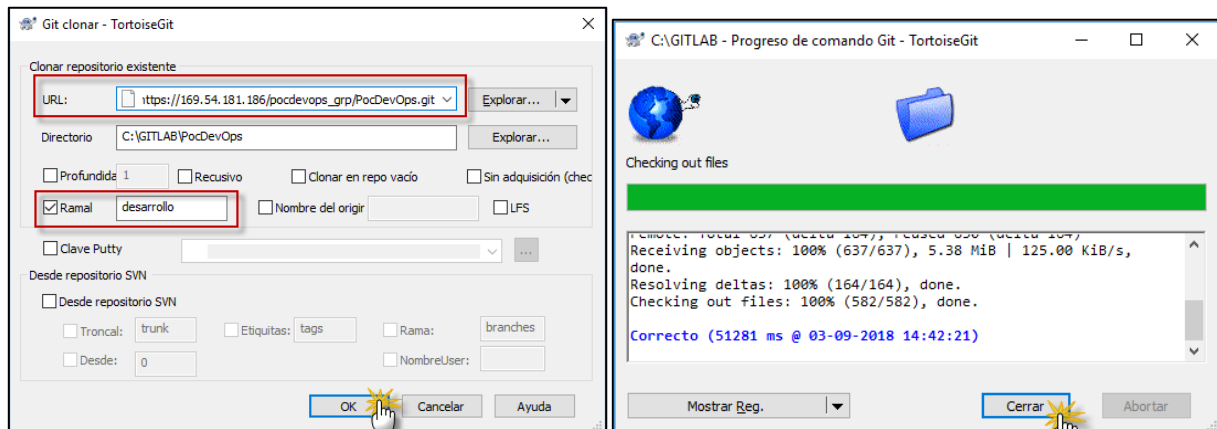
- 7.4.1.1. Crear carpeta donde desea tener sus repositorios, en éste caso crearemos una carpeta llamada: GITLAB.
- 7.4.1.2. Dentro de la carpeta creada en el paso anterior, haremos click derecho y seleccionaremos "Git Clonar..."



7.4.1.3. En la parte web de **GitLab**, debemos ingresar al repositorio y “Copiar URL” que se indica en la siguiente imagen:



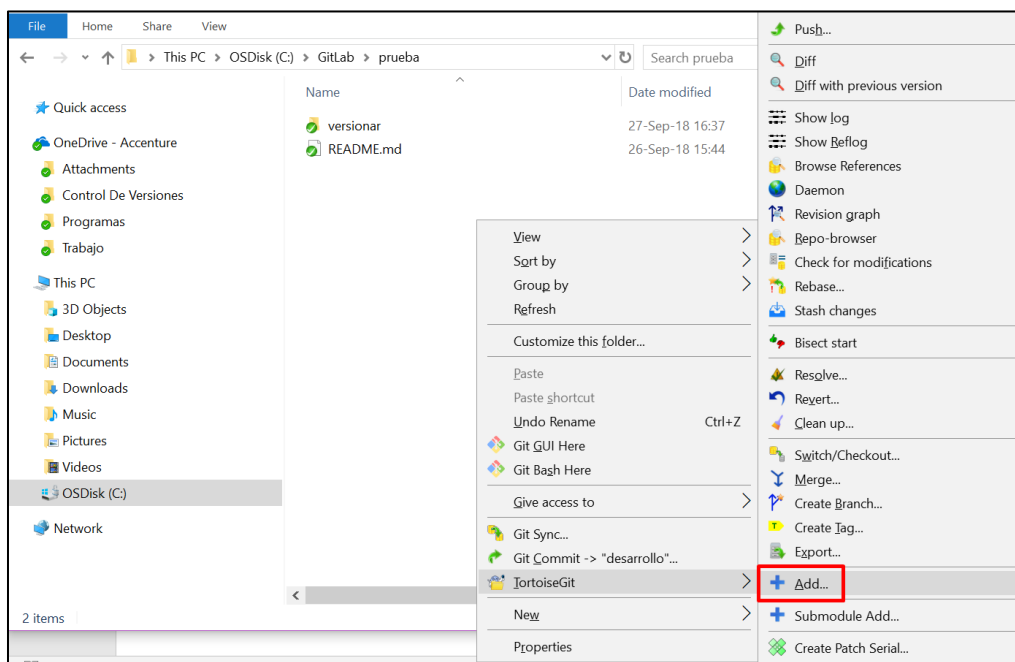
7.4.1.4. Luego pegarla en “URL” verificando que tenga el host indicado (10.186.228.19), en el campo “Ramal” debe indicar la branch que quiere clonar de forma local, para finalizar haga clic en “OK”, finalmente click en “Cerrar” ¡Listo!, repositorio clonado.



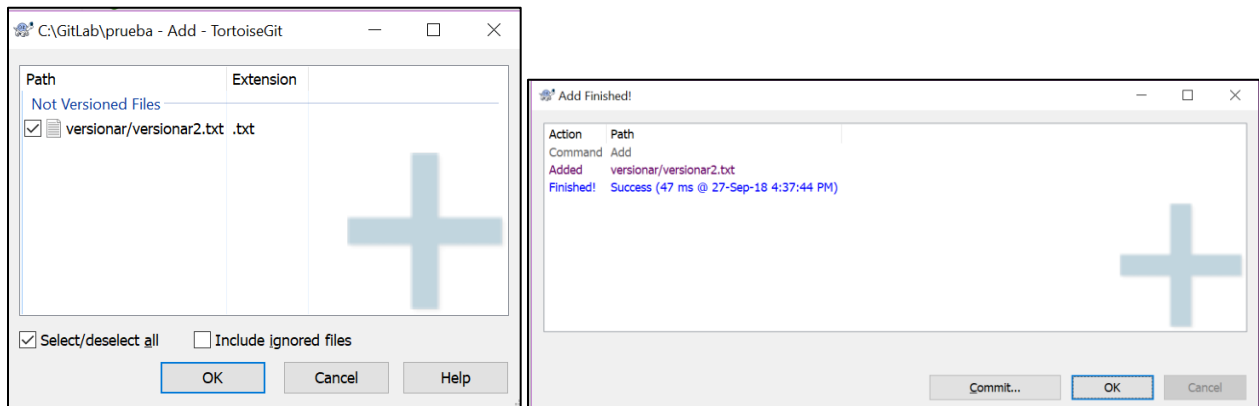
7.5. Versionando en GitLab con TortoiseGit

7.5.1. Seguir los siguientes pasos:

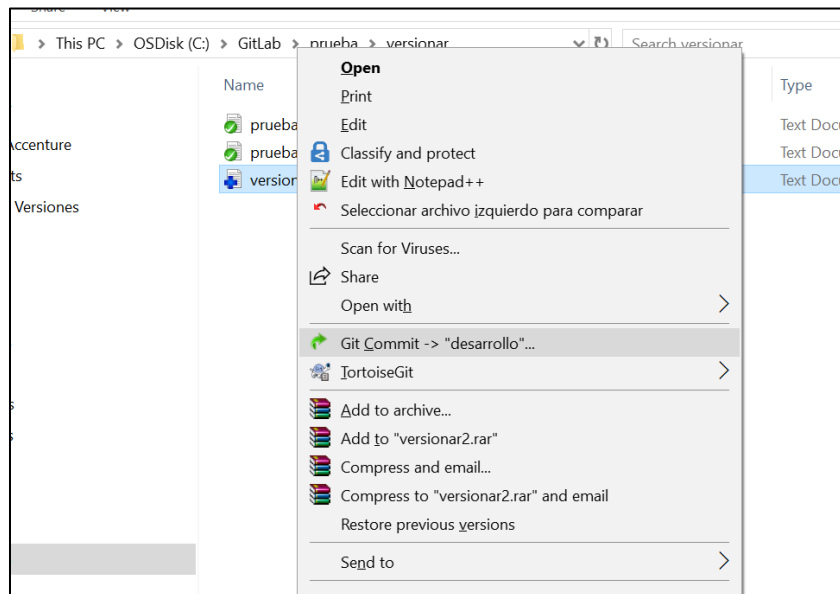
- 7.5.1.1. Ya teniendo tu repositorio clonado debes añadir tus cambios nuevos, en este caso utilizaremos la carpeta “versionar”, click derecho seleccionar “TortoiseGit”, y luego “Add”



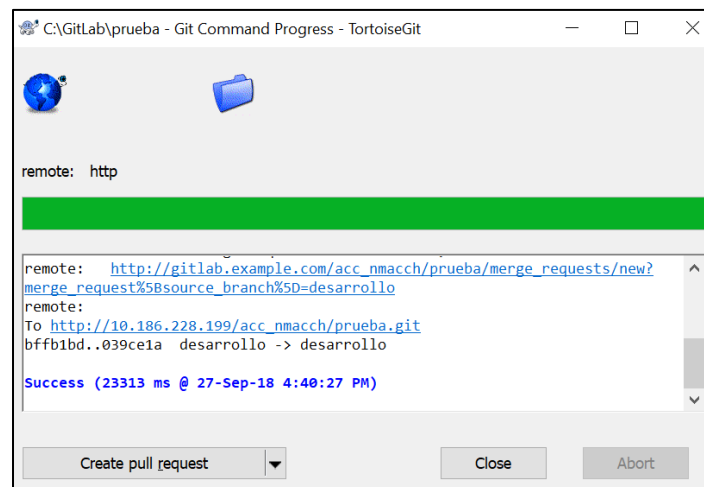
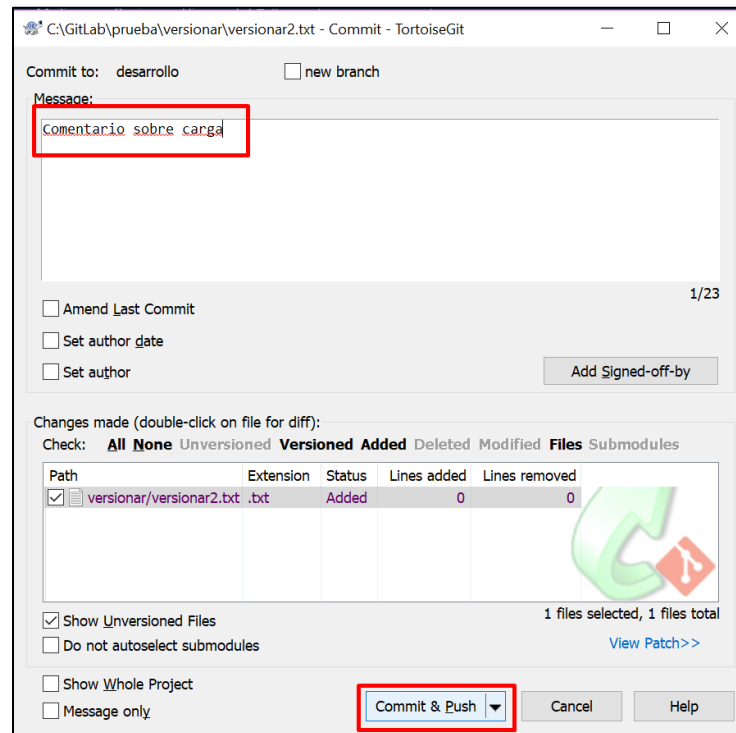
- 7.5.1.2. Al añadir te reconoce los nuevos cambios, haga clic en “OK” y “OK”.



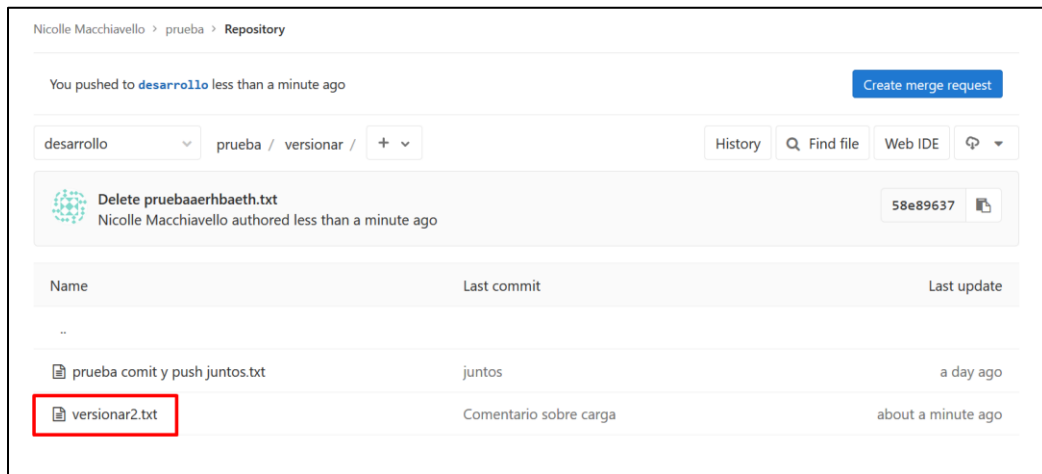
7.5.1.3. Ahora vamos directo a los cambios nuevos y presionamos botón secundario sobre estos cambios y seleccionamos “Git Commit -> {Branch}”



7.5.1.4. Agregamos un comentario y presionamos “Commit & Push”



Y listo tenemos versionado en GitLab desde nuestro repositorio local.



8. UTILIZACION DE GIT BATCH (OPCIONAL)

8.1. ¿Qué es GIT BATCH?

Anterior mente instalamos el cliente de GIT para Windows, este a su vez instala GIT BATCH que corresponde al acceso a GIT a modo de comandos a través de una consola.

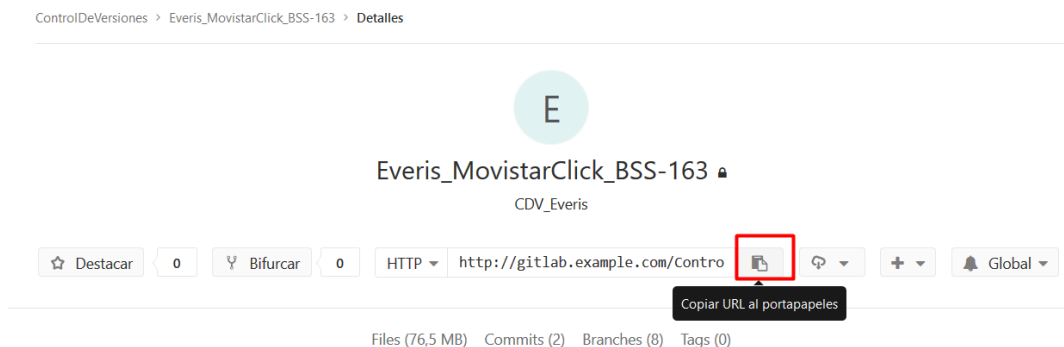
8.2. ¿Cómo conectarnos a GIT BATCH?

8.2.1. Descargar desde la siguiente URL: <https://github.com/git-for-windows/git/releases/download/v2.18.0.windows.1/Git-2.18.0-64-bit.exe>

8.2.2. Seguir los pasos del punto 7.3.

8.3. Utilizando comandos



A modo de ejemplo utilizamos el repositorio de MovistarClick, rama Release_PruebaOSI2018-00189







Copiamos el URL como lo muestra imagen anterior y verificamos que salga el host indicado http://10.186.228.199/ControlDeVersiones/Everis_MovistarClick_BSS-163.git , vemos el contenido de rama desde GIT web, para poder verificar que subiremos bien el archivo de prueba.

ControlDeVersiones > Everis_MovistarClick_BSS-163 > Repositorio

Release_PruebaOS... Everis_MovistarClick_BSS-163 / + v Historial Q Buscar archivo Web IDE ↻

 Carga Fuentes desde Stream MovistarClick_OSI2018-00189 Rational, Carga Inicial GIT
Nicolle Macchiavello authored hace un mes ac5a90ac 

Nombre	Último cambio	Última actualización
 MovistarClick_batch	Carga Fuentes desde Stream MovistarClick_OSI2018-001...	hace un mes
 MovistarClick_sql	Carga Fuentes desde Stream MovistarClick_OSI2018-001...	hace un mes
 MovistarClick_web	Carga Fuentes desde Stream MovistarClick_OSI2018-001...	hace un mes
 README.md	Agregar README.md	hace 2 meses

8.3.1. Clonar un repositorio: `Git Clone -b {rama} {URL}`

```
nicolle.malvino@CPX-RLSDRZEGRH7 MINGW64 ~
$ git clone -b Release_PruebaOSI2018-00189 http://10.186.228.199/ControlDeVersiones/Everis_MovistarClick_BSS-163.git
Cloning into 'Everis_MovistarClick_BSS-163'...
remote: Counting objects: 1411, done.
remote: Compressing objects: 100% (1051/1051), done.
remote: Total 1411 (delta 479), reused 1149 (delta 265)
Receiving objects: 100% (1411/1411), 21.77 MiB | 1.86 MiB/s, done.
Resolving deltas: 100% (479/479), done.
Checking out files: 100% (700/700), done.
```

Ingresamos al repositorio y verificamos lo mismo que vimos por WEB.

```
nicolle.malvino@CPX-RLSDRZEGRH7 MINGW64 ~
$ cd Everis_MovistarClick_BSS-163/

nicolle.malvino@CPX-RLSDRZEGRH7 MINGW64 ~/Everis_MovistarClick_BSS-163 (Release_PruebaOSI2018-00189)
$ ls
MovistarClick_batch/  MovistarClick_sql/  MovistarClick_web/  README.md
```

8.3.2. Agregar archivo: **Git Add {archivo.extensión}** o en el caso que sean varias utilizamos **Git Add** .

```
nicolle.malvino@CPX-RLSDRZEGRH7 MINGW64 ~/Everis_MovistarClick_BSS-163 (Release_PruebaOSI2018-00189)
$ git add prueba.txt
warning: LF will be replaced by CRLF in prueba.txt.
The file will have its original line endings in your working directory.

nicolle.malvino@CPX-RLSDRZEGRH7 MINGW64 ~/Everis_MovistarClick_BSS-163 (Release_PruebaOSI2018-00189)
$ ls -ltr
total 5
drwxr-xr-x 1 nicolle.malvino 1049089 0 Jan 16 12:29 MovistarClick_batch/
drwxr-xr-x 1 nicolle.malvino 1049089 0 Jan 16 12:29 MovistarClick_sql/
drwxr-xr-x 1 nicolle.malvino 1049089 0 Jan 16 12:29 MovistarClick_web/
-rw-r--r-- 1 nicolle.malvino 1049089 0 Jan 16 12:29 README.md
-rw-r--r-- 1 nicolle.malvino 1049089 9 Jan 16 12:33 prueba.txt
```

8.3.3. Guardar cambios: **git commit -m "comentario relacionado al commit"** o **git commit -a -m "Es una buena práctica que los comentarios del commit sean significativos"**

Además, puedes agregar la opción -a al comando git commit, lo cual prepara automáticamente los archivos rastreados antes de confirmarlo, lo cual te ahorra el paso de realizar git add.

```
nicolle.malvino@CPX-RLSDRZEGRH7 MINGW64 ~/Everis_MovistarClick_BSS-163 (Release_PruebaOSI2018-00189)
$ git commit -m "carga prueba"
[Release_PruebaOSI2018-00189 34f6c99] carga prueba
1 file changed, 2 insertions(+)
create mode 100644 prueba.txt
```

8.3.4. Subir los cambios: **Git Push**


```
nicolle.malvino@CPX-RLSDRZEGRH7 MINGW64 ~/Everis_MovistarClick_BSS-163 (Release_PruebaOSI2018-00189)
$ git push
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), 280 bytes | 280.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote:
remote: To create a merge request for Release_PruebaOSI2018-00189, visit:
remote: http://gitlab.example.com/ControlDeVersiones/Everis_MovistarClick_BSS-163/merge_requests/new?merge_request%5Bsource_branch%5D=Release_PruebaOSI2018-00189
remote:
To http://10.186.228.199/ControlDeVersiones/Everis_MovistarClick_BSS-163.git
ac5a90a..34f6c99 Release_PruebaOSI2018-00189 -> Release_PruebaOSI2018-00189
```






Ahora podemos verificar que el archivo de prueba fue subido en la parte web.

ControlDeVersiones > Everis_MovistarClick_BSS-163 > Repositorio

You pushed to [Release_PruebaOSI2018-00189](#) hace menos de 1 minuto [Crear solicitud de fusión](#)

Release_PruebaOS... Everis_MovistarClick_BSS-163 / + Historial Q Buscar archivo Web IDE

 **carga prueba**
Nicolle Macchiavello authored hace alrededor de 1 minuto 34f6c99b

Nombre	Último cambio	Última actualización
 MovistarClick_batch	Carga Fuentes desde Stream MovistarClick_OSI2018...	hace un mes
 MovistarClick_sql	Carga Fuentes desde Stream MovistarClick_OSI2018...	hace un mes
 MovistarClick_web	Carga Fuentes desde Stream MovistarClick_OSI2018...	hace un mes
 README.md	Agregar README.md	hace 2 meses
 prueba.txt	carga prueba	hace alrededor de 1 minuto

8.3.5. Borrar archivos: `git rm {archivo.extension}` o bien si es una carpeta `git rm -r {carpeta}`

```
nicolle.malvino@CPX-RLSDRZEGRH7 MINGW64 ~/Everis_MovistarClick_BSS-163 (Release_PruebaOSI2018-00189)
$ git rm prueba.txt
rm 'prueba.txt'

nicolle.malvino@CPX-RLSDRZEGRH7 MINGW64 ~/Everis_MovistarClick_BSS-163 (Release_PruebaOSI2018-00189)
$ ls -ltr
total 4
drwxr-xr-x 1 nicolle.malvino 1049089 0 Jan 16 12:29 MovistarClick_batch/
drwxr-xr-x 1 nicolle.malvino 1049089 0 Jan 16 12:29 MovistarClick_sql/
drwxr-xr-x 1 nicolle.malvino 1049089 0 Jan 16 12:29 MovistarClick_web/
-rw-r--r-- 1 nicolle.malvino 1049089 0 Jan 16 12:29 README.md
```

```
nicolle.malvino@CPX-RLSDRZEGRH7 MINGW64 ~/Everis_MovistarClick_BSS-163 (Release_PruebaOSI2018-00189)
$ git rm -r prueba/
rm 'prueba/archivo.txt'
```

Luego de borrar hacemos commit y Push para ver los cambios reflejados en la web.

```
nicolle.malvino@CPX-RLSDRZEGRH7 MINGW64 ~/Everis_MovistarClick_BSS-163 (Release_
PruebaOSI2018-00189)
$ git commit -m "borrado de archivo prueba"
[Release_PruebaOSI2018-00189 da49d6a] borrado de archivo prueba
1 file changed, 2 deletions(-)
delete mode 100644 prueba.txt

nicolle.malvino@CPX-RLSDRZEGRH7 MINGW64 ~/Everis_MovistarClick_BSS-163 (Release_
PruebaOSI2018-00189)
$ git push
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% (2/2), 230 bytes | 230.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote:
remote: To create a merge request for Release_PruebaOSI2018-00189, visit:
remote: http://gitlab.example.com/ControlDeVersiones/Everis_MovistarClick_BSS-
163/merge_requests/new?merge_request%5Bsource_branch%5D=Release_PruebaOSI2018-00
189
remote:
To http://10.186.228.199/ControlDeVersiones/Everis_MovistarClick_BSS-163.git
34f6c99..da49d6a Release_PruebaOSI2018-00189 -> Release_PruebaOSI2018-00189
```

8.3.6. Comandos de Utilidad:

- Si te quieres cambiar de rama: **git checkout {nombrerama}**
- Para traer los cambios del repositorio: **git pull**
- Para conocer más comandos disponibles: **git help**

Como buena práctica, luego de hacer commit a tus cambios, justo antes de subir tus cambios, hagas pull.

***Commit - pull - push**

9. CONTROL DE VERSIONES DEL DOCUMENTO

Versión	Fecha	Autor	Revisado por	Principales Cambios
0.1	03/10/2018	Nicolle Macchiavello	Yasna Albornoz	Versión Inicial
0.2	08/10/2018	Nicolle Macchiavello	Yasna Albornoz	Ramas Git
0.3	09/10/2018	Nicolle Macchiavello	Yasna Albornoz	TortoiseGit
1.0	09/10/2018	Nicolle Macchiavello	Yasna Albornoz	Estructura
1.1	10/10/2018	Nicolle Macchiavello	Yasna Albornoz	Modificaciones Versión inicial
1.2	29/11/2018	Nicolle Macchiavello	Yasna Albornoz	Cambio de url git
1.3	18/12/2018	Nicolle Macchiavello	Yasna Albornoz	Modificación del Merge
1.4	16/01/2019	Nicolle Macchiavello	Yasna Albornoz	Comandos