**Primera Parte: Introducción**

El objetivo de esta guía es dar a conocer unos conocimientos de nivel básico/intermedio sobre el funcionamiento de Git, las diversas posibilidades que ofrece Github y por último aplicar todo lo aprendido de manera práctica siguiendo una metodología.

**¿Qué es Git?**

Git es un software de **control de versiones**. El control de versiones, resumiéndolo mucho, es la gestión de los diversos cambios que se realizan sobre un repositorio (un repositorio es el nombre que recibe el lugar donde se aloja el código de un proyecto de desarrollo en algún lenguaje de programación).

Git usa cadenas de 40 caracteres llamadas **SHA** para identificar los cambios realizados. Ej: 74be3c2b93c8267de64e6dfde8658d31d7d2fe6c

**Objetos de Git.**

Hay 4 tipos de objetos en git (el más importante a entender es el commit).

1. Blob: se usa para almacenar datos de archivos, es generalmente un archivo.
2. Tree: es, básicamente, como un directorio, hace referencia a un conjunto de otros *trees* y/o *blobs* (por ej. archivos y subdirectorios).
3. Commit: apunta a un determinado *tree*, marcando como era en un momento determinado (quien no haya entendido lo que es un *tree*, sustituya la palabra *tree* por *archivo*). Contiene información sobre ese momento determinado, los cambios del autor desde el último commit, el commit anterior (conocido como *parent*), etc. También se puede entender un commit, de una forma más imprecisa y coloquial, como la modificación o el conjunto de modificaciones a uno o varios archivos del repositorio. Otra forma de entenderlo también sería, como una "foto" de uno o varios archivos del repositorio en un momento determinado.
4. Tag: es una forma de marcar un commit como específico de alguna forma. Se usa normalmente para marcar algunos commits como releases específicos o algo destacable en esas lineas.

Para más información sobre los objetos de git, dirigirse a: [Git Community Book](http://book.git-scm.com/1_the_git_object_model.html)

**Qué es**[**Github**](http://github.com/)**?**

Github es una plataforma de desarrollo colaborativo de software para **alojar proyectos** usando el sistema de control de versiones Git. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago. También se pueden obtener repositorios privados (de pago) si se es estudiante.

Github no sólo ofrece alojamiento del código si no muchas más posibilidades asociadas a los repos como son, forks, issues, pull requests, diffs, etc. Se verán todos con detalle más adelante.

**Qué es una metodología de desarrollo de software en el mundo de la programación y de qué forma vamos a aplicarla?**

Citaré a wikipedia aquí: Metodología de desarrollo de software en ingeniería de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.

Básicamente el punto aquí es controlar el proceso de desarrollo para que sea ordenado y de esta forma lo más **productivo**posible.

Esto a primera vista es algo muy simple pero se complica extremadamente dando lugar a diversas metodologías y hay libros muy extensos sobre ello además de mucha controversia sobre cual es la mejor. A nosotros esto nos importa más bien poco pero a donde quiero llegar con esto es a la implementación de una metodología de desarrollo del emulador.

Este método de desarrollo estaría basado en Git y Github ofreciéndonos una enorme versatilidad y posibilitando que todo el equipo de desarrollo esté al día en cuanto a novedades en el desarrollo así como multitud de ventajas que más adelante explicaré. Esta metodología estaría basada en las branches y pull request.

Para concluir esta introducción o presentación de la guía me gustaría mencionar que todo esto que he resumido arriba es en lo que se basan todos los proyectos de desarrollo de software profesionales con unas bases firmes.

**Segunda Parte: Git&Github -> Setup**

En primer lugar ha de crearse una cuenta en [Github](https://github.com/signup/free).

El siguiente paso será instalar git.

Para Mac OS X -> [Git](http://code.google.com/p/git-osx-installer/downloads/list?can=3)

Para Windows -> [Git](http://code.google.com/p/msysgit/downloads/list?can=3)

Antes de pasar a explicar el proceso de enlace del pc con Github quiero dejar algo claro. Git es un programa que se instala en nuestro pc (hasta ahí todo claro supongo :), pero hay que tener en cuenta que no todos los programas tienen una interfaz gráfica (gui). En el caso de git en mac no hay gui (aunque sí hay programas externos que permite manejar git a través de una gui) y en el caso de windows git sí viene incluido con una gui. Sin embargo, tanto en mac como en windows, **no** es recomendable trabajar con git mediante una interfaz gráfica. En vez de eso se trabajará mediante línea de comandos por terminal (en mac) / cmd (en windows).

**Set Up Clave SSH**

Las claves SSH se usan para establecer una conexión segura entre el pc y Github.

Para **crear la clave** se ha de seguir los siguientes pasos en terminal (en mac) o en cmd (en windows).

ssh-keygen -t rsa -C "your\_email@youremail.com" (sustituir "[your\_email@youremail.com](mailto:your_email@youremail.com)" por el email que se puso al crear la cuenta de github).

Esto devolverá lo siguiente:

Generating public/private rsa key pair. Enter file in which to save the key. C/Users/your\_user\_directory/.ssh/id\_rsa) Hacemos click en enter porque queremos que guarde la clave en el directorio por defecto. Guardará la clave ssh en el directorio .ssh dentro de nuestra carpeta de usuario.

Ahora nos dirá: Enter passphrase (empty for no passphrase) No hay por qué indicar nada. Enter.

Y nos dirá que la volvamos a introducir o darle a enter si lo dejamos en blanco antes.

Esto devolverá algo así.

Your identification has been saved in /Users/your\_user\_directory/.ssh/id\_rsa. Your public key has been saved in /Users/your\_user\_directory/.ssh/id\_rsa.pub

Debemos **copiar la clave ssh**. Para ello debemos introducir en la terminal lo siguiente o seguir los pasos indicados en windows:

En Mac: pbcopy < ~/.ssh/id\_rsa.pub

En Windows: dirigirse la gui de git (llamada Git Extensions), ir a Help (Ayuda), Show Key (Mostrar Clave) y entonces presionar Copy to Clipboard (Copiar al portapales) para copiar la clave ssh al portapeles

**Situar la clave SSH en Github.**

Se deberá dirigir a [Github](https://github.com/account/ssh) y hacer click en "Add another pubic key". Control+v (Cmd+v en mac) en la zona de "Key" para situar ahí nuestra clave ssh. Se pone cualquier título en "Title". Por ej: algo para reconocer el pc del que es esa clave (Ej: Macbook Pro de Jesús).

**Comprobación**

Para comprobar que todo se ha hecho correctamente se debe introducir lo siguiente en la línea de comandos (cmd en windows, terminal en mac)

ssh -T git@github.com

Lo cual devolverá algo así

The authenticity of host ´github.com (123.123.123.123) can't be established. RSa key fingerprint is....... Are you sure you want to continue connecting (yes/no)

No te preocupes, esto debe ser así. Escribe *yes*.

Y eso devolverá

Hi 'username'! You've succesfully authenticated, ....

Llegados a este punto nuestro pc ya será capaz de conectarse al servidor de github y poder realizar todo tipo de acciones.

Por último se configurará nuestra info.

**Set Up Config**

Se habrá de poner lo siguiente en la línea de comandos. Sustituyendo "Firstname Lastname" por nuestro nombre y primer apellido y "[your\_email@youremail.com](mailto:your_email@youremail.com)" por el email con el que se registró en github.

git config --global user.name "Firstname Lastname"

git config --global user.email "your\_email@youremail.com"

**Segunda Parte: Git&Github -> Crear y Clonar un Repositorio**

**Crear un Repositorio**

En primer lugar se deberá crear el repositorio en Github, haciendo click en donde dice [New Repository](https://github.com/repositories/new).

Ahí se indicará el nombre del proyecto/repositorio (Project Name) y opcionalmente una descripción del mismo (Description) y un link a la web del proyecto (Homepage Url). También se puede indicar si el repositorio será público, siendo visible para todo el mundo, o bien privado, que solo será accesible para el usuario que lo crea y el o los contribuidores.

Una vez creado el Repo en Github se deberá crear en el PC siguiendo estos pasos:

1. Crear el directorio (carpeta) en donde se alojará el repo. Para ello recomiendo crear una carpeta en nuestra carpeta de usuario (mac) / disco local c (windows) y alojar ahí todos los repos.
2. En línea de comandos (terminal/cmd) navegar hasta el repo. Deberemos llegar hasta el repositorio mediante la línea de comandos, usando el comando *cd*. Ej:

<Macbook-Pro-de-Jesus:~ Jesus$ cd github>

<Macbook-Pro-de-Jesus:github Jesus$ cd Guia-sobre-Git-Github-y-Metodologia-de-Desarrollo-de-Software-usando-Git-y-Github>

<Macbook-Pro-de-Jesus:Guia-sobre-Git-Github-y-Metodologia-de-Desarrollo-de-Software-usando-Git-y-Github Jesus$>

1. 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**.

<git init>

1. Añadir la dirección remota del repositorio de github. Se deberá indicar dónde estará alojado el repo.

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

Ej: [git@github.com](mailto:git@github.com):Hispano/test.git

**Clonar un Repositorio**

Esto es muy simple. Una vez se llegue al directorio donde se va a clonar el repo vía línea de comandos, se debe indicar lo siguiente.

<git clone linkhttp>

Ej: git clone <https://github.com/Hispano/test>

**Tercera Parte: Git-> Realizar cambios, revertirlos y subirlos al repositorio remoto**

Esta parte comprende el workflow que se realiza con normalidad cuando se está trabajando en un repositorio manejando Git. La dividiré en las siguientes partes:

1. Añadir uno o varios archivos a un commit.
2. Realizar un commit con su correspondiente mensaje.
3. Subir los cambios al repositorio remoto alojado en los servidores de GitHub.
4. Revertir los cambios realizados por dicho commit.

Para apreciar los cambios que se producen en el repositorio mientras se realizan estas acciones se utilizará el comando <git status> Si no hemos realizado ningún cambio en el repositorio, al usar ese comando nos debería mostrar algo así:

<➜Guia-sobre-Git-Github-y-Metodologia-de-Desarrollo-de-Software-usando-Git-y-Github git:(master) git status> <On branch master> <nothing to commit (working directory clean)>

**Añadir archivos a un commit**

Esto se realiza con el comando <git add> Seguido del nombre de la ruta y nombre del archivo que se quiere añadir al commit, de forma que quedaría así: <git add directorio/directorio/archivo.extensión>

Sólo podremos usarlo para archivos modificados o añadidos al repositorio.

Ej: git add src/server/game/CMakeLists.txt

Lo cual añadirá el archivo CMakeLists.txt a nuestro commit.