|  |  |  |
| --- | --- | --- |
| **REGISTRO HISTÓRICO DEL DOCUMENTO** | | |
| **VERSIÓN** | **FECHA** | **DESCRIPCIÓN CAMBIO REALIZADO Y VALIDADO** |
| **1.0** | 19/02/2018 | Versión inicial, Documentación del GIT |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Administración del GIT**

**Documentación Técnica**

**GIT**

**SkillNet Ltda**

**19/02/2018**

Contenido

[Objetivo 5](#_Toc507083029)

[Alcance 5](#_Toc507083030)

[Justificación 5](#_Toc507083031)

[Requisitos Mínimos GIT 6](#_Toc507083032)

[Crear Repositorio Remoto 6](#_Toc507083033)

[Crear una cuenta 6](#_Toc507083034)

[Instalación GIT 8](#_Toc507083035)

[Flujo de trabajo 8](#_Toc507083036)

[Repositorio 8](#_Toc507083037)

[Configuración 9](#_Toc507083038)

[Crear Checkout 9](#_Toc507083039)

[Añadir y actualizar 9](#_Toc507083040)

[Envió de cambios 10](#_Toc507083041)

[Clonar repositorio remoto 10](#_Toc507083042)

[Ramas 12](#_Toc507083043)

[Crear una Rama 12](#_Toc507083044)

[Borrar una Rama 12](#_Toc507083045)

[Actualizar y fusionar 13](#_Toc507083046)

[Etiquetas 14](#_Toc507083047)

[Reemplazar cambios locales 14](#_Toc507083048)

[Datos adicionales 15](#_Toc507083049)

[Deploy Automático con Git 16](#_Toc507083050)

[Conclusiones 19](#_Toc507083051)

[Enlaces 20](#_Toc507083052)

[Clientes gráficos 20](#_Toc507083053)

[Guías 20](#_Toc507083054)

Tabla Ilustraciones

[Ilustración 1 Nuevo Repositorio 6](#_Toc507082983)

[Ilustración 2 Crear repositorio Remoto 7](#_Toc507082984)

[Ilustración 3 Repositorio 8](#_Toc507082985)

[Ilustración 4 Checkout 9](#_Toc507082986)

[Ilustración 5 add and commit 9](#_Toc507082987)

[Ilustración 6 añadir 10](#_Toc507082988)

[Ilustración 7 commit 10](#_Toc507082989)

[Ilustración 8 Enviar cambios 10](#_Toc507082990)

[Ilustración 9 Clonar repositorio 10](#_Toc507082991)

[Ilustración 10 URL GitHub 11](#_Toc507082992)

[Ilustración 11 Subir Repo a Remoto 11](#_Toc507082993)

[Ilustración 12 Crear rama 12](#_Toc507082994)

[Ilustración 13 Rama principal 12](#_Toc507082995)

[Ilustración 14 Push 12](#_Toc507082996)

[Ilustración 15 Borrar rama 12](#_Toc507082997)

[Ilustración 16 Actualizar repositorio 13](#_Toc507082998)

[Ilustración 17 Activar Rama 13](#_Toc507082999)

[Ilustración 18 añadir al index 13](#_Toc507083000)

[Ilustración 19 Revisar cambios de fusión 13](#_Toc507083001)

[Ilustración 20 Agregar Etiquetas 14](#_Toc507083002)

[Ilustración 21 Log 14](#_Toc507083003)

[Ilustración 22 Reemplazar cambios 14](#_Toc507083004)

[Ilustración 23 Deshacer cambios 14](#_Toc507083005)

[Ilustración 24 Deshacer cambios rama 14](#_Toc507083006)

[Ilustración 25 Interfaz grafica 15](#_Toc507083007)

[Ilustración 26 Colores de consola 15](#_Toc507083008)

[Ilustración 27 Linea por commit 15](#_Toc507083009)

[Ilustración 28 Archivos Interactivos 15](#_Toc507083010)

[Ilustración 29 Hooks para scripts 16](file:///C:\Users\skillnet1105\Desktop\Fo7502.Vv2-%20Formato%20%20Diagnostico%20INGE.docx#_Toc507083011)

[Ilustración 30 Crear Script 16](file:///C:\Users\skillnet1105\Desktop\Fo7502.Vv2-%20Formato%20%20Diagnostico%20INGE.docx#_Toc507083012)

[Ilustración 31 Código script 17](file:///C:\Users\skillnet1105\Desktop\Fo7502.Vv2-%20Formato%20%20Diagnostico%20INGE.docx#_Toc507083013)

[Ilustración 32 Dar permisos de script 18](file:///C:\Users\skillnet1105\Desktop\Fo7502.Vv2-%20Formato%20%20Diagnostico%20INGE.docx#_Toc507083014)

[Ilustración 33 Agregar usuario repositorio remoto 18](file:///C:\Users\skillnet1105\Desktop\Fo7502.Vv2-%20Formato%20%20Diagnostico%20INGE.docx#_Toc507083015)

[Ilustración 34 Permisos en el directorio Web 18](file:///C:\Users\skillnet1105\Desktop\Fo7502.Vv2-%20Formato%20%20Diagnostico%20INGE.docx#_Toc507083016)

[Ilustración 35 Ejecutar script automático 18](file:///C:\Users\skillnet1105\Desktop\Fo7502.Vv2-%20Formato%20%20Diagnostico%20INGE.docx#_Toc507083017)

# 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

# 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

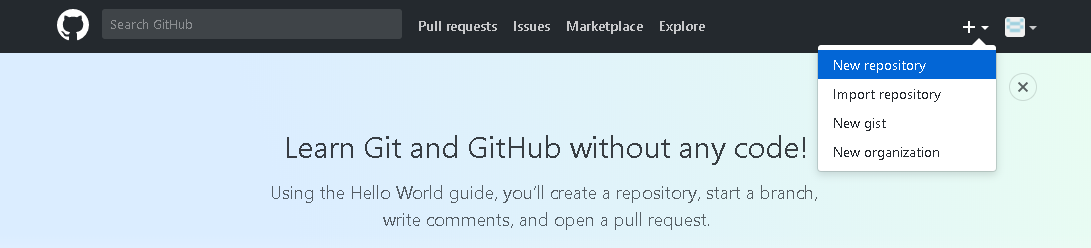


Ilustración 1 Nuevo Repositorio

* 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.

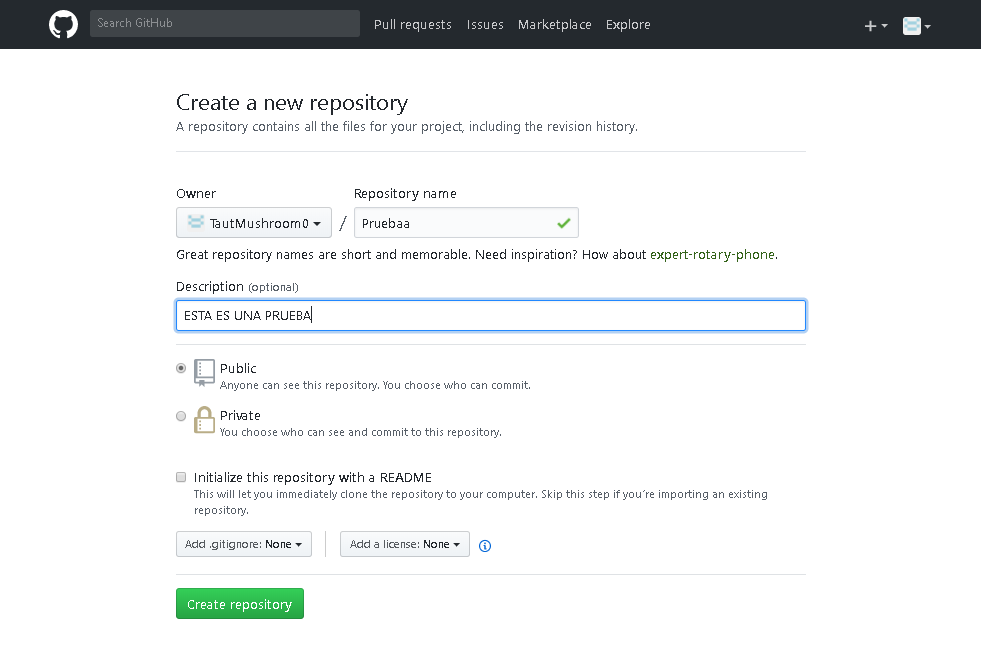


Ilustración 2 Crear repositorio Remoto

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).

# 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 Índex 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:

C:\Users\skillnet1105\Pictures\EDIT3.png

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

C:\Users\skillnet1105\Pictures\EDIT1.png

Ilustración 4 Checkout

Añadir y actualizar

Para poder añadir registros al índex es usando el comando:

C:\Users\skillnet1105\Pictures\EDIT2.png

Ilustración 5 add and commit

o



Ilustración 6 añadir

Para añadir todos los registros alojados en la carpeta GIT

Ahora para actualizar los cambios hay que registrarlos con un commit



Ilustración 7 commit

Envió 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.



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:



Ilustración 9 Clonar repositorio

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:

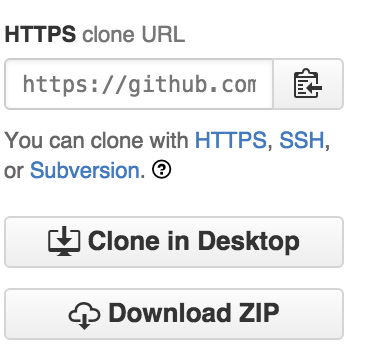


Ilustración 10 URL GitHub

* Es importante tener en cuenta que para poder interactuar con el Git a otros repositorios web hay que crear una llave de confianza para que permita la conexión de los mismos, Para verificar que existe una correcta conexión ingresamos:

C:\Users\skillnet1105\Pictures\git push.png

Ilustración 11 Conexión github

En caso que salga correcta la conexión, omitir los siguientes pasos debido a que ya puede subir archivos al repositorio remoto por medio de **git push,** Lo único que hay que verificar es que funcione el git pull, si sale error ingrese **Git pull origin master --allow-unrelated-histories** Y luego continúe subiendo los repositorios.

En caso contrario, si sale error, la forma de realizar una llave de confianza es con:

Ilustración 12 Generar llave de confianza

Ssh-keygen –t rsa –b 4096 –c “Su [Email@ejemplo.com](mailto:Email@ejemplo.com)”

Y asegurarse que el agente de ssh está funcionando.

Eval $(ssh-agent -s)

Ilustración 13 Validar agente ssh

Ilustración 14 Validar agente ssh

Agregue su clave privada SSH al agente ssh. Si creó su clave con un nombre diferente, o si está agregando una clave existente que tiene un nombre diferente, reemplace *id\_rsa* en el comando con el nombre de su archivo de clave privada.

ssh-add ~/.ssh/id\_rsa

Ilustración 15 Agregar clave privada a agente ssh

Es hora de agregar la llave publica al repositorio, para esto nos dirigimos al Github, donde posteriormente tendremos que estar logueado y dentro del repositorio correspondiente.

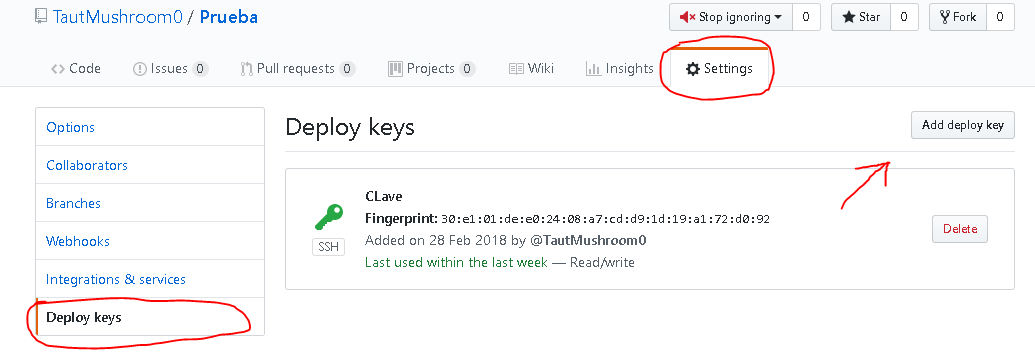


Ilustración 16 insertar llave publica a repositorio

Vamos a Settings, Deploy Keys y Add deploy key, a continuación, se ingresa un título de la llave y en el espació ingresamos el código de la llave publica, que la podemos encontrar en la ruta donde se generó **id\_rsa**.**pub** **(En caso que se haya puesto el mismo nombre),** copiamos la clave y la pegamos, posteriormente aceptamos y ya estará enlazado la configuración para que suba archivos automáticamente al repositorio.

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:



Ilustración 17 Crear rama

Ahora volver a la rama principal ejecutando:



Ilustración 18 Rama principal

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



Ilustración 19 Push

Borrar una Rama

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



Ilustración 20 Borrar rama

# Actualizar y fusionar

Para actualizar el repositorio local al commit mas nuevo, ejecuta



Ilustración 21 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:



Ilustración 22 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



Ilustración 23 añadir al index

Antes de fusionar los cambios pueden revisarse usando:



Ilustración 24 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



Ilustración 25 Agregar Etiquetas

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



Ilustración 26 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:



Ilustración 27 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:



Ilustración 28 Deshacer cambios



Ilustración 29 Deshacer cambios rama

Datos adicionales

Interfaz gráfica por defecto



Ilustración 30 Interfaz grafica

Colores especiales para la consola



Ilustración 31 Colores de consola

Mostrar sólo una línea por cada commit en la traza



Ilustración 32 Linea por commit

Agregar archivos de forma interactiva



Ilustración 33 Archivos Interactivos

# Deploy Automático con Git

Suponiendo que hemos montado un [repositorio git](https://www.linuxito.com/programacion/750-como-montar-un-servidor-git) en un servidor Web, los desarrolladores enviaran 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”.

Ilustración 34 Hooks para scripts

Cd .git/

Cd hooks/

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

Vi post-commit

Ilustración 35 Crear Script

Y ingresamos lo siguiente:

!/bin/sh

Git push origin master

Ilustración 36 Código script

Este script correrá automáticamente cada vez que se haga un **Commit** 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 37 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 38 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 39 Permisos en el directorio Web

Y ejecutamos con:

Ilustración 40 Ejecutar script automático

Git add .

Git commit –m “Nombre Version”

# 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.

# Enlaces

Clientes gráficos

* [GitX (L) (OSX, open source)](http://gitx.laullon.com/)
* [Tower (OSX)](http://www.git-tower.com/)
* [Source Tree (OSX, free)](http://www.sourcetreeapp.com/)
* [GitHub for Mac (OSX, free)](http://mac.github.com/)
* [GitBox (OSX)](https://itunes.apple.com/gb/app/gitbox/id403388357?mt=12)

Guías

* [Git Community Book](http://book.git-scm.com/)
* [Pro Git](http://progit.org/book/)
* [Think like a git](http://think-like-a-git.net/)
* [GitHub Help](http://help.github.com/)
* [A Visual Git Guide](http://marklodato.github.com/visual-git-guide/index-en.html)
* [Deploy](https://www.linuxito.com/programacion/880-como-implementar-deploy-automatico-en-git)