
[illegible]**FECHA**

DESCRIPCIÓN CAMBIO REALIZADO Y VALIDADO

19/02/2018

Versión inicial, Documentación del GIT

	PROCESO OPERATIVO DE CALIDAD Formato Documentación GIT	
	Documento No. Fo7502.Vv2	Página 2 de 20

Administración del GIT

Documentación Técnica
GIT

Contenido

Objetivo.....	5
Alcance	5
Justificación	5
Requisitos Mínimos GIT	6
Crear Repositorio Remoto.....	6
Crear una cuenta	6
Instalación GIT	8
Flujo de trabajo	8
Repositorio	8
Configuración	9
Crear Checkout.....	9
Añadir y actualizar.....	9
Envío de cambios	10
Clonar repositorio remoto	10
Ramas.....	12
Crear una Rama.....	12
Borrar una Rama	12
Actualizar y fusionar.....	13
Etiquetas.....	14
Reemplazar cambios locales	14
Datos adicionales	15
Deploy Automático con Git	16
Conclusiones	19
Enlaces.....	20
Clientes gráficos	20
Guías.....	20


	PROCESO OPERATIVO DE CALIDAD Formato Documentación GIT	
	Documento No. Fo7502.Vv2	Página 4 de 20

Tabla Ilustraciones

Ilustración 1 Nuevo Repositorio	6
Ilustración 2 Crear repositorio Remoto.....	7
Ilustración 3 Repositorio	8
Ilustración 4 Checkout.....	9
Ilustración 5 add and commit	9
Ilustración 6 añadir.....	10
Ilustración 7 commit	10
Ilustración 8 Enviar cambios	10
Ilustración 9 Clonar repositorio.....	10
Ilustración 10 URL GitHub	11
Ilustración 11 Subir Repo a Remoto.....	11
Ilustración 12 Crear rama	12
Ilustración 13 Rama principal.....	12
Ilustración 14 Push	12
Ilustración 15 Borrar rama	12
Ilustración 16 Actualizar repositorio	13
Ilustración 17 Activar Rama	13
Ilustración 18 añadir al index	13
Ilustración 19 Revisar cambios de fusión	13
Ilustración 20 Agregar Etiquetas	14
Ilustración 21 Log	14
Ilustración 22 Reemplazar cambios	14
Ilustración 23 Deshacer cambios	14
Ilustración 24 Deshacer cambios rama	14
Ilustración 25 Interfaz grafica	15
Ilustración 26 Colores de consola	15
Ilustración 27 Linea por commit	15
Ilustración 28 Archivos Interactivos	15
Ilustración 29 Hooks para scripts	16
Ilustración 30 Crear Script.....	16
Ilustración 31 Código script.....	17
Ilustración 32 Dar permisos de script	18
Ilustración 33 Agregar usuario repositorio remoto	18
Ilustración 34 Permisos en el directorio Web	18
Ilustración 35 Ejecutar script automático	18

Objetivo


Asimilar, adaptar y generar tecnologías telemáticas para aplicarlas al desarrollo de la corporación y contribuir al avance del conocimiento en este campo.

Alcance

Obtener soluciones a los problemas planteados por el establecimiento de estas tecnologías en nuestro medio.

Justificación

Identificar y conocer los procesos necesarios para llevar a cabo una correcta administración y asesorar a las empresas en la adquisición de tecnologías relacionadas con estas áreas

	PROCESO MISIONAL DE CALIDAD Formato Documentación GIT	
	Documento No. Fo7502.Vv2	Página 6 de 20

Requisitos Mínimos GIT

3GB Espacio en Disco

1.0 GB de RAM

Conexión a internet

Crear Repositorio Remoto

Para este ejemplo se creará un repositorio remoto en **GitHub**, esta es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. Se la utiliza principalmente para la creación de código fuente de programas de computadora, por consiguiente, se procede con la breve explicación:

Crear una cuenta

- Ingresa a : <https://github.com/>
- Botón: Sign in.
- Ingresa datos como: Nombre Usuario, correo y contraseña.
- Verifica el correo electrónico.
- Clic en el signo + y NEW REPOSITORY

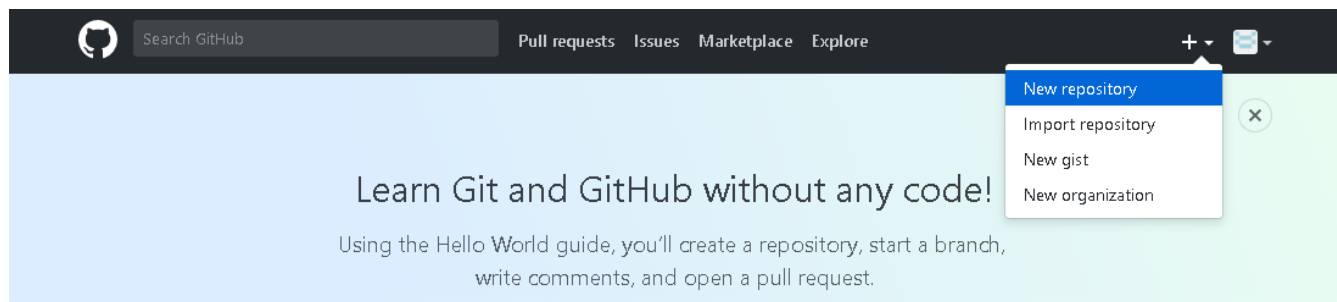
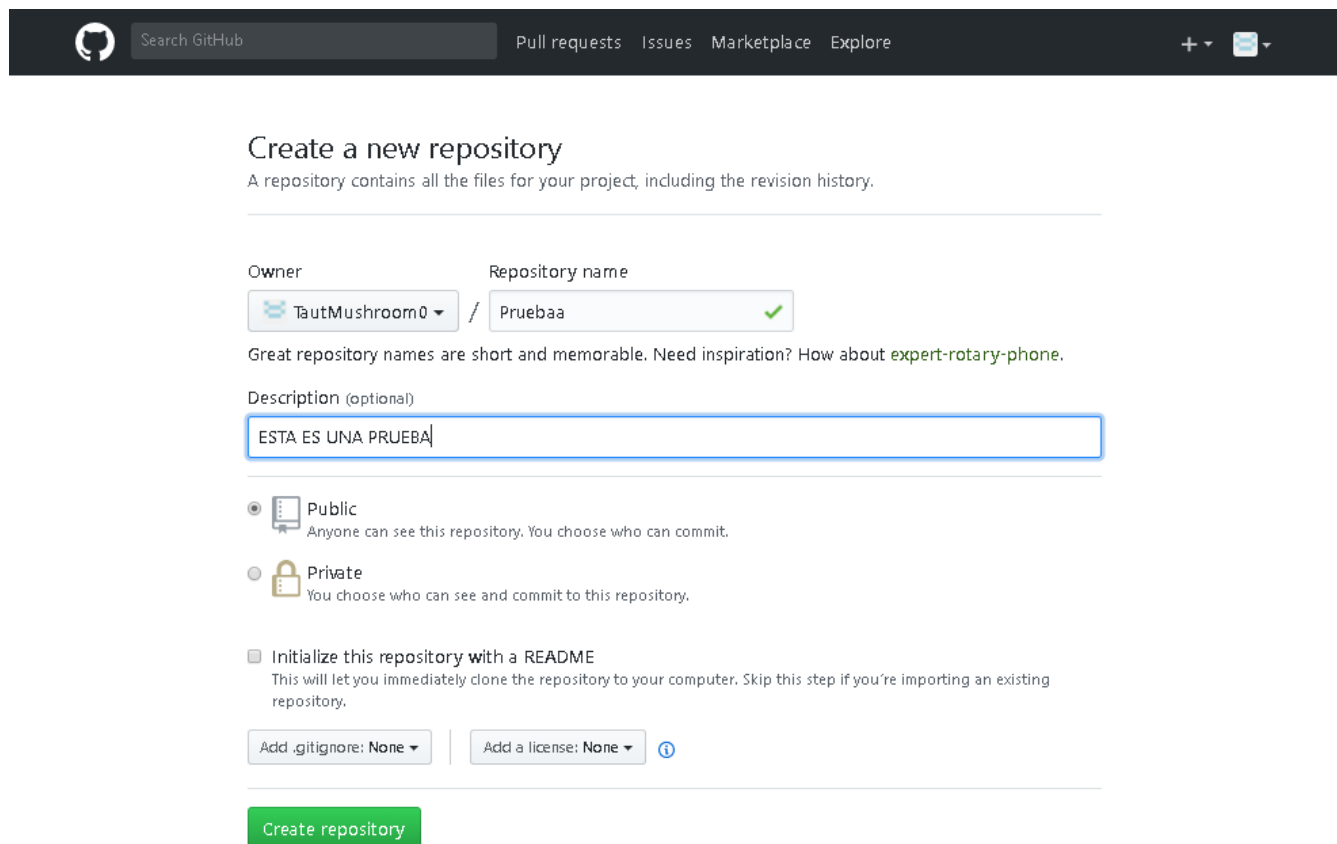


Ilustración 1 Nuevo Repositorio

- Una vez ahí, se asigna un nombre al repositorio con única restricción que no este repetido, luego una descripción de que se trata, se marca la casilla de público para que los demás puedan verlo, le damos en iniciar repositorio y al final en el botón verde para crear nuestro repositorio. PD: No se pueden crear repositorios privados a menos que pagues una suscripción a GitHub.



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: TautMushroom0 / Repository name: Pruebaa ✓

Great repository names are short and memorable. Need inspiration? How about [expert-rotary-phone](#).

Description (optional): ESTA ES UNA PRUEBA

☒ Public
Anyone can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.


☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None ⓘ

Create repository

Ilustración 2 Crear repositorio Remoto

Se puede activar o desactivar la casilla de verificación del Readme, dependiendo de si se quiere que el repositorio se inicie vacío, o si quiere que se inicie con archivo que describa más a fondo el archivo (usualmente para eso funciona el Readme).

	PROCESO OPERATIVO DE CALIDAD Formato Documentación GIT	
	Documento No. Fo7502.Vv2	Página 8 de 20

Instalación GIT

Para realizar la instalación hay varios links, lo primero es identificar para cual sistema operativo se desea instalar, para eso están los siguientes links.

Para OSX:

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

Para Windows:

<https://gitforwindows.org/>

Para Linux:

(CENTOS7 similares):

```
yum install git-core
```

(UBUNTU y similares):

```
apt-get install git
```

Flujo de trabajo

El repositorio local está compuesto por tres "árboles" administrados por Git. El primero es el Directorio de trabajo que contiene los archivos, el segundo es el Índice que actúa como una zona intermedia, y el último es el HEAD que apunta al último commit realizado.

Repositorio

Para poder crear un repositorio nuevo hay que abrir el GUIT CMD y ejecutar el comando:

```
git init
```

Ilustración 3 Repositorio

Configuración

Antes de empezar a utilizar Git, es recomendable que dediques un par de minutos a establecer algunas opciones de configuración útiles. Estas opciones se establecen mediante el comando **git config**. En primer lugar, define tu nombre y email de contacto ejecutando los siguientes comandos:

```
> git config --global user.name "Miguel Pinzón"
> git config --global user.email Miguel.pinzon@skillnet.com.co
```

Después, puedes modificar si quiere el editor que se utiliza para escribir los mensajes que acompañan a cada commit al servidor:

```
> git config --global core.editor vim
```

Otra opción muy útil es la que hace que se coloreen las diferencias en los *commits* para entender mejor los cambios:

```
> git config --global color.ui true
```

La manera más sencilla de ver la versión la cual se instalo es:

```
> git versión
```

Crear Checkout

Se crea una copia local del repositorio ejecutando

```
git clone /path/to/repository
```

Ilustración 4 Checkout

Añadir y actualizar

Para poder añadir registros al índice es usando el comando:

```
git add <filename>
```

Ilustración 5 add and commit

o

```
git add .
```

Ilustración 6 añadir

Para añadir todos los registros alojados en la carpeta GIT

Ahora para actualizar los cambios hay que registrarlos con un commit

```
git commit -m "Commit message"
```

Ilustración 7 commit

Envío de cambios

Una vez los cambios que anteriormente quedaron guardados en el **HEAD** de la copia local, si se desea enviar estos cambios a un repositorio remoto ya existente se ejecuta el siguiente comando.

```
git push origin master
```

Ilustración 8 Enviar cambios


La palabra **master** se debe reemplazar por la rama a la que se desea enviar los cambios

Clonar repositorio remoto

Si no se ha clonado un repositorio ya existente y se desea conectar un repositorio local a un repositorio remoto se ejecuta:

```
git remote add origin <server>
```

Ilustración 9 Clonar repositorio

	PROCESO OPERATIVO DE CALIDAD Formato Documentación GIT	
	Documento No. Fo7502.Vv2	Página 11 de 20

Y con esto se podrán subir los cambios al repositorio seleccionado.

La URL del repositorio remoto la obtienes de tu repositorio en Github, de la siguiente imagen, puedes ver el lugar en el que puedes obtener la URL, para copiarla sólo necesitas hacer clic en el botón a un lado de la URL:

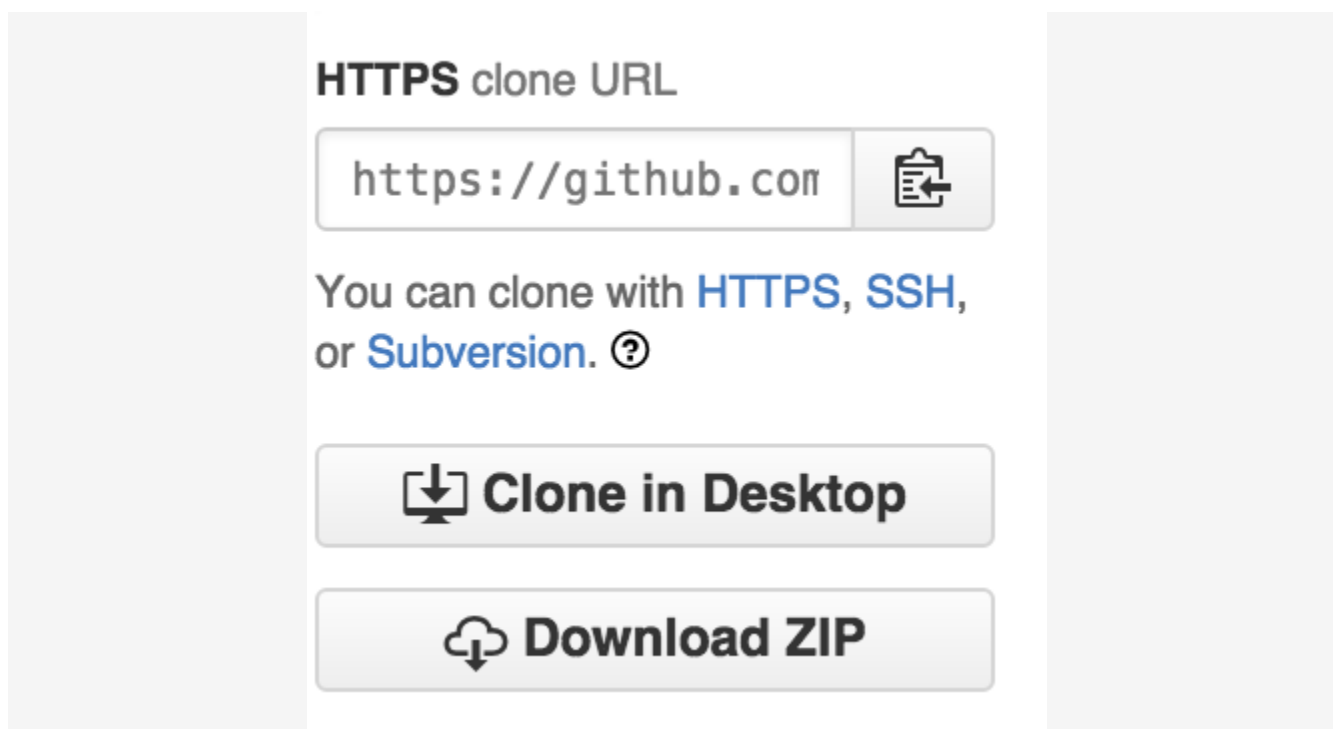


Ilustración 10 URL GitHub

- Introducimos este comando para que se suban nuestros archivos al repositorio remoto (en GitHub).

```
git push -u origin master
```

Ilustración 11 Subir Repo a Remoto

Ramas

Las ramas son utilizadas para desarrollar funcionalidades aisladas unas de otras. La rama master es la rama "por defecto" cuando se crea un repositorio. Crea nuevas ramas durante el desarrollo y debe fusionarlas a la rama principal cuando haya terminado.

Crear una Rama

Para crear una nueva rama llamada "feature_x" y cambiarse a la misma, se debe ejecutar:

```
git checkout -b feature_x
```

Ilustración 12 Crear rama

Ahora volver a la rama principal ejecutando:

```
git checkout master
```

Ilustración 13 Rama principal

Una rama nueva no estará disponible para los demás a menos que se suba (push) la rama del repositorio remoto.

```
git push origin <branch>
```

Ilustración 14 Push

Borrar una Rama

En el caso que se desee eliminar una rama se debe ejecutar el comando:

```
git branch -d feature_x
```

Ilustración 15 Borrar rama

Actualizar y fusionar

Para actualizar el repositorio local al commit mas nuevo, ejecuta

```
git pull
```

Ilustración 16 Actualizar repositorio

en el directorio de trabajo para bajar y fusionar los cambios remotos.

Para fusionar otra rama a su rama activa (Por ejemplo, master), se utiliza:

```
git merge <branch>
```

Ilustración 17 Activar Rama

en ambos casos GIT intentará fusionar automáticamente los cambios. Desafortunadamente, no siempre será posible y se podrán producir conflictos. El administrador es responsable de fusionar esos conflictos manualmente al editar los archivos mostrados por GIT. Después de modificarlos, necesita marcarlos como fusionados con

```
git add <filename>
```

Ilustración 18 añadir al index

Antes de fusionar los cambios pueden revisarse usando:

```
git diff <source_branch> <target_branch>
```

Ilustración 19 Revisar cambios de fusión

Etiquetas

Se recomienda crear etiquetas para cada nueva versión publicada de un software. Este concepto no es nuevo, ya que estaba disponible en SVN. Puede crear una nueva etiqueta llamada 1.0.0 ejecutando

```
git tag 1.0.0 1b2e1d63ff
```

Ilustración 20 Agregar Etiquetas

1b2e1d63ff se refiere a los 10 caracteres del commit id al cual se quiere referir con la etiqueta. Puede obtener el commit id con

```
git log
```

Ilustración 21 Log

también se puede usar menos caracteres que el commit id, pero debe ser un valor único.

Reemplazar cambios locales

En caso de que se haya hecho mal una solución es reemplazar cambios locales usando el comando:

```
git checkout -- <filename>
```

Ilustración 22 Reemplazar cambios

Este comando reemplaza los cambios en el directorio de trabajo con el último contenido de HEAD. Los cambios que ya han sido agregados al Index, así como también los nuevos archivos, se mantendrán sin cambio.

Por otro lado, si quiere deshacer todos los cambios locales y commits, puede traer la última versión del servidor y apuntar a su propia local principal de esta forma:

```
git fetch origin
```

Ilustración 23 Deshacer cambios

```
git reset --hard origin/master
```

Ilustración 24 Deshacer cambios rama

Datos adicionales

Interfaz gráfica por defecto

```
gitk
```

Ilustración 25 Interfaz grafica

Colores especiales para la consola

```
git config color.ui true
```

Ilustración 26 Colores de consola

Mostrar sólo una línea por cada commit en la traza


```
git config format.pretty oneline
```

Ilustración 27 Linea por commit

Agregar archivos de forma interactiva

```
git add -i
```

Ilustración 28 Archivos Interactivos

	PROCESO OPERATIVO DE CALIDAD Formato Documentación GIT	
	Documento No. Fo7502.Vv2	Página 16 de 20

Deploy Automático con Git

Suponiendo que hemos montado un repositorio git en un servidor Web, los desarrolladores enviarán cambios a las diferentes ramas del repositorio periódicamente. Ahora bien, cuando los cambios alcanzan un punto estable, es deseable pasarlos al sitio Web (lo que se conoce comúnmente como deploy) para que sean visibles a los usuarios.

Para ello se requiere ejecutar un **checkout**, actualizar los archivos en el directorio de trabajo. En este punto surgen dos alternativas: darle acceso y permiso a los desarrolladores para que puedan correr el **checkout** o intervenir el **SysAdmin** (correr el **checkout** manualmente y configurar los permisos según corresponda).

Sin embargo, en ciertos entornos (especialmente en entornos de desarrollo/testing) es deseable que el checkout se realice de forma automática y sin intervención manual (tanto de los desarrolladores como del SysAdmin/DevOps). Para ello se necesita pasar automáticamente los cambios al sitio cada vez que un desarrollador actualiza (push) una rama específica del repositorio. Una especie de mecanismo de auto-deploy para que el SysAdmin no deba intervenir manualmente en el proceso.

Git posee un mecanismo que permite ejecutar scripts cuando los eventos importantes ocurren, para esto encontramos los **hooks**, aquí hay que configurar un **checkout** o un **deploy** para que corra automáticamente en el servidor cada vez que un desarrollador realiza un push sobre la rama “testing”.

```
Cd .git/  
Cd hooks/
```

Ilustración 29 Hooks para scripts

En el directorio **hooks/** se crea el siguiente archivo “post-receive”:

```
Vi post-receive
```

Ilustración 30 Crear Script

Y ingresamos lo siguiente:

```
#!/bin/sh

GIT_DIR="/var/repo/repositorio1.git"
WORK_TREE="/var/www/demo/aplicacion1"
TARGET_BRANCH="testing"

while read OLDREV NEWREV REFNAME
do
    BRANCH=$(git rev-parse --symbolic --
abbrev-ref $REFNAME)
    if [ -n "$BRANCH" ] && [
"$TARGET_BRANCH" == "$BRANCH" ]; then
        git --work-tree=$WORK_TREE --git-
dir=$GIT_DIR checkout -f
    fi
done
```

Ilustración 31 Código script

Hay que configurarlo de la siguiente forma, según tu repositorio:

- **GIT_DIR** (directorío del repositorio).
- **WORK_TREE** (directorío del sitio Web).
- **TARGET_BRANCH** (rama que dispara la actualización del sitio Web).

Este script correrá automáticamente cada vez que se haga un **push** sobre el repositorio, por ende, es importante determinar sobre que rama se hace.

Para continuar, hay que dar permiso de ejecución al Script:

```
chmod +x post-receive
```

Ilustración 32 Dar permisos de script

Para evitar tener que asignar permisos cada vez que se cambian archivos en el sitio, es conveniente agregar al usuario del repositorio remoto (en este ejemplo “Webmaster”) al grupo que utiliza el servidor HTTP (“www-data” en Debian y derivados). Luego, configurar en el repositorio los permisos necesarios para el sitio Web: lectura para el grupo “www-data” y escritura sólo donde sea específicamente necesario (directorio para subida de archivos, sesiones, temporales, etc.).

```
cd ..  
chown -R webmaster:www-data  
chmod -R 750
```

Ilustración 33 Agregar usuario repositorio remoto

Repetir el esquema de permisos en el directorio Web y comprobar su funcionamiento:

```
cd /var/www/demo/aplicacion1/  
chown -R webmaster:www-data  
chmod -R 750
```

Ilustración 34 Permisos en el directorio Web


Y ejecutamos con:

```
man git-push  
man git-checkout
```

Ilustración 35 Ejecutar script automático

Conclusiones

Este manual es con el fin de orientar y dar a conocer las funcionalidades a disposición de un Git para poder llevar a cabo una correcta administración y poder aprovechar con totalidad los beneficios que contienen las nuevas tecnologías.

	PROCESO OPERATIVO DE CALIDAD Formato Documentación GIT	
	Documento No. Fo7502.Vv2	Página 20 de 20

Enlaces

Cientes gráficos

- [GitX \(L\) \(OSX, open source\)](#)
- [Tower \(OSX\)](#)
- [Source Tree \(OSX, free\)](#)
- [GitHub for Mac \(OSX, free\)](#)
- [GitBox \(OSX\)](#)

Guías

- [Git Community Book](#)
- [Pro Git](#)
- [Think like a git](#)
- [GitHub Help](#)
- [A Visual Git Guide](#)
- [Deploy](#)