# **UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

## **Linux Sistemas Embebidos**

##### 

### GitHub

#### 

P R E S E N T A N

González Pérez Cesar Israel

Ochoa Ortiz Abel

Reyes González David Antonio

García Silva Adán Alberto

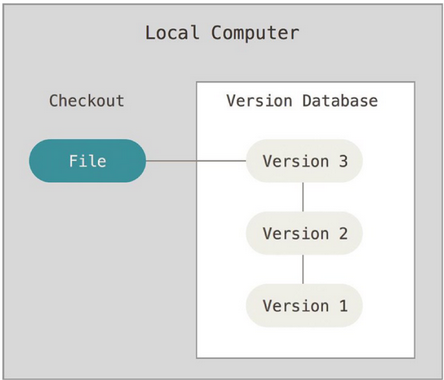
**PROFESOR: Iván Chavero**

**MÉXICO, D.F. ABRIL, 2016**

## **Introducción**

## ¿Qué es GitHub?

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



GitHub aloja tu repositorio de código y te brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto. Algo muy importante, con esto podemos contribuir a mejorar el desarrollo de los demás.

## **Control de versiones**

El control de versiones hoy en día es una de las actividades primordiales en la administración de proyectos para el desarrollo de software de forma general. De este modo surge la necesidad de mantener y llevar control del código que vamos programando, manteniendo los cambios hechos al mismo. Esto agiliza el desarrollo en equipo y los cambios impactados a un desarrollo.

Es necesario gestionar los cambios sin importar el número de involucrados en un desarrollo logrando tener un control en las versiones y en el avance del mismo.

En cuanto a equipos de trabajo se refiere, todavía se hace más necesario disponer de un control de versiones.

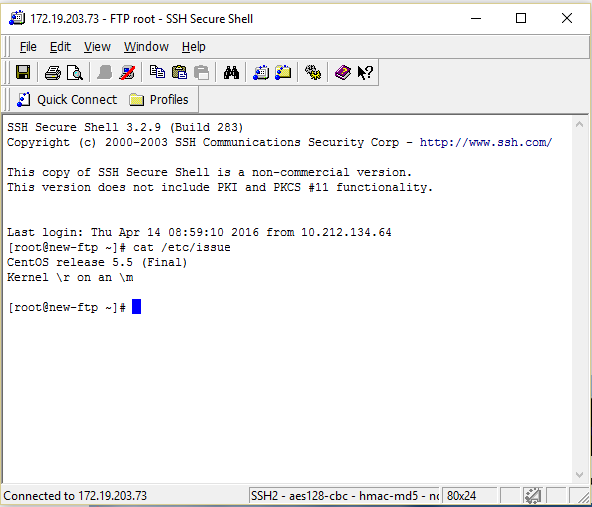
**Desarrollo**

A continuación, se explicará a detalle lo realizado con esta herramienta demostrando las grandes capacidades de la misma, así como los problemas comunes y como resolverlos.

Partimos de colocar un repositorio maestro en el que todos los involucrados en el equipo realicen sus updates a las partes correspondientes de sus actividades. Para este repositorio decidimos realizarlo en un servidor en producción con una IP publica para el acceso de cada uno de los integrantes.

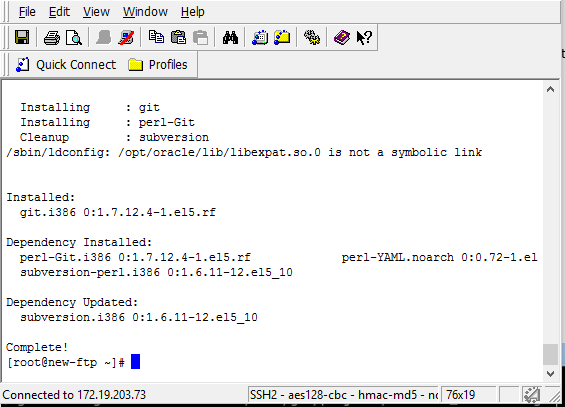
Partiendo de la instalación del servidor, se realizó lo siguiente:

1. Se realizará en una instancia Linux, en nuestro caso lo implementamos sobre CentOS.

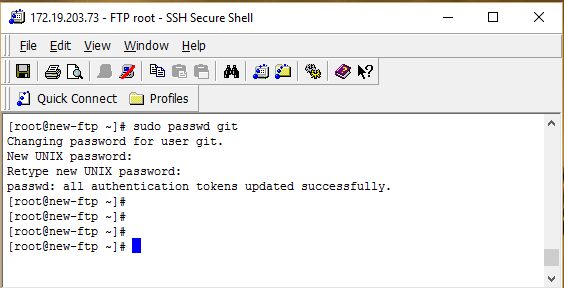


1. Procedemos a instalar una instancia de Git. Para hacerlo en CentOS instalaremos el paquete de Git de este modo, para lo que ejecutamos lo siguiente:

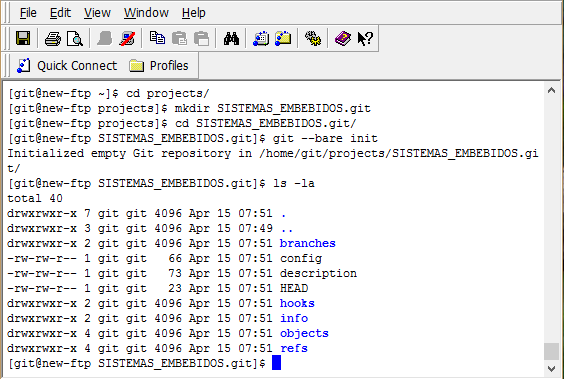
yum install git



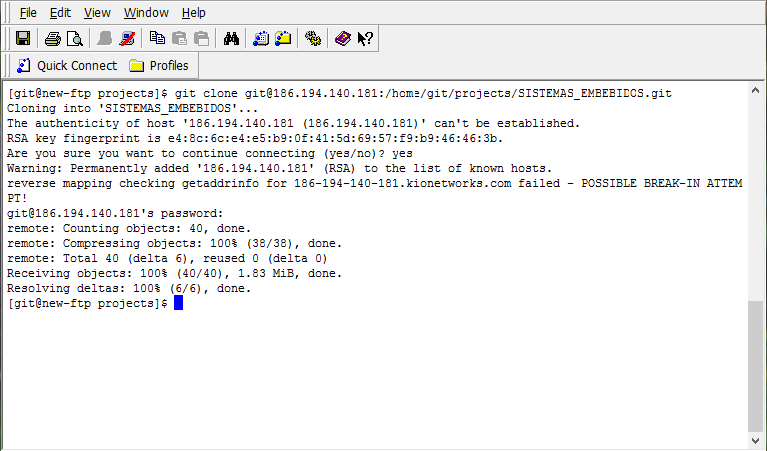
1. Para la gestión con Git crearemos un usuario y le asignamos en la siguiente forma de la siguiente forma:



1. Ingresamos con el usuario creado y posteriormente creamos una carpeta donde se localizarán los proyectos que manejemos en Git, nombraremos nuestro proyecto y accederemos a esa carpeta, una vez dentro de este directorio vamos a crear el proyecto de Git con el comando:

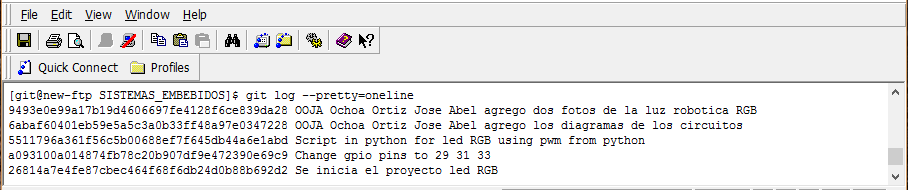


1. Para probar el repositorio, desde la misma máquina o desde otra máquina cliente, podemos hacer un clone del proyecto con el comando:



Podemos analizar los cambios realizados en el repositorio mediante el comando:

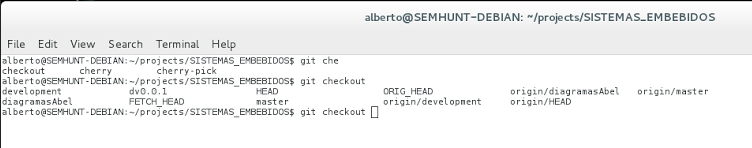
git log --pretty=oneline



**Aplicación *RoboticLedLight***

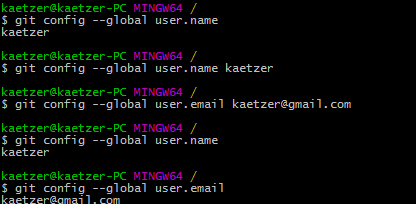
Comenzamos a desarrollar una aplicación en la que se involucró el desarrollo de software, la implementación de hardware, así como el manejo de control de versiones mediante Git en un repositorio propio que se encuentra en línea.

Se realizaron las siguientes ramificaciones en las que cada uno deposito lo que se aportó en el desarrollo de este embebido.

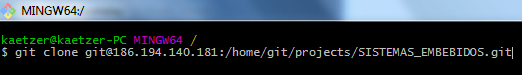


A continuación, se detalla el desarrollo de la aplicación realizando lo siguiente:

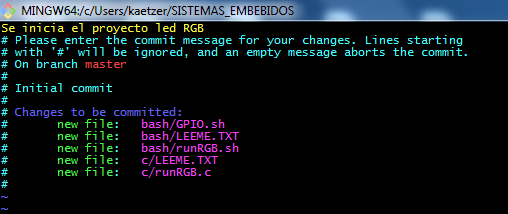
Se configuran las variables globales del repositorio para estar identificados en el repositorio.



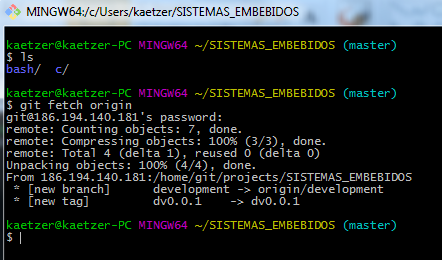
Se clona el repositorio con el siguiente comando



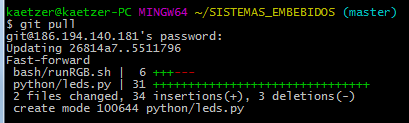
Se trabajaron los primeros prototipos para el proyecto módulo de espacio de usuario



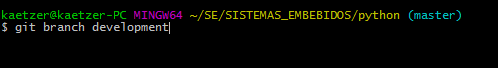
Se buscan los cambios que existen en el repositorio remoto



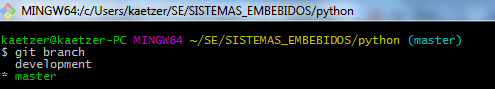
Damos pull para actualizar nuestro repositorio local



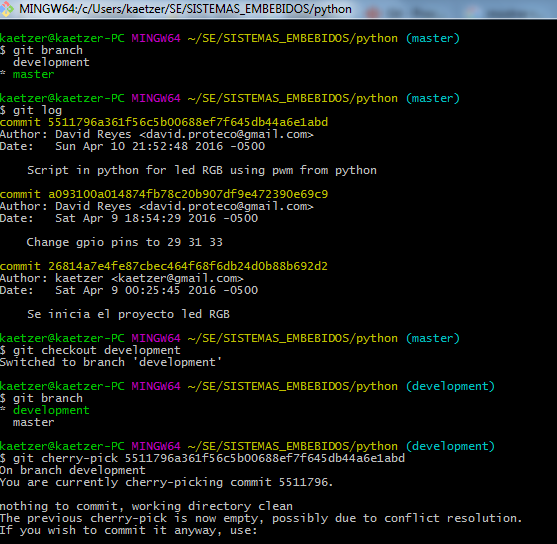
Creamos una rama de desarrollo



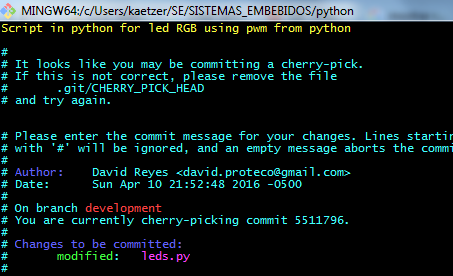
Cambiamos de rama a desarrollo



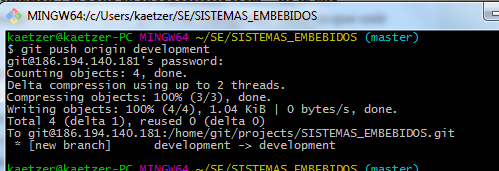
Buscamos en log para hacer un cherry-pick a la rama de desarrollo para continuar los cambios de la rama master



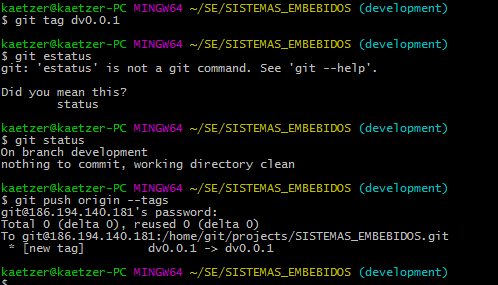
Hacemos el commit de la rama de desarrollo



Mandamos nuestros cambios al repositorio remoto

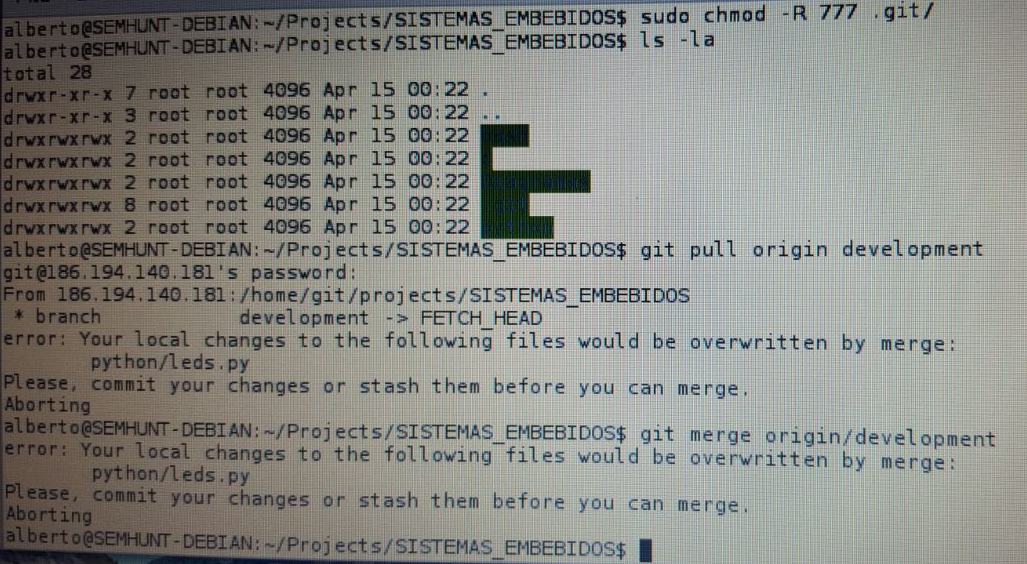


Creamos el primer tag de la primera versión de nuestro código de desarrollo

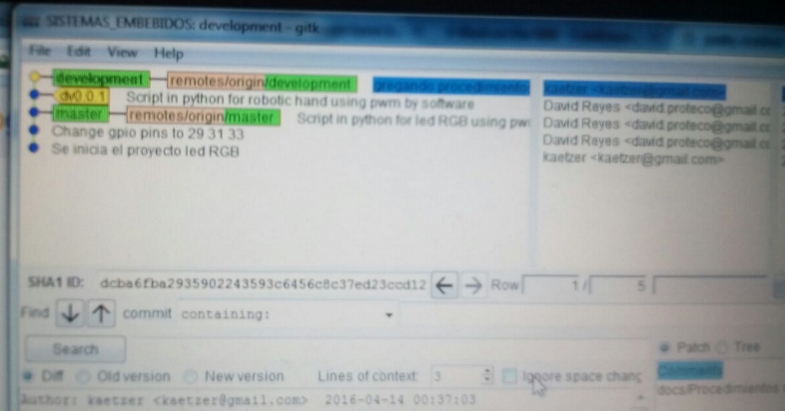


**Problemas**

Durante el desarrollo de esta tarea, nos encontramos con una situación en la que al tratar de hacer un pull, este marcaba el siguiente error, indicando que se tenía que hacer un commit antes de hacer el pull, esto fue ocasionado debido a que en el repositorio (servidor) se estaban realizando cambios.



Para solucionar este incidente se debió acceder al repositorio y concluir los cambios para que en los equipos (usuarios) se pudiera continuar con el desarrollo.



**Conclusión**

Hemos visto la importancia de mantener un versionado ya que podemos tener un control total y absoluto de los avances, cambios, mejoras, updates, etc. Git tiene una serie de otros beneficios que listaremos a continuación:

* Los repositorios y ramas Git son mucho más pequeños.
* Las ramas Git llevan toda su historia completa.
* Los formatos de archivo de los repositorios son simples, la reparación es fácil y la corrupción es rara.
* El clonado de repositorios actúa como repositorio de copias de seguridad completas.
* Proporciona una mejor auditoría de eventos de ramificación (branch) y fusión (merge).