

**Título**

Manual Técnico

**Breve Descripción**

Manual técnico para la herramienta GIT dedicada al control de versiones en los distintos proyectos.

**Propósito**

La creación de este manual se realiza con el propósito de dar a conocer a detalle el proceso de instalación y configuración de la herramienta, así como estándares e infraestructura para el uso adecuado.

**Elaborado por**

SAXA

**Última revisión**

11/03/2020

**Referencias****Versión**

1.0

**Válido**De: *Diciembre/2020*A: *Diciembre/2021***Sustituye al Documento****Estado**☐ ☐ Draft☐ ☒ Definitive**Lista de Distribución**

SC Manager	RGA
Scrum Master	COMS
Software Project Manager	COMS
Technical Leader	MORR
SCM Group	BEGI, HEOX
SQA Group	RURL, TOBF
Software Engineering Group	MORR, MALZ
Test Group	ANBM, SARE, RITL, LORO
Data Analyst	MROM

**Autorización****Elaborado por**

SAXA

**Revisado por****Revisado por /  
Autorizado por**

<b>1. INTRODUCCIÓN .....</b>	<b>3</b>
1.1 GIT .....	3
1.2 ÁREAS DE GIT .....	3
<b>2. COMANDOS BÁSICOS .....</b>	<b>4</b>
<b>3. REQUERIMIENTOS .....</b>	<b>5</b>
<b>4. INSTALACIÓN DE GIT.....</b>	<b>6</b>
DESDE CÓDIGO FUENTE KERNEL GIT .....	6
DESDE EL REPOSITORIO DE IUS.....	7
INSTALAR DESDE LA FUENTE.....	9
INSTALACIÓN SENCILLA CON YUM .....	10
<b>5. ACTUALIZAR VERSIÓN DE GIT .....</b>	<b>11</b>
<b>6. CONFIGURACIÓN DE GIT .....</b>	<b>11</b>
<b>7. BRANCHING.....</b>	<b>12</b>
<b>8. CONEXIÓN A ALOJADOR DE REPOSITORIOS.....</b>	<b>13</b>
4.1 GitLAB .....	13
4.2 CODECOMMIT AWS .....	16
4.3 GitHub.....	17

## 1. INTRODUCCIÓN

En el presente documento se dan a conocer las instrucciones para el uso adecuado de la herramienta Git, la cual es requerida para llevar a cabo el control de versiones de los proyectos en Praxis.

El manual presenta una breve descripción acerca de Git, posteriormente se explican las áreas que lo componen, así como una serie de comandos más comunes a utilizar, después de ello se desglosan las funciones y su uso correspondiente.

**Este manual está basado en la versión de Git 2.25.1 instalado en un sistema operativo CentOS 7.**

### 1.1 GIT

Git es un sistema distribuido de control de versiones para rastrear cambios en el código fuente durante el desarrollo de software. Está diseñado para coordinar el trabajo entre programadores, pero se puede usar para rastrear cambios en cualquier conjunto de archivos. Sus objetivos incluyen velocidad, integridad de datos y soporte para flujos de trabajo no lineales distribuidos.

### 1.2 ÁREAS DE GIT

- + Directorio de Trabajo: Es donde se trabajan todos los archivos.
- + Área de prueba: Se preparan los archivos siendo versionados.
- + Repositorio: Son los cambios guardados.

## 2. COMANDOS BÁSICOS

**git init:** Indica que comenzaremos a utilizar GIT.

**git add <file>:** Pasa los archivos del Working Directory al Staging área.

**git status:** Ver en qué área se encuentran los archivos.

**git commit:** Pasa los archivos de Staging área al Repositorio.

**git push:** Sube los archivos a un repositorio remoto.

**git pull:** Trae los cambios que han hecho los colaboradores.

**git clone:** Hace una copia desde el servidor central donde está el código a nuestras computadoras para comenzar a trabajar.

**pwd:** Indica en qué directorio está ubicado.

**ls:** Lista los archivos que se encuentran dentro del directorio.

### 3. REQUERIMIENTOS

#### Hardware

En realidad, esta herramienta no necesita tantos recursos del equipo debido a que su manejo es solo por consola y es apoyada por los servicios web GitLab y GitHub.

500 MB son más que suficientes en el espacio de disco duro y cubre los sistemas operativos de Windows y Linux.

#### Software

Necesita un servidor CentOS 7 instalado y configurado con un usuario no root que tenga privilegios como “sudo”.

Cuenta con algunas dependencias que se instalan automáticamente junto con Git:

- Curl 7.68
- Expat 2.2.9
- Grep 14.0.2
- Openssl 1.1.1
- pcre2 10.34
- Perl 5.30.1

## 4. INSTALACIÓN DE GIT

Existen distintos métodos para instalar la herramienta Git en CentOS, sin embargo, cada uno se acopla a las necesidades y al contexto del uso que se vaya a dar.

### Desde código fuente Kernel Git

Esta primera alternativa es **la más recomendable** para instalar en los servidores con herramientas actualizadas dentro de Praxis, debido a que permite elegir exactamente la versión que se desea instalar y la encuentra sin ningún problema.

- Antes de instalar Git, asegúrese de que ya haya instalado los paquetes necesarios en su sistema:

```
yum install curl-devel expat-devel gettext-devel openssl-devel zlib-devel  
yum install gcc perl-ExtUtils-MakeMaker
```

- Use el siguiente comando para descargar y descomprimir Git 2.25.1:

```
cd /usr/src  
wget https://mirrors.edge.kernel.org/pub/software/scm/git/git-2.25.1.tar.gz  
tar xzf git-2.25.1.tar.gz
```

- Después de descargar y extraer el código fuente de Git, use el siguiente comando para compilar el código fuente:

```
cd git-2.25.1
```

```
make prefix=/usr/local/git all  
make prefix=/usr/local/git install
```

- Luego de la instalación del cliente git, ahora solo necesita configurar el binario en el entorno del sistema. Establezca la variable PATH con el binario git recién instalado en / etc / bashrc ejecutando el siguiente comando. Además, vuelva a cargar los cambios en el entorno actual:

```
echo 'export PATH=/usr/local/git/bin:$PATH' >> /etc/bashrc  
source /etc/bashrc
```

- Finalmente, use el siguiente comando para verificar la versión actual de Git:

```
git --version
```

## Desde el repositorio de IUS

Esta opción contiene una gama personalizada de opciones e instala las últimas versiones.

Puede preferir instalar Git desde IUS, una fuente de paquetes de calidad gestionados por la comunidad almacenados en el formato de archivo .rpm (paquetes RPM).

Inline with Upstream Stable (IUS) proporciona versiones actualizadas del software clave para CentOS y Red Hat Enterprise Linux (RHEL).

**Nota: No puede actualizar directamente desde paquetes de stock a paquetes IUS y solo toma una última versión como referencia, es decir, puede que no instale exactamente la 2.25.1 sino alguna anterior a partir de la 2.0.**

- Comience por instalar las herramientas de desarrollo con el siguiente comando:

**sudo yum groupinstall "Development Tools"**

- Luego ejecute:

**sudo yum install gettext-devel openssl-devel perl-CPAN perl-devel zlib-devel**

- Instale CentOS 7 repo desde IUS usando el comando:

**sudo yum install <https://centos7.iuscommunity.org/ius-release.rpm>**

- Obtenga en paquete de instalación de Git:

**sudo yum install git2u-all**

- Ahora se debe tener la última versión de Git, para comprobarlo, ingrese el comando:

**git --version**



## Instalar desde la fuente

En este método, tendrá la tarea de construir Git desde el código fuente. Puede especificar la versión requerida. El único riesgo de esta alternativa es que puede que la versión que escriba no exista en el repositorio.

- Comience por instalar las herramientas de desarrollo con el siguiente comando:

```
sudo yum groupinstall "Development Tools"
```

- Luego descargue las dependencias:

```
sudo yum -y install wget perl-CPAN gettext-devel perl-devel openssl-devel zlib-devel
```

- Descargue Git:

```
export VER="2.25.1"  
wget https://github.com/git/git/archive/v${VER}.tar.gz
```

- Descomprima el archivo descargado, copie y pegue el directorio y en la ruta deseada:

```
tar -xvf v${VER}.tar.gz  
rm -f v${VER}.tar.gz
```

- Entrar a la carpeta traída de Git y realizar la instalación.

```
cd git-*  
sudo make install
```

- Ahora se debe tener la última versión de Git, para comprobarlo, ingrese el comando:

**git --version**

## Instalación sencilla con YUM

Esta alternativa es sumamente simple, no obstante, no cuenta con muchas de las ventajas en la instalación, tal como se ha visto anteriormente; debido a que no es posible determinar la versión requerida a descargar y por lo tanto, obtiene la versión por default, trayendo así Git 1.8.

- Busque e instale la última versión con el comando:

**yum install git**

- Verifique la instalación con el comando:

**git --version**

## 5. ACTUALIZAR VERSIÓN DE GIT

En ocasiones, es necesario actualizar a la última versión de Git de uno ya instalado, para eso solo debe seguir estos sencillos pasos:

- Primero, debe eliminar la versión ya instalada con el comando:

**`sudo yum remove git`**

**NOTA\*\*** Si se hizo la instalación desde algún repositorio, no eliminará ningún archivo ya que la nueva fuente de código puede sobrescribirse sobre la anterior, por lo tanto puede omitir el comando de eliminado.

- Prosiga con alguna de las instalaciones ya antes mencionadas, aunque se recomienda aplicar para la actualización desde [Código Fuente Kernel Git](#), ya que es más precisa y cubre hasta la última versión.

## 6. CONFIGURACIÓN DE GIT

- Antes que nada, después de instalar Git, lo primero que debería hacer es lanzar un par de comandos de configuración.

**`git config --global user.name "Tu nombre aquí"`**

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

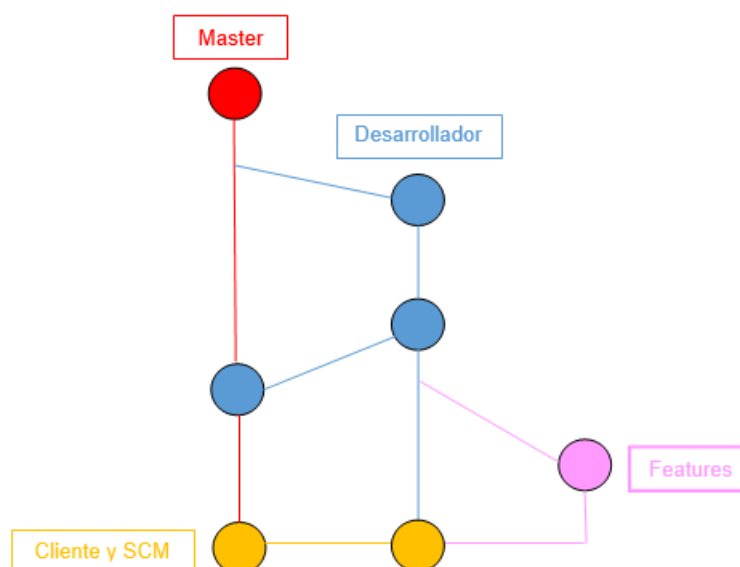
- Con estos comandos indica su nombre de usuario (se usa nombre y apellidos generalmente) y el email. Esta configuración sirve para que cuando haga commits en el repositorio local, éstos se almacenen con la referencia a usted mismo. Gracias a ello, más adelante cuando obtenga información de los cambios realizados en el los archivos del repositorio local, le va a aparecer como responsable de esos cambios a este usuario y correo que ha indicado.

## 7. BRANCHING

Para el control de versiones en Git, se debe contar con una rama llamada “**Master**”, la cual será siempre la versión base y original del proyecto. En caso del área de desarrollo de software, a partir de la ramificación o branch “Master”, se derivan otras, una es dedicada para el **desarrollador**, esta persona, realiza los cambios al proyecto y una vez listo, puede realizar sus commits y plasmarlos sobre la rama principal. Mientras que el otro branch es para **el cliente y SCM**, de igual forma, los cambios que se realicen y se aprueben, pueden visualizarse en la ramificación base.

Desde la rama del desarrollador, puede crearse otra para las modificaciones llamadas “**Feature**”, sin modificar a la original, es decir, sus cambios se suben primero al branch del desarrollador y una vez que esté aprobada y no contenga errores, puede subirse a la rama “Master”.

En realidad, se requieren de pocos branches, lo único que no se debe perder de vista jamás es la ramificación “Master”, ya que será la que lleve todo el proyecto completo y en correcto funcionamiento. Cada branch base pertenece a un solo proyecto.



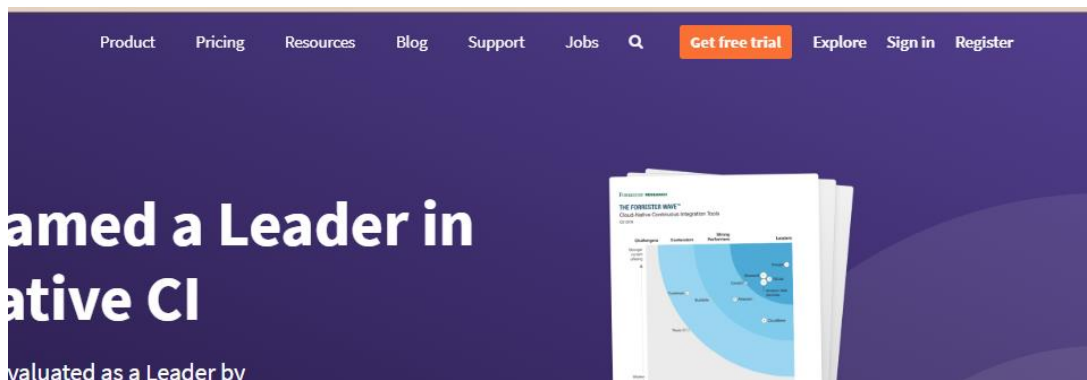
## 8. CONEXIÓN A ALOJADOR DE REPOSITORIOS

El controlador de versiones de Git, debe ser conectado a un hosting donde los usuarios puedan alojarlos y manipularlos a través de servicios. Existen varios alojadores de repositorios que ayudan a llevar a cabo dichos procedimientos, los cuales se describen a continuación.

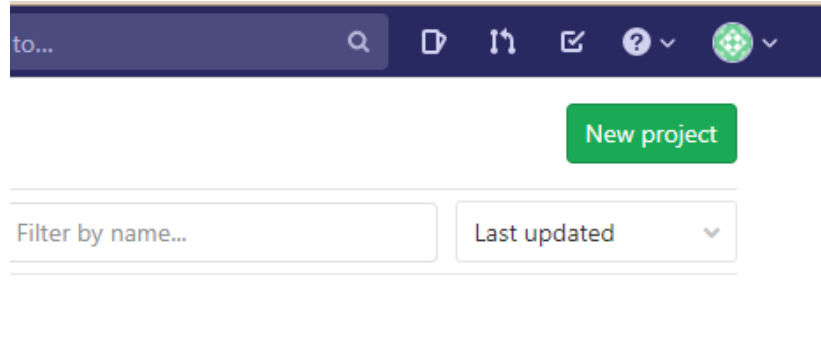
### 4.1 GitLab

GitLab es una plataforma de hosting enfocada especialmente a nivel empresarial para desarrollar y compartir proyectos a partir del controlador de versiones GIT, sus precios de servicios van desde **\$70.00 mxn hasta \$1,900.00 mxn**.

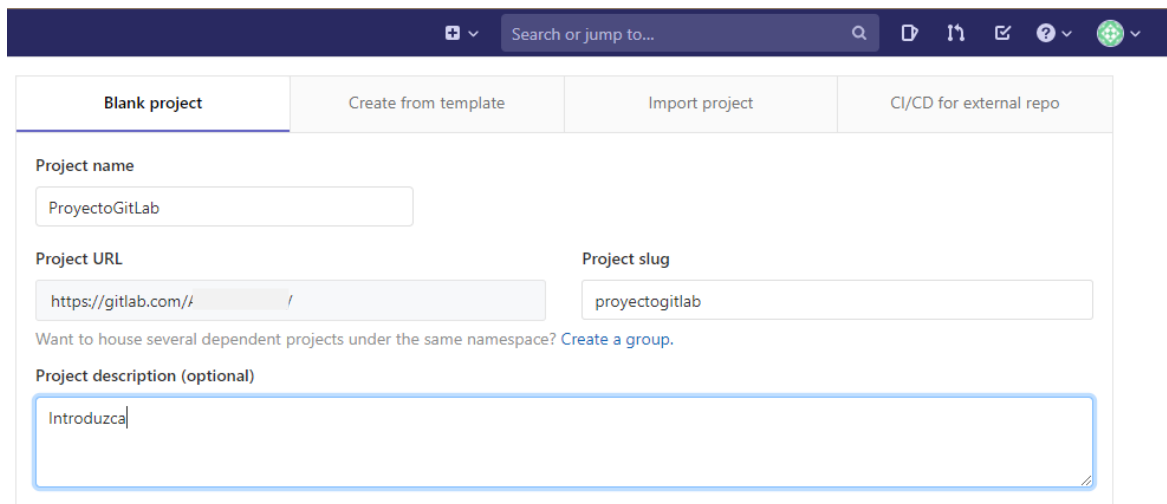
- Para subir y compartir sus proyectos con otros colaboradores, primero debe conocer de qué manera está implementado GitLab, si ha sido instalado localmente en un servidor debe tener acceso a la dirección web que se le haya asignado. En caso de no haber sido instalado y estar disponible por medio de la nube basta solo con ingresar a la página <https://gitlab.com>
- Si antes ya se ha registrado en esta página, solo haga clic en la esquina superior derecha en el botón “Sign in” para acceder a su cuenta.
- En caso de no estar registrado, haga clic en el botón “Register” para crear un nuevo perfil.



- Una vez que haya ingresado con su perfil a GitLab, en la pantalla principal diríjase al botón verde de “Nuevo Proyecto” y haga clic.



- Abrirá una ventana con una serie de campos a llenar, en la cual se debe colocar el nombre del proyecto y una descripción de este.

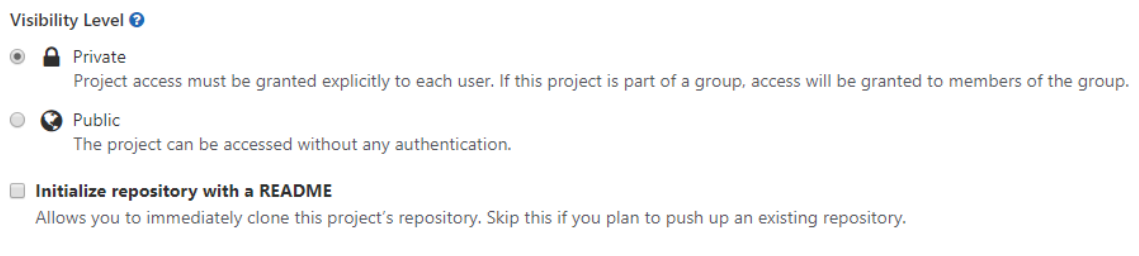


The screenshot shows the 'New project' form in GitLab. The form has four tabs: 'Blank project' (selected), 'Create from template', 'Import project', and 'CI/CD for external repo'. The 'Blank project' tab contains the following fields:

- Project name:** A text input field with the value 'ProyectoGitLab'.
- Project URL:** A text input field with the value 'https://gitlab.com/'.
- Project slug:** A text input field with the value 'proyectogitlab'.
- Project description (optional):** A large text area with the placeholder text 'Introduzca'.

Below the 'Project URL' and 'Project slug' fields, there is a link that says 'Want to house several dependent projects under the same namespace? [Create a group.](#)'

- En la parte de abajo puede elegir la modalidad en la que desea su proyecto, público o privado.



The screenshot shows the 'Visibility Level' section of the 'New project' form. It includes the following options:

- Private:** Selected by default. Description: 'Project access must be granted explicitly to each user. If this project is part of a group, access will be granted to members of the group.'
- Public:** Description: 'The project can be accessed without any authentication.'
- Initialize repository with a README:** A checkbox that is currently unchecked. Description: 'Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.'

- Finalmente haga clic en el botón inferior que dice “Crear repositorio”.
- Mostrará una serie de líneas de comandos que deben insertarse en la consola de GIT para llevar a cabo el repositorio de su proyecto, pero solo debe enfocarse en algunos de ellos.
- El recuadro para crear un nuevo repositorio es en caso de haber sido invitado a modificar el código y debe descargar el proyecto por primera vez, si es así, coloque las líneas de comando de este apartado una por una.

#### Create a new repository

```
git clone https://gitlab.com/AlmaSauceda/proyectgitlab.git
cd proyectgitlab
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

- El siguiente recuadro es para compartir un folder o proyecto existente dentro del equipo de cómputo, al igual que el anterior, se coloca línea por línea en la ventana de comandos de GIT.

#### Push an existing folder

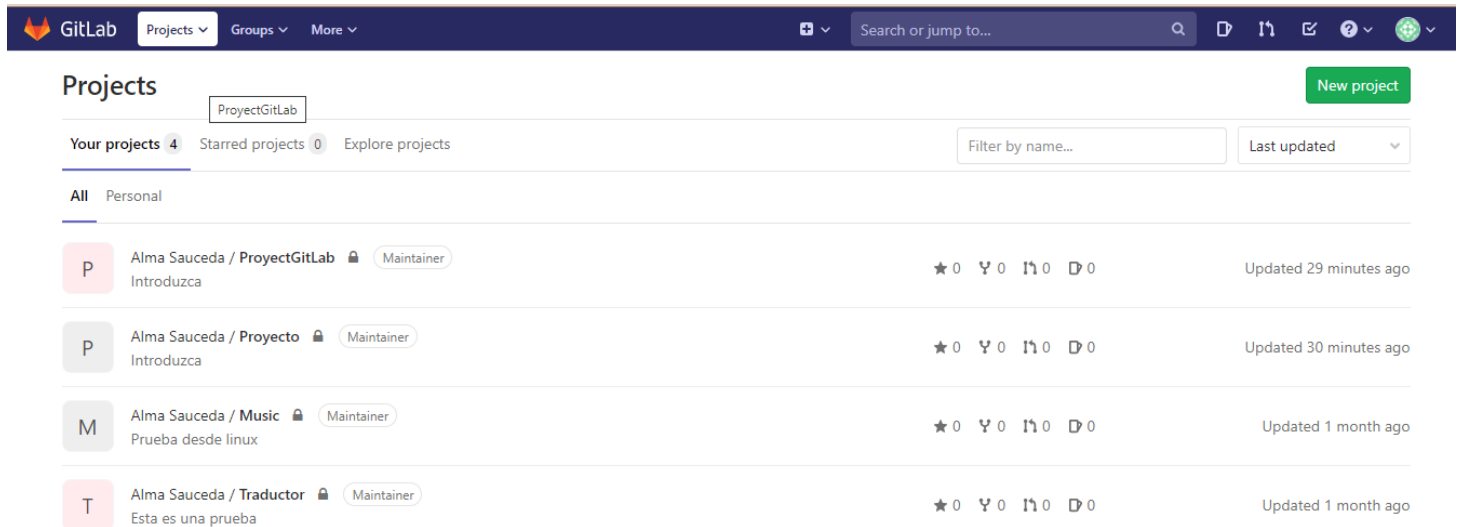
```
cd existing_folder
git init
git remote add origin https://gitlab.com/AlmaSauceda/proyectgitlab.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

- Existe otro recuadro en la parte inferior, este es para subir un repositorio que ya tengamos creado en GIT a la plataforma de hosting.

#### Push an existing Git repository

```
cd existing_repo
git remote rename origin old-origin
git remote add origin https://gitlab.com/AlmaSauceda/proyectgitlab.git
git push -u origin --all
git push -u origin --tags
```

- Luego de eso, su repositorio habrá sido agregado y vinculado con éxito y al refrescar la pantalla de inicio de GitLab notará que se ha añadido a la lista de sus proyectos.



**\*\*\*NOTA:** GitLab cuenta con distintos protocolos para la conexión servidor-cliente, puede ser sencilla por medio de HTTPS o por medio de SSH, el cual es un protocolo más seguro sobre todo a nivel empresarial.

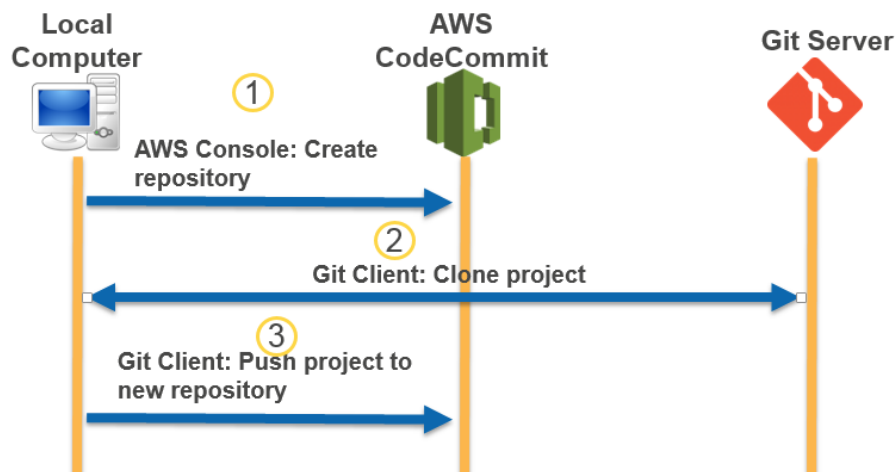
## 4.2 CodeCommit AWS

AWS CodeCommit es un servicio completamente administrado de control de código fuente que aloja repositorios basados en Git seguros. Simplifica la colaboración en el código por parte de los equipos, en un ecosistema seguro y con alta escalabilidad. Con CodeCommit no necesita utilizar su propio sistema de control de código fuente ni preocuparse por el escalado de la infraestructura de dicho sistema. CodeCommit, que funciona perfectamente con las herramientas de Git existentes, se puede utilizar para almacenar de forma segura cualquier elemento, ya sea código fuente o binario.



Este servicio, facilita el uso de la herramienta, debido a que todo lo trabaja a través de la nube, por lo tanto no es necesario tener distintas instancias de GIT instaladas, sino todo lo maneja en una sola sin riesgo de saturación de usuarios.

Su precio también es una de sus ventajas, ya que los primeros 5 usuarios son totalmente gratis, rebasando ese número el costo a pagar es de **\$1 dólar al mes**, brindando repositorios ilimitados, 10 GB/mes de almacenamiento por usuario activo y 2 000 solicitudes Git/mes por usuario activo.

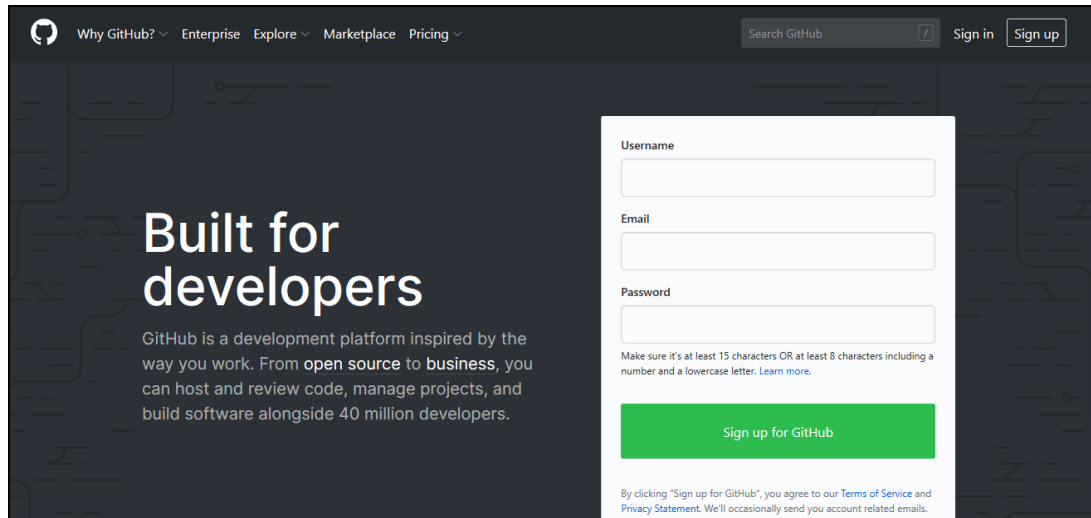


### 4.3 GitHub

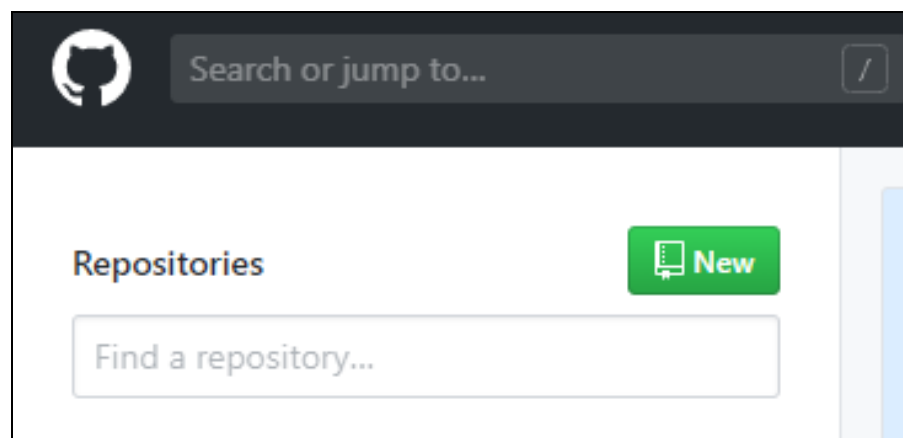
Para este paso, se requiere de una herramienta muy relacionada a GIT llamada GitHub, la cual es una plataforma para el hosting de los proyectos entre una comunidad de personas que desarrollan y comparten, usando GIT. **Sus precios van desde \$130.00 mxn hasta \$170.00 mxn al mes, aún tienen un paquete para equipos más grandes, pero para ello, es necesario contactar al personal de ventas.**

- Para subir y compartir sus proyectos con otros colaboradores, tiene que ingresar en su navegador web a la página <https://github.com>

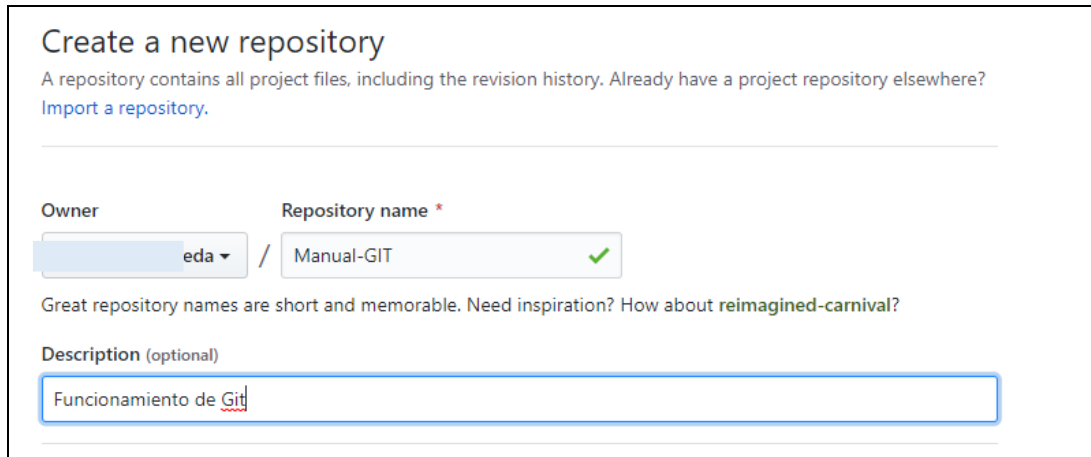
- Si ya se ha registrado en dicha página anteriormente solo haga clic en la esquina superior derecha en el botón “Sign in” para acceder a su cuenta.
- En caso de no estar registrado, cree un nuevo perfil llenando los datos que le piden en la página principal.



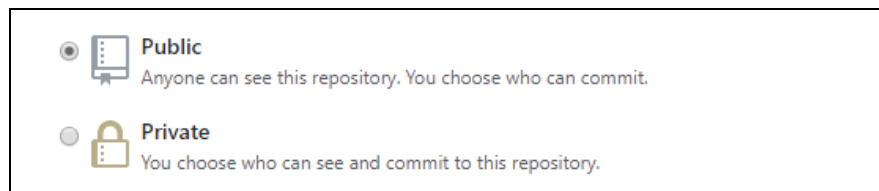
- Una vez que haya ingresado con su perfil a GitHub, diríjase al apartado de “Repositorios” y de clic en “Nuevo”.



- En el nombre del repositorio escriba el que usted prefiera, solo cuide el no repetir el nombre de algún repositorio ya existente (En caso de haber contado con perfil de GitHub anteriormente).
- Debajo puede colocar una descripción de su proyecto.



- Después, elija si quiere que sus archivos sean públicos o privados, es decir, si su proyecto es público cualquier persona podrá ver el contenido de la carpeta, sin hacer commits. Mientras que en privado, puede determinar quiénes pueden visualizar el contenido y hacer commits. No obstante, esta última opción solo aplica si tiene una suscripción especial, y por lo tanto lleva un costo monetario de por medio.



- Finalmente haga clic en el botón inferior que dice “Crear repositorio”.

- Mostrará una serie de líneas de comandos que deben insertarse en la consola de GIT para llevar a cabo el repositorio de su proyecto, pero solo debe enfocarse en algunos de ellos.
  - El primero es:

```
git remote add origin https://github.com/AlmaSauceda/Manual-GIT.git
```

Copie y péguelo en consola para indicar al proyecto en dónde se va almacenar.

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git remote add origin https://github.com/AlmaSauceda/Manual-GIT.git
```

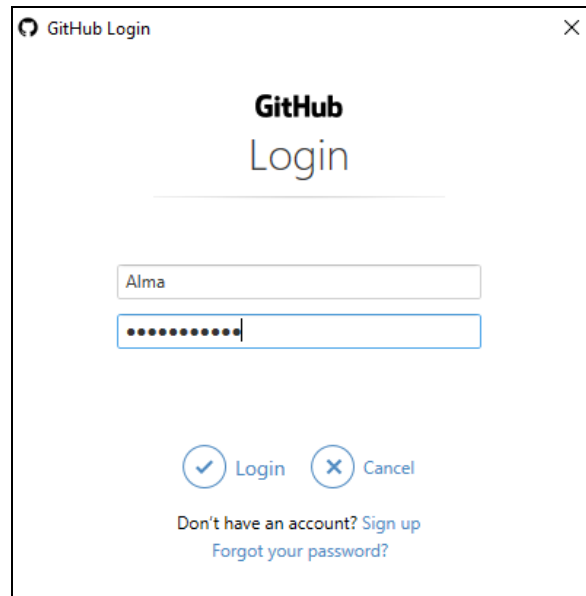
- El segundo es:

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

Copie y péguelo en consola para lanzar su proyecto a GitHub.

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git push -u origin master
```

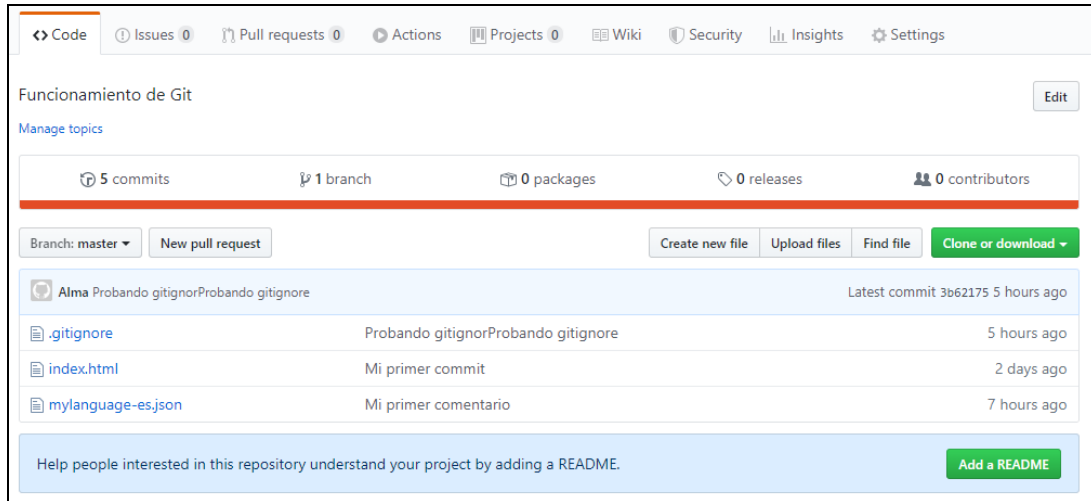
- Le abrirá una pequeña ventana de GitHub, donde le pedirá el usuario y contraseña de su perfil de GitHub.



- Presione “Login”.
- De inmediato en su consola de GIT comenzará a subir todos los archivos de su proyecto a GitHub.

```
B.Alma@201701775-FRT MINGW64 ~/documents/traductor (master)
$ git push -u origin master
Logon failed, use ctrl+c to cancel basic credential prompt.
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 4 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (12/12), 1.61 KiB | 548.00 KiB/s, done.
Total 12 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/AlmaSauceda/Manual-GIT.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

- Luego, refresque la página de su repositorio en GitHub, y aparecerá todo el proyecto y los cambios que ha realizado.



- Al hacer clic en cada archivo podrá visualizar el código o el contenido en cada uno de ellos.

