



# Introducción Git y GitHub

Hecho por: Daniel y Rodrigo



```
        self.autoDetermineLanguageFromString(inputString)
    if lang is None:
        raise Exception("Input language could not be determined")
    return None
    parsedInput = self.parseInputToLanguageModel(inputString, inputLanguage, context)
    if not parsedInput or not self.model:
        return None
    context.append(parsedInput) # Add new conversation entry to context
    return (self.model.generateLLMOutput(parsedInput), context)

def parseInputToLanguageModel(inputString, inputLanguage, context):
    if self.model is None or self.model.language != inputLanguage:
        # LLM is not initialised or has wrong language, load LLM
        self.model = self.loadAILanguageModelFromDatabase(inputLanguage)
        if self.model is None or not self.runModelSelfDiagnosticTests():
            raise Exception("AI language model load failed")
        return None
    self.model.setLLMContext(context) # Put past conversation context into it
    llmInputParser = self.model.getInputParser()
    return llmInputParser.parseInput(inputString)
```

# Introducción

¿Qué vamos a aprender el día de hoy?

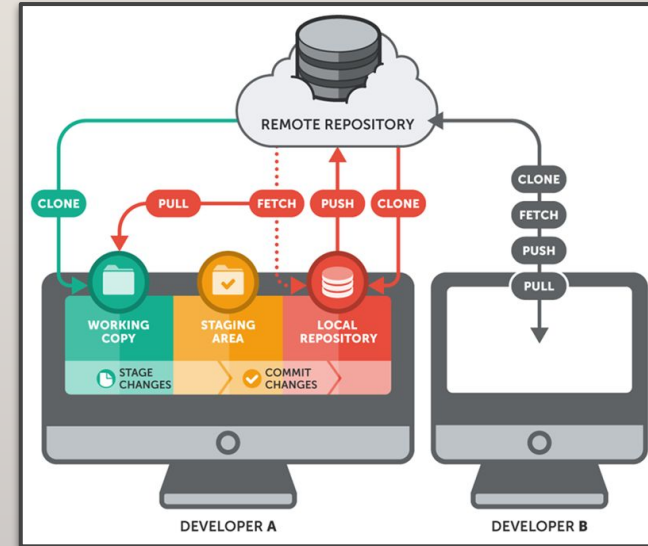
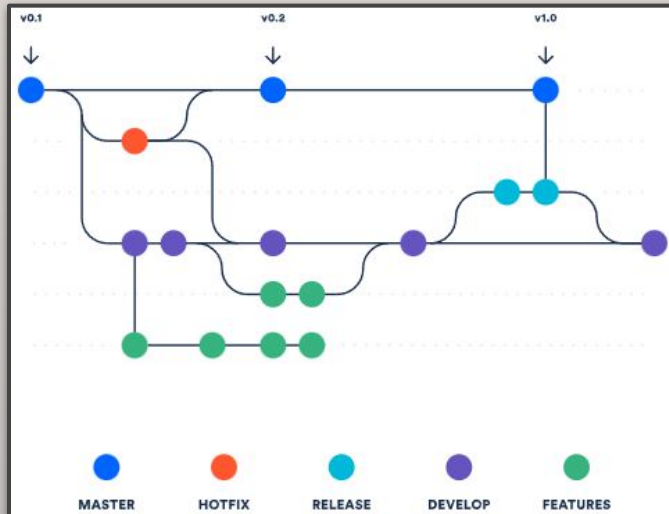
- Sistema de control de versiones - Funcionalidades y Características [1]
- Git y GitHub - Diferencias [2]
- Control de versiones - Ventajas y Beneficios [3]
  - Flujo de trabajo - Git y GitHub [4]

Webgrafía/Bibliografía



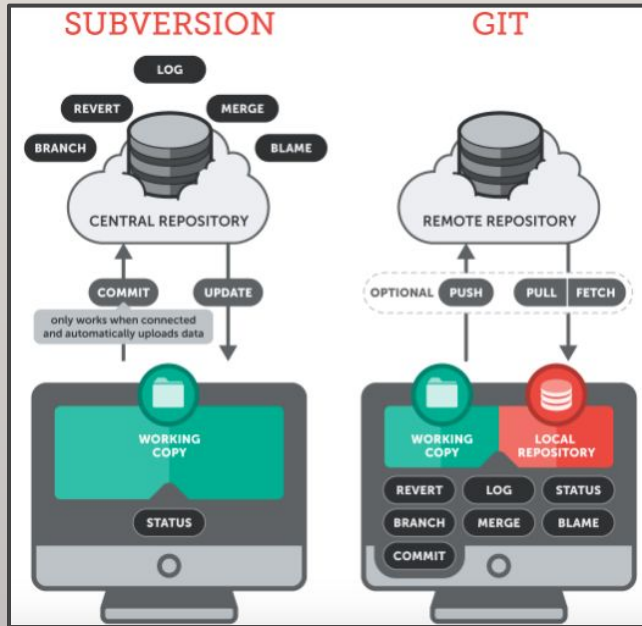
# Sistema de control de versiones - Funcionalidades y Características

Un sistema de control de versiones es una herramienta que tiene como funciones principales gestionar y monitorizar un archivo o conjunto de estos, proyectos.



También se caracteriza permite crear, añadir, modificar y eliminar archivos y directorios.

# Sistema de control de versiones - Funcionalidades y Características



Hay que destacar que una de las características por la que este tipo de sistemas son considerados imprescindibles por los programadores, es la capacidad de trabajo colaborativo.

# Sistema de control de versiones - Funcionalidades y Características

Esquema/Resumen visual

Trabajo individual ➡ Commit ➡ Historial de versiones ➡ Compartir con el equipo ➡ Merge de cambios

Commit  
=  
compromiso/confirmación  
sobre un conjunto de  
cambios provisionales de  
forma permanente, es decir,  
conversión de cambios  
provisionales a permanentes



Merge  
=  
Unión de dos o más  
ramas en una sola

## Git y GitHub - Diferencias



GIT



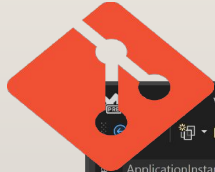
GITHUB

GitHub es una plataforma de desarrollo colaborativo donde se albergan diferentes proyectos utilizando el sistema de control de versiones Git.

Git es un sistema de control de versiones de código fuente, que, gracias a su diseño eficiente y con capacidad en la contabilidad y computabilidad en el mantenimiento de versiones de aplicaciones hacen posible el trabajo colaborativo y simultáneo de varios programadores en el mismo proyecto.



# Git y GitHub - Diferencias



The screenshot shows the Visual Studio IDE with the Git extension. The interface is divided into several panes:

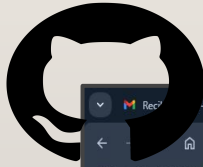
- Left Pane (1):** Shows the 'Git Repositories' view. The 'featurebranch3' branch is selected, and the 'Local History' is visible.
- Top Pane (2):** Shows the 'Git Changes' view for the 'featurebranch3' branch. It displays a list of commits, including 'A new commit' and 'closes #2 similar to #10'.
- Bottom Pane (3):** Shows the 'Commit 5b95b5ac' view. It displays the commit message 'closes #2 similar to #10' and the code changes for the 'Resize.cs' file. The changes include adding a new 'ResizeMode' enum value and updating the 'Height' property.

The code changes in the bottom pane are as follows:

```
36 [DefaultValue(0), Description('
37 public int Height { get; set; }
38
39 - [DefaultValue(ResizeMode.Resize
40 + [DefaultValue(ResizeModeNew.Res
41 public ResizeMode Mode { get; s
42 + public ResizeModeNew Mode { get
43
44 public Resize()
45 {
```



# Git y GitHub - Diferencias



The screenshot shows the GitHub dashboard for a user named 'Rodrigo'. The interface is in dark mode and includes a top navigation bar with the GitHub logo and a search bar. The main content area is divided into several sections:

- Top repositories:** A list of repositories owned by 'Rodrigo-cyber17', including 'Men-horizontal-con-subniveles-mediante-float', '1.CFG5-desarrollo-de-aplicaciones-multiplataforma---Rodrigo', 'Festivales-de-M-sica', 'Marcadores-y-contenido-incrustado', 'Tablero-de-Ajedrez', 'Formulario-de-Registro', and 'Marcadores-favoritos'.
- Hogar (Home):** A central area with several cards and links:
  - Introducción a GitHub:** A card explaining how to get started with GitHub.
  - Codificar con Copilot:** A card about using GitHub Copilot for code suggestions.
  - Páginas de GitHub:** A card about creating a site or blog from GitHub repositories.
  - Hola, acciones de GitHub:** A card about creating a GitHub Action.
  - Ver más proyectos tutoriales:** A link to more tutorial projects.
  - Empieza a escribir código:** A link to start writing code.
  - Crear un nuevo repositorio para Rodrigo-cyber17:** A section for creating a new repository, with options for 'Público' (Public) and 'Privado' (Private).
  - Preséntate con un README de perfil:** A section for creating a profile README file.
- GitHub Copilot:** A promotional banner for GitHub Copilot, highlighting its extensions and AI-powered features.
- Latest changes:** A section showing recent updates and improvements, such as 'Improved filtering for secret scanning alerts' and 'GitHub Models introduces AI-powered system prompt enhancement [GA]'.
- Explorar repositorios:** A section for exploring other repositories, featuring 'apache/camel' as a highlighted project.



## Git y GitHub - Diferencias

### Resumen

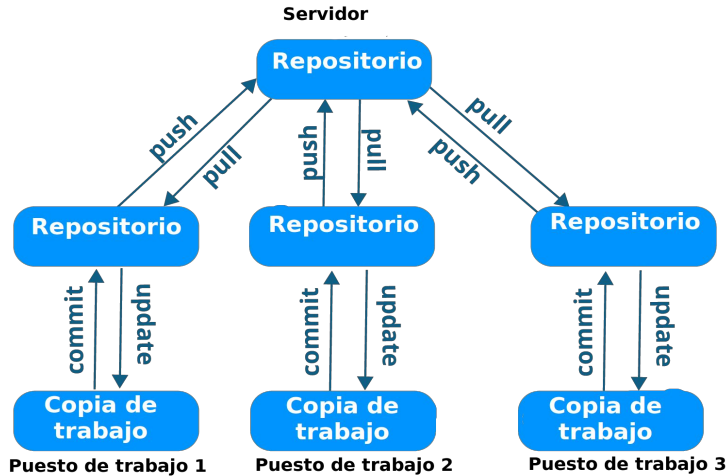
Mientras que Git es la herramienta local encargada de la gestión de versiones y cambios en el proyecto, GitHub es la plataforma en la nube con la que facilitar al programador/es el trabajo en equipo, la colaboración entre equipos y/o miembros de éste y, el alojamiento de repositorios

### Esquema visual

Trabajo Local con Git ➡ Sincronización con GitHub (Remoto) ➡ Colaboración con otros desarrolladores

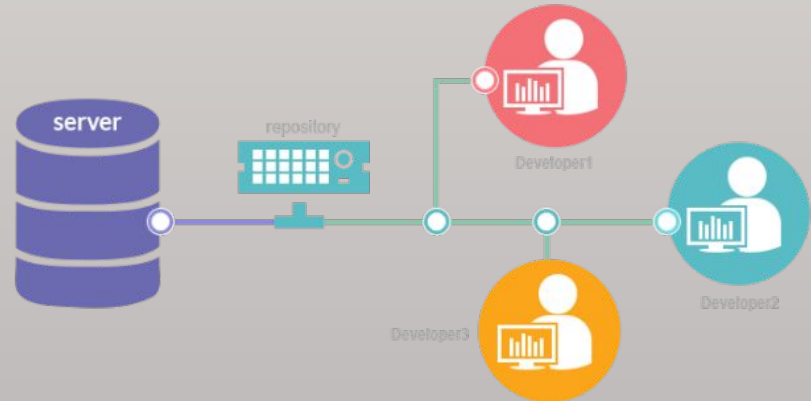
# Control de versiones - Ventajas y Beneficios

Un sistema de control de versiones aporta bastantes ventajas al programador a la hora de trabajar. Para ello las dividiremos en 3 puntos clave a mencionar:



- Trabajo en equipo -

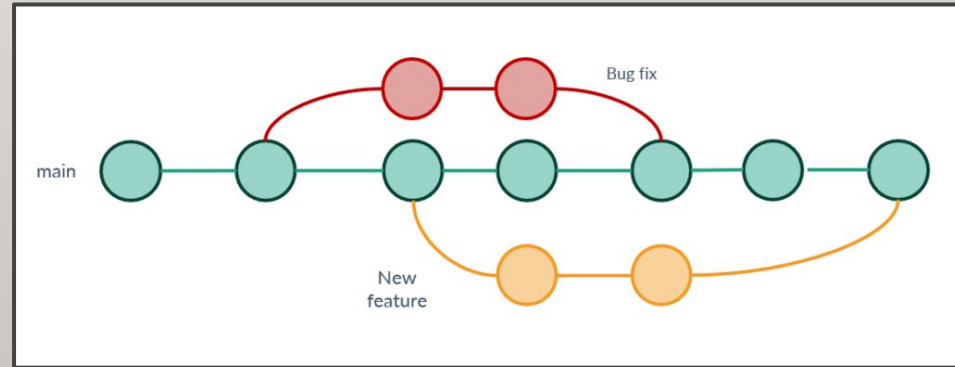
trabajo colaborativo-simultáneo dentro de la gestión de proyectos.



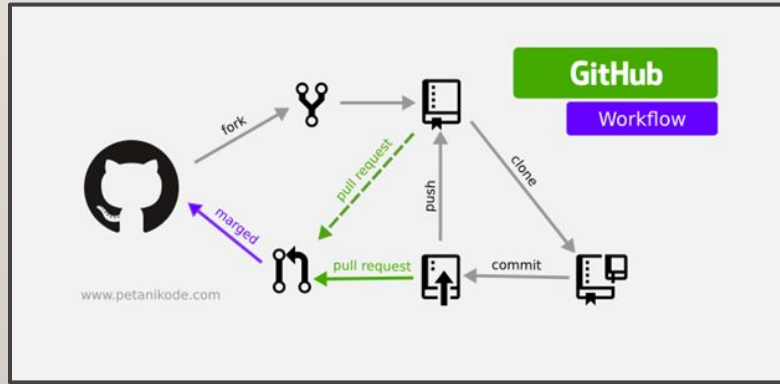
# Control de versiones - Ventajas y Beneficios

- Historial de cambios -

Hace un guardado sobre el registro completo de las modificaciones, facilitando así el rastreo y la corrección de errores.

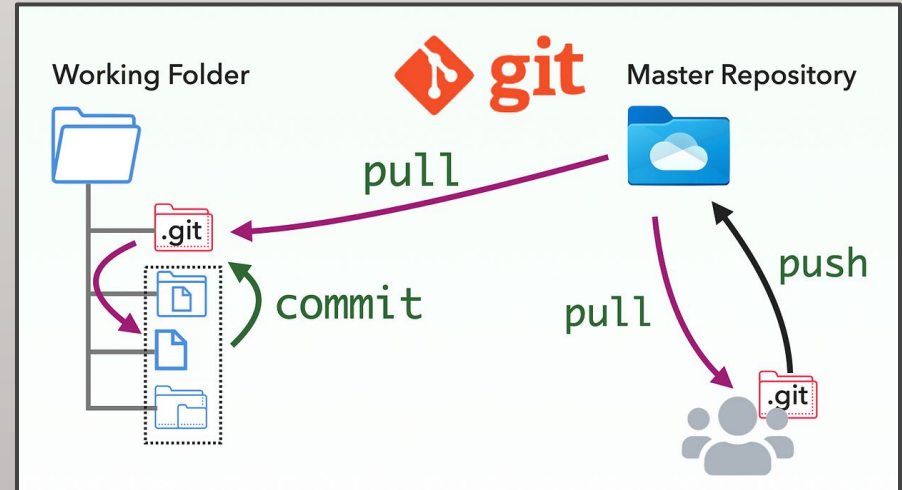
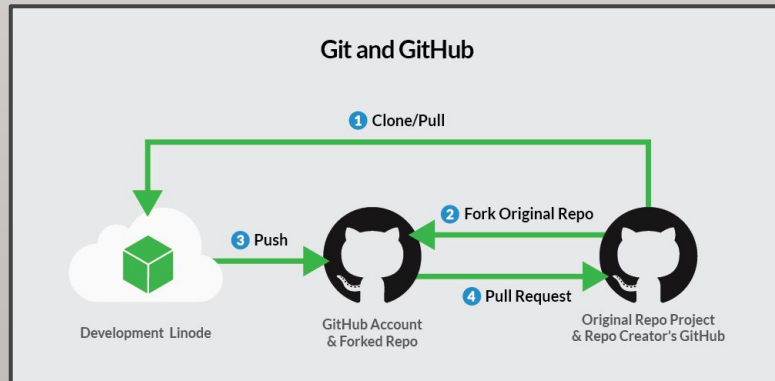


# Control de versiones - Ventajas y Beneficios



- Uso de ramas -

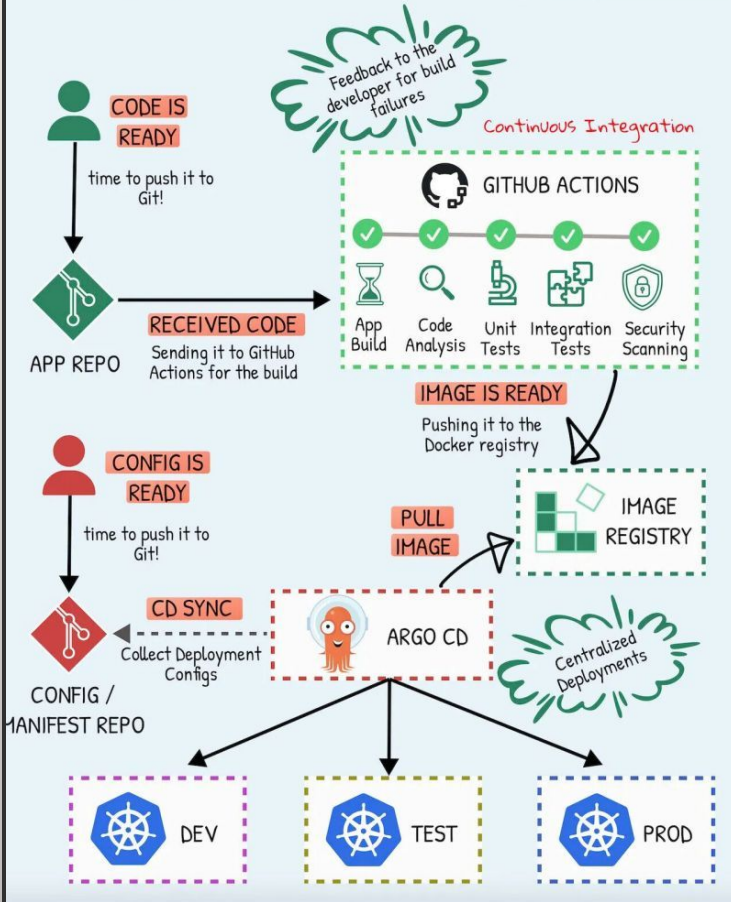
Permite desarrollar nuevas funcionalidades o pruebas sin afectar la versión principal del proyecto.



# GitOps WORKFLOW

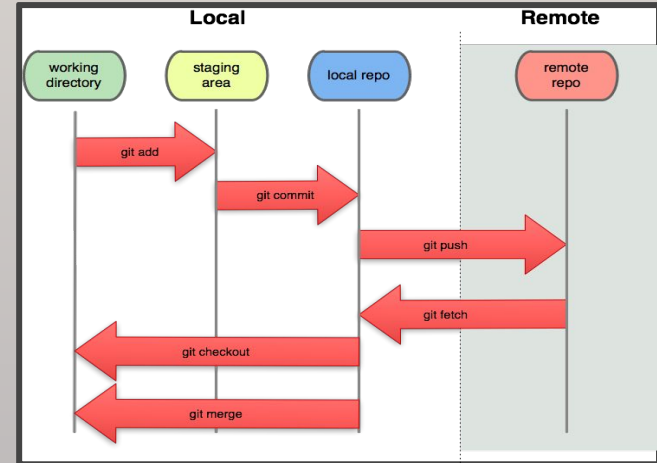
## Simplified Visual Guide

blog.bytebytego.com



## Flujo de trabajo - Git y GitHub

Podemos entonces entender un flujo de trabajo en materia de Git y GitHub mediante lo que denominamos como 6 pilares fundamentales, estos los dividiremos en 2 grupos de 3. Donde los 3 primeros serán sobre la creación y modificación inicial en el desarrollo; mientras que, los otros 3 reflejarán los últimos pasos, el almacenamiento en remoto y el trabajo compartido.



# Flujo de trabajo - Git y GitHub

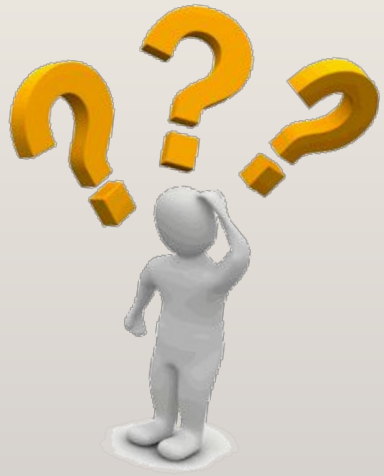
1. Trabajar localmente: Se genera una copia donde cada persona puede realizar cambios sin alterar el proyecto o la versión original de este.
2. Guardar cambios con commits: Se registran los cambios importantes en el historial del proyecto.
3. Ramas para desarrollo paralelo: Se crean ramas separadas para trabajar en nuevas características sin afectar la versión principal. Este punto puede verse semejante o similar al primero.



## Flujo de trabajo - Git y GitHub

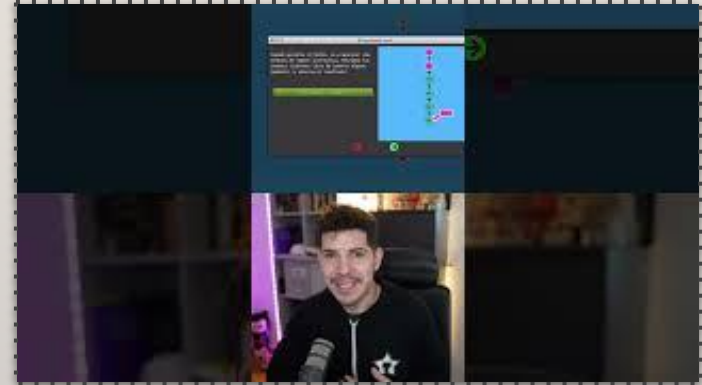
5. Subir y sincronizar cambios: Los cambios se envían al repositorio remoto en GitHub para ser almacenados y compartirlos.
6. Integrar cambios: Se revisan y fusionan los cambios realizados por los colaboradores para obtener un resultado final o previo a este.
7. Mantener un historial: Se obtiene acceso a todas las versiones del proyecto a fines de recuperar cambios base o modificaciones eliminadas sobre versiones más antiguas a la del punto anterior.

## Resumen del tema



¿Pero qué es lo que hemos  
aprendido con esto?

# Resumen del tema



# Webgrafía/Bibliografía

- [1]  
[Wikipedia - Control de versiones](#)  
[Bitbucket - Software de control de versiones para equipos profesionales](#)
- [2]  
[GIT- Git](#)  
[Wikipedia - Git](#)  
[GITHUB- Github](#)  
[Wikipedia - GitHub](#)  
[GitHub - Acerca de GitHub y Git](#)
- [3]  
[Atlassian - Qué es el control de versiones](#)  
[GitLab - ¿Qué es el control de versiones?](#)  
[KeepCoding - ¿Por qué usar sistemas de control de versiones de software?](#)
- [4]

# Enlace repositorio GitHub

- <https://github.com/Rodrigo-cyber17/Presentaci-n-ETS.git>

