

2020

# Manual básico GitHub

El manual se centra en la creación de un repositorio y el uso habitual.

Despliegue de aplicaciones web

Daniel Carmona Gilibert  
Jesús Manuel Tortolero Martín  
Daniel Gómez Reina  
Juan Carlos Alvarado Salguero  
Yasmina Torres Elena



## 1 Autores

- **Daniel Carmona Gilibert** -> Puntos 5 y 6
- **Jesús Manuel Tortolero Martín** -> Puntos 4 y 7
- **Daniel Gómez Reina** -> Puntos 2 y 3
- **Juan Carlos Alvarado Salguero** -> Puntos 7 y 8
- **Yasmina Torres Elena** -> Puntos 9 y 1

## 2 Índice

1	Autores .....	2
2	Índice .....	2
3	Tabla de contenidos .....	3
4	Introducción .....	4
5	Instalar Git .....	4
6	Crear una cuenta en GitHub .....	4
7	Gestión de proyectos .....	5
7.1	Tipos de ramas .....	5
7.2	Crear repositorio .....	6
7.3	Crear un proyecto .....	8
7.4	Subir proyecto .....	8
7.5	Colaborar en un proyecto .....	8
8	Glosario .....	10
9	Bibliografía .....	10

### 3 Tabla de contenidos

<i>Ilustración 1. Comando de instalación de git para linux.....</i>	<i>4</i>
<i>Ilustración 2. Tipos de cuentas que se pueden crear en GitHub .....</i>	<i>4</i>
<i>Ilustración 3. Formulario de creación de una cuenta en GitHub .....</i>	<i>5</i>
<i>Ilustración 4. Creación de un nuevo repositorio .....</i>	<i>6</i>
<i>Ilustración 5. Formulario de creación de nuevo repositorio .....</i>	<i>6</i>
<i>Ilustración 6. Inicialización del repositorio git apuntando al repositorio GitHub .....</i>	<i>7</i>
<i>Ilustración 7. Comandos de creación de proyecto .....</i>	<i>8</i>
<i>Ilustración 8. Comandos para subir un archivo .....</i>	<i>8</i>
<i>Ilustración 9. Añadir colaborador a un proyecto .....</i>	<i>8</i>
<i>Ilustración 10. Clonar un proyecto.....</i>	<i>9</i>
<i>Ilustración 11. Comandos para subir cambios a máster desde rama local .....</i>	<i>9</i>
<i>Ilustración 12. Comandos para subir cambios a una rama y luego hacer merge a la master.....</i>	<i>9</i>

## 4 Introducción

**GitHub** es una plataforma de desarrollo colaborativo de software para **alojar proyectos** usando el sistema de control de versiones Git.

**Git** es un software de **control de versiones**. El control de versiones, gestiona los diversos cambios que se realizan sobre un repositorio

El código se almacena de forma pública, aunque también se puede hacer de forma privada.

GitHub no sólo ofrece alojamiento del código si no muchas más posibilidades asociadas a los repositorios como son, forks, issues, pull requests, etc. Se detallan en el gloriario final.

## 5 Instalar Git

```
sudo apt-get install git
```

Ilustración 1. Comando de instalación de git para linux

En Windows es un ejecutable dependiendo de la versión <https://git-scm.com/download/win>

## 6 Crear una cuenta en GitHub

1. Entre en <https://github.com/pricing#feature-comparison>

Pricing

### Plans for all developers

**GitHub is now free for teams**  
GitHub Free gives teams private repositories with unlimited collaborators at no cost. GitHub Team is now reduced to \$4 per user/month. [Try GitHub Free](#)

Free	Team	Enterprise	GitHub One
Basics for teams and developers	Advanced collaboration and support for teams	Security, compliance, and flexible deployment for enterprises	All of our best tools, support, and services
<ul style="list-style-type: none"><li>Unlimited public/private repositories</li><li>Unlimited collaborators</li><li>2,000 Actions minutes/month</li><li>Free for public repositories</li><li>500MB of GitHub Packages storage</li><li>Free for public repositories</li><li>Community Support</li></ul>	<ul style="list-style-type: none"><li>Unlimited public/private repositories</li><li>Required reviewers</li><li>3,000 Actions minutes/month</li><li>Free for public repositories</li><li>500MB of GitHub Packages storage</li><li>Free for public repositories</li><li>Code owners</li></ul>	<ul style="list-style-type: none"><li>Everything included in Team</li><li>SAML single sign-on</li><li>50,000 Actions minutes/month</li><li>Free for public repositories</li><li>500MB of GitHub Packages storage</li><li>Free for public repositories</li><li>Advanced auditing</li></ul>	<ul style="list-style-type: none"><li>Everything included in Enterprise</li><li>Community-powered security</li><li>Actionable metrics</li><li>24/7 support</li><li>Continuous learning</li></ul>
\$0/month	\$4 per user/month	\$21 per user/month	<a href="#">Learn more</a>
<a href="#">Join for free</a>	<a href="#">Continue with Team</a>	<a href="#">Contact Sales</a>	<a href="#">Contact Sales</a>

Ilustración 2. Tipos de cuentas que se pueden crear en GitHub

2. Se selecciona la opción que se adapte a las necesidades, como caso básico usaremos la gratuita. Finalmente, solo habrá que llenar un pequeño formulario

Ilustración 3. Formulario de creación de una cuenta en GitHub

## 7 Gestión de proyectos

### 7.1 Tipos de ramas

**Github suele usarse usando el modelo de branching**, que propone una estrategia de ramificación y generación de versiones de productos usando un repositorio Git. +

Para entornos pequeños se recomienda usar solo una rama master y ramas features. A continuación se hace un resumen de las ramas, aunque en el desarrollo del proyecto se ha usado inicialmente una sola rama master para más tarde

Tipo de ramas	
<b>Principales</b> <ul style="list-style-type: none"> <li>• Solo hay una de cada tipo, no hay más instancias</li> <li>• Nadie puede hacer commit directamente en ellas</li> <li>• Solo es posible incorporar código a través de otra rama</li> <li>• Su nombre es igual al tipo de rama, master y develop</li> </ul>	
<b>Master</b>	Siempre que crea un nuevo proyecto, se crea una rama predeterminada llamada máster. Esta es la rama en la que su trabajo debería terminar eventualmente, una vez que esté listo para pasar a producción.
<b>Develop</b>	Se genera a partir de la master. Contiene el código que aún no está en producción. No debe hacerse commit directamente sobre develop.
<b>Auxiliares</b> <ul style="list-style-type: none"> <li>• Se pueden instanciar todas las veces que se considere</li> <li>• El objetivo de instanciarlas es para trabajar directamente sobre ellas</li> <li>• Su nombre está compuesto por el tipo y por su objetivo, funcionalidad, versión, etc.</li> </ul>	
<b>Feature</b>	Siempre que cree una nueva función, creará una rama para trabajar en ella. Eso se llama rama feature.
<b>Release</b>	Si tiene un proceso de control de calidad manual o tiene que admitir versiones antiguas de su software para sus clientes, es posible que necesite una rama de lanzamiento como un lugar para realizar las correcciones o actualizaciones necesarias. No existe una diferencia técnica entre una feature y una realease, pero la distinción es útil

	cuando se habla de un proyecto con su equipo.
<b>Hotfix</b>	Se crea una rama hotfix cuando se detecta un defecto que debe ser solucionado de forma urgente, se copia el código de la rama master. Se solventa la incidencia y se vuelca el cambio en máster y develop.

## 7.2 Crear repositorio

1. Botón “+” *del nav superior de la web*, habiendo realizado login en GitHub. Para crear un repositorio en GitHub, solo hay que seleccionar el botón “New repository” del desplegable que se muestra

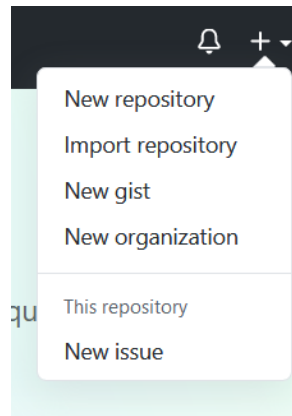


Ilustración 4. Creación de un nuevo repositorio


2. Rellenar el nombre del repositorio y opcionalmente la descripción del repositorio

Create a new repository

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

---


Owner \*      Repository name \*


 juank94daw /

Great repository names are short and memorable. Need inspiration? How about [congenial-guide](#)?

Description (optional)

---

☒  **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:

Skip this step if you're importing an existing repository.

☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

---

Ilustración 5. Formulario de creación de nuevo repositorio

Una vez creado el repositorio en Github se deberá crearlo en el PC siguiendo estos pasos:

1. Crear la carpeta en donde se alojará el repositorio, se recomienda alojar todos los repositorios bajo un mismo directorio raíz. Puedes usar el comando **pwd** para comprobar en que ubicación te encuentras, **cd** para situarte en un directorio en concreto, **ls** para listar lo que hay dentro del directorio. Para crear la carpeta puedes usar el comando **mkdir**.
2. Inicializar el repo de git. Una vez se llega al que será el directorio del repo se debe introducir lo siguiente para iniciar el repositorio en git y apuntar al repositorio de GitHub

```
<git init>  
  
<git remote add origin git@github.com:nombreusuario/nombrerepo.git>
```

**Ilustración 6. Inicialización del repositorio git apuntando al repositorio GitHub**

### 7.3 Crear un proyecto

1. Acceder al repositorio donde se quiera crear el proyecto. Si el proyecto no contiene aún datos muestra únicamente una ayuda para subir archivos y proyectos. Para crear un proyecto desde cero, habrá que comenzar creando los archivos del mismo y luego subiéndolos a la página. En el primer recuadro de la ayuda, verás una serie de comandos para el terminal.

```
touch README.md
git init
git add README.md
git commit -m "comentario"
git remote add origin
https://github.com/juank94daw/Repositorio.git
git push -u origin master
```

Ilustración 7. Comandos de creación de proyecto

2. Ejecutar los comandos listados en la ilustración 7

### 7.4 Subir proyecto

Ubicarse en la carpeta del repositorio y realizar los siguientes comandos, teniendo en cuenta que el comentario es obligatorio o no se realizará el cambio. Se debe añadir los archivos, hacer el commit con comentario y realizar la subida.

```
git add archivo
git commit -m "comentario"
git push
```

Ilustración 8. Comandos para subir un archivo

### 7.5 Colaborar en un proyecto

Para colaborar en un proyecto el propietario debe darle permisos:

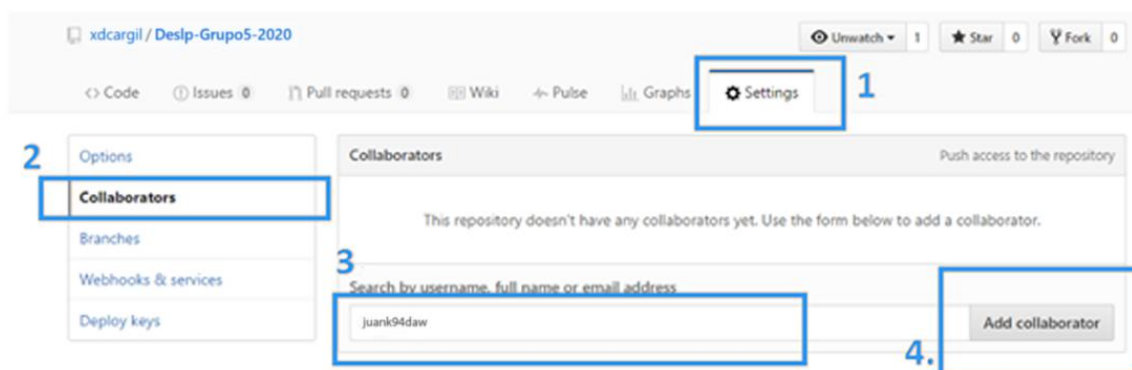


Ilustración 9. Añadir colaborador a un proyecto



1. Situar en el directorio local donde colocaremos el repositorio, clonar el repositorio, colocarse en él y crear una nueva rama tipo feature que denominará como deseé para colaborar desde tu usuario:

```
git init
git clone https://github.com/xdcargil/Deslp-Grupo5-2020.git
cd directorio del repositorio
git branch juank94daw
git checkout juank94daw
```

**Ilustración 10. Clonar un proyecto**

Una vez creada la rama se puede comprobar en que rama se está con el comando `git branch` el cual lista las ramas y destaca la actual.

2. Una vez con el repositorio en local de manera actualizada hay que realizar los siguientes pasos que para subir un proyecto. El `add .` permite añadir todos los archivos que hayan sido modificados sin necesidad de especificarlos. A continuación se detallan los pasos a seguir para añadir los cambios de la rama local `'juank94daw'` a `master` realizando antes un `pull` para asegurarnos de tener actualizado el contenido de `master`.

```
git add .
git commit -m "comentario"
git checkout master
git pull
git merge juank94daw
git push
```

**Ilustración 11. Comandos para subir cambios a máster desde rama local**

En caso de querer crear en el github la rama y no solo localmente.

```
git add .
git commit -m "comentario"
git push origin juank94daw
git checkout master
git pull
git merge juank94daw
git push origin master
```

**Ilustración 12. Comandos para subir cambios a una rama y luego hacer merge a la master**

## 8 Glosario

Glosario de términos	
<b>Repositorio</b>	Un repositorio es el nombre que recibe el lugar donde se aloja el código de un proyecto de desarrollo en algún lenguaje de programación
<b>Commit</b>	Acción de guardar cambios en uno o más archivos, puede crear un nuevo registro también. Un commit es como una instantánea de los archivos en ese momento.
<b>Commit message</b>	Cada vez que realiza un commit, debe proporcionar obligatoriamente un mensaje que describa por qué se realizó el cambio, sino no permite hacer el commit. Ese mensaje es muy útil cuando se trata de comprender más tarde por qué se implementó cierto cambio.
<b>Push</b>	Publica en el servidor remoto de Github lo que se encuentra en nuestro servidor local.
<b>Pull</b>	Obtiene los cambios del repositorio remoto y actualiza nuestro repositorio local.
<b>Branch</b>	Una rama 'Branch' es una serie independiente de confirmaciones a un lado que puede utilizar para probar un experimento o crear una nueva función.
<b>Merge</b>	Un merge es una forma de tomar el trabajo completado de una rama e incorporarlo a otra rama. Por lo general, fusionará una rama de funciones con la rama maestra.
<b>Tag</b>	Una etiqueta es una referencia a un commit histórica específica. Las etiquetas se utilizan con mayor frecuencia para documentar las versiones de producción para que sepa exactamente qué versiones del código entraron en producción y cuándo.
<b>Checkout</b>	La salida le permite ir a una versión diferente del historial del proyecto y ver los archivos a partir de ese momento. Por lo general, verá una rama para ver todo el trabajo que se ha realizado en ella, pero cualquier confirmación puede ser algo que verifique.
<b>Wiki</b>	Desarrollados originalmente por Ward Cunningham, los wikis son una forma liviana de crear páginas web con enlaces simples entre ellos. Los proyectos de GitHub suelen utilizar wikis para la documentación.
<b>Clone</b>	A menudo, hay que descargar una copia de un proyecto de GitHub para poder trabajar en él localmente. El proceso de copiar el repositorio a su ordenador se llama clonación. Comando para clonar: <code>&lt;git clone linkhttp&gt;</code>
<b>Fork</b>	A veces, no tienes los permisos necesarios para realizar cambios directamente en un proyecto. Quizás sea un proyecto de código abierto escrito por personas que no conoce, o un proyecto escrito por otro grupo de su empresa con el que no trabaja mucho. Si desea enviar cambios a dicho proyecto, primero debe hacer una copia del proyecto en su cuenta de usuario en GitHub. Ese proceso se llama bifurcación del repositorio 'Fork'. Luego puede clonarlo, realizar cambios y enviarlos de nuevo al proyecto original mediante una solicitud de extracción 'Pull request'.
<b>Pull request</b>	Originalmente, se usaba para solicitar que otra persona revisara el trabajo que había completado en una rama y luego lo fusionara con el maestro. Ahora, se utilizan a menudo al principio del proceso para iniciar una discusión sobre una posible función.

## 9 Bibliografía

<https://conociendogithub.readthedocs.io/en/latest/data/dinamica-de-uso/>

<https://github.com/Hispano/Guia-sobre-Git-Github-y-Metodologia-de-Desarrollo-de-Software-usando-Git-y-Github>

<https://gist.github.com/ElenaMLopez/ca17cb107b6a3c51946279f8fd67327d>

[https://www.youtube.com/playlist?list=PLPI81lqbj-4l8i-x2b5\\_MG58tZfgKmJls](https://www.youtube.com/playlist?list=PLPI81lqbj-4l8i-x2b5_MG58tZfgKmJls)