

Agenda

sábado, 9 de septiembre de 2017 13:07



ÍNDICE

Duración: 16 horas

Objetivo: Al finalizar la acción formativa, los asistentes:

- Conocerán los elementos avanzados de la arquitectura de GIT
- Podrán aplicar determinadas técnicas a la gestión y organización de proyectos GIT

Requisitos:

Familiaridad con los repositorios de código fuente.
Idealmente, conocimientos básicos de GIT.

- Propósito y alcance
- Arquitectura
- Componentes fundamentales
- Operaciones CRUD
- Operaciones de organización de código
- Gestión de ramas y etiquetas
- Resolución de conflictos
- Integración con npm, Maven...
- Configuración de GIT
- Introducción a GIT en el lado servidor
- Programas cliente para operar con GIT

GIT AVANZADO

ESCUELA:
TECNOLOGÍA



 Indra Open University

- NOMBRE APELLIDO PROFESOR
Alejandro Cerezo

- VER PERFIL COMPLETO:



<https://www.linkedin.com/company/icono-training-consulting/>

<https://www.linkedin.com/in/alejandrocerezo/>

- CONTACTO



training@iconotc.com

alce65@hotmail.es

Título del documento | 3



Avda. de Bruselas 35
28108 Alcobendas,
Madrid España
T +34 91 480 50 00
F +34 91 480 50 80
www.indracompany.com

ij



[Icono Training Consulting: iconotc.com](http://iconotc.com)

ACCIÓN FORMATIVA: GIT AVANZADO

Duración: 16 horas

Requisitos Asistentes: Familiaridad con los repositorios de código fuente. Idealmente, conocimientos básicos de GIT.

Objetivos:

Al finalizar la acción formativa, los asistentes:

- Conocerán los elementos avanzados de la arquitectura de GIT
- Podrán aplicar determinadas técnicas a la gestión y organización de proyectos GIT

Contenido:

- ❖ Propósito y alcance
- ❖ Arquitectura
- ❖ Componentes fundamentales
- ❖ Operaciones CRUD
- ❖ Operaciones de organización de código
- ❖ Gestión de ramas y etiquetas
- ❖ Resolución de conflictos
- ❖ Integración con Maven
- ❖ Configuración de GIT
- ❖ Introducción a GIT en el lado servidor
- ❖ Programas cliente para operar con GIT

Desarrollo de la Agenda

lunes, 8 de enero de 2018 13:06

- Propósito y alcance
- Configuración de GIT
- Programas cliente para operar con GIT
- Arquitectura
- Componentes fundamentales
- Introducción a GIT en el lado servidor
- Integración con npm, Maven...
- Operaciones CRUD
- Operaciones de organización de código
- Gestión de ramas y etiquetas
- Resolución de conflictos
- Introducción: Propósito y alcance
 - Git: Control de versiones distribuido
 - Repositorios en la nube: GitHub, Bitbucket, GitLab
- Instalación y Configuración de GIT
 - Git. Terminales en Windows
 - Configuraciones
 - Clientes gráficos para operar con GIT
 - SourceTree
 - Cliente de GitHub
 - Integración en IDEs y Editores. Visual Studio Code
- Arquitectura y Componentes fundamentales
 - Áreas: Stashing - Working - Steaging - Repository
 - Creación de un repositorio, La carpeta .git
 - Punteros: HEAD
 - Commits
 - Readme.md y gitignore
- Introducción a GIT en el lado servidor
 - Repositorios bare
 - GitHub
 - GitLab
 - Bitbucket
- Integración con npm, Maven...
- Operaciones CRUD sobre los commits
 - C -> git add / git commit
 - R -> git status / git log
 - U -> git commit --amend
 - D -> git reset
- Operaciones de organización de código
 - Stashing
 - Reset
 - Revert
 - Repositorios remotos
- Gestión de ramas y etiquetas
 - Ramas
 - Merge
 - Resolución de conflictos
 - Tags
 - Rebase
- Workflow.
-

Introducción

lunes, 8 de enero de 2018 13:31

Introducción: Propósito y alcance

- Git: Control de versiones distribuido
- Repositorios en la nube:
 - GitHub - <https://github.com/>
 - Bitbucket <https://bitbucket.org/>
 - GitLab <https://about.gitlab.com/>

Sistemas de control de versiones (SCV)

martes, 25 de abril de 2017 0:01

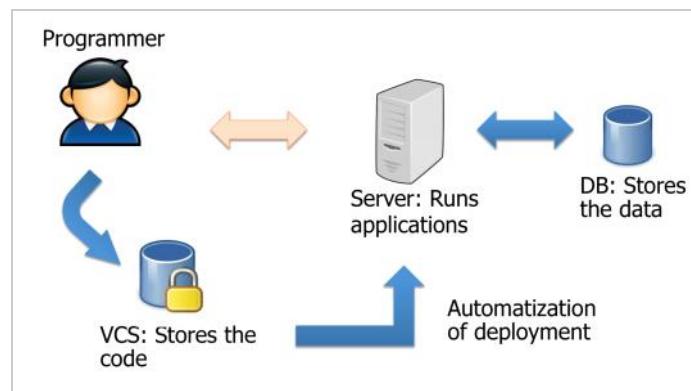
Los **sistemas de control de versiones** (SCV) son una herramienta esencial para manejar proyectos de software; lo que ha hecho de ellos herramientas de uso habitual en el desarrollo profesional de software desde hace décadas.

Proporcionan una serie de funcionalidades claves para el desarrollo de proyectos como es

- el control de cambios en el código,
- la reversibilidad de dichos cambios,
- la posibilidad de colaborar en el desarrollo del código.

Además, los SCV permiten tener en paralelo varias **versiones o ramas** del proyecto. Las ramas se utilizan para desarrollar funcionalidades aisladas de los cambios en otras partes del proyecto que posteriormente pueden integrarse en la rama principal.

Version Control System



Tipos de SVC

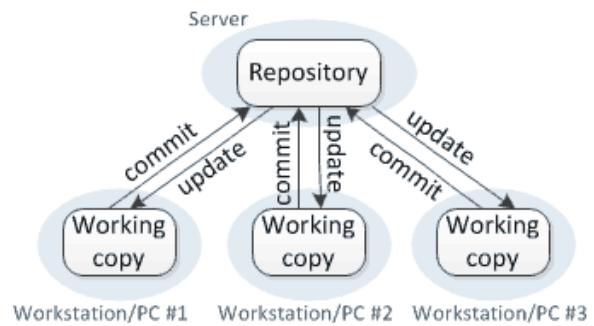
Lunes, 8 de enero de 2018 14:00

Los SCV se pueden clasificar en dos grandes familias:

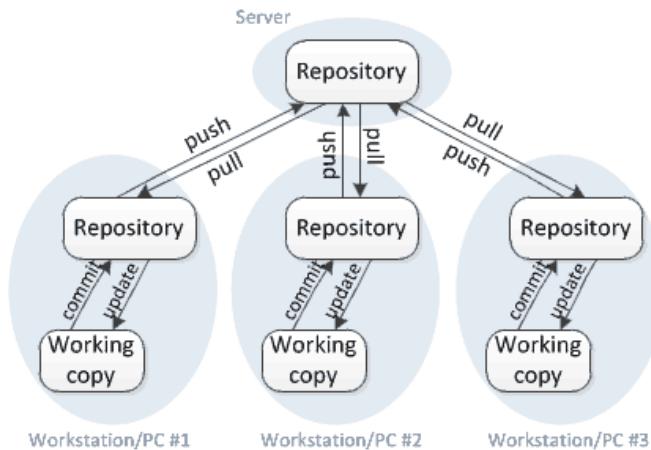
- SCV centralizados y
- SCV distribuidos.

Los SCV **centralizados** como CVS y Subversion (SVN) son sistemas cliente-servidor donde hay un repositorio canónico en el servidor que contiene toda la información de los cambios mientras que los clientes solo tienen copias de trabajo.

Centralized version control



Distributed version control



En sistemas **distribuidos** como Mercurial y Git no existe el concepto de repositorio canónico por lo que cada cliente ha de tener una copia completa del repositorio.

Desde 2010, la tendencia es utilizar cada vez más SCV distribuidos, en particular Git

SVC habituales

lunes, 8 de enero de 2018 14:02

Centralizados

- RCS (el original, de Pardue University luego GNU)
- CVS
- Microsoft Visual SourceSafe / Team Foundation Version Control (TFVC)
- Subversion (SVN) <https://subversion.apache.org/>

Distribuidos

- BitKeeper <http://www.bitkeeper.org/>
- Git <https://git-scm.com/>
- Mercurial (hg) <https://www.mercurial-scm.org/>
- Bazaar (bzr) <http://bazaar.canonical.com/en/>

“CVS and SVN are remote backups that you use to save your changes.
Git is an editor that you use to write your code’s biography.”

— isaacs (@izs) May 14, 2010

Cronología

1972	<i>Source Code Control System (SCCS)</i>	<i>Closed source</i> , libre con Unix
1982	<i>Revision Control System (RCS)</i>	<i>Cross platform</i> . <i>Open Source</i> . Más características y más rápido Capaz de trabajar solamente con un fichero cada vez
1986-1990	<i>Concurrent Versions System (CVS)</i> :	<i>Open Source</i> . Usuarios concurrentes pueden trabajar simultáneamente con un mismo fichero
2000	<i>Apache Subversion</i> .	Instantáneas (<i>Snapshot</i>) del directorio, y no solo de un fichero.. <i>Commits</i> transaccionales .
2000	<i>BitKeeper SCM</i> .	<i>Closed source, proprietary</i> . Sistema distribuido (<i>Distributed version control</i>) "Community version" libre. Usado para el código fuente del <i>kernel</i> de Linux entre 2002-2005
2005, abril		La "community version" deja de ser de uso libre
2005	<i>Git</i>	Desarrollado por el equipo responsable del <i>kernel</i> de Linux, encabezado por Linus Torvalds. Intenta ser una mejora de <i>BitKeeper</i> .

Git. El SVC distribuido más utilizado

martes, 25 de abril de 2017 0:09

Git es un SCV distribuido diseñado para la gestión eficiente de flujos de trabajo distribuido no lineales.
Git fue diseñado y desarrollado inicialmente por Linus Torvalds en 2005 para el desarrollo del kernel de Linux.

Git es un software de **control de versiones** diseñado por Linus Torvalds, pensando en la **eficiencia** y la **confiabilidad** del mantenimiento de versiones y aplicaciones cuando éstas tienen un gran número de archivos de **código fuente**.

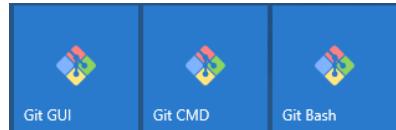
- Control de versiones
- Código fuente

automatizar la gestión de los cambios que se producen en archivos de todo tipo, tanto de código como de cualquier otro tipo, e.g. imágenes....

La licencia de Git es libre y hay distribuciones oficiales para los sistemas operativos

- Mac OS X,
- Windows,
- Linux y
- Solaris.

La distribución de Git incluye herramientas de **línea de comando** y de **escritorio**.



Además, hay disponibles herramientas proporcionadas por terceros que permiten una mayor integración con el escritorio o con entornos de desarrollo.

Orígenes

lunes, 8 de enero de 2018 14:15

Git fue diseñado y desarrollado inicialmente en 2005 con el objetivo concreto de dar soporte al desarrollo del kernel de Linux liderado por Linus Torvalds.

La experiencia del equipo en la gestión de la integración de las diferentes aportaciones en un proyecto distribuido de esta magnitud determinó los objetivos del proyecto

- Rapidez
- Simplicidad de uso
- Multiplicidad de versiones (Ramas)
- Distribuido:
capaz de trabajar sin conexiones
- Preparado para grandes proyectos

Estos objetivos se reflejaron en las siguientes decisiones de implementación:

Versiones no incrementales. Git almacena cada cambio como una instantánea de todos los archivos del proyecto. Para ser eficiente, si el archivo no ha sido modificado, sólo se almacena un enlace al archivo idéntico previamente almacenado. Los SCV anteriores a Git habitualmente almacenaban solo una versión base y las modificaciones hechas en cada cambio por archivo.

Trabajo fuera de línea. Por ser un sistema distribuido, cada repositorio de Git es un repositorio completo capaz de funcionar sin acceso a la red o al resto de los repositorios distribuidos gracias a que contiene una copia local de la historia completa del desarrollo del proyecto. Los cambios en la historia pueden copiarse de un repositorio a otro como nuevas ramas de desarrollo y se pueden copiar de la misma manera que una rama de desarrollo local.



- Cada operación se realiza en el repositorio local.
- No necesita conexión con un servidor.
- Se puede trabajar normalmente sin conexión (*offline*)
- Es más rápido que los repositorios centralizados

Autenticación criptográfica de la historia. El identificador de un cambio se computa utilizando un algoritmo criptográfico que utiliza como entrada el cambio y la historia completa de cambios. Esto permite que cualquier cambio de la información durante la transmisión o en sistema de archivos sea detectado por Git.

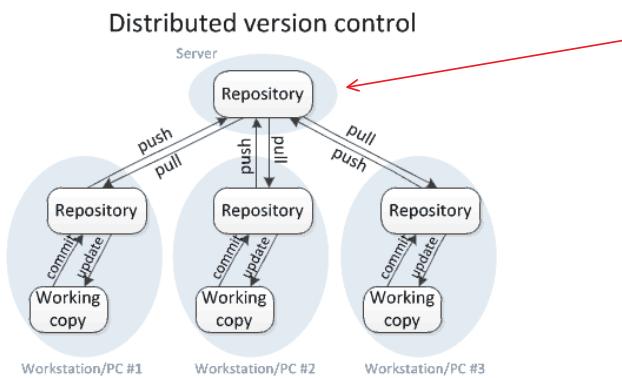


- Todo está identificado por secuencias *hash*.
- Es imposible cambiar el contenido de cualquier archivo o directorio sin que Git lo sepa.
- No se puede perder información ni dañar el archivo sin que Git pueda detectarlo.
- Ejemplo de secuencia de comprobación (*checksum*)

24b9da6552252987aa493b52f8696cd6d3b00373

Repositorios remotos

lunes, 8 de enero de 2018 15:57



Uno de los equipos actúa como repositorio de referencia, pero en realidad es solo una variación similar a cualquier otro

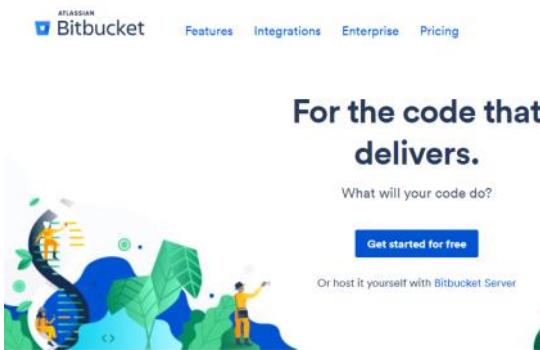
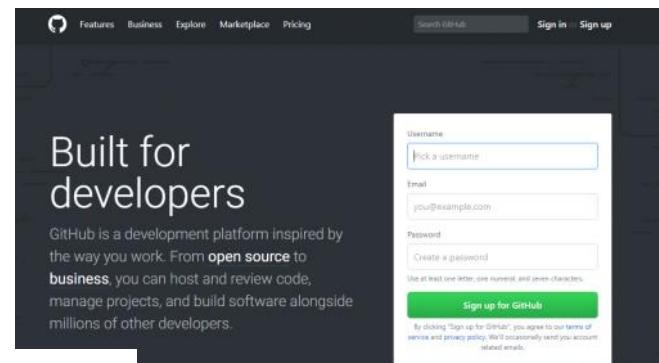
En la red de los usuarios

Existen versiones servidor de Git como

- *gitolite* <http://gitolite.com/gitolite/index.html>
- *gitosis* <https://git-scm.com/book/es/v1/Git-en-un-servidor-Gitosis> o incluyendo una completa interfaz Web,
- *GitLab*, para Linux <https://about.gitlab.com/equipos>

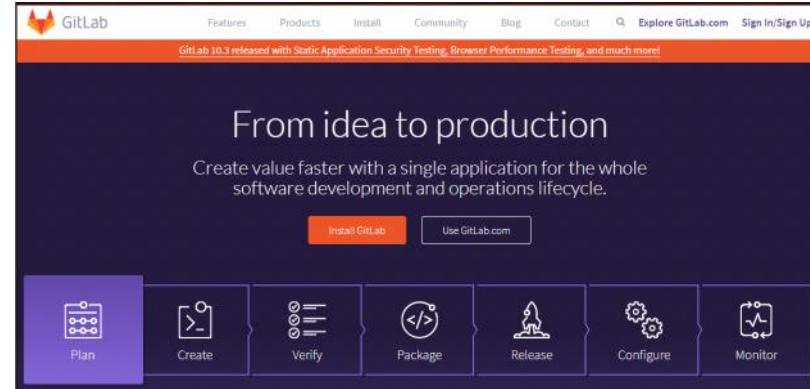
En Internet

GitHub - <https://github.com/>



Bitbucket <https://bitbucket.org/>

GitLab <https://about.gitlab.com/>



Información Extra

domingo, 29 de octubre de 2017 20:59

The screenshot shows a visual guide for Git. At the top, it says "git - la guía sencilla" and provides a link: <http://rogerdudler.github.io/git-guide/index.es.html>. Below this, there's a large downward arrow pointing from a small icon to a larger one. The text "want a simple but powerful git client for your mac?" is visible.

Una referencia visual de Git

Otros Idiomas: [Deutsch](#) [English](#) [Français](#) [Italiano](#) [日本語](#) [한국어](#) [Polski](#) [Português](#) [Русский](#) [Slovenčina](#) [Tiếng Việt](#) [简体中文](#) [正體中文](#)

Si las imágenes no funcionan, puedes intentar la [versión sin-SVG](#) de esta página.

Esta página da una referencia breve y visual para los comandos más comunes en git. Una vez que conozcas un poco sobre la forma de trabajo de git, este sitio puede fortalecer tu entendimiento. Si estás interesado en cómo se creó este sitio, mira mi [repositorio de GitHub](#).

Contenidos

1. [Uso Básico](#)
2. [Convenciones](#)
3. [Comandos en Detalle](#)
 - a. [Diff](#)
 - b. [Commit](#)
 - c. [Checkout](#)
 - d. [Committeando con un HEAD Detachado](#)
 - e. [Reset](#)
 - f. [Merge](#)
 - g. [Cherry Pick](#)
 - h. [Rebase](#)
4. [Notas Técnicas](#)

<http://marklodato.github.io/visual-git-guide/index-en.html>

Otro resumen, sin un diseño tan cuidado

The screenshot shows a grid-based reference for Git commands. It includes sections for Resource, Create Git, Local Changes, History, Merge/Rebase, Branching/Tagging, and Useful Commands. Each section lists specific git commands and their descriptions.

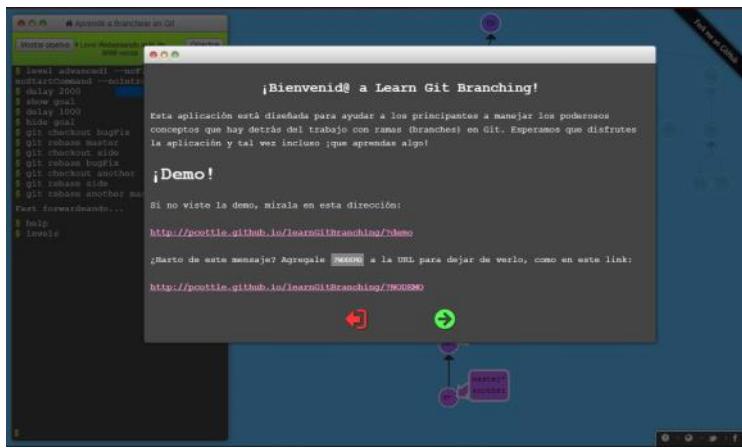
<http://overapi.com/git>

Referencia a los comandos de Git

The screenshot shows a diagram of the Git repository structure. It highlights the WORKSPACE, INDEX, LOCAL REPOSITORY, and REMOTE REPOSITORY. Arrows point from various commands to their corresponding sections in the repository. A legend at the bottom defines symbols for branches, tags, and merges.

<https://ndpssoftware.com/git-cheatsheet.html#loc=workspace>

Referencia interactiva con los comandos de Git superpuestos en la estructura del repositorio



<https://learngitbranching.js.org/?demo>

Animación del proceso de creación de ramas en Git

<https://try.github.io/levels/1/challenges/1>

Tutorial interactivo, en el que OCTOCAT nos enseña a manejar Git

Libros

Junes, 8 de enero de 2018 18:30

<https://git-scm.com/book/es/v2>

<https://git-scm.com/book/en/v2>

Original en inglés

Download Ebook



Book

The entire Pro Git book, written by Scott Chacon and Ben Straub and published by Apress, is available here. All content is licensed under the Creative Commons Attribution Non Commercial Share Alike 3.0 license. Print versions of the book are available on Amazon.com.



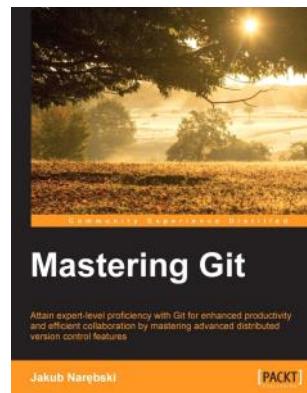
2nd Edition (2014)
Switch to 1st Edition

1. Inicio - Sobre el Control de Versiones

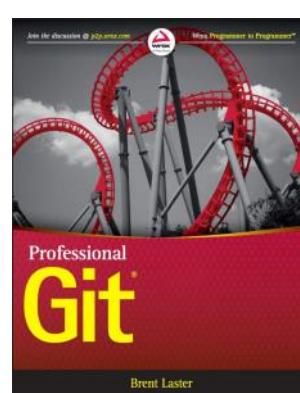
- 1.1 Acerca del Control de Versiones
- 1.2 Una breve historia de Git
- 1.3 Fundamentos de Git
- 1.4 La Línea de Comandos
- 1.5 Instalación de Git
- 1.6 Configurando Git por primera vez
- 1.7 ¿Cómo obtener ayuda?
- 1.8 Resumen



Git for Teams
Emma Jane Westby
O'Reilly, 2015



Mastering Git
Jakub Narębski
Packt Publishing, 2016



Professional Git
Brent Laster
John Wiley & Sons, 2016

Instalación y Configuración

lunes, 8 de enero de 2018 13:33

Instalación y Configuración de GIT

- Git. Terminales en Windows
 - Configuraciones
- Clientes gráficos para operar con GIT
 - SourceTree
 - Cliente de GitHub
- Integración en IDEs y Editores. Visual Studio Code

Instalación de Git

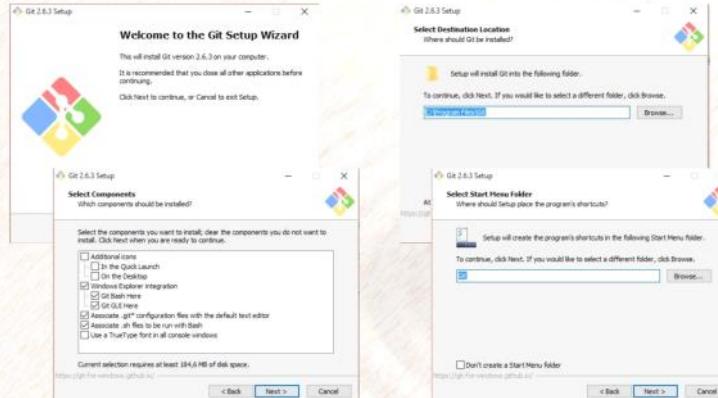
martes, 25 de abril de 2017 0:24

The screenshot shows the official Git website at git-scm.com. At the top right is a search bar. Below it is a diagram illustrating the distributed nature of Git. A central stack of books is connected by red arrows to several other stacks of books, symbolizing the branching and merging of code. To the left of the diagram, there's a link to "Learn Git in your browser for free with Try Git." Below the diagram are four main navigation links: "About", "Documentation", "Downloads", and "Community". Under "Downloads", there's a section for "Pro Git" with a link to its online version and an Amazon.com link. In the center, a large box highlights the "Latest source Release 2.14.1" with a "Downloads for Windows" button. Below this are links for "Windows GUIs", "Tarballs", "Mac Build", and "Source Code". A red arrow points from the "Downloads for Windows" button down to the first step of the installation wizard.

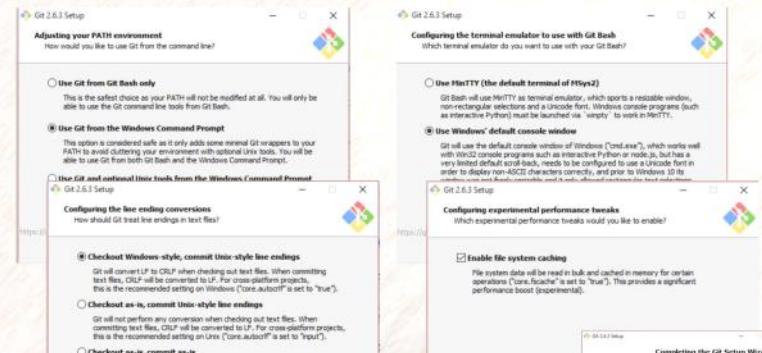
Git. Instalación (1)

<http://git-scm.com/download/win> → Git-2.6.3-64-bit.exe

Puede ser necesario ejecutar como administrador



Git. Instalación (2)





Resultado

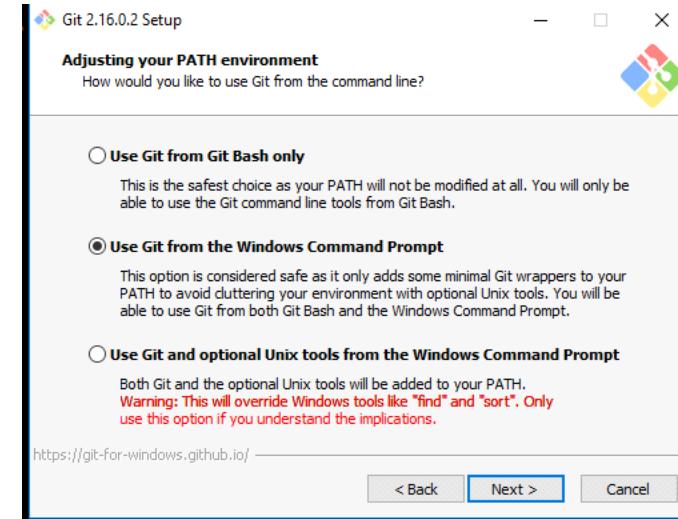
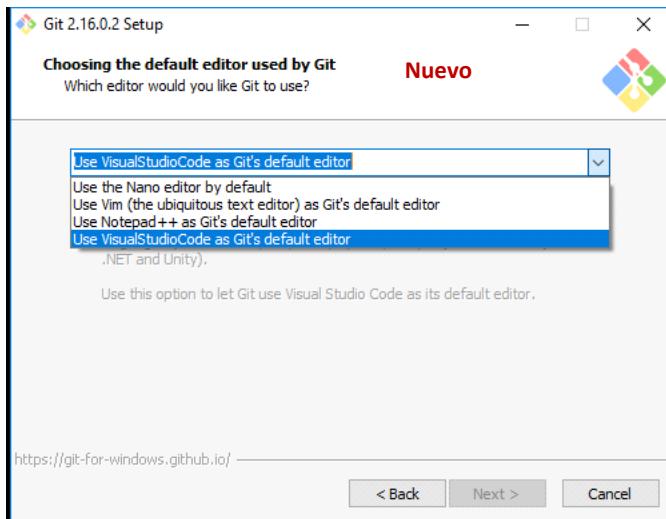
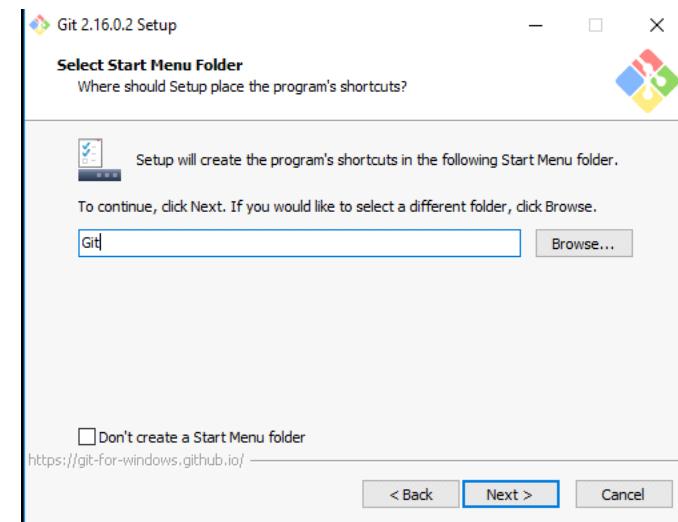
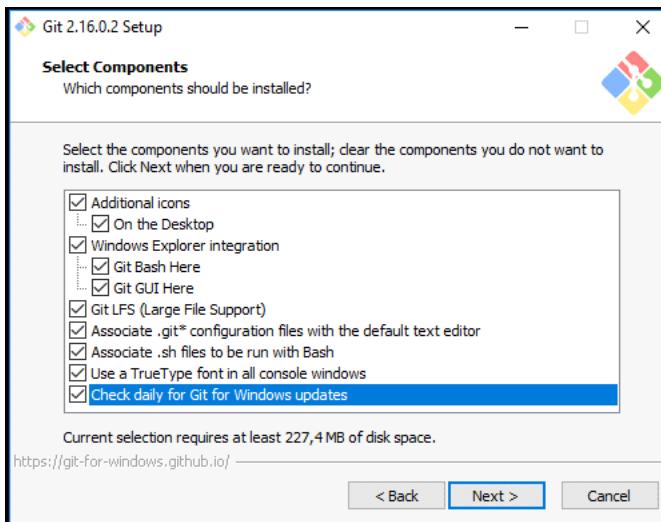
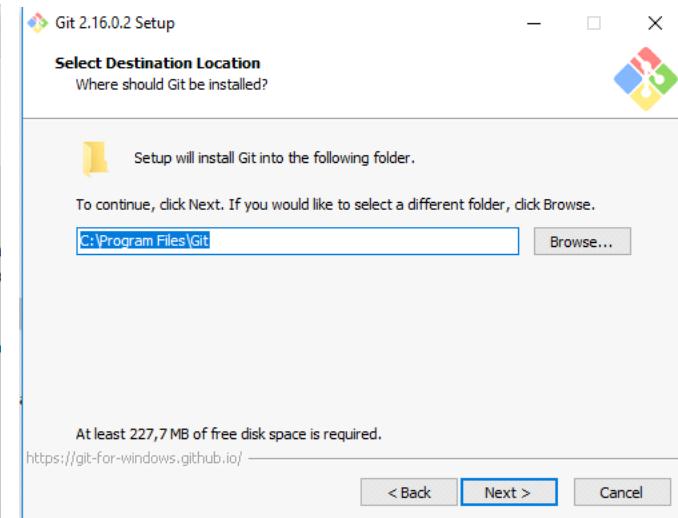
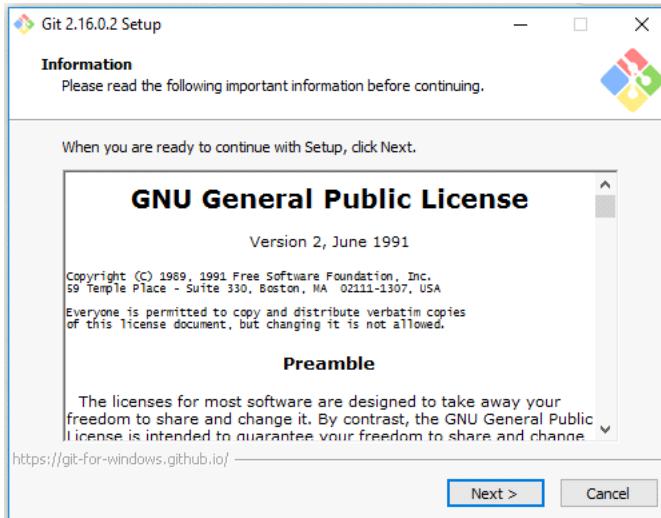


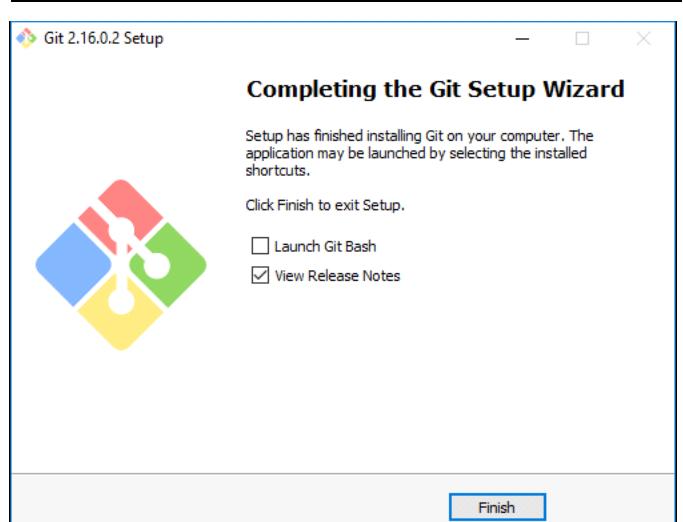
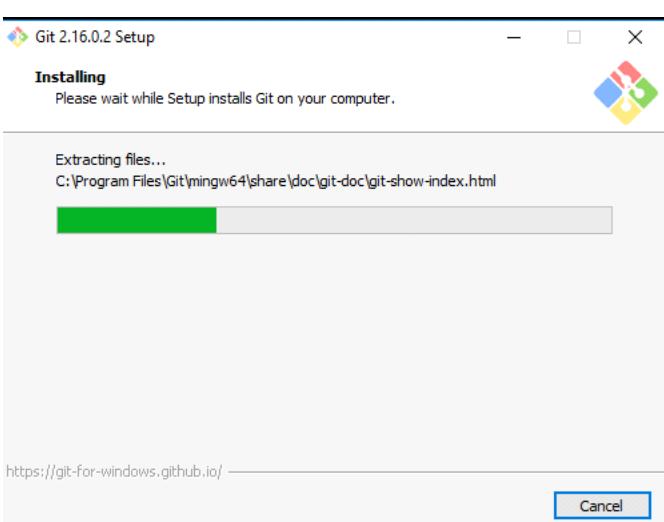
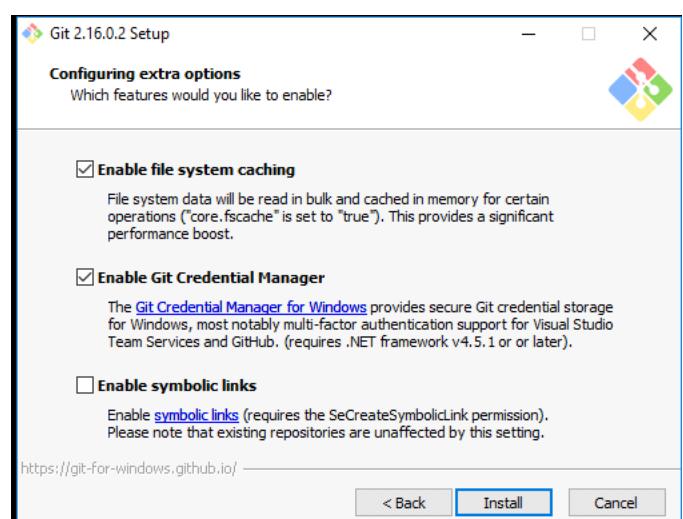
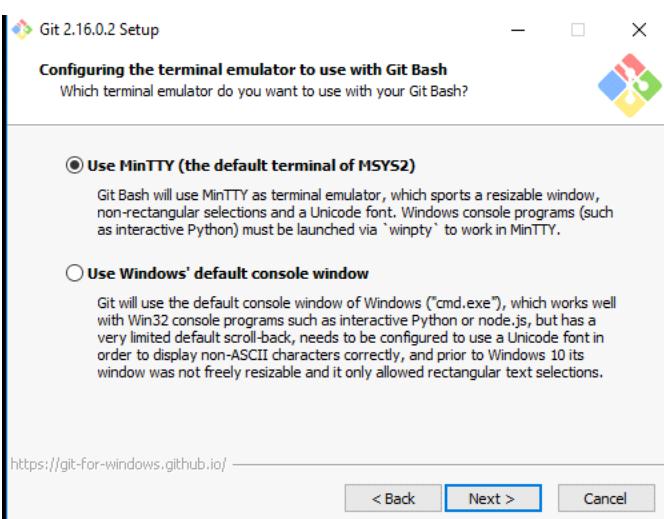
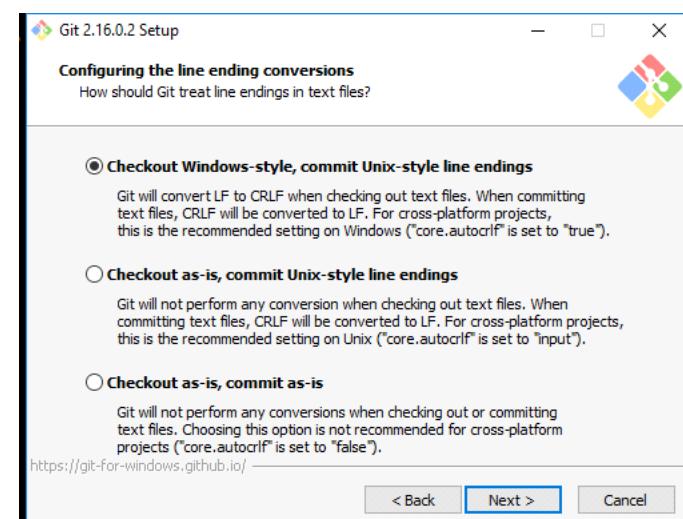
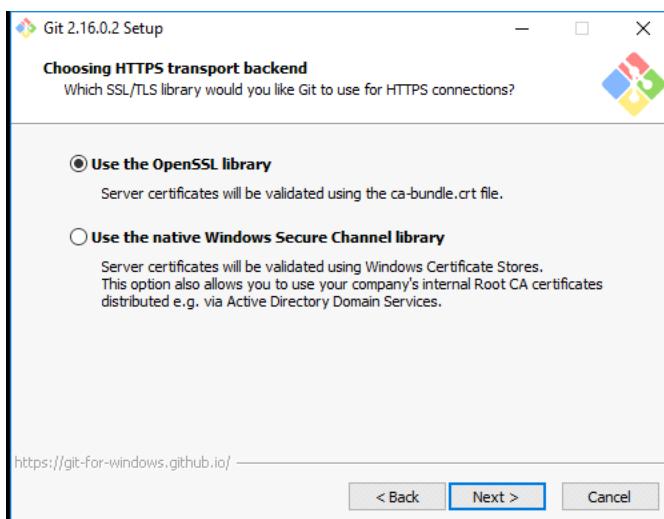
Comprobación

```
MINGW64:/d/Users/Alejandro
Alejandro@Cyborg8 MINGW64 ~
$ git --version
git version 2.12.1.windows.1
Alejandro@Cyborg8 MINGW64 ~
$ |
```

2.16.x

domingo, 21 de enero de 2018 22:08





Terminales

lunes, 8 de enero de 2018 14:40

Git Bash: shell Linux incluida en la instalación de Git

```
alce6@Cyborg10 MINGW64 ~
$ git --version
git version 2.14.1.windows.1
alce6@Cyborg10 MINGW64 ~
$ -
```

Símbolo del sistema (consola) de Windows

```
C:\Users\alce6>git --version
git version 2.14.1.windows.1
C:\Users\alce6>
```

Símbolo del sistema

```
Microsoft Windows [Versión 10.0.16299.192]
(c) 2017 Microsoft Corporation. Todos los derechos reservados.
```

```
C:\Users\alce6>git --version
git version 2.14.1.windows.1
C:\Users\alce6>
```

Power Shell configurado especialmente para Git

```
posh~git ~ Git_Basic [master]
D:\desarrollo\Gits\Git_Basic [master]> git --version
git version 2.14.1.windows.1
D:\desarrollo\Gits\Git_Basic [master]> -
```

Se utiliza el *plugin* posh-git

<https://github.com/dahlbyk/posh-git>

En la propia Web de GitHub se indica como instalarlo directamente o después de instalar *GitHub for Windows*, que ya incluye el plugin

<https://git-scm.com/book/sr/v2/Appendix-A%3A-Git-in-Other-Environments-Git-in-Powershell>

Para ello hay que crear / editar el fichero

%user%\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1

En él se incluirá

- . (Resolve-Path "\$env:LOCALAPPDATA\GitHub\shell.ps1")
- . \$env:github_posh_git\profile.example.ps1

Para la creación del script de PowerShell el proceso es el siguiente

<https://technet.microsoft.com/en-us/library/ff461033.aspx>

*At the Windows PowerShell prompt, type the following command, and then press ENTER:
Test-path \$profile*

*If the results of the previous command are false, Type the following command, and then press ENTER.
New-item -type file -force \$profile*

*If the results of the test are true, type the following command, and then press ENTER.
Notepad \$profile*

Configuración

domingo, 10 de septiembre de 2017 13:01

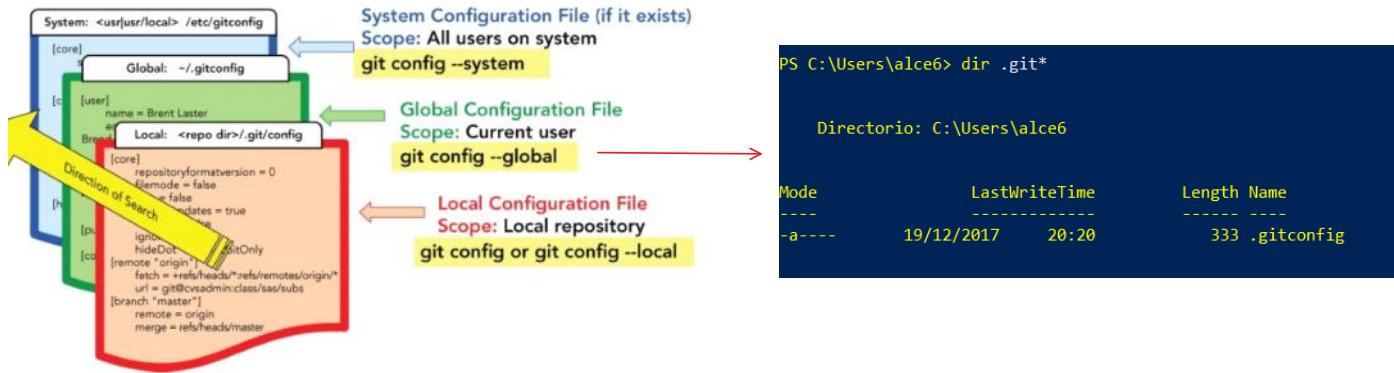
Git trae una herramienta llamada **git config** que te permite obtener y establecer variables de configuración, que controlan el aspecto y funcionamiento de Git.

Estas variables pueden almacenarse en tres sitios distintos, dependiendo de los parámetros del comando:

- **Archivo /etc/gitconfig**: Contiene valores para todos los usuarios del sistema y todos sus repositorios. Si pasas la opción **--system** a git config, lee y escribe específicamente en este archivo.
- **Archivo ~/.gitconfig**: Específico a tu usuario. Puedes hacer que Git lea y escriba específicamente en este archivo pasando la opción **--global**.
- **Archivo config "local"**, en el directorio de Git (es decir, **.git/config**) del repositorio que estés utilizando actualmente: Específico a ese repositorio. Cada nivel sobrescribe los valores del nivel anterior, por lo que los valores de **.git/config** tienen preferencia sobre los de **/etc/gitconfig**

Como en cualquier entorno Linux, los modificadores utilizan
-- seguido de una palabra
- seguido de una letra, muchas veces como

Ámbito (Scope) de los ficheros de configuración en Git



Usuario

lunes, 8 de enero de 2018 14:27

Identidad del usuario

```
$ git config --global user.name "Pepe Perez"  
$ git config --global user.email pperez@example.com
```

Username

```
$ git config --global user.name "Your Name Here"  
# Sets the default name for git to use when you commit
```

Email

```
$ git config --global user.email "your_email@example.com"  
# Sets the default email for git to use when you commit
```

Editor por defecto

lunes, 8 de enero de 2018 14:31

Configuración del editor por defecto

git config --global core.editor "\"C:\Program Files (x86)\Microsoft VS Code\Code.exe\""

git config --global core.editor "code.exe --wait"

Propiedades configuradas

miércoles, 27 de diciembre de 2017 11:02

```
PS C:\Users\alce6> git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
diff.astextplain.textconv=astextplain
rebase.autosquash=true
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.required=true
filter.lfs.process=git-lfs filter-process
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
http.sslbackend=openssl
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.required=true
filter.lfs.process=git-lfs filter-process
credential.helper=manager
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.required=true
user.name=alce65
user.email=alce65@hotmail.es
core.autocrlf=true
core.excludesfile=D:\alce65\Documents\gitignore_global.txt
core.editor="C:\Program Files (x86)\Microsoft VS Code\Code.exe"
PS C:\Users\alce6>
```

Alias

martes, 9 de enero de 2018 0:36

Los alias fueron añadidos en Git 1.4.0

Evitan tener que teclear repetidas veces el mismo comando completo

Se pueden configurar

- en cualquiera de los ámbitos de configuración mencionados
- en cualquier caso,
 - o por comandos
 - o editando el fichero correspondiente

`git config --global alias.co commit`

- Editar `Programs/git/gitconfig`
- Editar `$HOME/<user>/.gitconfig` (**específico de usuario**)
- Editar `.git//config` (específico del repositorio)

`[alias]
co = commit`

Ejemplo de uso

`git config --global alias.hist git log --pretty=format:"%h %ad | %s%d [%an]" --graph --date=short`

Configuración de proxies

lunes, 7 de agosto de 2017 21:38

Configuración git

```
git config --global http.proxy http://user:passw@proxy.empresas.es:8080
```

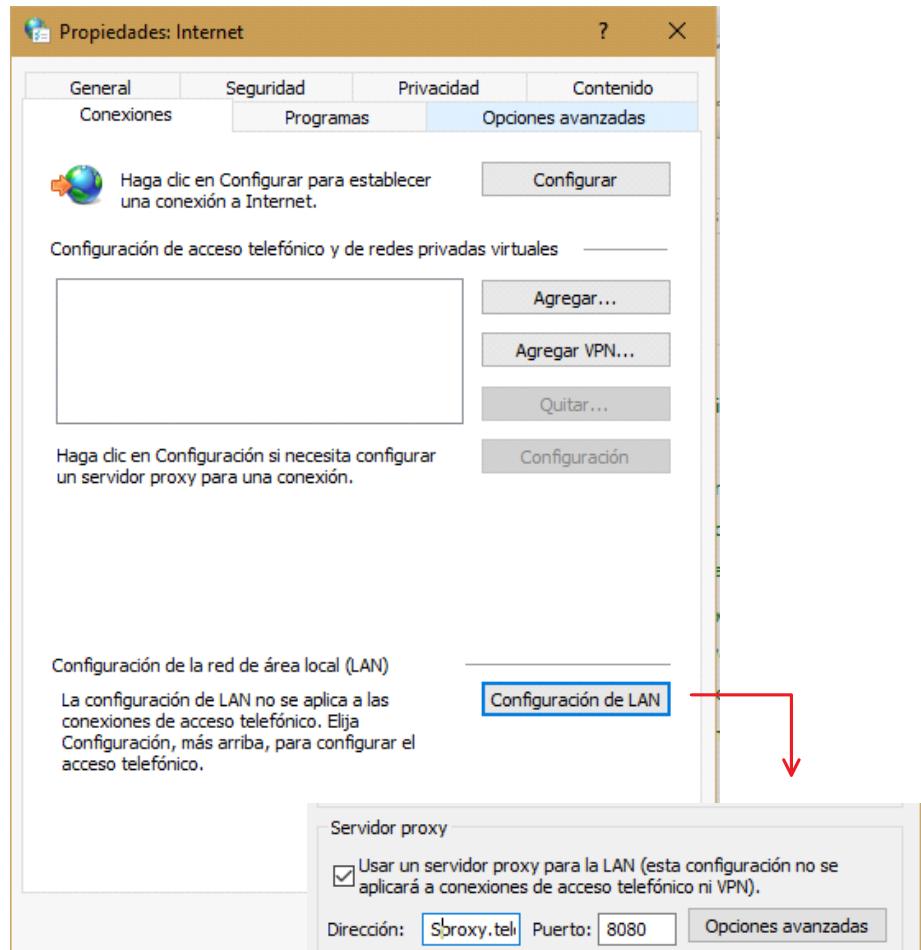
Otros aspectos de la configuración de conexiones

Configuración npm

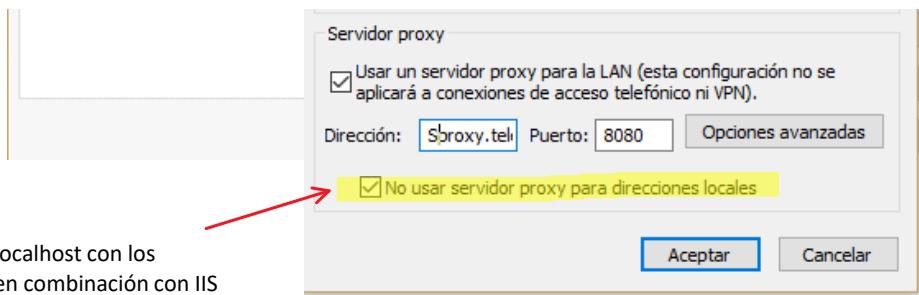
```
$>npm config set proxy http://user:passw@proxy.empresas.es:8080
```

```
$>npm config set https-proxy http://user:passw@proxy.empresas.es:8080
```

Propiedades de internet



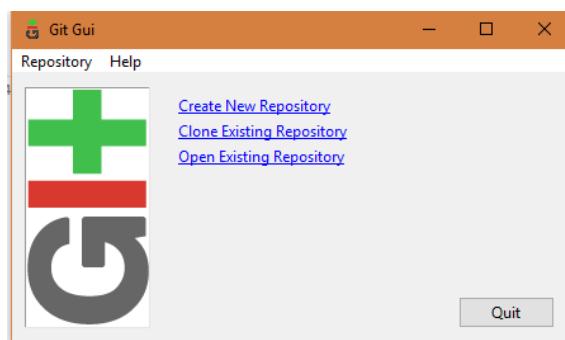
Permite usar localhost con los navegadores en combinación con IIS para probar el código desarrollado



Clientes gráficos

lunes, 8 de enero de 2018 14:34

Cliente incluido en la instalación de Git, prácticamente sin funcionalidad



Otras opciones desde la propia Web de Git

<https://git-scm.com/download/gui/windows>

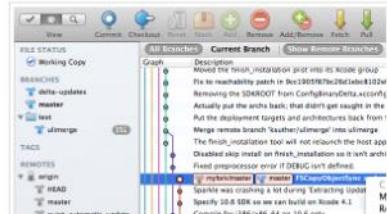
GUI Clients

Git comes with built-in GUI tools for committing ([git-gui](#)) and browsing ([gitk](#)), but there are several third-party tools for users looking for platform-specific experience.

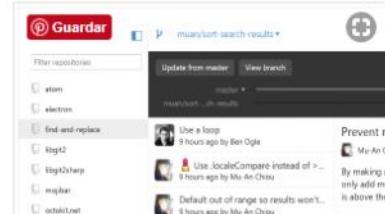
If you want to add another GUI tool to this list, just [follow the instructions](#).

All Windows Mac Linux Android iOS

16 Windows GUIs are shown below ↓



SourceTree
Platforms: Mac, Windows
Price: Free
License: Proprietary



GitHub Desktop
Platforms: Mac, Windows
Price: Free
License: MIT

- SourceTree
- GitHub Desktop
- TortoiseGit
- Git Extensions
- GitKraken
- SmartGit
- Tower
- GitEye
- gitg
- ungit
- git-cola
- Cycligent Git Tool
- Aurees
- CodeReview
- gmaster
- GitAhead

SourceTree

Platforms: Mac, Windows
Price: Free
License: Proprietary

SmartGit

Platforms: Linux, Mac, Windows
Price: \$79/user / Free for non-commercial use
License: Proprietary

Git Extensions

Platforms: Linux, Mac, Windows
Price: Free
License: GNU GPL

ungit

Platforms: Linux, Mac, Windows
Price: Free
License: MIT

Aurees

GitHub Desktop

Platforms: Mac, Windows
Price: Free
License: MIT

Tower

Platforms: Mac, Windows
Price: \$79/user (Free 30 day trial)
License: Proprietary

GitKraken

Platforms: Linux, Mac, Windows
Price: Free for non-commercial use
License: Proprietary

git-cola

Platforms: Linux, Mac, Windows
Price: Free
License: GNU GPL

CodeReview

TortoiseGit

Platforms: Windows
Price: Free
License: GNU GPL

GitEye

Platforms: Linux, Mac, Windows
Price: Free
License: Proprietary

gitg

Platforms: Linux, Windows
Price: Free
License: GNU GPL

Cycligent Git Tool

Platforms: Linux, Mac, Windows
Price: Free
License: Proprietary

gmaster

Platforms: Windows

Price: Free
License: MIT

Aurees

Platforms: Windows
Price:
License: Proprietary

Price: Free
License: GNU GPL

CodeReview

Platforms: Linux, Mac, Windows
Price: Free
License: GNU GPL

License: Proprietary

gmaster

Platforms: Windows
Price: Beta / Free for non-commercial use
License: Proprietary

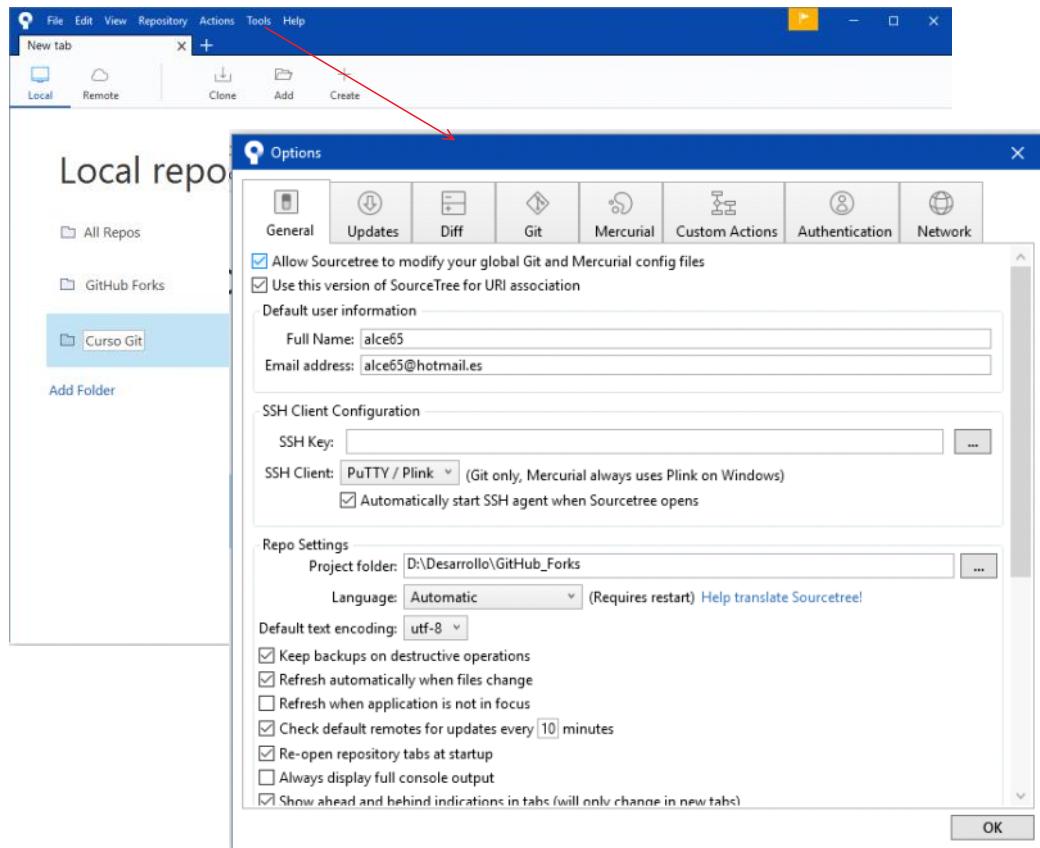
GitAhead

Platforms: Linux, Mac, Windows
Price: Free for non-commercial use
License: Proprietary

SourceTree

Junes, 8 de enero de 2018 14:36

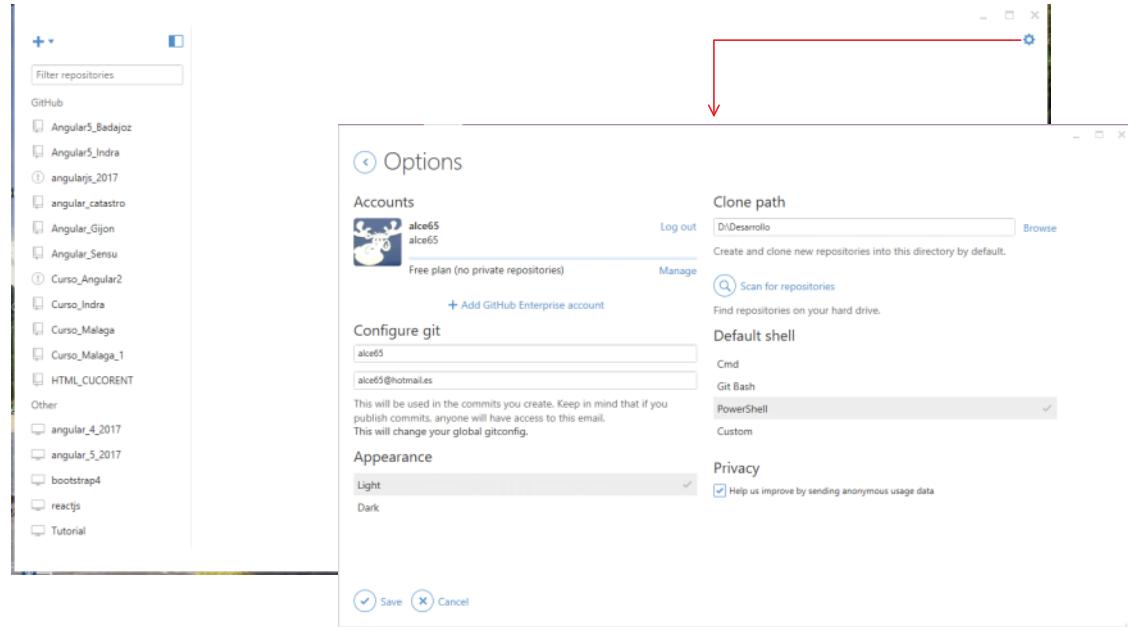
<https://www.sourcetreeapp.com/>



GitHub Desktop

lunes, 8 de enero de 2018 14:38

<https://desktop.github.com/>



Entornos de desarrollo

lunes, 8 de enero de 2018 23:18

Editores de código



IDEs

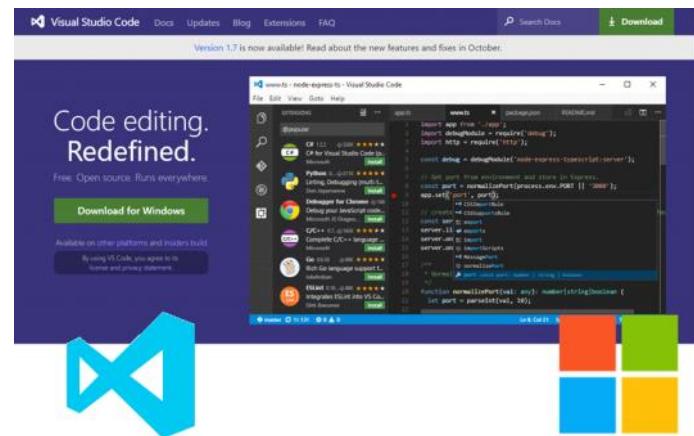
- Eclipse
- NetBeans
- Visual Studio



Editor de código: VSC

sábado, 20 de enero de 2018 12:10

<https://code.visualstudio.com/>



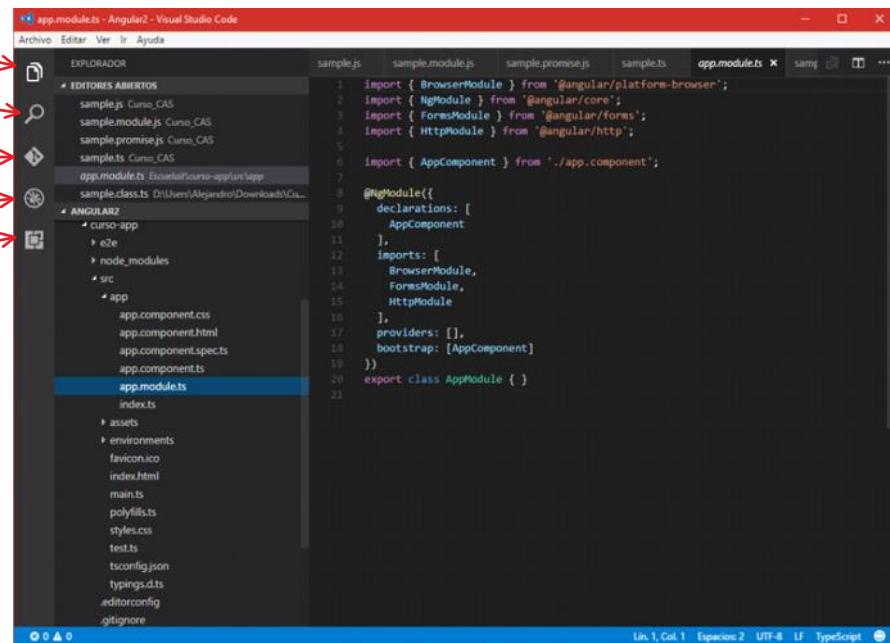
Carpetas y ficheros

Búsquedas en el proyecto

Git - GitHub integrado

Depurador

Extensiones



IDE: Eclipse

Junes, 8 de enero de 2018 23:44

<https://www.eclipse.org/ide/>

The screenshot shows the Eclipse IDE for Java Developers website and a screenshot of the Eclipse IDE interface.

Eclipse IDE for Java Developers

Package Description
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle Integration

This package includes:

- Git Integration for Eclipse
- Eclipse Java Development Tools
- Maven Integration for Eclipse
- Mylyn Task List
- Code Recommenders Tools for Java Developers
- Eclipse XML Editors and Tools

Detailed features list

Download Links
Windows 32-bit
Windows 64-bit
Mac OS X (Cocoa) 64-bit
Linux 32-bit
Linux 64-bit
Downloaded 145,600 Times
Checksums...
Bugzilla
Open Bugs: 35
Resolved Bugs: 114

Maintained by: Eclipse Packaging Project

Eclipse IDE for Java Developers

Git Repositories

java_desk - Java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Klondike.java Descarte.java

```
1 package Klondike;
2
3 public class Descarte {
4     ...
5     public void mostrar() {
6         ...
7     }
8
9     public void moverA(Palo[] palos) {
10        ...
11    }
12
13    public void moverA(Columna[] columnas) {
14        ...
15    }
16
17
18
19
20
21
22
23 }
```

Git Repositories > java_desk [master t3 i2 - Merged]

- Branches
 - Local
 - Remote Tracking
 - Tags
- References
- Remotes
 - origin
- Working Tree - D:\Desarrollo\Back

Outline Klondike Descarte

- mostrar(): void
- moverA(Palo[]): void
- moverA(Columna[]): void
- moverA(Columna): void
- moverA(Palo): void
- poner(Carta): void

En resumen

sábado, 20 de enero de 2018 12:27

- Instalación de Git
- (Creación de cuenta en GitHub)
- Instalación de GitHub Desktop
- Configuración del usuario en Git
- Configuración de *proxies* en Git
- Configuración de PowerShell para Git
- Instalación de SourceTree
- Instalación de VSC
- (Instalación de Eclipse)

Arquitectura y Componentes

lunes, 8 de enero de 2018 13:36

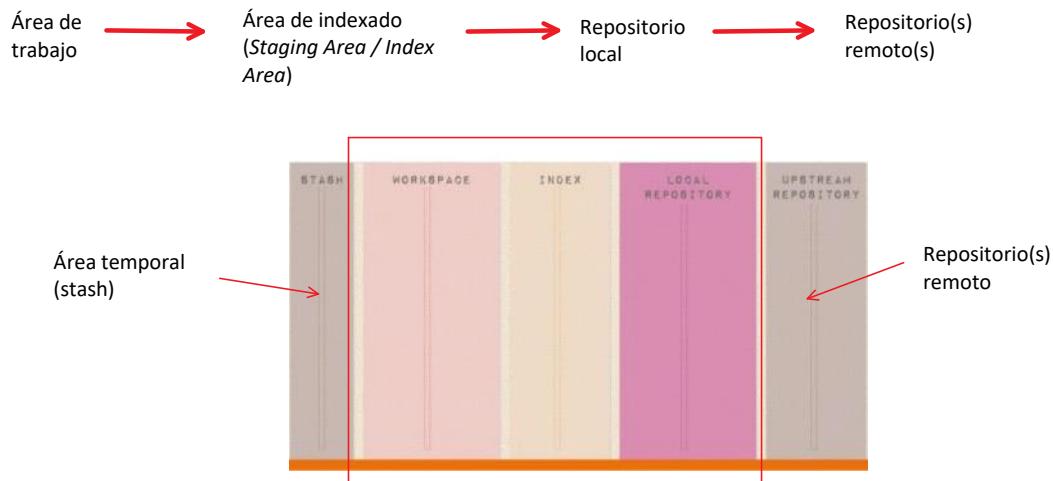
Arquitectura y Componentes fundamentales

- Áreas
 - Stashing
 - Working
 - Steaging
 - Repository
- Commits
- Punteros: HEAD
- Creación de un repositorio, La carpeta .git
 - Readme.md y gitignore

Estructura

jueves, 4 de enero de 2018 19:06

Git puede concebirse como un sistema de niveles en el que se promocionan los ficheros:

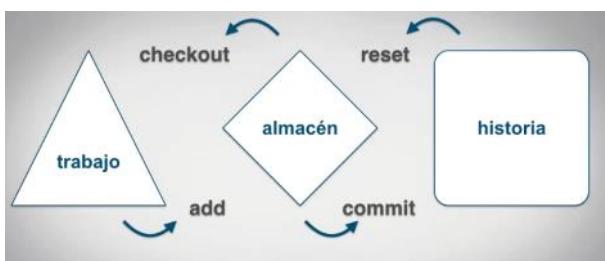


El repositorio creado tiene tres partes diferenciadas. El directorio es lo que se llama directorio o espacio de trabajo, y dentro de la carpeta .git que ha sido creada por el comando se encuentra el área de preparación, que actúa como una zona intermedia, y el historial de cambios.

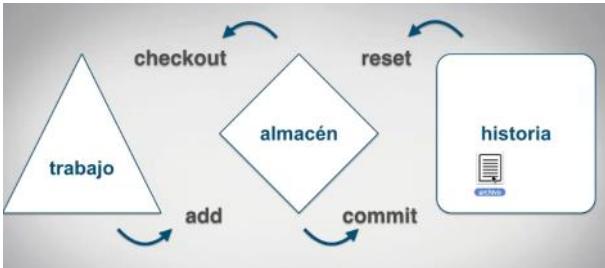
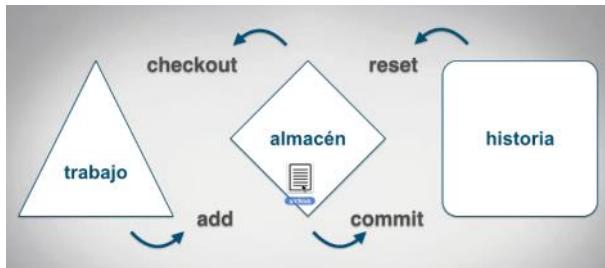
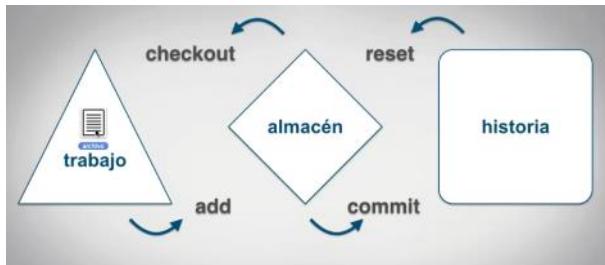
Etapas y flujo

jueves, 11 de mayo de 2017 0:08

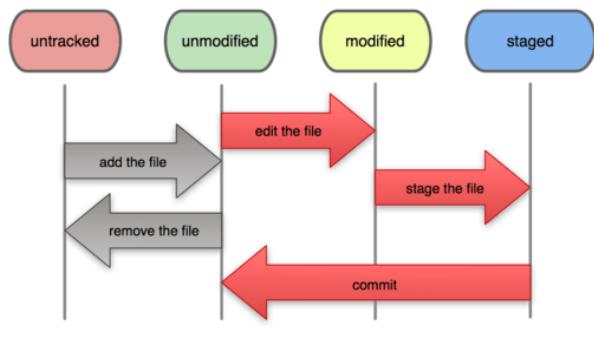
Estructura del repositorio



Ciclo de vida del fichero



File Status Lifecycle



Tracked

En su ciclo de vida (*Lifecycle*) los ficheros pasan por diversos estados

Cada fichero en el directorio de trabajo (*working directory* o *working area*) estará en uno de los dos estados posibles: rastreado (*tracked*) o sin seguimiento (*untracked*).

Tracked: ficheros que aparecen en la última instantánea (*snapshot*) del repositorio, pudiendo encontrarse modificados, preparados (*staged*) o no modificados y confirmados, .

Estados del archivo



Modificado

Preparado

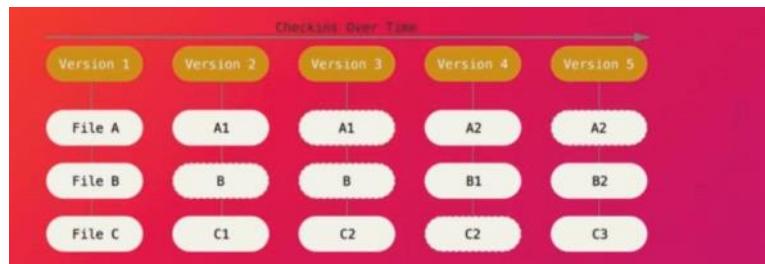
Confirmado

Untracked: cualquier otro fichero, del que el repositorio no tiene ninguna información.

Almacenamiento

sábado, 20 de enero de 2018 17:42

Git SVC: *Snapshot Storage Model*



OTROS SVC: *Delta Storage Model*

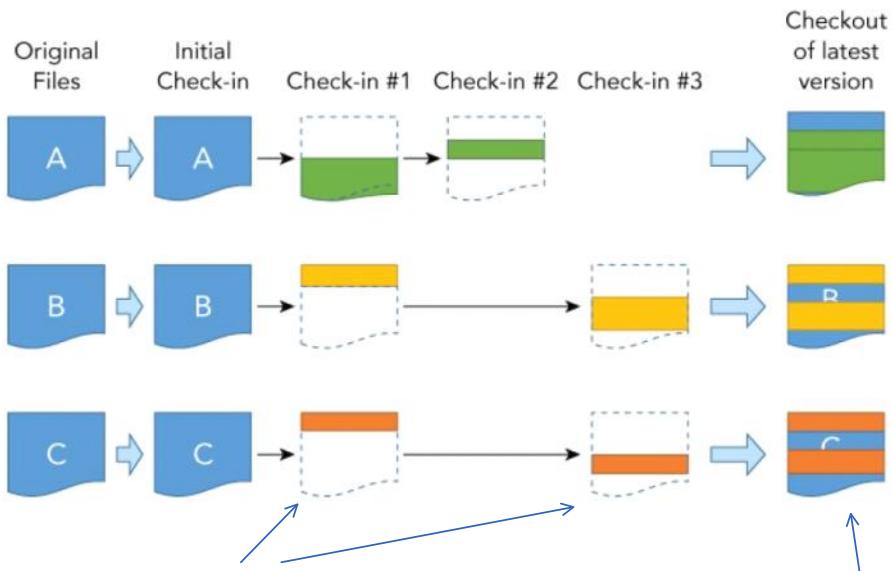
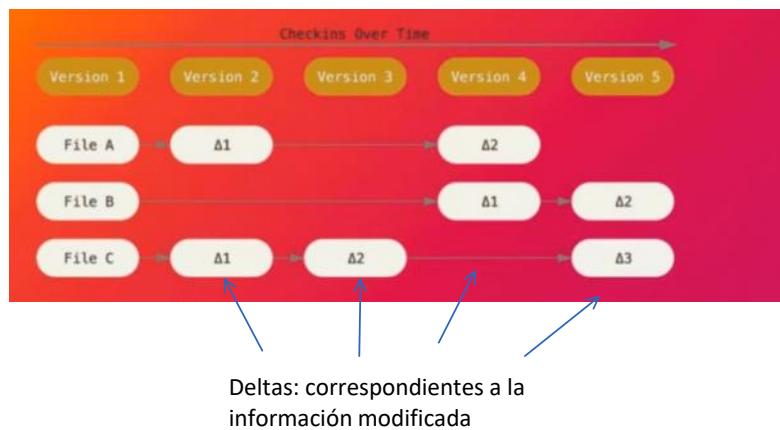


Otros sistemas SVC

sábado, 20 de enero de 2018 17:35

Delta Storage Model

Almacenamiento en base a ficheros



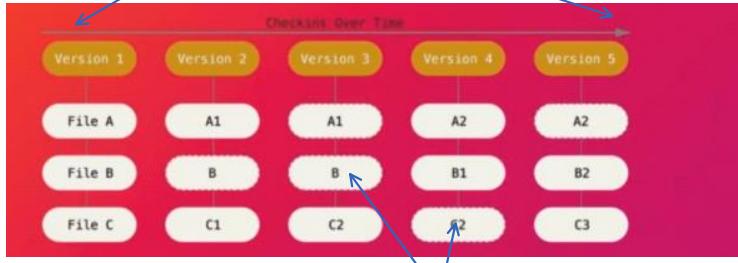
Professional Git
Brent Laster
John Wiley & Sons, 2016

GIT

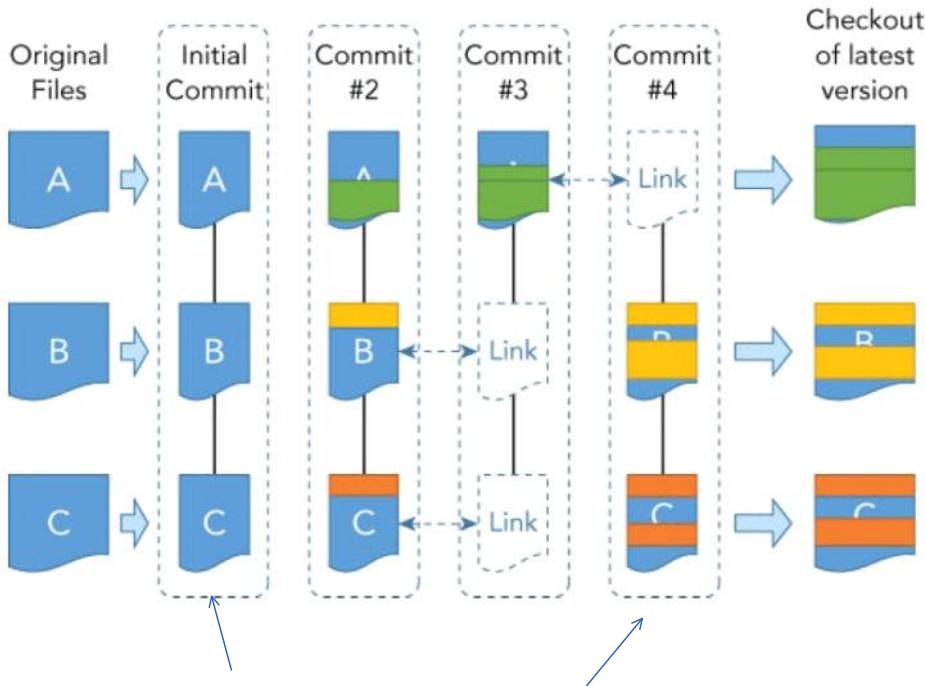
Git almacena *snapshots* (instantáneas) completas

Todos los procesos inicialmente son locales

OTROS SVC: *Delta Storage Model*



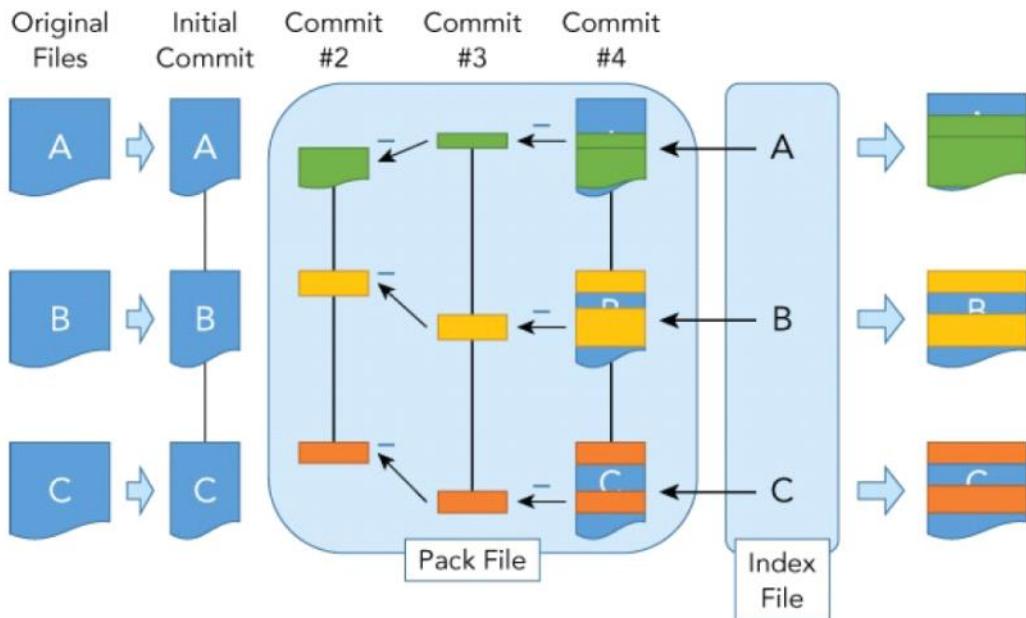
si no hay cambios se almacena un puntero al fichero sin cambios



Cada instantánea incluye toda la información de los ficheros a los que se refiere, con los últimos cambios producidos y un enlace a los ficheros que no han cambiado

Empaqueado

Periódicamente, en ciertos momentos (*trigger points*), como cuando se ejecuta la funcionalidad de recolección de elementos no utilizados (*garbage collection*), Git busca contenido que sea muy similar entre revisiones y empaqueta esas revisiones juntas para formar un archivo de paquete comprimido (*compressed pack file*)



Desde su origen, este modelo está pensado para versionar el código, es decir ficheros de un tamaño bastante limitado. Posteriormente se han desarrollado extensiones que mejoran el tratamiento de ficheros muy grandes (imágenes, audio, video...) como son

- git-annex (<https://git-annex.branchable.com/>)
- Git Large File Storage (Git LFS) (<https://git-lfs.github.com/>)

Git Large File Storage

Docs Downloads Source

An open source Git extension for versioning large files

Git Large File Storage (LFS) replaces large files such as audio samples, videos, datasets, and graphics with text pointers inside Git, while storing the file contents on a remote server like GitHub.com or GitHub Enterprise.

Download v2.3.4 (Windows)

Remote

Local

Large File Storage

file.psd

GitHub.com support now available. [Install the client to get started.](#)

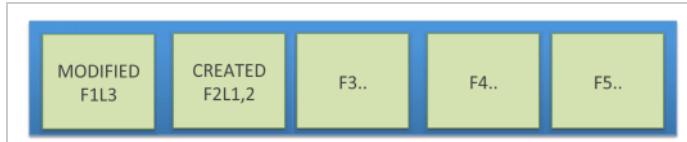
Commits

miércoles, 27 de diciembre de 2017 20:00

Para registrar información sobre los cambios en el repositorio se utilizan **commits** → registros de la DB

Contenido de un *commit*:

- el *changeset* (i.e. los cambios) de cada uno de los ficheros afectados
- punteros a todos los ficheros que no han cambiado



Un *commit* es el único método de introducir cambios en un repositorio.

Los *commits* generalmente son generados de forma explícita por el usuario pero también pueden ser generados por Git, como sucede en el caso del *merge*.

Un *commit* se hace visible con el comando git show

Carácter transaccional de los *commits*

Cada *commit* en Git representa un conjunto de cambios (*change set*) único y → DB transaccionales

Se aplican todos los cambios incluidos en un *commit* o ninguno (atomicidad)

Se puede estar seguro de que Git no deja un repositorio en un estado transitorio entre la instantánea de un *commit* y la del siguiente.

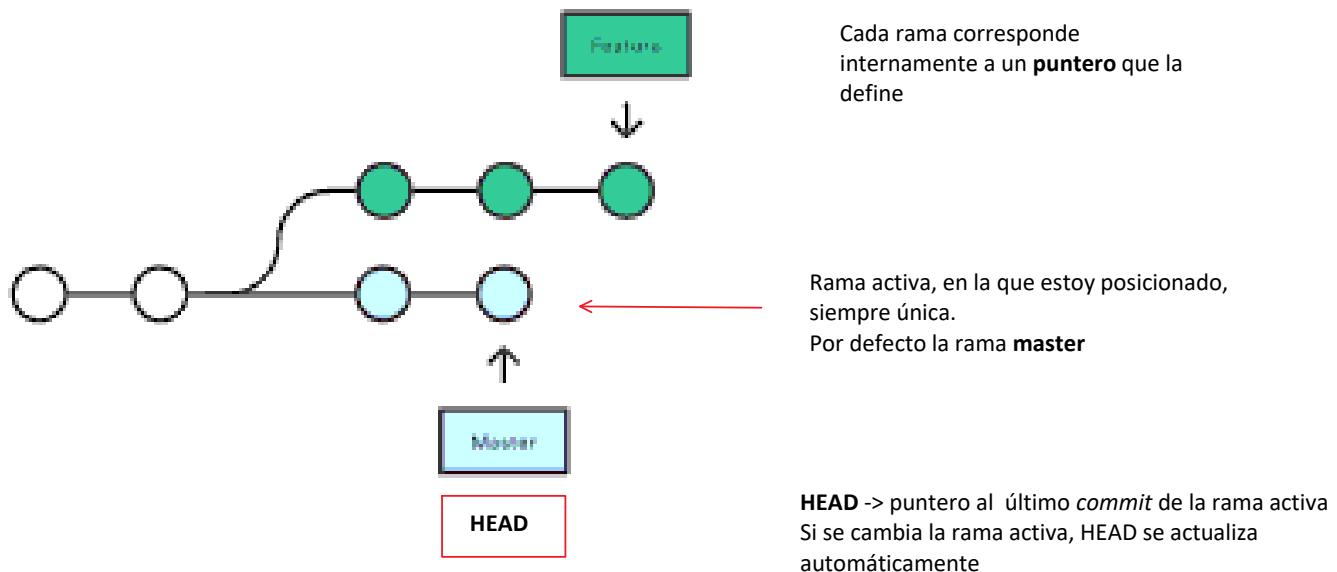
Cada *commit* tiene un *checksum* que garantiza la integridad de la información que contiene

Punteros: ramas y HEAD

lunes, 8 de enero de 2018 16:02

Los *commits* se organizan en secuencias temporales conocidas como ramas (*branch*)

Git internamente se basa en punteros



Cuando se añade un *commit*, HEAD cambia para seguir apuntando al último *commit* de la rama activa

Cuando se cambia a una rama diferente, HEAD cambia para seguir apuntando al último *commit* de la nueva rama.

Desde el S.O es posible comprobar como de produce este desplazamiento de punteros

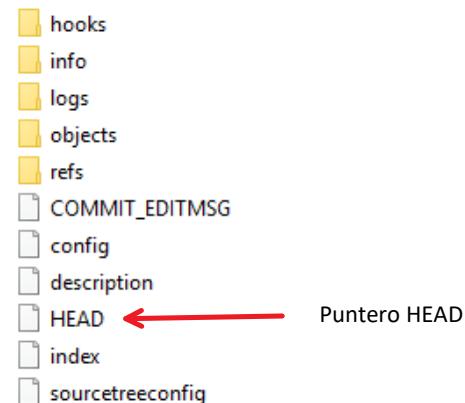
En *bash-shell*

```
$ cd .git  
$ cat HEAD -> ref: refs/heads/master  
$ git checkout -b newFeature  
$ cat HEAD -> ref: refs/heads/newFeature
```

Estructura de un repositorio (.git)

En Windows

```
cd .git  
type HEAD -> ref: refs/heads/master  
git checkout -b newFeature  
type HEAD -> ref: refs/heads/newFeature
```



Otras referencias

lunes, 22 de enero de 2018 0:57

Ciertas operaciones, como *merge* y *reset*, guardan la versión previa de HEAD como ORIG_HEAD justo antes de asignarle un nuevo valor a la primera.

El valor ORIG_HEAD puede usarse para revertir una operación recuperando el estado previo o para realizar las comparaciones necesarias.

Los nombres de las ramas, los de las ramas remotas o los tags son todos referencias.

En algunos casos se distinguen las referencias simbólicas (*symrefs*), como aquellas que apuntan indirectamente a los objetos en Git, pero en definitiva no dejan de ser referencias

Comandos

martes, 25 de abril de 2017 0:16

El diseño de Git se corresponde con la orientación a caja de herramientas, donde el comando git "contiene" un conjunto de subcomandos para las distintas tareas.

git <git-options> <command> <command-options> <operands>

Ejemplo git commit

Git está implementado como un conjunto de programas y scripts de *shell* que son fácilmente encadenables para formar nuevos comandos.

Además, Git cuenta con mecanismos para lanzar scripts de usuario cuando suceden ciertos eventos en el flujo de trabajo denominados puntos de enganche (*hooks*).

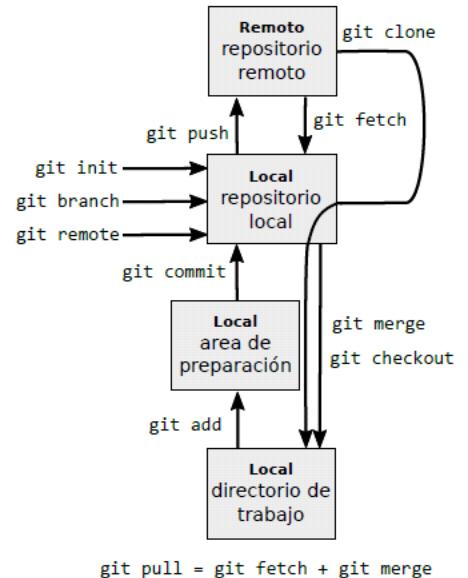
Resumen de comandos orientados al usuario
(*user friendly o Porcelain Commands*)

git init - crear un repositorio local en un directorio.
git clone - crear un repositorio local haciendo una copia de otro (local o remoto)
git add. - registra los ficheros del directorio de trabajo cuyos cambios se quieren
git commit - confirma los cambios de los ficheros registrados
git remote - configura los repositorios remotos

git branch - crear y destruir ramas
git checkout - permite cambiar de rama en el directorio de trabajo (Por defecto se trabaja en la rama denominada master)
git push - enviar los cambios a un repositorio remoto en la rama indicada
git pull - actualizar el repositorio con los cambios de un repositorio local

Este comando es esencialmente un encadenamiento de dos comandos:

git fetch - obtiene los cambios de una rama remota
git merge - fusiona si es posible estos cambios con una rama local.



Otros comandos

bisect
cherry
cherry-pick
config
diff
grep
help
log
mv
rebase
rerere
reset
revert
rm

*show
status
submodule
subtree
tag
worktree*

Mucho menos habituales son los comandos de bajo nivel (*Plumbing Commands*)

*cat-file
commit-tree
count-objects
diff-index
for-each-ref
hash-object
ls-files
merge-base
read-tree
rev-list
rev-parse
show-ref
symbolic-ref
update-index
update-ref
verify-pack
write-tree*

```
~\Documents> git --help
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p] [--paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  reset     Reset current HEAD to the specified state
  rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect    Use binary search to find the commit that introduced a bug
  grep      Print lines matching a pattern
  log       Show commit logs
  show      Show various types of objects
  status    Show the working tree status

grow, mark and tweak your common history
  branch   List, create, or delete branches
  checkout Switch branches or restore working tree files
  commit   Record changes to the repository
  diff     Show changes between commits, commit and working tree, etc
  merge   Join two or more development histories together
  rebase   Reapply commits on top of another base tip
  tag     Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch   Download objects and refs from another repository
  pull    Fetch from and integrate with another repository or a local branch
  push    Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
```

Función de autocompletado (`<TAB>`)

El doble guion --

Tiene una doble funcionalidad:

Separar opciones de una lista de argumentos
`$ git diff -w master origin -- tools/Makefile`

Separar un nombre de archivo explícitamente identificado

`# Checkout the tag named "main.c"`
`$ git checkout main.c`

`#Checkout the file named "main.c"`
`$ git checkout -- main.c`

Ejercicio. Primer repositorio

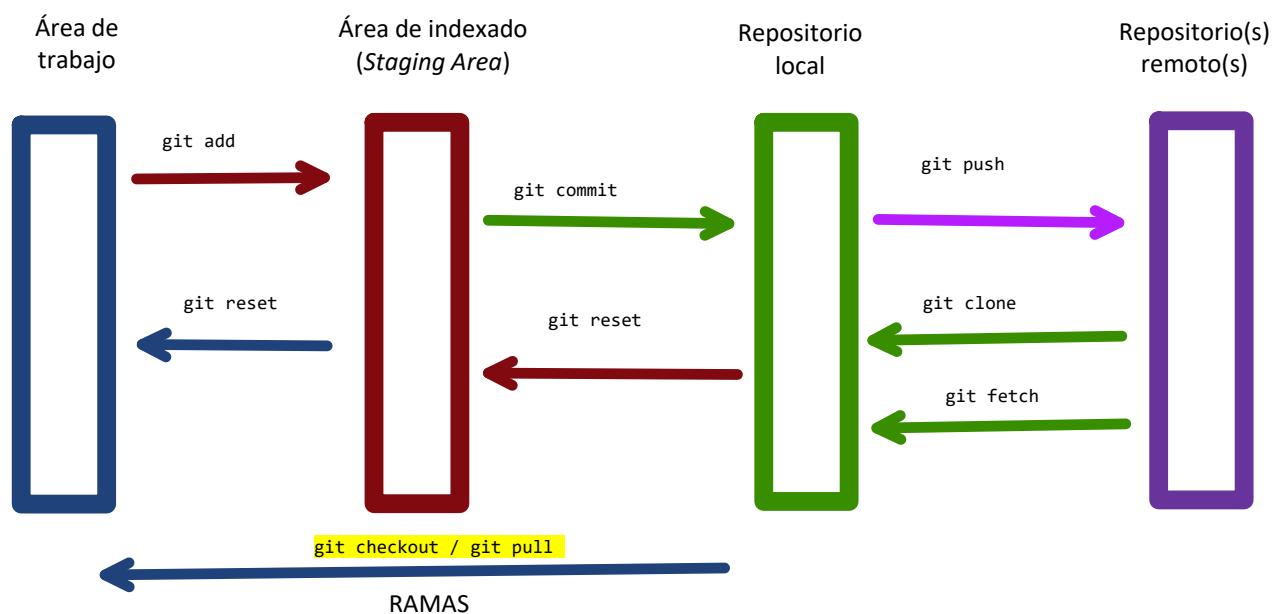
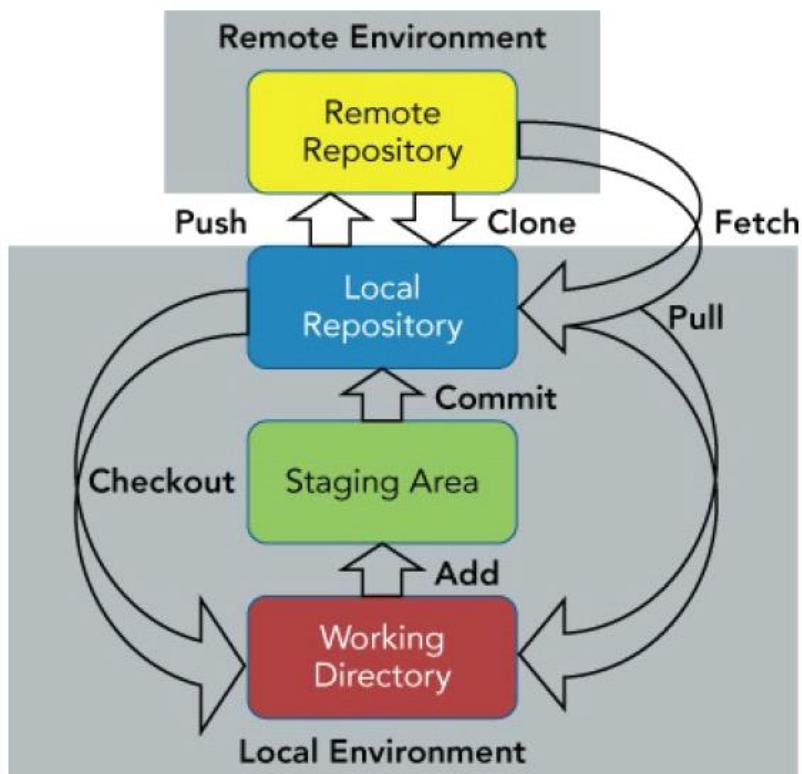
martes, 26 de diciembre de 2017 21:01

1. Se crea la carpeta raíz del proyecto, en la que se creará el repositorio
2. Se crea / inicializa el repositorio con el comando
`git init`

```
D:\desarrollo\Gits> cd .\Git_Basicc\  
D:\desarrollo\Gits\Git_Basicc> dir  
D:\desarrollo\Gits\Git_Basicc> git init  
Initialized empty Git repository in D:/desarrollo/Gits/Git_Basicc/.git/  
D:\desarrollo\Gits\Git_Basicc [master]>
```

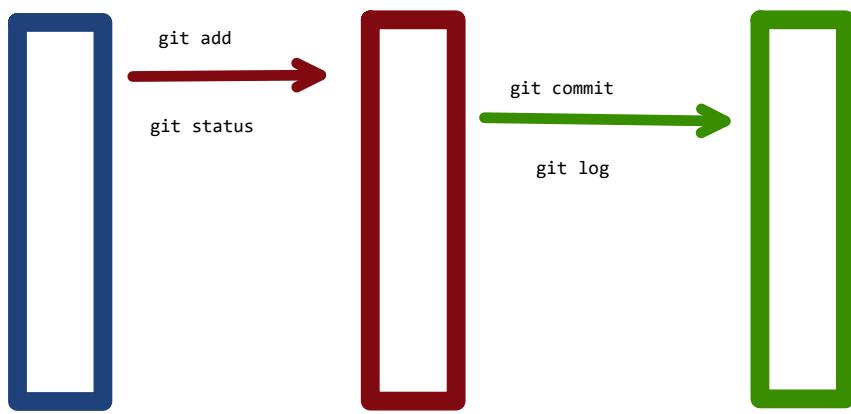
Flujos básicos

sábado, 20 de enero de 2018 18:22



Primeros comandos

Área de trabajo Área de indexado
(Staging Area)



Ejercicio: Primer commit

lunes, 8 de enero de 2018 20:48

```
D:\desarrollo\Gits\Git_Basic [master]> echo reaadme > README.md  
D:\desarrollo\Gits\Git_Basic [master +1 ~0 -0 !]> dir
```

```
Directorio: D:\desarrollo\Gits\Git_Basic  
  
Mode LastWriteTime Length Name  
---- ----- ---- -  
-a--- 04/01/2018 18:31 20 README.md
```

Buena práctica: que este fichero del primer *commit* sea el [README.md](#)

```
D:\desarrollo\Gits\Git_Basic [master +1 ~0 -0 !]> git add README.md  
D:\desarrollo\Gits\Git_Basic [master +1 ~0 -0 ~]> git status  
On branch master
```

No commits yet

```
Changes to be committed:  
(use "git rm --cached <file>..." to unstage)  
  
 new file: README.md
```

```
D:\desarrollo\Gits\Git_Basic [master +1 ~0 -0 ~]> git commit -m "Readme inicial"  
[master (root-commit) e4b3552] Readme inicial  
 1 file changed, 0 insertions(+), 0 deletions(-)  
 create mode 100644 README.md
```

```
D:\desarrollo\Gits\Git_Basic [master]> git log  
commit 833329b3f9bc80d44dc49931681d9f2f11906fd5 (HEAD -> master)  
Author: alce65 <alce65@hotmail.es>  
Date: Thu Jan 4 18:35:23 2018 +0100
```

```
 Readme inicial  
D:\desarrollo\Gits\Git_Basic [master]> git log --oneline  
833329b (HEAD -> master) Readme inicial  
D:\desarrollo\Gits\Git_Basic [master]> -
```

```
D:\desarrollo\Gits\Git_Basic [master]> git status  
On branch master  
nothing to commit, working tree clean  
D:\desarrollo\Gits\Git_Basic [master]> -
```

Para todos los ficheros de la *workarea*:
git add .

git status

git commit

Hash: identificador único de cada *commit*
Sus 7 primeros dígitos es la versión
abreviada del identificador

git log --oneline
desencadena una versión compacta del
histórico

Elementos de un repositorio

domingo, 21 de enero de 2018 9:52

- carpeta git
- Buenas prácticas para el primer commit
 - Fichero Readme
 - Fichero gitignore

La carpeta .git

lunes, 8 de enero de 2018 21:45

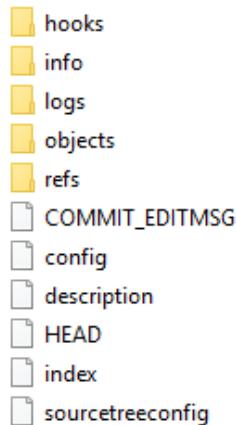
Comando : git init

```
D:\Desarrollo\Tools\GItRepo>git init
Initialized empty Git repository in D:/Desarrollo/Tools/GItRepo/.git/ ← Creación del
repository

D:\Desarrollo\Tools\GItRepo>dir /ad
El volumen de la unidad D es Data
El número de serie del volumen es: 10AD-0AF6

Directorio de D:\Desarrollo\Tools\GItRepo

28/10/2017 13:39 <DIR> .
28/10/2017 13:39 <DIR> ..
28/10/2017 13:39 <DIR> .git ← Carpeta oculta
                                0 archivos          0 bytes
                                3 dirs 197.347.762.176 bytes libres
```



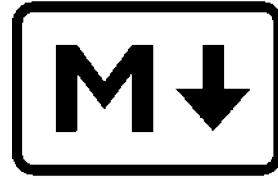
Fichero Readme

lunes, 8 de enero de 2018 20:45

Fichero de texto en formato *Markdown*

Contiene información descriptiva, instrucciones de uso, el historial de versiones

Se considera una buena práctica incluirlo en los repositorios, siendo prácticamente imprescindible en caso de repositorios compartidos, especialmente si se trata de proyectos de código libre / abierto o algún otro tipo de repositorio público.



Markdown

<https://es.wikipedia.org/wiki/Markdown>

The screenshot shows the ReText application interface. The title bar reads "Markdown syntax.md — ReText". The menu bar includes "File", "Edit", and "Help". Below the menu is a toolbar with icons for new file, open, save, preview, and other document operations. The main window has tabs for "New document" and "Markdown syntax". The left pane displays the raw Markdown code:

```
1 ## Some Markdown syntax examples
2
3 Welcome to ReText!
4
5 Here are some examples of Markdown syntax that you can use with
6 ReText.
7 To learn more, look at the [official documentation](http://
daringfireball.net/projects/markdown/syntax).
8
9 ### Basic formatting
10
11 **Bold text** | *Text in italics* | [Link](https://github.com/
retext-project/retext)
12
13 ### Bullet list
14
15 * Item one
16 * Item two
17
18 ### Ordered list
19
20 1. Item one
21 2. Item two
22
23 You can also use <u>some</u> HTML tags in your documents.
24
25 Happy editing!
```

The right pane contains rendered content corresponding to the code:

Some Markdown syntax examples

Welcome to ReText!

Here are some examples of Markdown syntax that you can use with ReText.

To learn more, look at the [official documentation](#).

Basic formatting

Bold text | **Text in italics** | [Link](#)

Bullet list

- Item one
- Item two

Ordered list

1. Item one
2. Item two

You can also use some HTML tags in your documents.

Happy editing!

Markdown en GitLab

jueves, 4 de enero de 2018 20:35



Motor diferente al que utiliza GitHub

- ▷ Newlines: dos saltos de linea (mínimo)
- ▷ Italic: underscore
- ▷ URL link: automático (con http, https, ftp)
- ▷ Quote multiline: entre >>>
- ▷ Code: con 'código' o con ``` para multilinea (también ``php)
- ▷ Inline diff: con {+ additions} o [- additions]
- ▷ Emoji: con :emoji: (ejemplo :zap:)
- ▷ @username para mencionar
- ▷ #123 para issue
- ▷ !123 para merge request
- ▷ \$123 para snippet
- ▷ ~123 para label (id)
- ▷ |README|doc/README referencia
- ▷ Task lists...

Lo vemos: <https://docs.gitlab.com/ee/user/markdown.html>

Fichero *GitIgnore*

martes, 26 de diciembre de 2017 21:02

Un archivo *gitignore* especifica intencionalmente archivos sin seguimiento (untracked) que git debe ignorar.

Los archivos ya rastreados (*tracked*) por Git no se ven afectados.

Debe estar situado en la carpeta raíz.

Para forzar que se incluya un fichero descartado en *gitignore* se puede usar
git add -f <ficheros>

Patrones

#: Comentarios
!: Negación del siguiente patrón
algo/: Directorio con el nombre indicado (no ficheros con ese nombre)
*: cualquier valor

.html , !foo.html, /.js

Ejemplos de *gitignore* para diversos lenguajes

<https://github.com/github/gitignore>

Se crea *gitignore* y se incorpora al repositorio

```
D:\desarrollo\Gits\Git_Basic [master +1 ~1 -0 !]> echo gitignore > .gitignore
D:\desarrollo\Gits\Git_Basic [master +1 ~1 -0 !]> code .
D:\desarrollo\Gits\Git_Basic [master +1 ~1 -0 !]> git status
```

```
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    .gitignore

nothing added to commit but untracked files present (use "git add" to track)
```

```
D:\desarrollo\Gits\Git_Basic [master +1 ~0 -0 ~]> git add .
D:\desarrollo\Gits\Git_Basic [master +1 ~0 -0 ~]> git commit -m "Creado gitignore"
[master cc925b5] Creado gitignore
 1 file changed, 0 insertions(+), 0 deletions(-)
  create mode 100644 .gitignore
D:\desarrollo\Gits\Git_Basic [master]> git status
On branch master
nothing to commit, working tree clean
```

```
1 # ejemplo de gitignore
2 *.class
3 node_modules/
```

Puede ser necesario crearlo con el editor de código

Commit del propio *.gitignore*

Funcionamiento de *gitignore*

```
D:\desarrollo\Gits\Git_Basic [master]> echo clase > clase.class
D:\desarrollo\Gits\Git_Basic [master]> dir

    Directorio: D:\desarrollo\Gits\Git_Basic

Mode           LastWriteTime       Length Name
<-----          -----          ----- 
d----  04/01/2018  18:46            mapas
-a---  05/01/2018  9:16           107 .gitignore
-a---  05/01/2018  9:19            16 clase.class
-a---  05/01/2018  8:31           168 index.js
-a---  04/01/2018  18:40            18 README.md

D:\desarrollo\Gits\Git_Basic [master]> git status
On branch master
nothing to commit, working tree clean
```

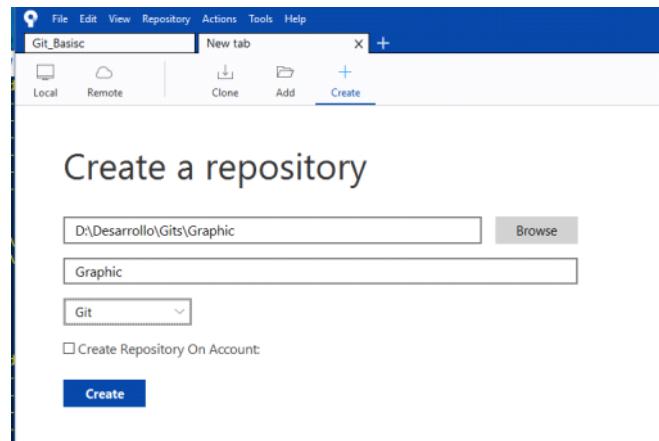
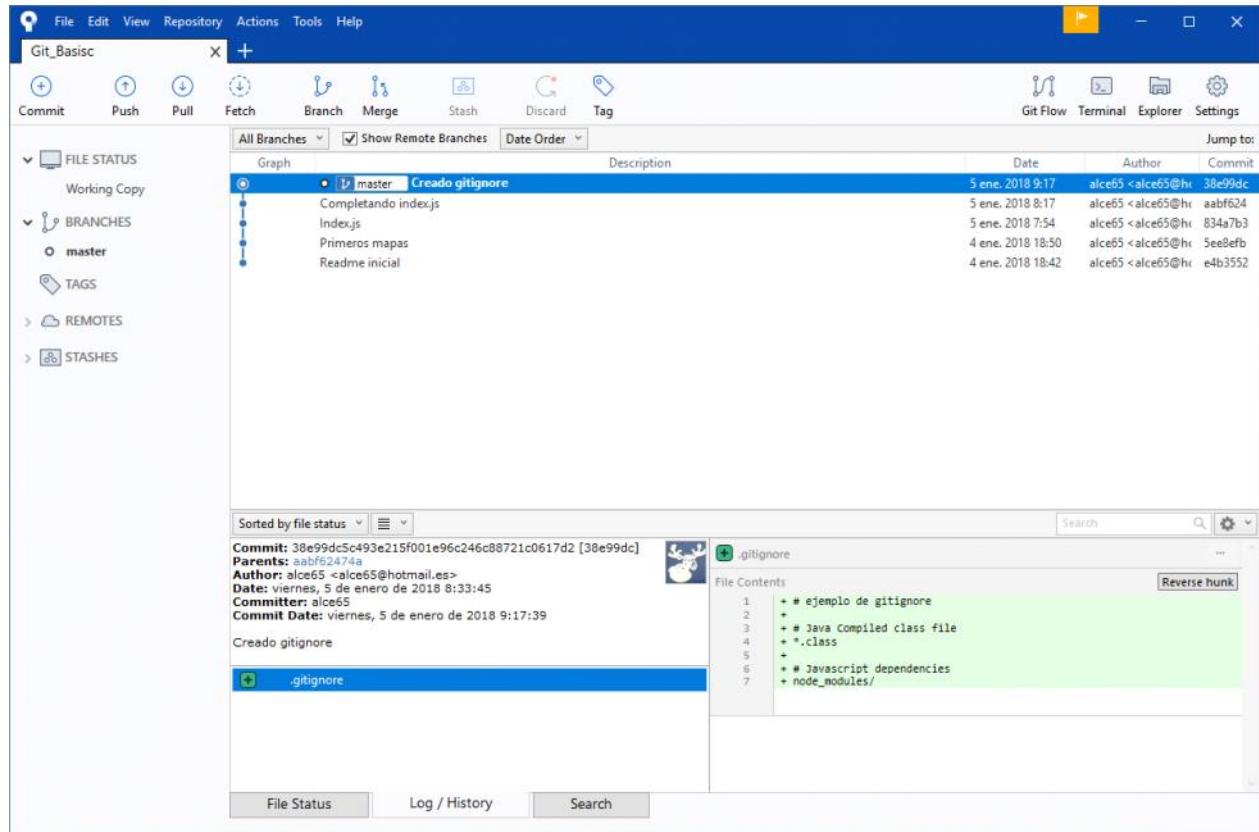
Se crea un fichero .class que no aparece como existente en la *working area* -> es completamente ignorado por el repositorio

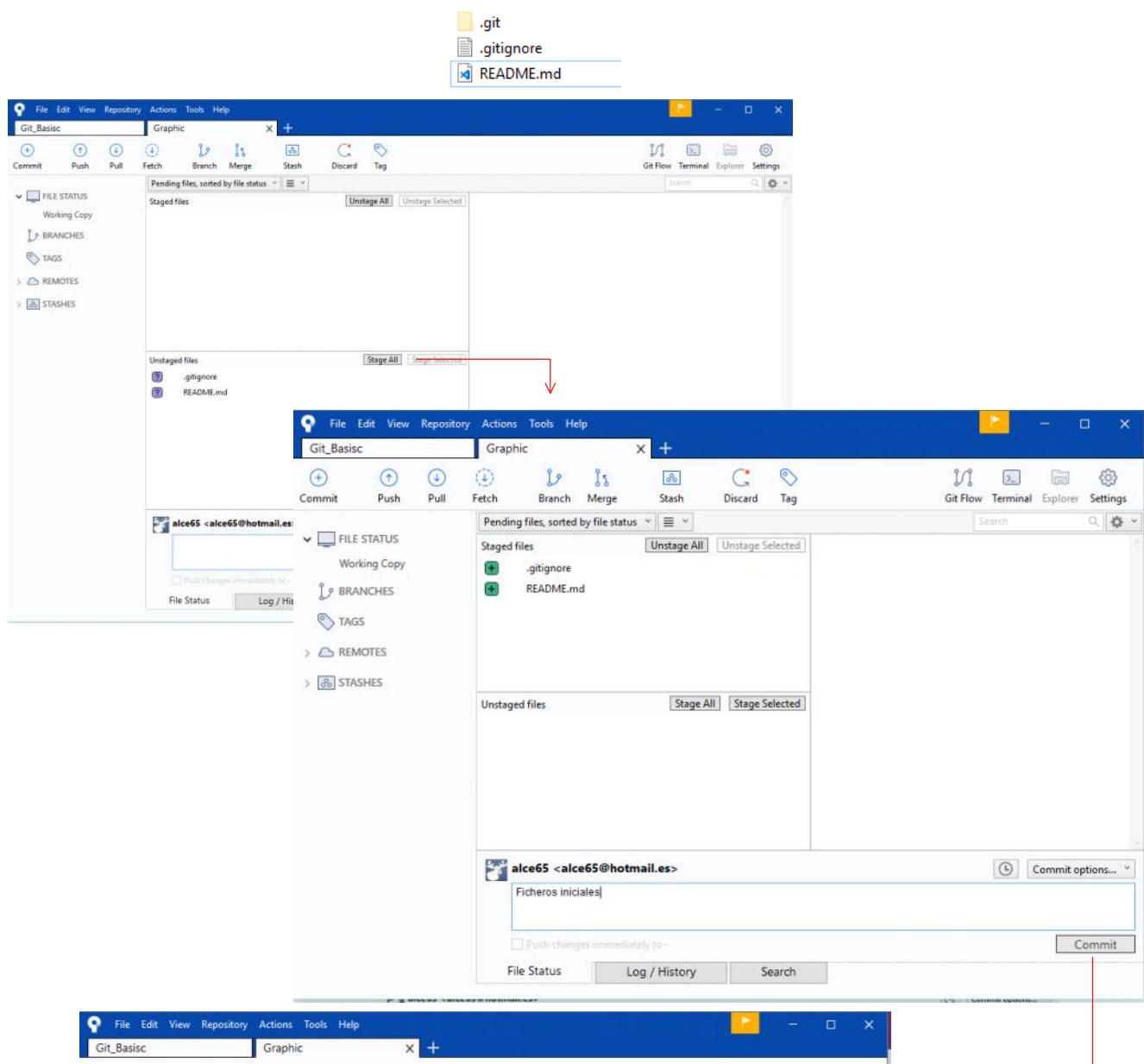
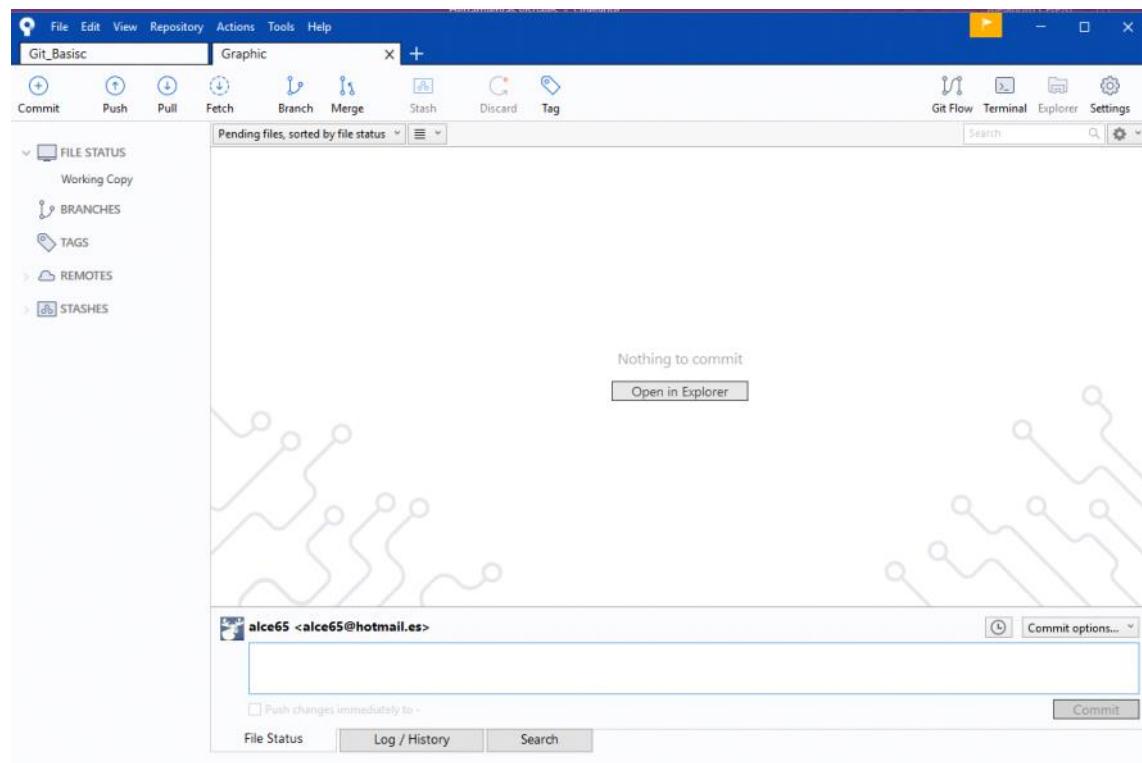
Herramientas visuales

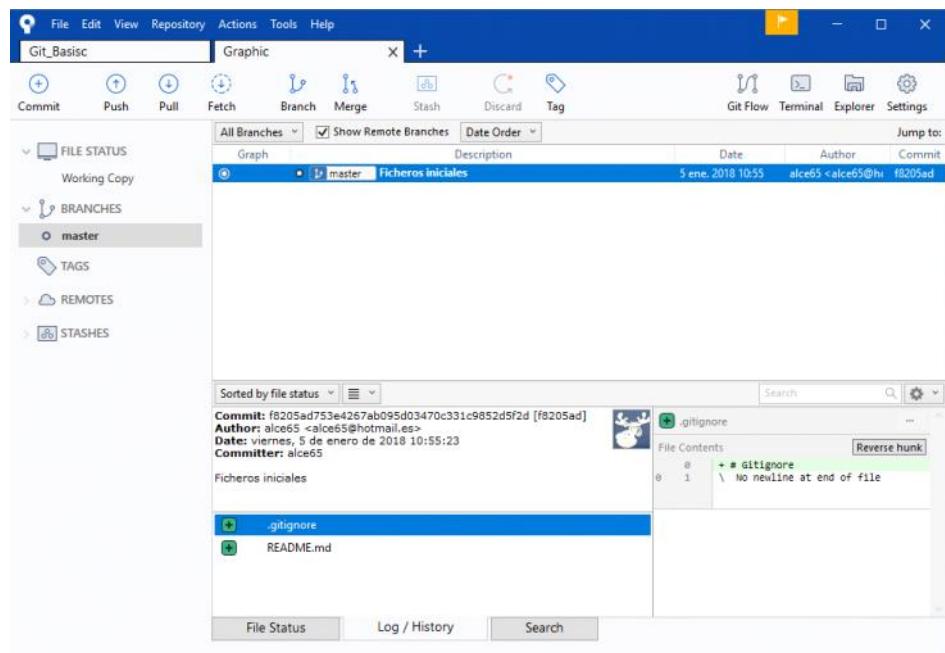
viernes, 5 de enero de 2018 10:41

Visualmente se pueden utilizar diversas herramientas, como *SourceTree*

<https://www.sourcetreeapp.com/>

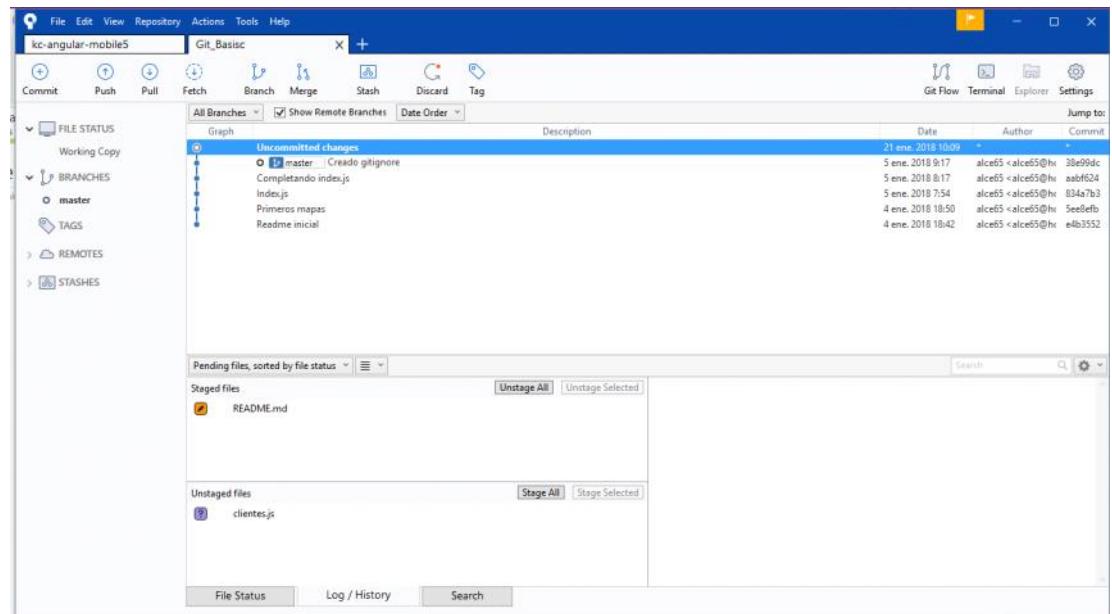






Ejercicio en SourceTree

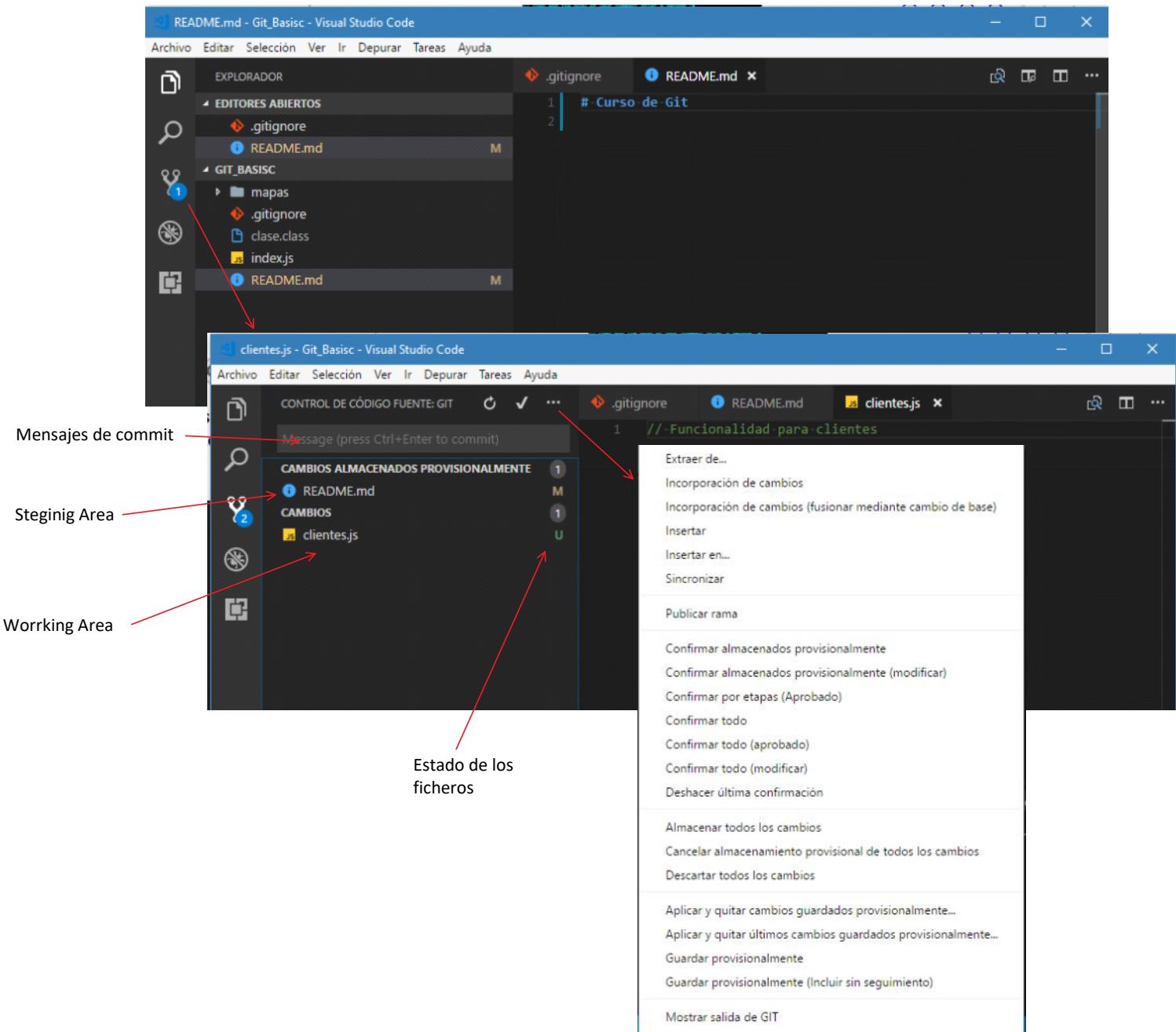
domingo, 21 de enero de 2018 9:54



Ejercicio en Visual Studio Code

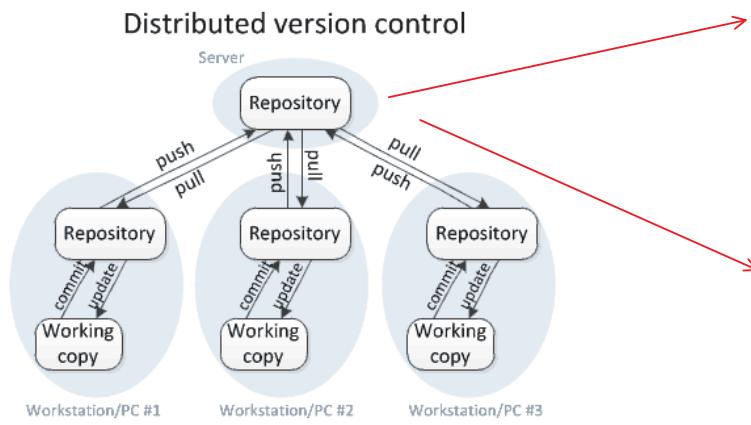
lunes, 8 de enero de 2018 23:31

Git Integrado



Introducción a GIT en el lado servidor

martes, 9 de enero de 2018 0:41



Servidores "propios"

- *gitolite* <http://gitolite.com/gitolite/index.html>
- *gitosis* <https://git-scm.com/book/es/v1/Git-en-un-servidor-Gitosis>
- *GitLab*, para Linux (interfaz Web,) <https://about.gitlab.com/equipos>

Hosting especializado de repositorios

- GitHub - <https://github.com/>
- Bitbucket <https://bitbucket.org/>
- GitLab <https://about.gitlab.com/>

• Repostorios *Bare*

miércoles, 3 de enero de 2018 13:37

Hay dos tipos diferentes de repositorios Git:

- *Bare* (desnudo, simple): no tiene directorio de trabajo. No se usa para el desarrollo directamente en el desarrollo de aplicaciones ya que en él no se pueden realizar directamente *commits*. De este tipo son los repositorios que se publican, en servidores estándar o especializados en el alojamiento (hosting) de repositorios
- Desarrollo: repositorio típico. Mantiene la rama en la que se trabaja, proporciona una copia extraída de dicha rama en el directorio de trabajo

Un repositorio *Bare* es crucial para el desarrollo colaborativo: otros desarrolladores clonian y recuperan de estos repositorios y los actualizan a partir de sus repositorios de desarrollo

Creación de un repositorio *Bare*

```
$ git init <repository> --bare
```

En el servidor donde se almacenan los repositorios *bare* son especialmente importantes algunos aspectos como

- la seguridad (por lo general basada en SSH)
- la gestión de las cuentas de usuarios

Aplicaciones

domingo, 21 de enero de 2018 12:21

gitolite

<http://gitolite.com/gitolite/index.html>

UNIX

The screenshot shows the official Gitolite website at <http://gitolite.com/gitolite/index.html>. The page title is "Hosting Git Repositories". The left sidebar contains links for "Hosting Git Repositories", "install/setup", "documentation", "TROUBLESHOOTING", "contact/support", "security issues", "mailing list(s)", "IRC", and "license". The main content area starts with a brief introduction: "Gitolite allows you to setup git hosting on a central server, with fine-grained access control and many more powerful features." Below this is a section titled "install/setup" with a note about source code and installation instructions. A sidebar on the right provides a tip for package managers. A callout box on the right side contains a message from April 2014 about a book on gitolite.

gitosis

<https://git-scm.com/book/es/v1/Git-en-un-servidor-Gitosis>

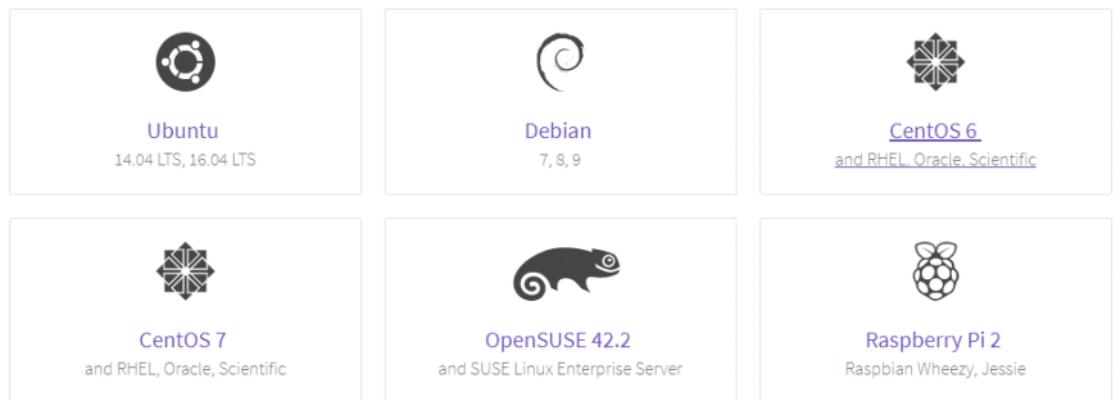
Es básicamente un conjunto de scripts que te ayudarán a gestionar el archivo 'authorized_keys', así como a implementar algunos controles de acceso simples.

instalación no demasiado simple
necesita de ciertas herramientas Python
se instala clonando el correspondiente repositorio Git

Versiones para diversas distribuciones Linux
Dotadas de un potente interfaz Web, como el del hosting de GitLab

<https://about.gitlab.com>

Omnibus package installation (recommended)





Ejemplo de hosting de repositorios Git
Puede actuar como remoto de cualquier repositorio Git

<https://github.com/>

Posibilidades del acceso anónimo

- Descarga / clonación de proyectos de software libre
- Perfil de los programadores (uso social / laboral)

Extensa documentación

<https://help.github.com/>

GitHub Help

Version ▾ Contact Support Return to GitHub

Sometimes you just need a little help.

How can we help?



Bootcamp

- › Set Up Git
- › Create A Repo
- › Fork A Repo
- › Be Social

Setup

- › Signing up for a new GitHub account
- › Verifying your email address
- › About commit email addresses
- › Setting your commit email address on GitHub
- › Setting your commit email address in Git
- › Blocking command line pushes that expose your personal email address
- › Setting your username in Git
- › Dealing with line endings
- › Supported browsers

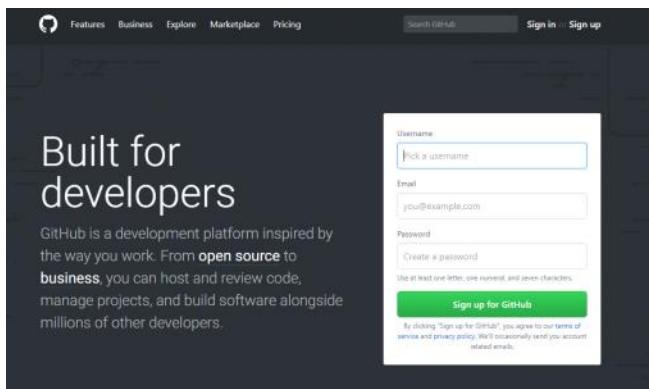
About GitHub

- › GitHub Glossary
- › Git and GitHub learning resources
- › Differences between user and organization accounts

Usuarios

domingo, 21 de enero de 2018 12:10

Alta de usuario



Perfil de usuario

GitHub Social

A screenshot of a GitHub user profile page for the user "alce65". The top navigation bar includes the GitHub logo, a search bar, and links for Pull requests, Issues, Marketplace, and Explore. Below the navigation, there is a profile picture and the user's name "alce65". The main content area shows the user's activity statistics: Overview, Repositories 33, Stars 2, Followers 29, and Following 9. A red box highlights the "Following 9" link. Below this, there is a section for "Popular repositories" featuring repositories like "Curso_Malaga_2" and "Angular_IT". Further down, there is a "GitHub Social" section with a "Add a bio" form containing fields for location ("Madrid, Spain") and email ("alce65@hotmail.es"). Red arrows point from the "Edit profile" button in the bio form to the "Edit profile" button in the GitHub Social form, and from the "Edit profile" button in the GitHub Social form to the "Edit profile" button in the GitHub header. The bottom of the page shows a heatmap titled "388 contributions in the last year" with a grid of colored squares representing contribution density by month and day.

Personal settings
Profile
Account
Emails
Notifications
Billing
SSH and GPG keys
Security
Blocked users
Repositories
Organizations
Saved replies
Applications
Developer settings
Organization settings
 Curso-Web

Public profile

Name

Public email

You can manage verified email addresses in your [email settings](#).

Bio

You can @mention other users and organizations to link to them.

URL

Company

You can @mention your company's GitHub organization to link it.

Location

Profile picture



[Upload new picture](#)

[Update profile](#)

Repositorios en GitHub

Lunes, 8 de enero de 2018 21:23

The screenshot shows the GitHub interface for creating a new repository. At the top, there are navigation links: Overview, Repositories 33 (which is highlighted in blue), Stars 2, Followers 29, and Following 9. Below these are search fields for 'Search repositories...', 'Type: All', 'Language: All', and a green 'New' button. The main section is titled 'Create a new repository' with the sub-instruction: 'A repository contains all the files for your project, including the revision history.' A red arrow points from the first list item to the 'Repository name' input field, which is currently empty and highlighted with a blue border. Another red arrow points from the second list item to the 'Create repository' button at the bottom.

Owner: alce65 / Repository name:

Great repository names are short and memorable. Need inspiration? How about [bug-free-broccoli](#).

Description (optional):

Public Anyone can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None | ⓘ

Create repository

- a. Solo se indica el nombre (y opcionalmente la descripción)

- b. Se Inicializa el repositorio con readme, gitignore y si procede información sobre la licencia

Enlace y clonado

Lunes, 8 de enero de 2018 21:11

Subir un repositorio a GitHub

Un **repositorio local**, después de ser creado puede ser **enlazado** con un **remoto vacío** (sin inicializar) para después por subirlo o "empujarlo" (*push*) siempre que sea necesario

```
git init  
...  
git remote add origin <url>  
...  
git push origin master
```

Las URL de los repositorios *bare* almacenados en GitHub terminan en .git
<https://github.com/alce65/GitRepo.git>

Añade una referencia al repositorio remoto identificada por un alias (*shorthand*), en este caso **origin**

Se puede definir cualquier número de remotos, cada uno con su alias o identificador propio

Podemos ver el resultado con el comando
git remote -v

Clonar un repositorio de GitHub

Un repositorio remoto, normalmente inicializado -> clonado a local

```
git clone <remote-url>
```

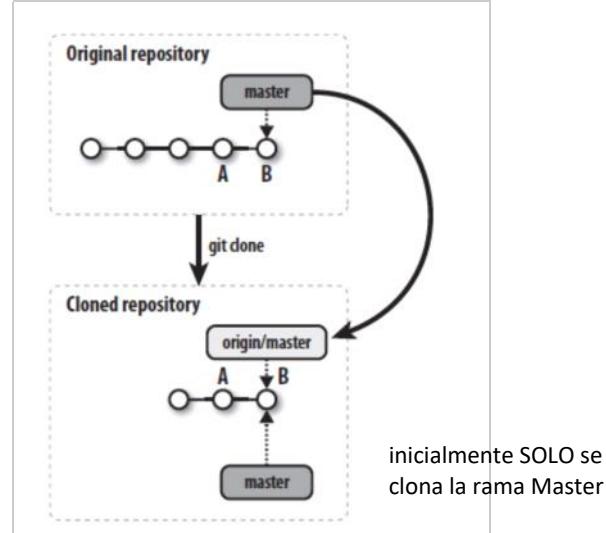
Crea el repositorio local con todo el contenido y le añade con el alias por defecto **origin** la url del remoto

El clonado se utiliza en 2 situaciones

- con nuestros propios repositorio
- con cualquier repositorio público

En este segundo caso, si no se dispone de permisos, no será posible subir nuevo contenido al remoto.

Por contra, para realizar esta operación ni siquiera es necesario tener sesión iniciada en GitHub ni tener una cuenta.



Un repositorio clonado queda vinculado al que lo origino:

- no reflejará automáticamente los cambios en el remoto
- podrá ser sincronizado mediante los comandos adecuados (*pull* - *fetch* - *merge*)

Ejercicios. Clonado

domingo, 21 de enero de 2018 20:07

Ejercicio.

- Creamos un remoto y lo enlazamos con el repositorio local que tenemos
- Creamos un repositorio en GitHub y lo clonamos en local

Comprobamos que el resultado es prácticamente el mismo

Desde Local a Remoto

- a. Creación del repositorio local
`git init`
- b. Primer *commit* para inicializarlo
`git add / git commit`
- c. Creación de remoto en GitLab vacío
- d. Conexión entre ambos
`git remote add origin <url>`
- e. Sincronización
`git push -u origin master`

Desde remoto a local

- a. Creación de un repositorio en GitHub añadiéndole un primer *commit* con los elementos iniciales
- b. Clonado del repositorio remoto creando así el repositorio local
`git clone <url>`

Forks

domingo, 21 de enero de 2018 12:09

Fork: clonado desde otra cuenta de GitHub a nuestro propio usuario de GitHub.

No añade nada en local

Puede hacerse desde cualquier repositorio público

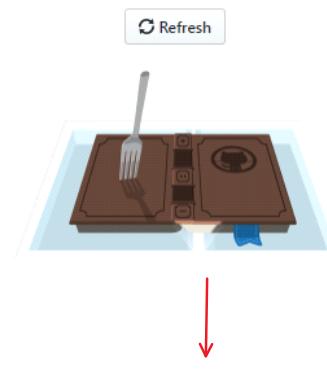
The screenshot shows the GitHub repository page for 'angular / angular'. At the top right, there is a 'Fork' button with a value of '7,957'. A red box highlights this button, and a red arrow points from it down to the forked repository page below.

Key elements visible on the page include:

- Repository name: angular / angular
- Watch: 2,850
- Star: 32,232
- Fork: 7,957 (highlighted)
- Issues: 1,813
- Pull requests: 309
- Projects: 9
- Insights
- Tags: angular, typescript, web, javascript, pwa, web-framework, web-performance
- Statistics: 9,352 commits, 30 branches, 196 releases, 570 contributors, MIT license
- Branch: master
- Create new file, Upload files, Find file, Clone or download

Forking miw-upm/IWVG

It should only take a few seconds.



The screenshot shows the forked repository 'alce65 / IWVG' on GitHub. It indicates that the repository was forked from 'miw-upm/IWVG'. Key details include:

alce65 / IWVG
forked from miw-upm/IWVG

Unwatch 1 Star 0 Fork 58

Code Pull requests 0 Projects 0 Wiki Insights Settings

No description, website, or topics provided. Edit

Add topics

50 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

This branch is even with miw-upm:master. Pull request Compare

setillo interpreter Latest commit b59dc11 on 8 Jul 2016

Commit	Message	Date
aco	Primera tanda de patrones	2 years ago
doo	interpreter	2 years ago

Actualizar remotos. Seguridad

domingo, 21 de enero de 2018 20:27

- Se puede clonar cualquier repositorio público
- Solo se pueden subir (*push*) actualizaciones a
 - los repositorios propios
 - en los que se está autorizado como contribuidor

Para garantizar la autenticación, los accesos a los repositorios emplean uno de los siguientes protocolos seguros

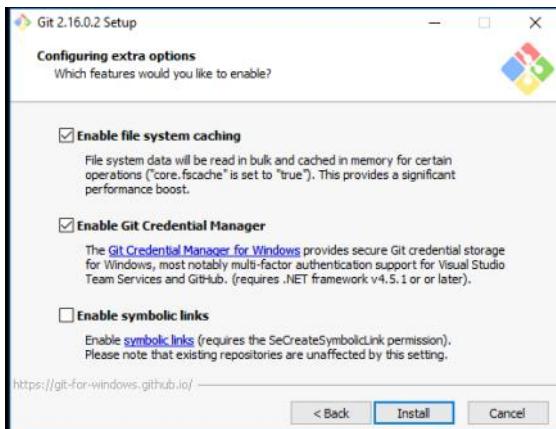
HTTPS:

- Habitual en entornos Windows
- Autenticación basada en usuario/clave

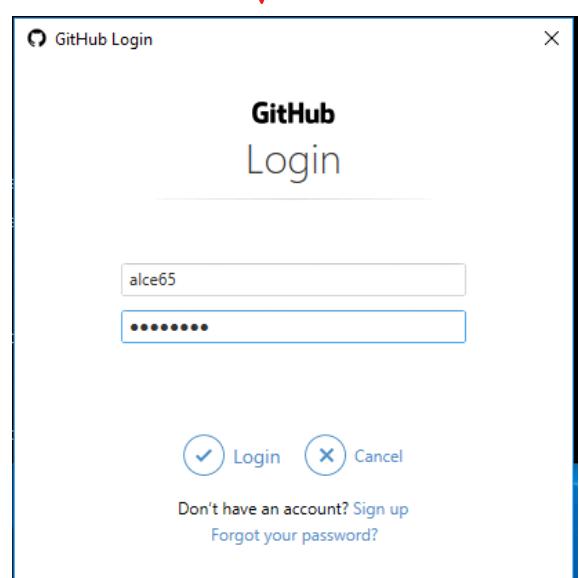
SSH (*Secure SHell*)

- Habitual en entornos UNIX/Linux
- Autenticación basada en diversas técnicas de cifrado, como la encriptación asimétrica y los hash

En la instalación



Al acceder por primera vez a una cuenta de GitHub para subir una actualización de cualquiera de sus repositorios



Panel de Control



Administrador de credenciales

Administrar credenciales

Vea y elimine su información de inicio de sesión guardada para sitios web, redes y aplicaciones conectadas.



Credenciales web



Credenciales de Windows

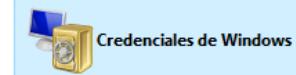
[Copia de seguridad de credenciales](#) [Restaurar credenciales](#)

Panel de Control

Administrador de credenciales

Administrar credenciales

Vea y elimine su información de inicio de sesión guardada para sitios web, redes y aplicaciones conectadas.



[Copia de seguridad de credenciales](#) [Restaurar credenciales](#)

Credenciales de Windows

[Agregar una credencial de Windows](#)

No hay credenciales de Windows.

Credenciales basadas en certificados

[Agregar una credencial basada en certificado](#)

No hay certificados.

Credenciales genéricas

[Agregar una credencial genérica](#)

MS.Outlook.15:alce65@outlook.com

Fecha de modificación: 21/11/2017

git:https://github.com

Fecha de modificación: Hoy

virtualapp/didlogical

Fecha de modificación: 21/11/2017

git:https://github.com

Fecha de modificación: Hoy

Dirección de red o Internet: git:https://github.com

Nombre de usuario: Personal Access Token

Contraseña:

Persistencia: Equipo local

[Editar](#) [Quitar](#)

Editar credencial genérica

Asegúrese de que el nombre de usuario y la contraseña que escriba se pueden usar para obtener acceso a la ubicación.

Dirección de red o Internet: git:https://github.com

Nombre de usuario:

Contraseña:

[Guardar](#)

[Cancelar](#)

Git Hub Social

domingo, 21 de enero de 2018 22:56

Actualizar remotos. Colaboraciones

domingo, 21 de enero de 2018 20:16

Se puede definir cualquier número de colaboradores autorizados a contribuir a un repositorio

The screenshot shows a GitHub repository page for 'alce65 / GitRepo'. At the top right, there are buttons for 'Unwatch' (with a count of 1), 'Star' (0), 'Fork' (0), and 'Edit'. Below these are navigation links: 'Code' (selected), 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Insights', and 'Settings'. The main content area displays the repository's details: 'Repository para prácticas con Git', 'Add topics', '1 commit', '1 branch', '0 releases', and '1 contributor'. A red box highlights the '1 contributor' link. Below this, there are buttons for 'Create new file', 'Upload files', 'Find file', and a green 'Clone or download' button. The file list shows 'alce65 Creado index.html' and 'index.html' with a timestamp of '3 months ago'. A red arrow points from the '1 contributor' link to the 'Contributors' section of the Insights tab.

Información sobre las contribuciones

The screenshot shows the 'alce65 / GitRepo' repository page with the 'Insights' tab selected. At the top right, there are buttons for 'Unwatch' (1), 'Star' (0), 'Fork' (0), and 'Edit'. Below these are navigation links: 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Insights' (selected), and 'Settings'. The main content area displays the title 'Oct 22, 2017 – Jan 21, 2018' and a chart titled 'Contributions to master, excluding merge commits'. The chart shows a single green bar peaking at approximately 1.0 on Oct 29, 2017. To the right, a 'Filter contributions' sidebar is open, showing options for 'Additions', 'Deletions', and 'Commits' (which is checked). A red arrow points from the 'Contributors' link in the left sidebar to the 'Filter contributions' sidebar.

En la configuración, se pueden añadir contribuidores

The screenshot shows the GitHub repository settings page for 'alce65 / GitRepo'. The 'Collaborators' option in the sidebar is highlighted with a red box and a red arrow pointing down to the main content area. The main content area displays the 'Collaborators' section, which includes a search bar, a message stating 'This repository doesn't have any collaborators yet.', and a search form with a placeholder 'Search by username, full name or email address' and a 'Add collaborator' button.

This repository doesn't have any collaborators yet. Use the form below to add a collaborator.

Search by username, full name or email address

You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

 Add collaborator

Ejercicio. Colaboración

domingo, 21 de enero de 2018 22:49

Grupos en GitHub

domingo, 7 de enero de 2018 20:02

Perfiles de personas

Equipos con determinados permisos

The screenshot shows the GitHub organization page for 'Escuelait'. At the top, there's a logo for 'escuelaIT' and a brief description: 'Esta es la escuela de la comunidad de DesarrolloWeb.com.'. Below the header, there are navigation links: 'Repositories' (selected), 'People 25', 'Teams 4', and 'Settings'. A red arrow points from the text 'Perfiles de personas' to the 'People' link. Another red arrow points from the text 'Equipos con determinados permisos' to the 'Teams' link. The main content area displays three repository cards:

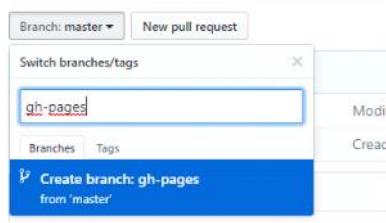
- Herramientas-Frontend-2015**: A repository for a Frontend course, last updated 16 days ago.
- HTML-CSS-2015**: A repository for HTML and CSS examples, last updated on 2 Jul.
- Curso-angularjs-FS-2015**: A JavaScript repository for an AngularJS course, last updated on 23 Apr.

On the right side, there's a sidebar titled 'People' showing 25 members, each with a small profile picture. Below the sidebar, there's a button labeled 'Invite someone'.

Pages

domingo, 21 de enero de 2018 22:52

En la página de nuestro repositorio vamos a dar clic en el botón Branch: master se nos desplegará un cuadro y dentro de la caja de texto escribimos gh-pages, de esta manera crearemos una nueva rama.



A continuación accederemos a la opción de settings y luego a branches.

Cambiaremos la rama por defecto "master" por "gh-pages", luego damos click en el botón "update" y aceptamos el mensaje que nos muestra.

The screenshot shows the 'Default branch' section of the GitHub repository settings. The 'gh-pages' branch is selected as the default. A red box highlights the 'Update' button in the dropdown menu. A red arrow points from this button to a confirmation dialog box titled 'Are you sure?' which contains the message: 'Changing your default branch can have unintended consequences that can affect new pull requests and clones.' and a button labeled 'I understand, update the default branch.'

Regresamos a Options y bajamos hasta el cuadro llamado "GitHub Pages", ahí abrimos el enlace que se nos muestra.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

The screenshot shows the GitHub Pages configuration interface. It includes sections for publishing status ('Your site is published at https://alce65.github.io/GitRepo/'), source branch ('gh-pages branch'), theme ('Choose a theme'), custom domain ('Custom domain'), and HTTPS enforcement ('Enforce HTTPS'). A red box highlights the 'Save' button in the source branch section.

<https://alce65.github.io/GitRepo/>



The screenshot shows a GitHub repository page for 'alce65 / GitRepo'. The repository has 4 commits, 1 branch, 0 releases, and 1 contributor. The latest commit was made 3 hours ago. The repository contains files: README.md and index.html. The README.md file content is:

```
Repository prueba

Linea 1 Linea 2
```

Ejercicio. Git Pages

domingo, 21 de enero de 2018 23:31

GitLab

Lunes, 8 de enero de 2018 12:23

Qué es GitLab

- Git / Basado en la nube ->Control de código y revisión
- Integración continua (integra el antiguo GitLab CI)
- Deploy continuo
- Usuarios, Proyectos

Versiones de GitLab

- GitLab.com
- Servidores descargables
 - CE (free)
 - EE: Starter - Premium - Ultimate

- Proyectos y grupos. con plantillas, importaciones y... repositorios
- Grupos y subgrupos
- Actividad
- Milestones (hitos): metodologías ágiles
- Snippets ("gist")
- Chat Mattermost (chat tipo slack)
- IDE
- Ganttlab (diagramas)
- Nueva navegación (en 9.4 y otra en 10.0)
- Nuevo 10.0
 - Auto DevOps
 - Nueva navegación
 - Nueva forma de colaboración entre grupos
 - Resolver merge request antiguos de forma automática
 - Mejoras en subgrupos
 - API para la WIKI
 - Más "quick actions"

Preentación GitLab Servers / GitLab.com

Grupos
Usuarios
Proyectos

Explorando proyectos

The screenshot shows the 'Explore projects' section of the GitLab.com website. The search bar contains 'gitlab-ce'. Projects listed include:

- Yale SDMP / Vesta**: A Ruby on Rails app to facilitate on-campus housing procedures. Developed for Yale's undergraduate housing process.
- 김진우 / training-jwkim**
- Moritz Bunkus / MKVToolNix**: Creating and working with Matroska files.
- eyeo / spec**: Functional specifications for all products at eyeo.
- Kostyantyn Matlaiyev / Flights**
- NeoMutt Project / neomutt**: Mirror of the NeoMutt Project -- https://github.com/neomutt -- https://www.neomutt.org/
- Anthony Pesch / redream**: Work In Progress SEGA Dreamcast emulator.
- python-devs / importlib_resources**: Design and implementation for a planned importlib.resources

Ejemplo: el propio gitLab

The screenshot shows the 'Explore projects' section of the GitLab.com website. The search bar contains 'gitlab-ce'. Projects listed include:

- GitLab.com / GitLab.com Support Tracker**: Support for GitLab.com issues only. See Getting Help for support with self-hosted installations and report GitLab CE bugs in the GitLab CE Issue Tracker.
- GitLab.org / GitLab Community Edition**: GitLab Community Edition (CE) is an open source end-to-end software development platform with built-in version control, issue tracking, code review, CI/CD... updated 29 minutes ago

Detalles
Actividad
Cycle Analytics

Repository

The screenshot shows the details page for the 'GitLab Community Edition' repository. The sidebar includes links for Overview, Details, Activity, Cycle Analytics, Repository, Issues (9,646), Merge Requests (490), CI / CD, Snippets, and Members. The main content area displays the repository's activity, including a merge request by Phil Hughes and a commit log.

Name	Last commit	Last update
.github	Address feedback about wording.	a year ago
.gitlab	Simplify the DB changes checklist	a month ago
app	Merge branch '38056-remove-unused-option' int...	31 minutes ago

Pages

martes, 9 de enero de 2018 0:30

The screenshot shows the GitLab Pages interface. At the top, there's a navigation bar with a speech bubble icon and the text "GitLab pages". Below this, there's a large orange header area containing several text snippets:

- Igual que GitHub pages
- Puedes hacer una página estática: HTML, CSS y JS
- Puedes hacerla a mano o...
- Static pages generator: <https://gitlab.com/pages/>
- Puedes hacerlo también para tipo grupo o usuario
- PASOS:

Under the "PASOS:" heading, there's a numbered list:

1. Fork de uno de los repositorios
2. Quitar el fork relationship (settings → general → advanced)
3. Activar "shared runners" (settings → CI/CD)
4. Editar algún archivo → push → auto build
5. Edit → pages → i la URL !
6. (Opcional) rename a namespace.gitlab.io (namespace del group)
7. (Opcional) editar configuración → baseUrl

To the right of the main content area, there's a sidebar with the text "Una rama para almacenar una página estática para el proyecto".

Más GitLab

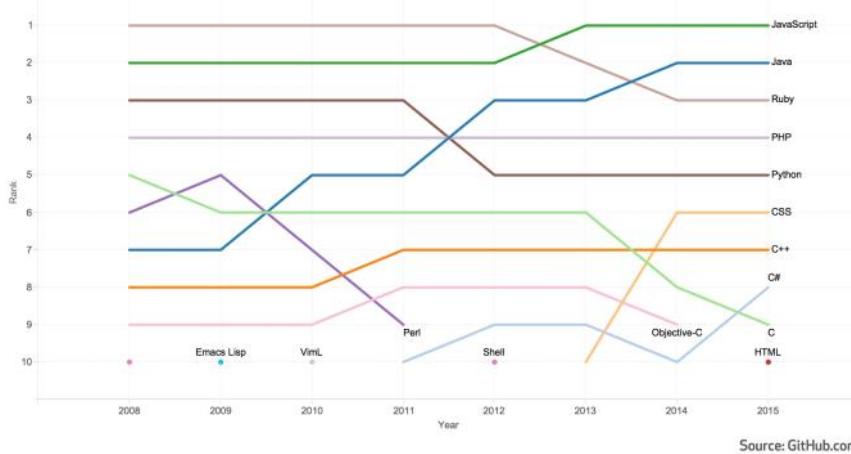
domingo, 21 de enero de 2018 9:51

Clonado de proyectos

domingo, 21 de enero de 2018 10:19

Los repositorios de GitHub
son fundamentalmente
código

Rank of top languages on GitHub.com over time



Source: GitHub.com

El uso correcto de `gitignore`

=> los repositorios no deben contener nunca

- librerías de terceros, compiladas o no
- código compilado



Es habitual tener que "instalar" los proyectos después de clonarlos



Entran en juego herramientas específicas de cada lenguaje de programación

- `npm`
- `Maven`
- `gem`
- `Composer`

Un ejemplo con `npm`

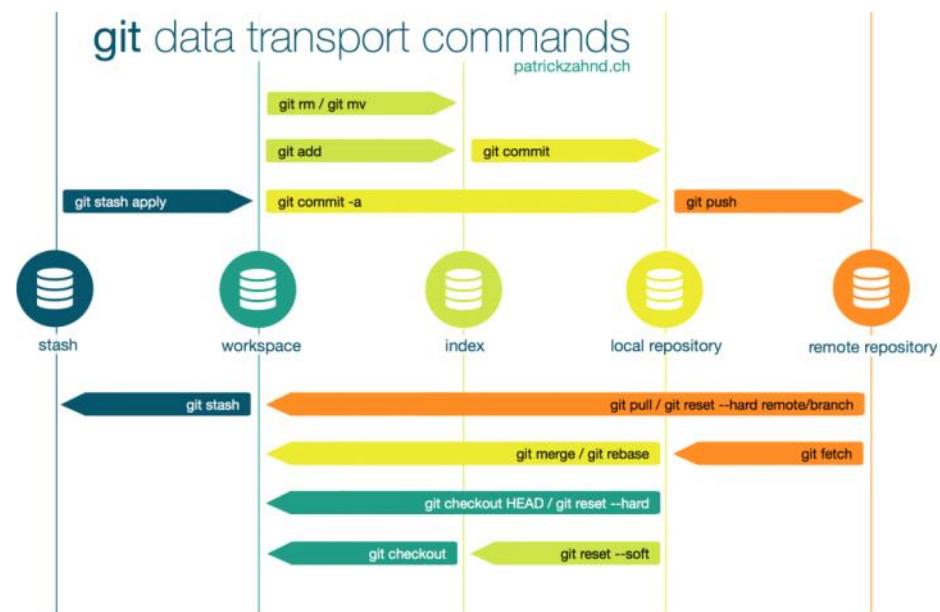
Operaciones básicas en Git

sábado, 28 de octubre de 2017 13:41

Por analogía con las bases de datos se puede hablar de CRUD

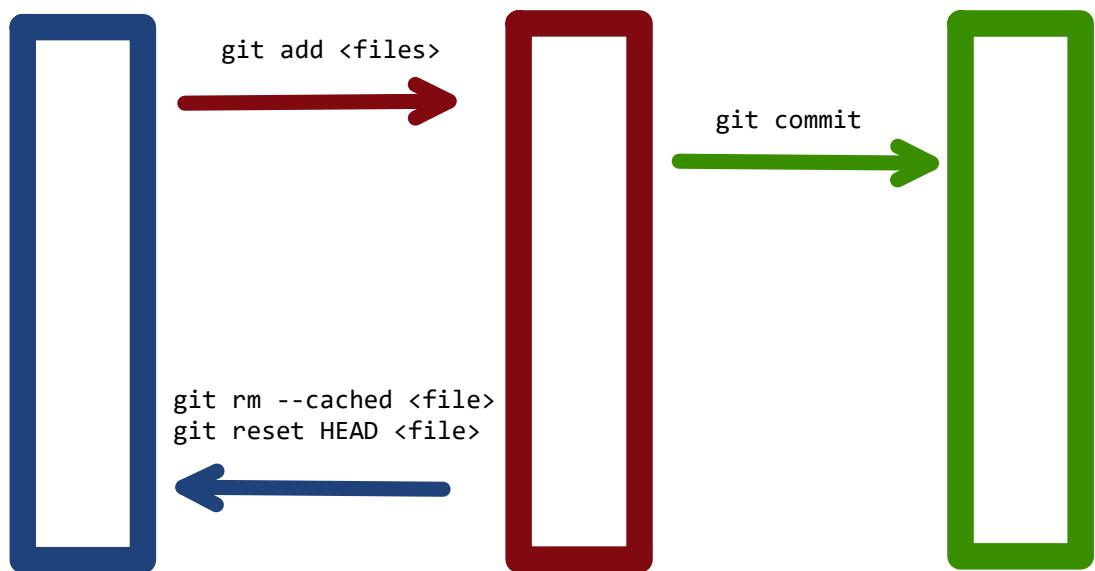
- *Create* -> `git add / git commit`.
- *Read* -> `git status / git log`.
- *Update* -> `git commit --amend`.
- *Delete*-> `git reset`.

```
git add  
git status  
git reset HEAD  
git commit  
git log  
git reset
```



Comandos

lunes, 8 de enero de 2018 23:17



Actualización del repositorio (Create)

sábado, 28 de octubre de 2017 13:44

El proceso de añadir un archivo al repositorio (*commit*) suele realizarse en 2 etapas

1. Añadir elementos a la zona de almacenamiento o indexación temporal (*staging area*) como paso previo para añadirlos al repositorio

- un fichero
 \$ git add <filename>
 \$ git add *.c
 \$ git add index.html
- un directorio
 \$ git add <directory>
- todo el contenido de la *working area*
 \$ git add <directory>

```
D:\Desarrollo\Tools\GItRepo>git add index.html
D:\Desarrollo\Tools\GItRepo>git status
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   index.html
```

2. Añadir al repositorio (*commit*) el contenido de la *staging area*, incorporando un mensaje que permita identificar cada actualización

```
$ git commit -m "message"
```

```
D:\Desarrollo\Tools\GItRepo>git commit -m "Creado index.html"
[master (root-commit) 2fb236e] Creado index.html
 1 file changed, 12 insertions(+)
 create mode 100644 index.html

D:\Desarrollo\Tools\GItRepo>git status
On branch master
nothing to commit, working tree clean

D:\Desarrollo\Tools\GItRepo>git log
commit 2fb236ec95010156687719187031031c94fd338f (HEAD -> master)
Author: alce65 <alce65@hotmail.es>
Date:   Sat Oct 28 14:56:57 2017 +0200

  Creado index.html
```

Información de todos los commits del repositorio, con su identificador único, los datos del usuario que hizo el comit ...

Más conciso mediante
git log --oneline

```
$ git commit -m "Message"
$ git commit -am "Message"
$ git commit -m "Message" <file>
$ git commit --amend
```

Ejemplo

lunes, 8 de enero de 2018 22:08

```
D:\desarrollo\Gits\Git_Basicc [master]> md mapas

Directorio: D:\desarrollo\Gits\Git_Basicc

Mode           LastWriteTime      Length Name
----           -----          ---- -
d----       04/01/2018     18:45           mapas

D:\desarrollo\Gits\Git_Basicc [master]> cd .\mapas\
D:\desarrollo\Gits\Git_Basicc\mapas [master]> echo mapa1 > mapa01.map
D:\desarrollo\Gits\Git_Basicc\mapas [master +1 ~0 -0 !]> echo mapa2 > mapa02.map
D:\desarrollo\Gits\Git_Basicc\mapas [master +1 ~0 -0 !]> dir

Directorio: D:\desarrollo\Gits\Git_Basicc\mapas

Mode           LastWriteTime      Length Name
----           -----          ---- -
-a---       04/01/2018     18:46        16 mapa01.map
-a---       04/01/2018     18:46        16 mapa02.map
```

Se crea una nueva carpeta

```
D:\desarrollo\Gits\Git_Basicc [master +1 ~0 -0 !]> git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    mapas/

nothing added to commit but untracked files present (use "git add" to track)
D:\desarrollo\Gits\Git_Basicc [master +1 ~0 -0 !]> git add mapas
D:\desarrollo\Gits\Git_Basicc [master +2 ~0 -0 ~]> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   mapas/mapa01.map
    new file:   mapas/mapa02.map
```

git status

git add

Ventajas de que exista esta zona:
poder seleccionar solo una parte del
contenido de la *workarea* para incorporarlo
a un *commit*

```
D:\desarrollo\Gits\Git_Basicc [master +2 ~0 -0 ~]> git commit -m "Primeros mapas"
[master 5ee8efb] Primeros mapas
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 mapas/mapa01.map
create mode 100644 mapas/mapa02.map
D:\desarrollo\Gits\Git_Basicc [master]> git log
commit 5ee8efb895f14abff8264e60e4cca392d99e7f4f (HEAD -> master)
Author: alce65 <alce65@hotmail.es>
Date:   Thu Jan 4 18:50:46 2018 +0100

    Primeros mapas

commit e4b35526d2e26ce5dff076a6c1233b7c015087e4
Author: alce65 <alce65@hotmail.es>
Date:   Thu Jan 4 18:42:07 2018 +0100

    Readme inicial
D:\desarrollo\Gits\Git_Basicc [master]> git log --oneline
5ee8efb (HEAD -> master) Primeros mapas
e4b3552 Readme inicial
D:\desarrollo\Gits\Git_Basicc [master]> -
```

git commit

git log

Eliminación de temporales

martes, 26 de diciembre de 2017 21:25

Eliminar elementos de la zona de almacenamiento o indexación temporal (*staging area*), i.e. revertir *add*:

```
# Antes del primer commit  
$ git rm --cached <file>
```

(antes del primer *commit* aún no existe el puntero HEAD y por tanto no puede hacerse referencia a él)

```
# Después del primer commit, cuando ya existe HEAD  
$ git reset HEAD <files>
```

```
D:\desarrollo\Gits\Git_Basic [master +1 ~0 -0 !]> echo funcionalidad > index.js  
D:\desarrollo\Gits\Git_Basic [master +1 ~0 -0 !]> git add .  
D:\desarrollo\Gits\Git_Basic [master +1 ~0 -0 ~]> git status  
On branch master  
Changes to be committed:  
  (use "git reset HEAD <file>..." to unstage)  
  
    new file:   index.js  
  
D:\desarrollo\Gits\Git_Basic [master +1 ~0 -0 ~]> git reset HEAD .  
D:\desarrollo\Gits\Git_Basic [master +1 ~0 -0 !]> git status  
On branch master  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
  
    index.js  
  
nothing added to commit but untracked files present (use "git add" to track)  
D:\desarrollo\Gits\Git_Basic [master +1 ~0 -0 !]> -
```

git add .

git reset HEAD .

Elimina un fichero de la *staging area*

Commit & Add

lunes, 8 de enero de 2018 23:24

Cuando un fichero ya está *track* (trazado), i.e. el repositorio ya conoce su existencia, pueden combinarse *add* y *commit* en una sola operación

```
$ git commit -am "message"
```

Staging & Commid unidos en caso de ficheros "trazados" (tracking)

```
D:\desarrollo\Gits\Git_Basicc [master +1 ~0 -0 !]> git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    index.js

nothing added to commit but untracked files present (use "git add" to track)
D:\desarrollo\Gits\Git_Basicc [master +1 ~0 -0 !]> git commit -am "Index.js"
On branch master
Untracked files:
  index.js

nothing added to commit but untracked files present
```

Un archivo nuevo "untracked" no puede ser objeto del comando *commit -am*

```
D:\desarrollo\Gits\Git_Basicc [master +1 ~0 -0 !]> git add .
D:\desarrollo\Gits\Git_Basicc [master +1 ~0 -0 ~]> git commit -m "Index.js"
[master 834a7b3] Index.js
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.js
```

En este caso es necesario ejecutar los dos comandos por separado

```
D:\desarrollo\Gits\Git_Basicc [master]> code .
D:\desarrollo\Gits\Git_Basicc [master]> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   index.js

no changes added to commit (use "git add" and/or "git commit -a")
D:\desarrollo\Gits\Git_Basicc [master +0 ~1 -0 !]> git commit -am "Modificando index.js"
[master 2c24170] Modificando index.js
 1 file changed, 0 insertions(+), 0 deletions(-)
```

Un fichero modificado, previamente trazado (*tracked*) en el repositorio SI puede ser objeto del comando *commit -am*

git commit -am

git checkout -- <file> elimina las modificaciones de un fichero en la *working area*

Modificación del último commit

viernes, 5 de enero de 2018 7:43

```
git commit -amend
```

```
D:\desarrollo\Gits\Git_Basic [master +0 ~1 -0 !]> git log --oneline  
2c24170 (HEAD -> master) Modificando index.js  
834a7b3 Index.js  
5ee8efb Primeros mapas  
e4b3552 Readme inicial  
D:\desarrollo\Gits\Git_Basic [master +0 ~1 -0 !]> git show  
commit 2c24170a890dace9900cb6225a7625d4d29e1103 (HEAD -> master)  
Author: alce65 <alce65@hotmail.es>  
Date: Fri Jan 5 07:57:21 2018 +0100  
  
    Modificando index.js  
  
diff --git a/index.js b/index.js  
index 08b0b6c..2e2ff1c 100644  
Binary files a/index.js and b/index.js differ
```

El último *commit* puede ser modificado, incorporándole nuevo contenido

```
D:\desarrollo\Gits\Git_Basic [master +0 ~1 -0 !]> git status  
On branch master  
Changes not staged for commit:  
  (use "git add <file>..." to update what will be committed)  
  (use "git checkout -- <file>..." to discard changes in working directory)  
  
        modified:   index.js  
  
no changes added to commit (use "git add" and/or "git commit -a")  
D:\desarrollo\Gits\Git_Basic [master +0 ~1 -0 !]> git commit --amend  
unix2dos: converting file D:/desarrollo/Gits/Git_Basic/.git/COMMIT_EDITMSG to DOS format...
```



El editor de textos configurado para Git permite modificar el mensaje de *commit*

```
dos2unix: converting file D:/desarrollo/Gits/Git_Basic/.git/COMMIT_EDITMSG to Unix format..  
. [master aabf624] Completando index.js  
  Date: Fri Jan 5 07:57:21 2018 +0100  
  1 file changed, 0 insertions(+), 0 deletions(-)  
D:\desarrollo\Gits\Git_Basic [master +0 ~1 -0 !]> git log --oneline  
aabf624 (HEAD -> master) Completando index.js  
834a7b3 Index.js  
5ee8efb Primeros mapas  
e4b3552 Readme inicial  
D:\desarrollo\Gits\Git_Basic [master +0 ~1 -0 !]> git show  
commit aabf62474ae968b89084b2bc6d4497165351938e (HEAD -> master)  
Author: alce65 <alce65@hotmail.es>  
Date: Fri Jan 5 07:57:21 2018 +0100  
  
    Completando index.js  
  
diff --git a/index.js b/index.js  
index 08b0b6c..2e2ff1c 100644  
Binary files a/index.js and b/index.js differ
```

El resultado es en realidad un nuevo *commit* con su nuevo identificador

Renombrado

miércoles, 27 de diciembre de 2017 19:38

Si se renombra un fichero desde fuera del repositorio, Git lo interpreta como dos acciones

```
PS D:\Desarrollo\Git_Basic> ren fichero.txt nuevo_nombre.txt
PS D:\Desarrollo\Git_Basic> git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    deleted:    fichero.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    nuevo_nombre.txt

no changes added to commit (use "git add" and/or "git commit -a")
PS D:\Desarrollo\Git_Basic>
```

El borrado de un fichero

La creación de un fichero

En versiones previas

La primera operación se recoge mediante un `git add`

La segunda, para transmitir la eliminación al repositorio, mediante un `git rm`

En la actualidad, es suficiente un `git add`

```
PS D:\Desarrollo\Git_Basic> git add .
PS D:\Desarrollo\Git_Basic> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    renamed:    fichero.txt -> nuevo_nombre.txt

PS D:\Desarrollo\Git_Basic>
```

Otra alternativa es indicarle a Git que lleve a cabo en renombrado, usando `git mv`

```
PS D:\Desarrollo\Git_Basic> git mv fichero.txt nuevo_nombre.txt
PS D:\Desarrollo\Git_Basic> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    renamed:    fichero.txt -> nuevo_nombre.txt

PS D:\Desarrollo\Git_Basic>
```

Acceso a los commits

martes, 2 de enero de 2018 18:09

Acceder a un de tres maneras diferentes:

1. Referencias absolutas
 - a. A través de su ID (1234567)

2. Referencias relativas

A través del nombre de la rama.
Sabiendo que por defecto la rama
está en el último *commit*
a. A través del puntero HEAD

~ -> altGr 4
-> alt 126

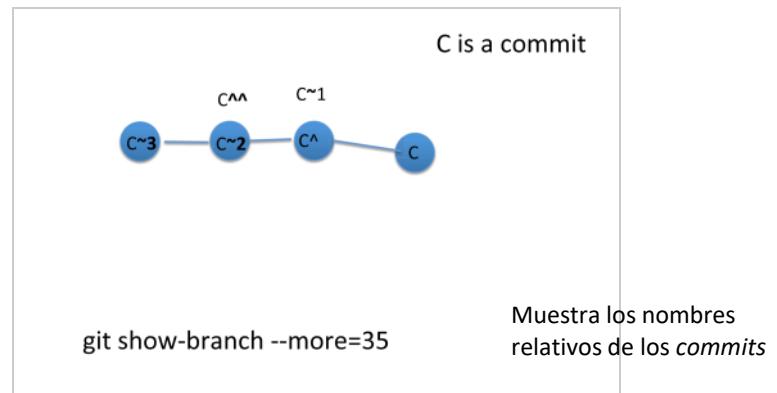
Cada *commit* en Git tienen un ID único correspondiente a un número hexadecimal basado en el algoritmo SHA1.

Cada ID es globalmente único, no sólo en un repositorio, sino a nivel de todos los existentes

Git proporciona mecanismos para identificar un *commit* con respecto a alguna de las referencias (punteros) a otro *commit*, como son HEAD o el nombre de rama:

[^]: Un *commit* antes (master[^], master^{^^}, HEAD[^], etc.)

~: N *commits* antes(master~2, HEAD~4)



Contenido de un commit

martes, 9 de enero de 2018 0:16

Para mostrar un *commit* se utiliza `git show <referencia>`

```
D:\desarrollo\Gits\timetravel [master]> git show master
commit 95431a3ffce7e8208293877a36e3ba63bb89f637 (HEAD -> master)
Author: alce65 <alce65@hotmail.es>
Date:   Fri Jan 5 14:06:41 2018 +0100

  Dia 1

diff -git a/prueba.txt b/prueba.txt
new file mode 100644
index 0000000..b474fba
Binary files /dev/null and b/prueba.txt differ
```

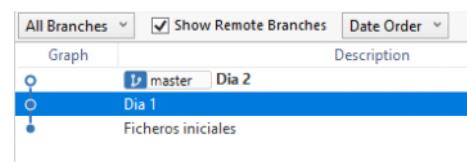
commit que recoge la creación de un fichero

```
D:\desarrollo\Gits\timetravel [master]> code .
D:\desarrollo\Gits\timetravel [master]> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   prueba.txt

no changes added to commit (use "git add" and/or "git commit -a")
D:\desarrollo\Gits\timetravel [master +0 ~1 -0 !]> git commit -am "Dia 2"
[master aada1bf] Dia 2
 1 file changed, 0 insertions(+), 0 deletions(-)
```

modificamos el fichero y creamos un nuevo *commit*



```
D:\desarrollo\Gits\timetravel [master]> git show master
commit aada1bf137c68e92cf5bf80ff6a44a88e6224cee (HEAD -> master)
Author: alce65 <alce65@hotmail.es>
Date:   Fri Jan 5 16:44:12 2018 +0100

  Dia 2

diff --git a/prueba.txt b/prueba.txt
index b474fba..e4ce670 100644
Binary files a/prueba.txt and b/prueba.txt differ
```

commit que recoge la modificación de un fichero

Histórico: log

martes, 2 de enero de 2018 18:11

Opciones avanzadas de git log

Se puede ver la sucesión histórica (*history*) de los *commits*:

`git log` (igual que `git log HEAD`)

Se puede indicar a partir de que *commit* debe de empezar la serie

`git log <commit-name>`
e.g. `git log master~2`

Se puede especificar un rango (desde .. hasta)

`$ git log master~12..master~10`

Se pueden mostrar los commits que afectan a un determinado path (carpeta, fichero...)

`$ git log -- <path>`

E.g. `git log -- README3.txt`

Se pueden mostrar los commits que coincidan con una expresión regular

`$ git log --grep='reg-exp'`

Se pueden excluir del listado los commits resultantes de un merge

`git log --no-merges`

Se pueden mostrar los cambios producidos durante un tiempo

`$ git log --since={2010-04-18}`

`$ git log --before={2010-04-18}`

`$ git log --after={2010-04-18}`

Se puede mostrar la salida en formato gráfico (dentro de las limitaciones de la consola)

`git log --decorate --graph --oneline`

```
*  17e4c5f (HEAD, develop) Merge branch 'hotfix-1.0.1' into develop
/ \
/ * f43bb33 (hotfix-1.0.1) Hotfix2
/ * 50a102d Hotfix1
/ * d2fe7d6 Version 1.0.1
/ *   d534111 (tag: 1.0) Merge branch 'release-1.0'
/ / \
* / \  77e7eac Merge branch 'release-1.0' into develop
/ \ \ \
/ / / /
/ / / /
/ * / 9261fad (release-1.0) Release10Fix2
/ * / 2562cb3 Release10Fix1
/ * / c51b802 Version 1.0
* / / fbdd638 work6
* / / 7353285 work5
/ / /
* / 538bb9a work4
* / cdef254 Merge branch 'feature1' into develop
/ /
```

Git Blame

martes, 9 de enero de 2018 0:37

Permite conocer el autor de la última modificación de cada línea de un fichero y en qué *commit* se incluyó el cambio

```
git blame <archivo>
```

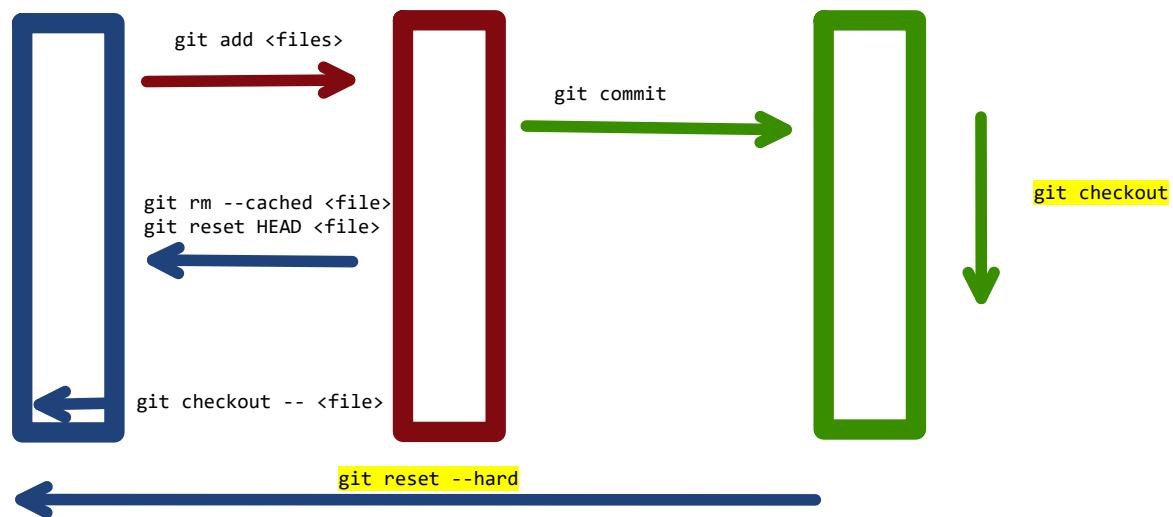
Esta también disponible cuando se accede al repositorio en GitHub

The screenshot shows a GitHub repository page for 'spring-boot-angular4-boilerplate / README.md'. At the top, there are tabs for 'Raw', 'Blame', and 'History'. A red arrow points from the text above to the 'Blame' tab. Below the tabs, there's a summary: '80 lines (54 sloc) | 3.18 KB'. The main area displays the blame output for the file, which includes commit history and line-level blame information.

Commit	Date	Blame Output
Upgrade to Angular 4.0.0-RELEASE and Angular CLI 1.0.0-RE...	10 months ago	1 # Spring Boot + Angular 4 Boilerplate
Add initial folder structure and boilerplate files	a year ago	2
docs: Update README	6 months ago	3 ![[spring-boot-angular4-boilerplate](https://raw.githubusercontent.com/Saka7/spring-boot-angular4-boilerplate/main/README.md)](https://raw.githubusercontent.com/Saka7/spring-boot-angular4-boilerplate/main/README.md)
Fix codeclimate badges	10 months ago	4
Add codecoverage badges	10 months ago	5 ![[Code Climate]](https://codeclimate.com/github/Saka7/spring-boot-angular4-boilerplate/badges/gpa)
Fix codeclimate badges	10 months ago	6 ![[Issue Count]](https://codeclimate.com/github/Saka7/spring-boot-angular4-boilerplate/badges/issue-count)
Modify sh and bat scripts	a year ago	7
Add initial folder structure and boilerplate files	a year ago	8 Quick start for Spring Boot + Angular 4 projects with JWT auth
		9
		10 ## Includes:
		11
		12 Front-end:
		13
Modify README	11 months ago	14 - angular-cli boilerplate files
docs: Update README	6 months ago	15 - JWT authentication service
Add initial folder structure and boilerplate files	a year ago	16
		17 Back-end:

Desplazamiento en el histórico

domingo, 21 de enero de 2018 10:31



`git checkout <commit>`

`git reset --hard`
`git checkout --<file>`

Checkout: Posicionar HEAD en un *commit*

`git checkout <id-commit al que quiero ir>`

Como consecuencia del desplazamiento del puntero HEAD pueden obtenerse distintos resultados:

- Posicionarme en un *commit* concreto en el pasado
- Cambiar de rama

Recordar

El doble guion --

Una de sus dos funcionalidades:

Separar un nombre de archivo explícitamente identificado

```
# Checkout the tag named "main.c"
$ git checkout main.c
```

```
#Checkout the file named "main.c"
$ git checkout -- main.c
```

Ejercicio: timeTravel

viernes, 5 de enero de 2018 13:58

Adelantamos el uso de ckeckout y reset

Proyecto timeTravel

```
D:\desarrollo\Gits\timetetravel [master +1 ~0 -0 !]> echo Dia1 > prueba.txt
D:\desarrollo\Gits\timetetravel [master +1 ~0 -0 !]> git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    prueba.txt

nothing added to commit but untracked files present (use "git add" to track)
D:\desarrollo\Gits\timetetravel [master +1 ~0 -0 !]> git add .
D:\desarrollo\Gits\timetetravel [master +1 ~0 -0 ~]> git commit -m "Dia 1"
[master 95431a3] Dia 1
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 prueba.txt
```

Commit 1

Contenido del fichero prueba.txt:
Dia1

```
D:\desarrollo\Gits\timetetravel [master]> code .
D:\desarrollo\Gits\timetetravel [master]> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   prueba.txt

no changes added to commit (use "git add" and/or "git commit -a")
D:\desarrollo\Gits\timetetravel [master +0 ~1 -0 !]> git commit -am "Dia 2"
[master aa20ad4] Dia 2
 1 file changed, 0 insertions(+), 0 deletions(-)
D:\desarrollo\Gits\timetetravel [master]> git log --oneline
aa20ad4 (HEAD -> master) Dia 2
95431a3 Dia 1
e3d80a7 Ficheros iniciales
```

Commit 2

```
D:\desarrollo\Gits\timetetravel [master]> type prueba.txt
Dia1
Dia2
```

Contenido del fichero prueba.txt:
Dia1
Dia2

Retorno al estado del commit 1:

```
D:\desarrollo\Gits\timetetravel [master]> git checkout 95431a3
Note: checking out '95431a3'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

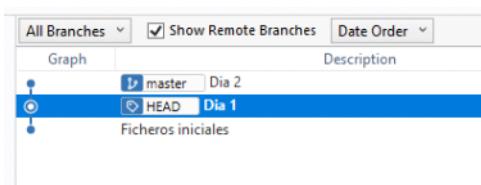
  git checkout -b <new-branch-name>

HEAD is now at 95431a3... Dia 1
D:\desarrollo\Gits\timetetravel [(95431a3...)]>
```

Desde este punto, sería incorrecto añadir nuevos *commits* sin crear una nueva rama, tal como indica el mensaje de Git

```
D:\desarrollo\Gits\timetetravel [(95431a3...)]> type prueba.txt
Dia1
```

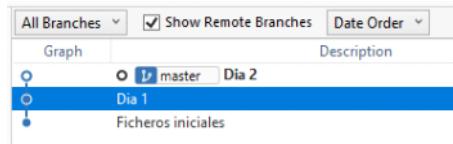
Editando el ficho, su contenido sería



El puntero HEAD ha retrocedido, y no está en el último *commit* de la rama activa o master (está *detached*)

```
D:\desarrollo\Gits\timetetravel [(95431a3...)]> git checkout master
Previous HEAD position was 95431a3... Dia 1
Switched to branch 'master'
```

Se puede volver nuevamente al último *commit* de la rama activa indicando *checkout* y su nombre (master)



El puntero HEAD vuelve a su posición "normal" en la rama master

```
D:\desarrollo\Gits\timetravel [master]> type prueba.txt
Dia1
Dia2
```

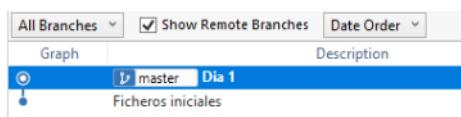
El contenido del fichero vuelve a estar completo

```
D:\desarrollo\Gits\timetravel [master]> git log --oneline
aa20ad4 (HEAD -> master) Dia 2
95431a3 Dia 1
e3d80a7 Ficheros iniciales
```

Otra operación que modifica más radicalmente el histórico es la eliminación de un *commit*

```
D:\desarrollo\Gits\timetravel [master]> git reset --hard master^
HEAD is now at 95431a3 Dia 1
D:\desarrollo\Gits\timetravel [master]> git log --oneline
95431a3 (HEAD -> master) Dia 1
e3d80a7 Ficheros iniciales
```

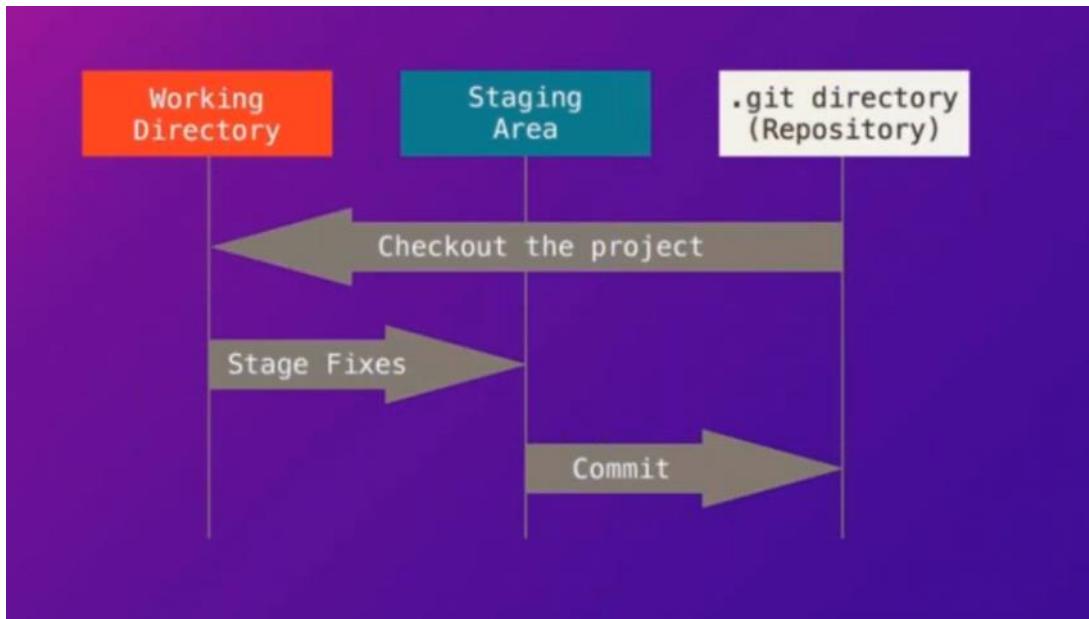
En este caso el *commit* es completamente irrecuperable



```
D:\desarrollo\Gits\timetravel [master]> type prueba.txt
Dia1
```

Resumen Git básico

miércoles, 3 de enero de 2018 19:20



Ejemplos.

- configuración de git:
 - o user / email
 - o editor asociado
 - o terminal
- creación de un fichero
 - o git add
 - o git status
 - o git commit
- clonación desde github
- el fichero .gitignore: configuraciones - passwords - dependencias - dist
 - o global .gitignore
- borrado de archivos. evitar hacerlo desde el S.O. seguido de add
 - o git rm
 - o git rm --cached (no borra del disco, sólo del repositorio)
- información de los commits y modificaciones
 - o git log
 - o formato de la salida git log --pretty=...
 - o git commit --amend
- eliminación de archivos de la staging area
 - o git rm --cached <file>
 - o git reset HEAD -- <file>
 - o git checkout -- <file> -> eliminación del archivo del área de trabajo
- repositorios remotos
- config alias

- ▷ git config
- ▷ git init
- ▷ git add
- ▷ git clone
- ▷ git status (ver siguiente slide)
- ▷ El .gitignore
- ▷ git diff
- ▷ git commit
- ▷ git rm
- ▷ git remote

<https://try.github.io/levels/1/challenges/1>