

# Terminal, Git, GitHub y GitHub Pages (Parte II)

---

## GitHub

---

### Introducción a GitHub

Existen varios tipos de repositorios remotos y empresas asociadas a proporcionarlos, las más usadas son **GitHub**, Bitbucket y gitlab. Todos funcionan de forma similar y lo importante es que soportan git, permitiéndonos de esta forma gestionar con los mismos comandos desde nuestros computadores, independiente del servicio que utilicemos.

Nosotros utilizaremos GitHub.

Como mencionamos, GitHub es un gestor de repositorios remotos, lo que quiere decir que podemos almacenar una copia de nuestro código en sus servidores. Así podemos trabajar colaborativamente y respaldar nuestro trabajo.

GitHub es gratis y no tiene restricciones de la cantidad de repositorios que podamos crear.

Así que si no tienes tu cuenta de GitHub creada ve a crearla a [GitHub.com](https://github.com)

¿Qué es un repositorio remoto?

Los repositorios remotos son versiones de tu proyecto que se encuentran alojados en Internet o en algún punto de la red.

## Configuración de GitHub

Una parte importante de conexión con cualquier servidor, es la autenticación, es decir, el procedimiento informático que permite asegurar que un usuario es auténtico o quien dice ser.

Existen varias formas de hacerlo, pero la mas fácil y segura es utilizar el protocolo **SSH** y sus llaves.

Las llaves, son un medio por el cual podemos identificar nuestro equipo con un servidor o página específica, incluso sin tener que ingresar una contraseña. Esto funciona mediante dos llaves, una privada y otra pública. La privada vive en nuestro equipo y la pública es la que se ingresa en el lado remoto.

Como posteriormente vamos a subir nuestros cambios a los servidores de GitHub, la idea es identificar nuestro equipo en su servicio, mediante nuestra llave **SSH**. Esto nos permitirá conectarnos de forma segura y sin mayores interacciones de nuestra parte.

Hay que indicar que esta no es la única forma de conectarnos con GitHub, pero nos proporcionara una conexión directa sin contraseñas, a través de nuestra terminal.

Entonces nuestro primer paso sera revisar si ya tenemos llaves generadas o si necesitamos crearlas.

# Creando y añadiendo clave SSH

---

## Obteniendo llaves SSH

La forma de obtenerlas puede variar en cada sistema operativo y configuración. GitHub a dispuesto de documentación para cada caso, la puedes encontrar [acá](#).

El primer paso consisten en verificar en nuestro sistema si ya las tenemos generadas.

1. En el caso de tener Linux o MAC abre la terminal, en el caso de Windows abre git bash.

Una vez al interior de la terminal ingresa el siguiente comando:

```
ls -al ~/.ssh
```

Esta opción nos listara en la terminal el contenido de la carpeta .ssh donde se encuentran las llaves. Regularmente por defecto el nombre de las llaves públicas puede ser como alguna de las siguientes:

- `d_dsa.pub`
- `id_ecdsa.pub`
- `id_ed25519.pub``
- `id_rsa.pub`

Si tienes alguno de estos archivos, puedes continuar el paso **Ingresando las llaves ssh al ssh-agent**.

También puede ocurrir que la consola nos indique un error de que no tenemos las claves creadas.

```
~/.ssh doesn't exist
```

Si este es el caso, o deseas generar un nuevo par de llaves exclusivas para GitHub, podemos hacerlo en el paso **Generando una llave SSH**.

## Generando una llave SSH

Si al listar las llaves SSH no encontramos ninguna u obtuvimos un mensaje de error, o tal vez deseas un par de llaves exclusivas para GitHub, debemos generarlas. Abre tu terminal y sigue los siguientes pasos:

1. Para generar el par de llaves utiliza el siguiente comando:

```
ssh-keygen -t rsa -b 4096 -C "Tu_email@example.com"
```

Debes remplazar tu correo electrónico entre las comillas.

Nos aparecerá el mensaje de que se están generando las llaves.

```
Generating public/private rsa key pair.
```

2. A continuación nos pedirá que decidamos la carpeta donde dejar las llaves.

```
Enter a file in which to save the key (/home/you/.ssh/id_rsa): [Press enter]
```

Presionaremos **enter** para mantener la sugerencia.

3. Luego nos pedirá que ingresemos una frase de contraseña, en nuestro caso solo presionaremos **enter**.

```
Enter passphrase (empty for no passphrase): [Type a passphrase]
Enter same passphrase again: [Type passphrase again]
```

Si decides utilizar una frase contraseña deberás seguir los pasos adicionales de esta [documentación](#).

Si seguimos los pasos anteriores se creará el archivo `id_rsa.pub` e `id_rsa`. El `.pub` es donde vive nuestra llave pública, mientras que el archivo `id_rsa` debe quedar dentro de nuestro computador.

El paso siguiente es ingresar la llave privada en nuestro `ssh-agent`.

# Añadiendo y probando SSH

## Ingresando las llaves ssh al ssh-agent

Si ya tenemos generadas las nuevas llaves **SSH** debemos realizar un paso previo antes de ingresarlas en el sitio de GitHub. Este paso es añadir la llave privada a nuestro **ssh-agent**, el cual es simplemente un programa que administra las conexiones con este protocolo en nuestro computador.

- En la terminal, debemos ingresar el siguiente comando

```
eval "$(ssh-agent -s)"
```

obtendremos una respuesta como esta:

```
Agent pid 59566
```

- Ahora debemos añadir el archivo de la llave privada con el siguiente comando:

```
ssh-add ~/.ssh/id_rsa
```

Si cambiaste los nombres de los archivos, o estas ingresando una llave anterior, debes ingresar el nombre correcto.

El paso siguiente es añadir la llave publica en GitHub.

# Ingresando clave pública ssh en GitHub

Primero debemos copiar el contenido de nuestra llave publica para poder ingresarla en GitHub. Existen algunas diferencias entre sistemas operativos, por lo tanto si algún paso no resulta, consulta la [documentación](#).

## MAC

Para este sistema operativo, si nuestra clave se llama `id_rsa.pub` utilizaremos el comando.

```
pbcopy < ~/.ssh/id_rsa.pub
```

Este comando copia todo el contenido del archivo y lo deja en el portapapeles, lo que debemos hacer ahora es pegarlo en la configuración de GitHub.

## Windows

Para este sistema operativo, si nuestra clave se llama `id_rsa.pub` utilizaremos el siguiente comando.

```
clip < ~/.ssh/id_rsa.pub
```

Este comando copia todo el contenido del archivo y lo deja en el portapapeles, lo que debemos hacer ahora es pegarlo en la configuración de GitHub.

## Linux

Para este sistema operativo, si nuestra clave se llama `id_rsa.pub` utilizaremos los siguientes comandos.

```
sudo apt install xclip
```

Se instalara una pequeña aplicación para copiar, por lo tanto tendremos ingresar nuestra clave del computador y esperar que se instale. Una vez instalada debemos utilizar el siguiente comando:

```
xclip -sel clip < ~/.ssh/id_rsa.pub
```

Este comando copia todo el contenido del archivo y lo deja en el portapapeles, lo que debemos hacer ahora es pegarlo en la configuración de GitHub.

**Recordemos que si nuestra clave tiene otro nombre, debemos ingresarla como corresponde**

**Recordemos que si tenemos algún problema copiando el archivo desde la terminal, podemos ir a la ruta del archivo y abrirlo con un editor de texto y copiar el contenido.**

# Ingresando clave pública SSH en GitHub

Partiremos por crear una nueva cuenta en **GitHub**. Ingresa a su [sitio](#) y si ya tienes una cuenta ingresa a ella, de lo contrario continúa con el registro inicial.

Una vez dentro de la cuenta de GitHub, vamos a añadir la clave en nuestra cuenta. Iremos a la pestaña de settings.

- Seleccionaremos clave SSH y GPG
- Veremos un botón a la derecha que nos indica añadir una nueva clave pública

## SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Vamos a asignarle un nombre y luego pegaremos la clave copiada.

Con este paso ya estamos identificados con GitHub.

## Probando la clave SSH

Si ingresamos correctamente la clave en GitHub, podemos utilizar el comando:

```
ssh -T git@github.com
```

lo que devolverá un mensaje:

```
"Hi tucorro! You've successfully authenticated, but GitHub does not provide shell access".
```

Si no ocurre esto, deberías repasar los pasos.

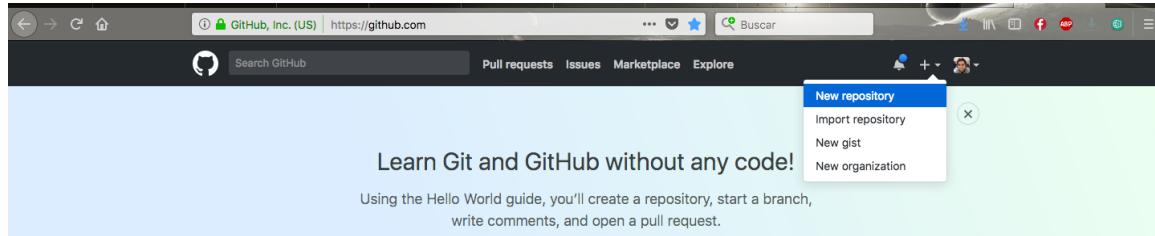
# Utilizando GitHub

Crearemos un repositorio remoto para el proyecto, lo añadiremos por consola y subiremos sus cambios.

## Nuestro primer repositorio remoto

Una vez que ya tenemos configurada nuestra cuenta en GitHub, podremos generar un nuevo repositorio.

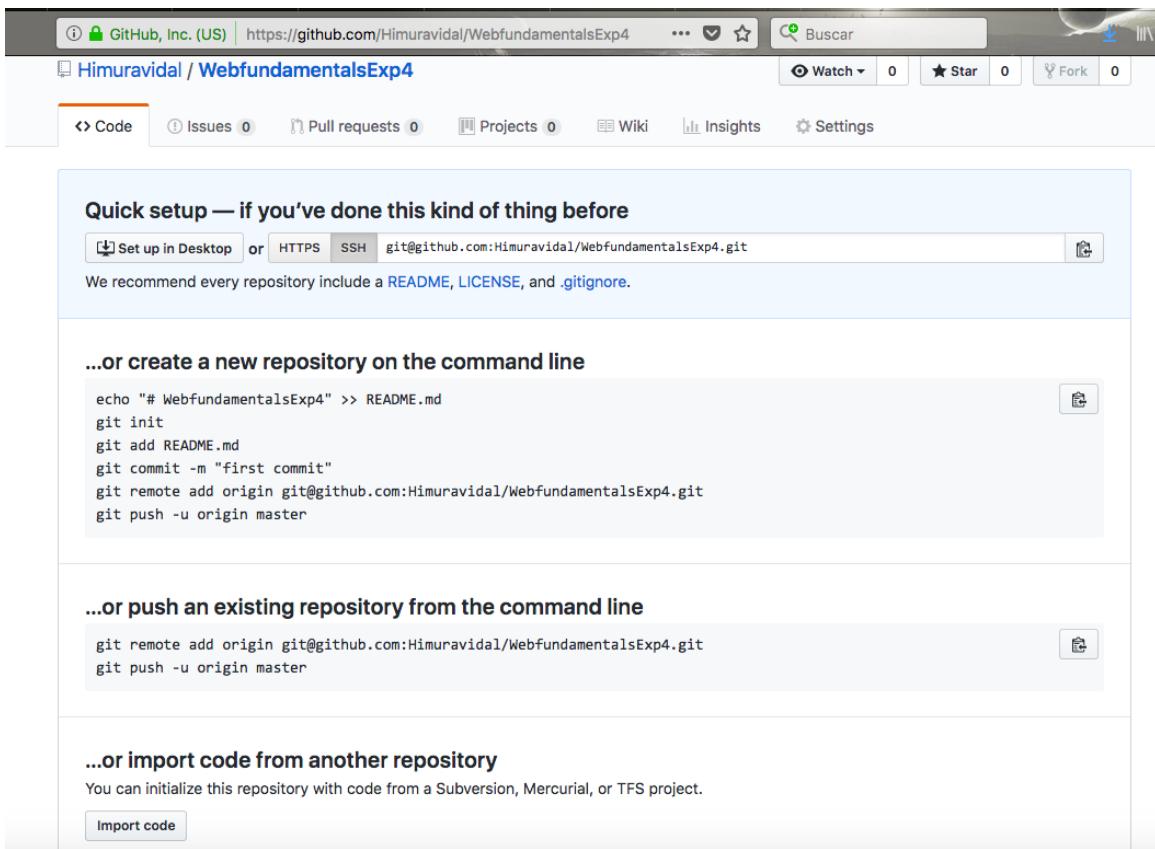
Para ello, dentro de nuestro perfil en GitHub presionaremos el botón **+** y seleccionaremos **new repository**.



Debemos indicarle un nombre y una descripción.

A screenshot of the GitHub 'Create a new repository' form. The URL in the address bar is 'https://github.com/new'. The form fields include: 'Owner' set to 'Himuravidal', 'Repository name' set to 'WebfundamentalsExp4', 'Description (optional)' containing the text 'Repositorio de prueba Web fundamentales exp 4', and 'Visibility' set to 'Public'. There is also a checkbox for 'Initialize this repository with a README' which is unchecked. At the bottom, there are buttons for 'Add .gitignore: None', 'Add a license: None', and a large green 'Create repository' button.

Luego de crearlo nos entregará algunos ejemplos de comando para subir nuestro código.



The screenshot shows a GitHub repository page for 'Himuravidal / WebfundamentalsExp4'. The top navigation bar includes links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. Below the navigation, there's a section titled 'Quick setup — if you've done this kind of thing before' with options for 'Set up in Desktop' or 'HTTPS' (selected) with the URL 'git@github.com:Himuravidal/WebfundamentalsExp4.git'. A note says 'We recommend every repository include a README, LICENSE, and .gitignore.' Below this, there are three sections: '...or create a new repository on the command line' with a code block containing git commands; '...or push an existing repository from the command line' with another code block; and '...or import code from another repository' with a 'Import code' button.

Utilizaremos estos pasos a continuación:

Si ya tenemos un proyecto con git iniciado como es nuestro caso, solo debemos indicarle el nuevo repositorio remoto y subir nuestro cambios.

Abramos la terminal y vamos a la carpeta del proyecto.

```
cd Desktop/meet\&coffee/
```

- Vamos aadir entonces el repositorio remoto recién creado a nuestro proyecto desde la terminal.

```
git remote add origin [dirección del repositorio remoto]
```

Remplazaremos dirección con la Url o git de nuestro repositorio recién creado.

Al presionar enter, no recibiremos ninguna confirmación, pero si queremos ver si esta ingresado el repositorio remoto, podemos usar el comando : `git remote -v` y veremos algo como esto:

```
adacher@adacher-ThinkPad-T440p:~/Desktop/newproject$ git remote add origin git@github.com:Himuravidal/testtest.git
adacher@adacher-ThinkPad-T440p:~/Desktop/newproject$ git remote -v
origin git@github.com:Himuravidal/testtest.git (fetch)
origin git@github.com:Himuravidal/testtest.git (push)
adacher@adacher-ThinkPad-T440p:~/Desktop/newproject$
```

3. Ahora podemos subir el proyecto repositorio remoto:

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

De esta forma todo lo que tenemos en local, será subido a nuestro repositorio recién creado y ya estamos compartiendo nuestro código con el mundo.

```
adacher@adacher-ThinkPad-T440p:~/Desktop/newproyect$ git push origin master
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 248 bytes | 248.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:Himuravidal/testtest.git
 * [new branch]      master -> master
adacher@adacher-ThinkPad-T440p:~/Desktop/newproyect$
```

# Trabajando con git y GitHub comandos básicos (Parte I)

## Subiendo y bajando cambios

Como hemos visto en el capítulo anterior, para subir los cambios al repositorio remoto debemos utilizar el comando `git push origin master`. De esta forma estoy subiendo todos los cambios registrados (comiteados) al repositorio remoto, en la branch (rama) en la que me encuentre, en este caso: `master`.

Ahora, si por el contrario necesito bajar los cambios que están en un remoto que tenemos registrado en nuestro proyecto, podremos utilizar el comando.

```
git pull origin master
```

Lo cual traerá a nuestro computador todos los cambios que se hayan realizado en el remoto, uniendo de forma automática los archivos que encuentre con cambios.

Recuerda que el nombre `origin` corresponderá al remoto registrado en tu proyecto y `master` se refiere a la rama master del remoto.

## Manejo de repositorios remotos

Si necesitamos saber si el proyecto en el que estamos trabajando ya contiene alguna referencia a un repositorio remoto, lo realizaremos con el comando `git remote`, el cual nos mostrará el nombre de los repositorios que tengamos añadidos.

```
adacher@adacher-ThinkPad-T440p:~/Documents/webfundamentals$ git remote  
origin  
adacher@adacher-ThinkPad-T440p:~/Documents/webfundamentals$ □
```

Si necesitamos saber las url de estos servidores, podemos utilizar `git remote -v`.

```
adacher@adacher-ThinkPad-T440p:~/Documents/webfundamentals$ git remote -v  
origin  git@github.com:gsanchezd/webfundamentals.git (fetch)  
origin  git@github.com:gsanchezd/webfundamentals.git (push)  
adacher@adacher-ThinkPad-T440p:~/Documents/webfundamentals$ □
```

## Añadiendo un repositorio remoto

Para añadir un repositorio remoto, simplemente debemos usar el comando

```
git remote add [nombre] [dirección del repositorio]
```

Si este comando se ejecuta bien, no muestra ningún resultado, solo debemos corroborarlo con `git remote` o `git remote -v`.

```
adacher@adacher-ThinkPad-T440p:~/Desktop/newproject$ git remote add origin git@github.com:Himuravidal/testtest.git  
adacher@adacher-ThinkPad-T440p:~/Desktop/newproject$ git remote -v  
origin  git@github.com:Himuravidal/testtest.git (fetch)  
origin  git@github.com:Himuravidal/testtest.git (push)  
adacher@adacher-ThinkPad-T440p:~/Desktop/newproject$ □
```

Generalmente al añadir un repositorio remoto se utiliza el nombre "origin". Sin embargo, podemos tener más de un servidor remoto configurado, solo tendríamos que utilizar nombres distintos.

# Trabajando con git y GitHub comandos básicos (Parte II)

## Obteniendo información de un repositorio remoto

Una vez que ya tengamos añadido un repositorio remoto en nuestro proyecto, podremos obtener información de él con el siguiente comando

**git remote show [nombre]**

cambiando el nombre por el de nuestro repositorio.

```
adacher@adacher-ThinkPad-T440p:~/Desktop/newproject$ git remote show origin
* remote origin
  Fetch URL: git@github.com:Himuravidal/testtest.git
  Push URL: git@github.com:Himuravidal/testtest.git
  HEAD branch: master
  Remote branch:
    master tracked
  Local ref configured for 'git push':
    master pushes to master (up to date)
adacher@adacher-ThinkPad-T440p:~/Desktop/newproject$
```

## Manejo de repositorios remotos

Si necesitamos renombrar un repositorio remoto que hemos añadido, podemos realizarlo de la siguiente forma:

**git remote rename nombreActual NuevoNombre**

```
adacher@adacher-ThinkPad-T440p:~/Desktop/newproject$ git remote rename origin origin_new
adacher@adacher-ThinkPad-T440p:~/Desktop/newproject$ git remote
origin_new
adacher@adacher-ThinkPad-T440p:~/Desktop/newproject$
```

Y por supuesto si necesitamos borrar un repositorio remoto, podemos realizarlo con el siguiente comando git:

**git remote rm NombreRepositorio**

```
adacher@adacher-ThinkPad-T440p:~/Desktop/newproject$ git remote rm origin_new
adacher@adacher-ThinkPad-T440p:~/Desktop/newproject$ git remote -v
adacher@adacher-ThinkPad-T440p:~/Desktop/newproject$
```

# Trabajando con git y GitHub comandos básicos (Parte III)

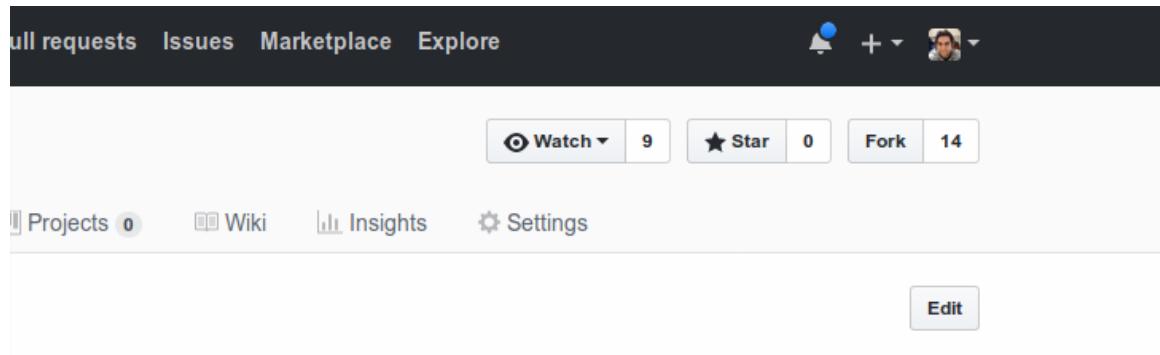
## Repositorios que ya están en GitHub, Fork o Clone

GitHub nos proporciona herramientas para utilizar repositorios que ya están creados. Para ellos existen dos herramientas que nos ayudarán: Fork y Clone.

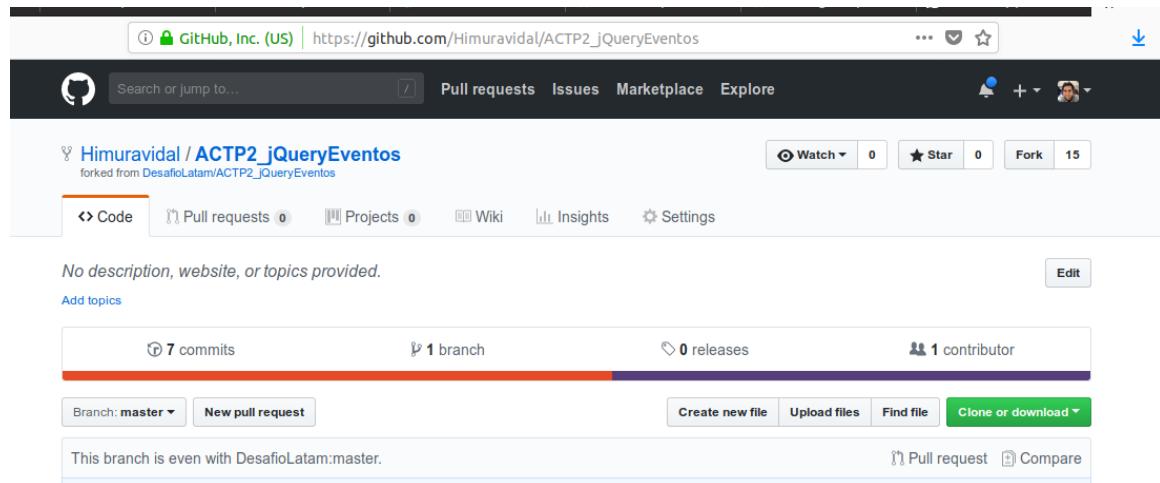
La primera es **Fork**, este comando creara un bifurcación del repositorio, en otras palabras realizará una copia del proyecto completo en nuestra propia cuenta de GitHub.

Esto es útil para realizar cambios sin afectar el proyecto original o partir un nuevo proyecto desde otro como base.

Para forkear un proyecto en GitHub, debemos presionar el botón fork ubicado en la esquina superior derecha de un repositorio.



Esto realizara la copia del repositorio en nuestra cuenta de GitHub.

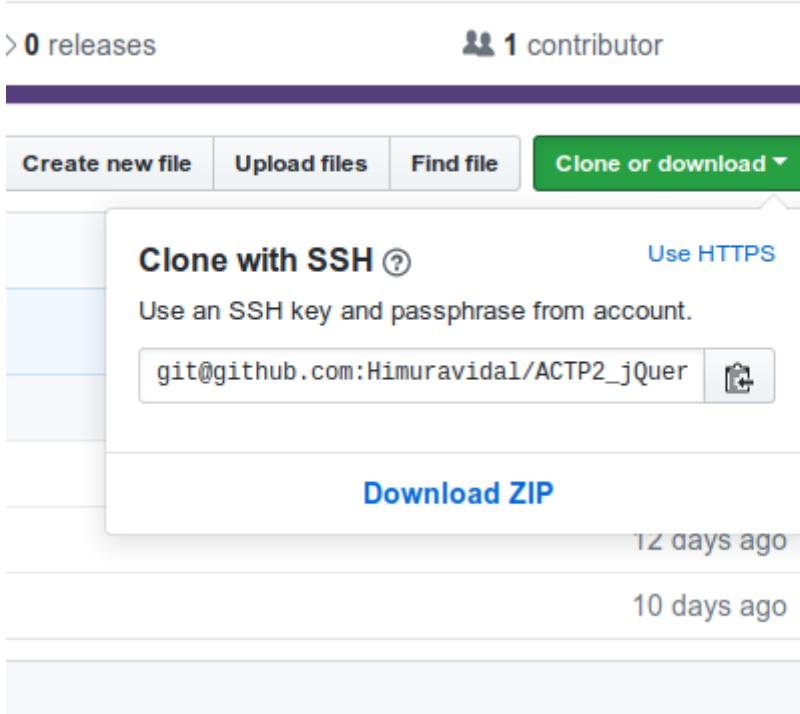


El siguiente paso para comenzar a trabajar seria bajar el contenido de este repositorio hacia nuestro computador local. Esto lo realizaremos **Clonando** el repositorio.

El comando para clonar un repositorio es el siguiente:

**git clone [dirección del repositorio]**

y la dirección del remoto la podemos obtener desde el botón **Clone or download**.



Debemos copiar la dirección que ahí aparece.

Atención con el tipo de enlace que esta seleccionado, si la dirección comienza con **git@** podremos utilizar **SSH** para lograr descargarlo, si es que tenemos las llaves configuradas , si nos saltamos este proceso, podremos descargarlo por **HTTPS** seleccionando **user HTTPS** en esta pestaña.

Cuando ejecutemos el comando `git clone`, veremos los siguiente en la terminal.

```
adacher@adacher-ThinkPad-T440p:~/Documents/webfundamentals$ git clone git@github.com:Himuravidal/ACTP2_jQueryEventos.git
Cloning into 'ACTP2_jQueryEventos'...
remote: Counting objects: 101, done.
remote: Compressing objects: 100% (93/93), done.
remote: Total 101 (delta 9), reused 98 (delta 6), pack-reused 0
Receiving objects: 100% (101/101), 21.98 MiB | 2.28 MiB/s, done.
Resolving deltas: 100% (9/9), done.
adacher@adacher-ThinkPad-T440p:~/Documents/webfundamentals$
```

La acción de clonar va añadir por defecto el remoto desde donde estamos copiando, lo que permitirá luego, subir los cambios con el comando `git push`. Esto funcionara siempre y cuando el repositorio clonado sea nuestro o tengamos permisos de escritura en el.

## git branch

Ahora que ya conocemos git y GitHub y su uso básico, vamos a profundizar más en sus herramientas.

Una de las principales ventajas de git es la utilización de branch (ramas).

Primero definiremos una rama simplemente como un camino donde está nuestro código, mientras que la definición técnica se refiere a un branch como un apuntador a donde van los commits que realizamos.

Cada vez que inicializamos git, de forma automática se genera un branch master, que es dónde se guardarán los nuevos commits.

git nos da la posibilidad de generar más de una rama para poder almacenar nuestros cambios, de modo que podamos probar nuevas funcionalidades y ordenar el trabajo colaborativo de forma más ordenada y sin dañar el trabajo ya realizado.

## Conociendo las ramas del proyecto

Veámoslo paso a paso. Al iniciar git en una carpeta, de forma automática se genera la rama master. Podremos conocer las ramas que tiene nuestro proyecto con el comando:

```
git branch
```

Nos mostrará en consola todos los branch del proyecto. Como solo tenemos una rama, nos aparecerá `master`.

## Cuándo generar una nueva rama

Imaginemos que estamos trabajando en un equipo y necesitamos que un compañero haga cambios en el código. Posiblemente necesite modificar muchas cosas para su nueva funcionalidad.

Si solo utilizamos una rama, estos cambios quedarán sobre lo que ya estaba funcionando, lo que hará difícil saber qué es lo nuevo y solventar algún posible problema.

Para estos casos será mejor segmentar los flujos de trabajo con ramas: una para la aplicación ya funcional en la rama master y una nueva para las otras funcionalidades.

Para crear una nueva branch utilizaremos el siguiente comando:

```
git branch nueva_branch
```

Una vez creado, no cambiará de forma automática a la nueva rama, sino que nos quedaremos en la rama original, así que para cambiarnos debemos hacerlo de la siguiente manera con el comando:

```
git checkout nueva_branch
```

De esta forma nos cambiamos al nuevo branch y podemos utilizar todos los comandos de git ya conocidos (git add, commit, etc.).

Así es como podemos lograr registrar los cambios de la nueva funcionalidad, sin afectar los de la rama master.

Si queremos crear una branch y situarnos enseguida en ella debemos ejecutar el siguiente comando:

```
git checkout -b otra_branch
```

Veamos con el comando

```
git branch
```

las ramas que hemos creado.

# GitHub Pages

---

## ¿Qué es GitHub pages?

Ahora que ya sabemos un poco cómo utilizar git y GitHub, vamos a conocer una utilidad para mostrar nuestros sitios construidos con HTML, CSS y JavaScript.

Se llama GitHub pages y renderiza nuestro código de una rama específica en un dominio y espacio dentro de GitHub.

Esto será de utilidad para poder mostrar nuestro trabajo de forma gratis, sin tener que recurrir a dominios o servidores externos.

## Subiendo una página a GitHub pages

Podemos utilizar un proyecto simple o el mismo que venimos utilizando.

Lo importante es que tengamos un repositorio en GitHub ya creado y, por supuesto, hayamos subido los cambios al repositorio remoto.

Hay dos formas de realizar la subida a GitHub pages. La primera es manual a través de la consola:

Crearemos la rama

```
git branch gh-pages
```

Recuerda que el nombre debe ser "gh-pages".

Ahora pasaremos todos los commits de la rama master a la nueva rama, para luego subir los cambios al repositorio remoto.

```
git checkout gh-pages
```

```
git merge master
```

```
git push origin gh-pages
```

Finalmente, visitemos la página (reemplaza con tu nombre de usuario y nombre de tu repositorio donde corresponde).

[https://TuNombreDeUsuario.GitHub.io/Nombre\\_de\\_tu\\_repositorio/](https://TuNombreDeUsuario.GitHub.io/Nombre_de_tu_repositorio/)

Podremos ver la página almacenada en GitHub pages, que es completamente online. Se la puedes mostrar a quien quieras.

## Dominio personalizado (Parte I)

Cuando pensamos en Internet, regularmente lo primero que viene a nuestra mente son nombres de sitios y servicios conocidos como por ejemplo: Google, Facebook o Amazon. Estas marcas son tan conocidas que si no existieran los buscadores, de igual forma recordaríamos como acceder a cualquiera de ellas.

Ya que lo único que deberíamos añadirle a su nombre seria el `www` y el `.com`.

Esta es la importancia de elegir y tener un buen dominio, facilitando que las personas reconozcan nuestro servicio simplemente con el nombre de nuestro sitio.

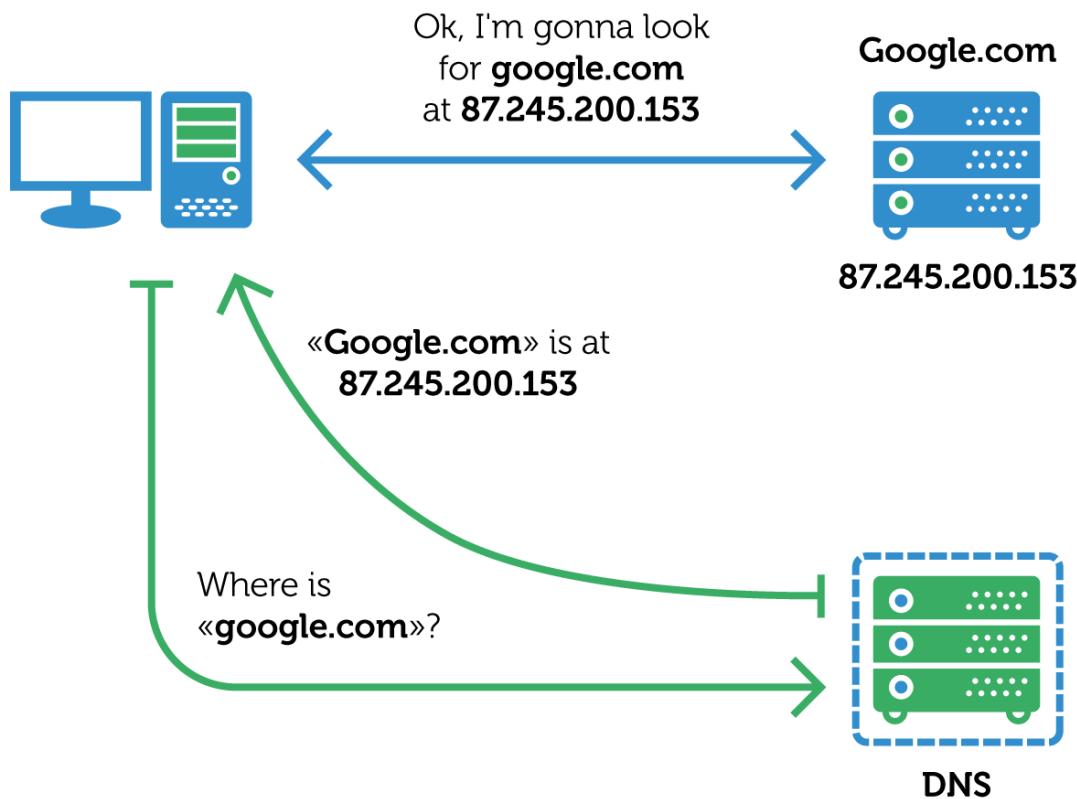
Durante esta experiencia hablaremos del dominio y como obtener uno. Lamentablemente para obtener un dominio `.com` necesitaríamos una tarjeta de crédito, así que para evitar este requisito, obtendremos uno gratuito `.tk`.

Pero primero definiremos algunos conceptos.

### ¿Qué es un dominio?

En internet, un dominio es el nombre único y exclusivo que se le da a un sitio web, con él podemos visitar el sitio desde nuestro navegador. Es la identidad del sitio.

El dominio será la identificación de nuestro sitio. Y para que éste funcione, sigue las reglas y procedimientos de un servicio llamado **DNS**, este significa **Sistema de nombres de dominio**.



© 2016 AO Kaspersky Lab. All Rights Reserved.

EL DNS se encarga de asociar un nombre de dominio con una **IP**.

Una **IP** o protocolo de Internet es una dirección numérica que identifica un dispositivo conectado a una red.

Como sería muy difícil que las personas se aprendieran números para acceder a un sitio, considerando la cantidad de sitios que existen, se determinó usar el sistema de dominios que permite que las páginas tengan nombres más memorables.

En nuestro caso, cuando estábamos subiendo nuestro proyecto a GitHub pages, es GitHub quién se encarga de hacer esto por nosotros. Es decir ellos proporcionan un dominio asociado a la dirección **IP** donde esta el proyecto, lo que permite que compartamos el enlace y cualquier persona puede ver nuestro sitio.

## Añadiendo un dominio personalizado

Un dominio es un recurso limitado, ya que no pueden haber dos sitios web diferentes con el mismo nombre. Es por este motivo que este recurso se adquiere por un periodo de tiempo específico y se tiene que ir renovando.

Para este propósito existen muchos sitios que ofrecen el servicio de asignarte un dominio y servicios asociados a este.

Dentro de los mas conocidos encontraremos:

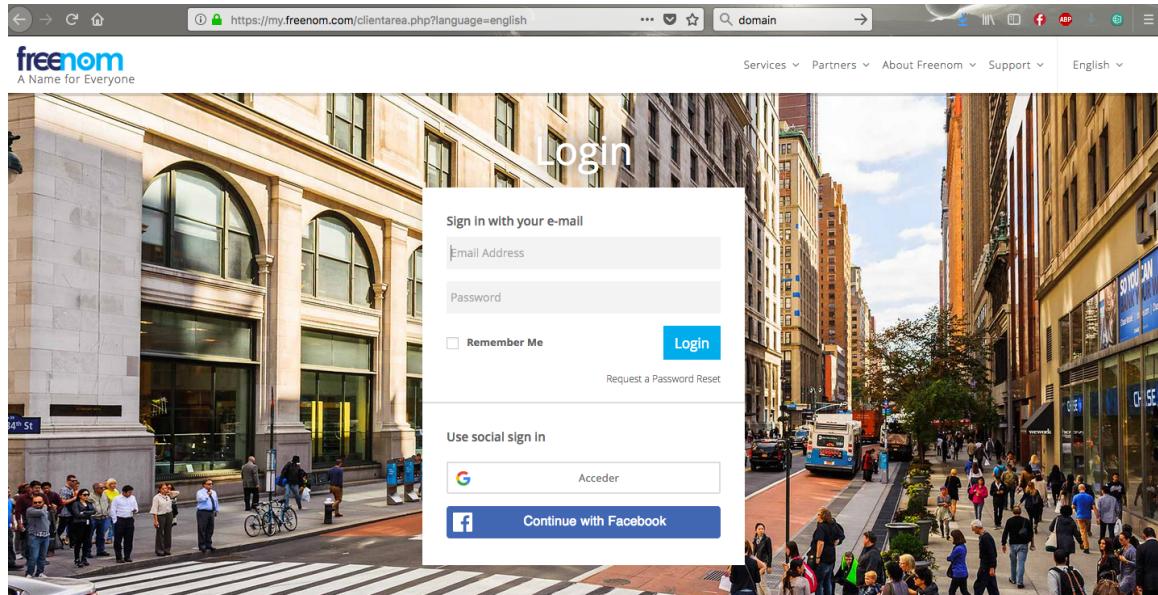
- [Godaddy](#)
- [Bluehost](#)
- [Nic.cl](#), para el [.cl](#) de Chile.

Todos ofrecen dominios por diferentes tarifas. En godaddy podremos encontrar dominios por 1 dolar hasta 7000, todo depende si está utilizado o no.

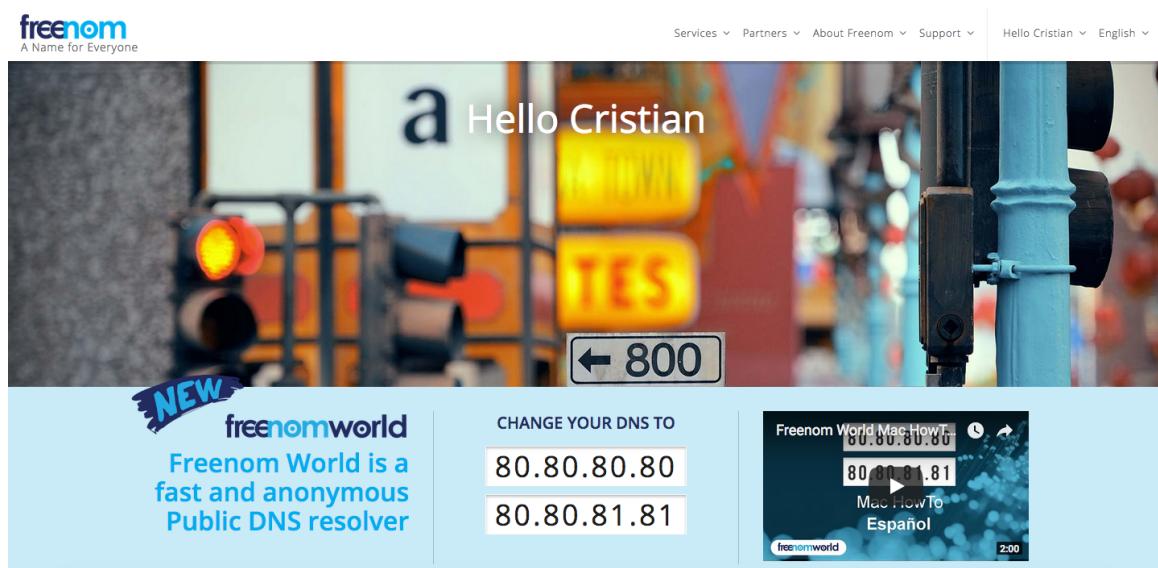
Existen sitios que nos ofrecen un dominio gratuito por un periodo corto de tiempo. Es el caso de [Freenom](#) donde podremos registrar un dominio con extension [.tk](#) de forma gratis por un periodo de tres meses renovables.

## Dominio personalizado (Parte II)

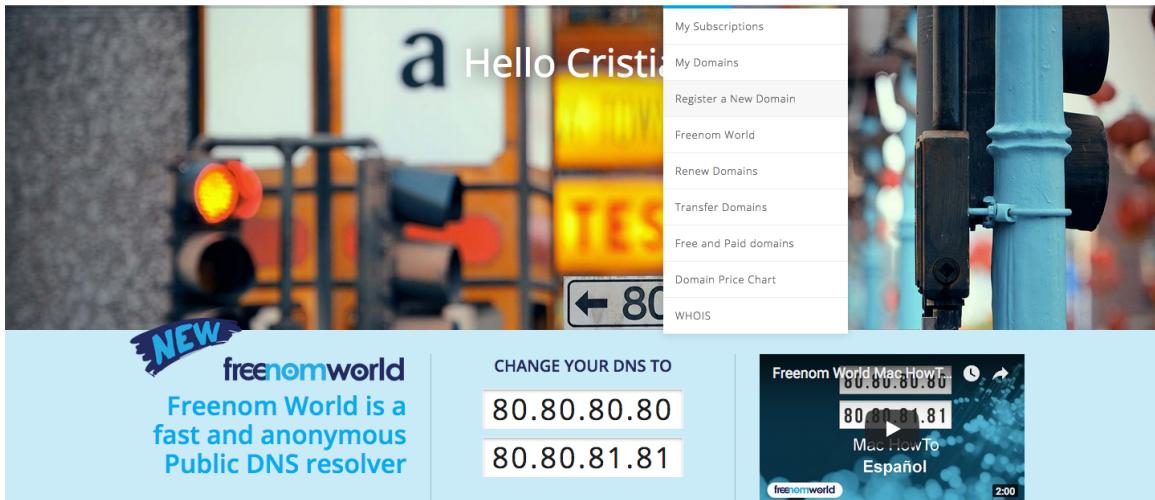
Registrémonos su sitio.



Comenzamos por ingresar nuestros datos y confirmar nuestra cuenta, luego entraremos a la plataforma.



Una vez dentro de la plataforma iremos a **services** y seleccionaremos **new domain**.



En este apartado podremos ingresar el nombre de dominio que queramos utilizar.

meetandcoffee .tk	•FREE	USD 0.00	<a href="#">Get it now!</a>
meetandcoffee .ml	•FREE	USD 0.00	<a href="#">Get it now!</a>
meetandcoffee .ga	•FREE	USD 0.00	<a href="#">Get it now!</a>
meetandcoffee .cf	•FREE	USD 0.00	<a href="#">Get it now!</a>

En este caso ingresaré el nombre Meetandcoffee y presionaré **check availability**. Si estuviera disponible, me indicará:

meetandcoffee

Check Availability

1 domain in cart **Checkout**

Get one of these domains. They are free!

meetandcoffee .tk	•FREE	USD 0.00	Selected
meetandcoffee .ml	•FREE	USD 0.00	<a href="#">Get it now!</a>
meetandcoffee .ga	•FREE	USD 0.00	<a href="#">Get it now!</a>
meetandcoffee .cf	•FREE	USD 0.00	<a href="#">Get it now!</a>

Domain Period

meetandcoffee.tk 3 Months @ FREE

Continue

Y para quedarnos con ese nombre debemos seleccionarlo y presionar **checkout**. Esto nos dirigirá a la siguiente pantalla.

Find a new FREE domain

Check Availability

Domain Use your new domain Period

meetandcoffee.tk Use DNS 3 Months @ FREE

Continue

Debemos presionar "Use DNS" y luego "Use Freenom DNS service".

Find a new FREE domain

Check Availability

Domain Use your new domain Period

meetandcoffee.tk Use DNS 3 Months @ FREE

Use Freenom DNS Service Use your own DNS

Enter your A record here

Hostname	meetandcoffee.tk	IP address	192.30.252.153
Hostname	www.meetandcoffee.tk	IP address	192.30.252.154

Continue

Ingresaremos las **IP** del servidor de GitHub.

**192.30.252.153**

**192.30.252.154**

Estas direcciones **IP** las obtuve desde la siguiente [documentación](#) y son distintas para cada usuario de GitHub.

Si nuestro sitio estuviera en un servidor propio, bastaría con poner la **IP** de esa máquina para que la redirección funcionara, pero como nuestro sitio esta en el servidor de GitHub, todavía debemos configurar algunas cosas.

Continuaremos añadiendo nuestros datos.

The screenshot shows a web browser window with the URL <https://my.freenom.com/cart.php?a=view>. The page title is "Your Details". There is a checkbox labeled "Lock profile. Updates to your details will affect your profile information." followed by a list of input fields:

First Name	Cristian
Last Name	Vidal
Company Name	
Address 1	José Miguel Claro 1074
Zip Code	
City	Providencia
Country	Chile
State/Region	Región Metropolitana de Santiago
Phone Number	+56
Email Address	cristian.vidal.lopez@gmail.com

At the bottom, there is a checkbox labeled "I have read and agree to the Terms & Conditions" and a blue button labeled "Complete Order".

Guardaremos y si todo salió correctamente veremos una orden de confirmación.

## Order Confirmation

Thank you for your order. You will receive a confirmation email shortly.

Your Order Number is: 3160177622

If you have any questions about your order, please open a support ticket from your client area and quote your order number.

[Click here to go to your Client Area](#)

A continuación configuraremos bien el servicio **DNS** de **freenom** en la zona de configuración de nuestro dominio.

Services ▾ Partners ▾ About Freenom ▾ Support ▾ Hello Cristian ▾

**My Domain:** Domains you have registered

Expiry date	Status
18/07/2018	ACTIVE
25/07/2018	ACTIVE

My Subscriptions

My Domains

Register a New Domain

Freenom World

Renew Domains

Transfer Domains

Free and Paid domains

Domain Price Chart

WHOIS

Get GoSitel NEW!

Get GoSitel NEW!

En nuestro dominio presionaremos, manage domain.

## My Domains

View & manage all the domains you have registered with us from here...

Enter Domain to Find

Filter

2 Records Found, Page 1 of 1

Domain	Registration Date	Expiry date	Status	Type		
adacher.tk	18/04/2018	18/07/2018	ACTIVE	Free	Manage Domain	Get GoSite! NEW!
meetandcoffee.tk	25/04/2018	25/07/2018	ACTIVE	Free	Manage Domain	Get GoSite! NEW!

Results Per Page: 10 ▾



**Managing meetandcoffee.tk**

Information Upgrade Management Tools ▾ Manage Freenom DNS

**Information**

To the right you can find the details of your domain. You can manage your domain using the tabs above.

Domain: meetandcoffee.tk ACTIVE  
Registration Date: 25/04/2018  
Expiry date: 25/07/2018

[+ Back to Domains List](#)



Dentro de esta configuración borraremos el registro de **WWW** de la segunda dirección IP. Presionamos guardar.

## DNS MANAGEMENT for meetandcoffee.tk

[Back to domain details](#)

Name	Type	TTL	Target	
	A	300	192.30.252.153	Delete
WWW	A	300	192.30.252.154	Delete

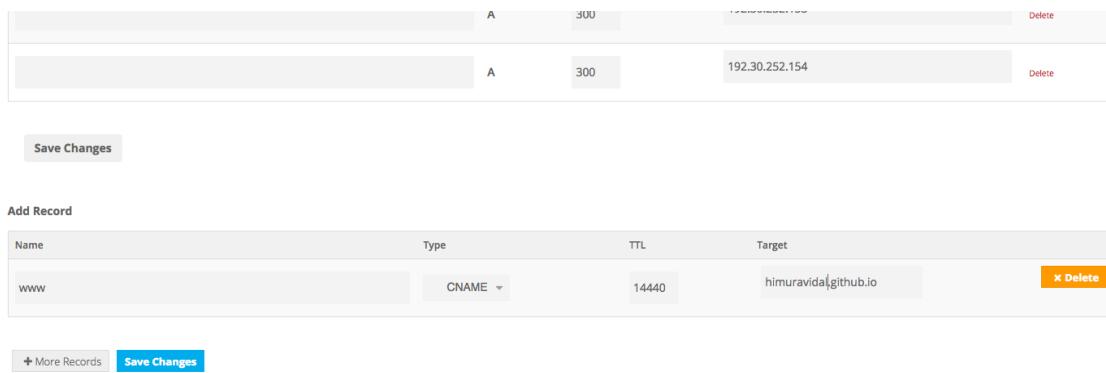
[Save Changes](#)

Add Record

Name	Type	TTL	Target	
	A	14440		X Delete

[+ More Records](#) [Save Changes](#)

Luego añadiremos un nuevo registro, **CNAME** con nombre **WWW** y apuntando a nuestra cuenta de GitHub.



The screenshot shows a DNS management interface with two sections: a list of existing records and a form for adding new ones.

**Existing Records:**

Name	Type	TTL	Target	Action
	A	300	192.30.252.154	Delete

**Add Record:**

Name	Type	TTL	Target	Action
www	CNAME	14440	himuravida.github.io	X Delete

Buttons at the bottom: **+ More Records**, **Save Changes**.

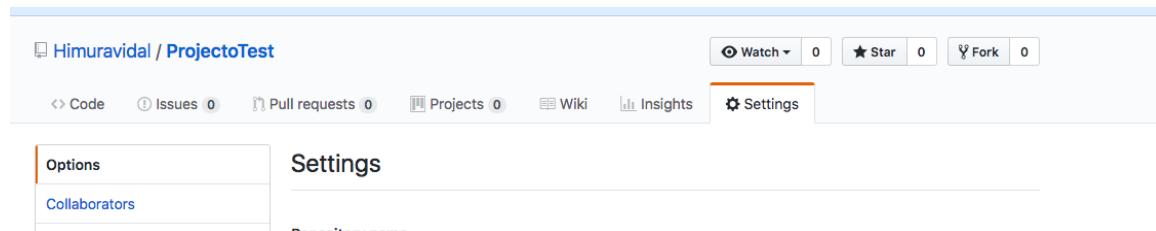
Debes remplazar mi nombre de usuario por el tuyo.

# Añadir el dominio al proyecto en GitHub pages

Configuraremos nuestro dominio en GitHub.

## Configuración dominio personalizado en GitHub

Vamos al repositorio que tenemos configurado con GitHub pages. Luego iremos hacia settings.

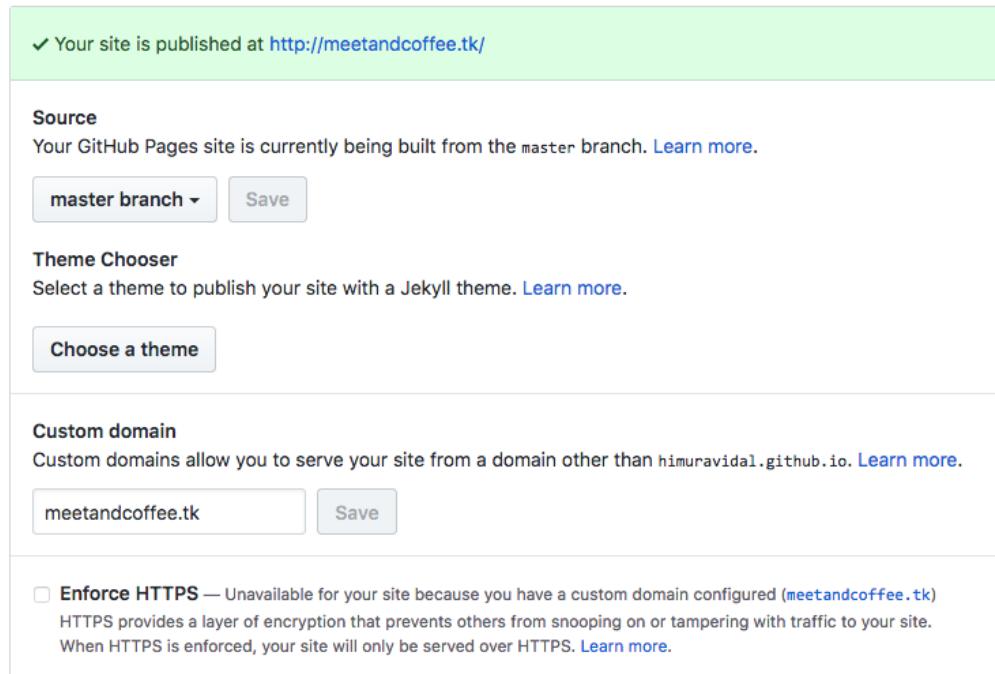


The screenshot shows a GitHub repository page for 'Himuravidal / ProjectoTest'. The 'Settings' tab is highlighted. On the left, there are tabs for 'Options' and 'Collaborators', with 'Options' currently selected. The main area displays the GitHub Pages configuration, which is currently set to publish at <http://meetandcoffee.tk>.

Bajaremos hasta la configuración de GitHub pages.

### GitHub Pages

[GitHub Pages](#) is designed to host your personal, organization, or project pages from a GitHub repository.



The GitHub Pages settings interface is shown. It includes sections for 'Source' (set to 'master branch'), 'Theme Chooser' (button to 'Choose a theme'), 'Custom domain' (set to 'meetandcoffee.tk'), and 'Enforce HTTPS' (checkbox is unchecked). A green banner at the top indicates that the site is published at <http://meetandcoffee.tk>.

Ingresaremos nuestro nuevo dominio y le daremos a guardar.

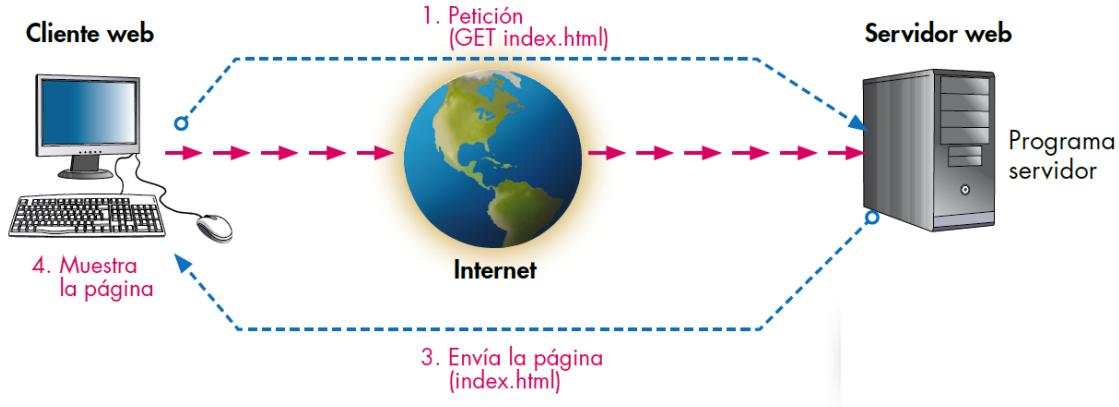
Mientras se guardan los cambios, vamos a mencionar que el proceso de añadir un nuevo dominio a una página a veces no ocurre de inmediato. Para que todos los operadores de internet les llegue la actualización de que la IP específica se encuentra en nuestro sitio, puede tomar varios minutos.

Si tuvimos suerte, podremos visitar nuestro sitio y comprobar que funciona con el dominio personalizado.

## Cliente servidor

Cuando ingresamos un dominio en la barra de direcciones en nuestro navegador y luego presionamos **enter**, ocurren muchos procesos invisibles para nosotros. Casi todo internet trabaja bajo un esquema de **cliente-servidor**.

Siguiendo el mismo ejemplo anterior, cuando escribimos un dominio en la barra y presionamos enter, nuestro navegador enviará los datos de nuestra búsqueda hacia un servidor. Es decir nos conectamos como cliente y le pedimos servicios a esa máquina (servidor).



Created by Paint X

Como estamos utilizando internet, el protocolo que usa nuestro navegador para el requerimiento es **HTTP** y lo definimos al momento de poner en la url **http://**.

Por el momento, necesitamos saber que cada vez que ingresamos una url y presionamos enter, nuestro navegador envía una petición llamada **get** hacia el servidor, que a su vez responde enviando la página hacia nuestro equipo.

En internet existen otros protocolos y servicios funcionando de esta forma.