



Introducción a Git

¿Qué es Git?

Git es un sistema de control de versiones distribuido, gratuito y de código abierto, diseñado para gestionar desde proyectos pequeños a muy grandes con rapidez y eficacia.

¿Por qué distribuido?

Git es un sistema de control de versiones distribuido, por lo tanto, los clientes no solo descargan la última instantánea de los archivos; replican completamente el repositorio en local.

Git permite recuperar y enviar archivos a cualquiera de los clientes que tengan en local el repositorio. Dado que existen varios orígenes para recuperar la información, Git ofrece una gran robustez.

¿Qué resuelve Git?

- Seguimiento de los cambios.
- Autoridad de los cambios.
- Información sobre el cambio.
- Compartir código.
- Resolución de conflictos.
- ...

Los tres estados de Git



Los tres estados de Git

El flujo de trabajo básico de Git puede ser el siguiente:

1. Se modifican los archivos en nuestro directorio.
2. Se añade al área Staging los cambios de los archivos del directorio de trabajo.
3. Se hace commit, entonces se leen los archivos que están en el área Staging y se crea una versión de esos archivos de forma permanente en el directorio Git.

Cómo funciona Git

Finalmente, esto nos lleva al objeto más importante: el **commit**.

Los commits pueden ser considerados como instantáneas: saben cómo eran los árboles en un momento dado. También tienen alguna otra información asociada, como el autor, la fecha y un mensaje.

Cómo funciona Git

Los commits se organizan en un gráfico acíclico dirigido. Básicamente significa que los commits "fluyen" en una dirección.

Cómo funciona Git

The screenshot shows the SourceTree application interface for the 'sourcetree-website' repository. The top toolbar includes buttons for Commit, Pull, Push, Branch, Merge, and Shelf. The sidebar on the left contains a 'WORKSPACE' section with 'File status', 'History' (selected), and 'Search'. Below this are 'BRANCHES', 'BOOKMARKS', 'TAGS', 'REMOTES', 'SHELVED', and 'SUBREPOSITORIES'. The main area displays a commit graph and a table of commits.

| Graph | Commit | Author | Description | Date |
|-------|----------------|----------------|---|----------------------|
| | b7358c7 | Rahul Chha... | origin/master origin/HEAD Removing ol... | Mar 3, 2016, 11:... |
| | bdb8bef | Rahul Chhab... | Merged in update-google-verification (pull request #14) | Feb 18, 2016, 1:3... |
| | dfe975d | Tyler Tadej... | origin/update-google-verification Update google verificati... | Feb 11, 2016, 2:2... |
| | 3bc3290 | Tyler Tadej... | Replace outdated Atlassian logo in footer with base-64 en... | Feb 11, 2016, 2:1... |
| | dba47f9 | Tyler Tadej... | Add gitignore | Feb 11, 2016, 1:3... |
| | ff67b45 | Mike Minns... | Updated Mac min-spec to 10.10 | Feb 15, 2016, 11:... |
| | 72d32a8 | Michael Min... | Merged in hero_images (pull request #13) | Feb 15, 2016, 10:... |
| | 246c4ff | Joel Unger... | origin/hero_images hero_images Used Tinypng to c... | Feb 11, 2016, 3:3... |
| | 9d9438c | Joel Unger... | Replacing hero images with new version of SourceTree | Feb 9, 2016, 2:59... |
| | ce75b63 | Michael Min... | Merged in bug/date-https (pull request #12) | Feb 15, 2016, 10:... |
| | 85367bb | Patrick Tho... | origin/bug/date-https fixed date and https errors | Jan 7, 2016, 12:2... |
| | 4f9b557 | Joel Unger... | New Favicon | Feb 8, 2016, 3:55... |
| | 384e6d5 | Rahul Chhab... | origin/search-console-access search console google ver... | Feb 3, 2016, 2:09... |
| | 6fa47a9 | Mike Minns... | updated to move supported version to OSX 10.9+ | Dec 15, 2015, 2:0... |
| | 8dd87bb | Mike Minns... | remove extra , when a line is skipped due to empty server | Nov 23, 2015, 2:2... |
| | faa195e | Mike Minns... | Skip records with empty server/user id as gas rejects them | Nov 23, 2015, 2:1... |
| | 0cdf96 | Mike Minns... | corrected paths after merge | Nov 23, 2015, 2:0... |
| | 051ab1b | Mike Minns... | corrected column counting | Nov 23, 2015, 1:5... |
| | a723bc2 | Mike Minns... | Merge branch 'au2gex' | Nov 23, 2015, 1:5... |
| | 65fd580 | Mike Minns... | deal with invalid instanceids | Nov 23, 2015, 1:5... |
| | 500a892 | Michael Min... | Merged in au2gex (pull request #11) | Nov 23, 2015, 1:0... |

Cómo funciona Git

Una vez has hecho los cambios en local, es probable que quieras almacenar los cambios en un repositorio remoto como GitHub, Bitbucket, Azure Devops, etc.

Para ello primero necesitas crear una cuenta en alguna de las plataformas y crear tu primer repositorio.

Comandos Git

Comandos Git

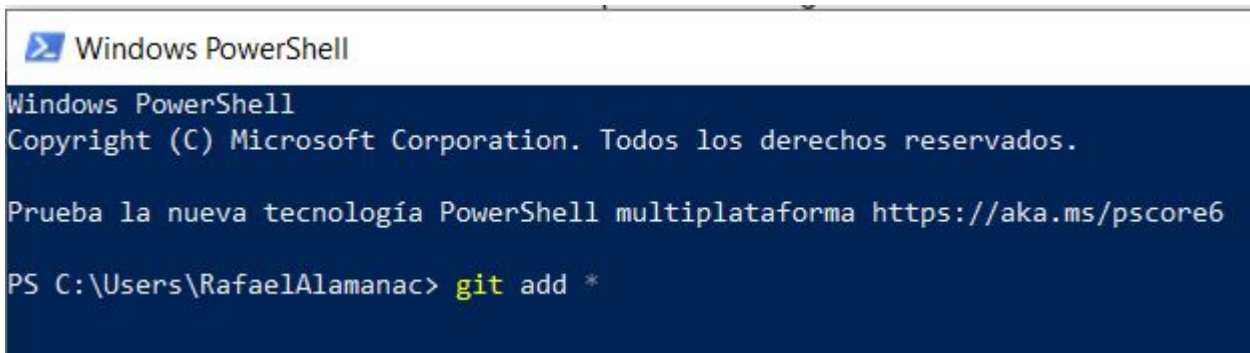
Existen dos tipos de comandos git, los que afectan al repositorio local o los que afectan al repositorio remoto.

- Los que afectan al repositorio local los llamaremos **comandos locales**.
- Los que afectan al repositorio remoto los llamaremos **comandos remotos**.

Es decir, **todo cambio** hecho en **local no afecta** al **remoto**.

Comandos locales: Add

Este comando actualiza el índice utilizando el contenido actual que se encuentra en el árbol de trabajo, para preparar el contenido preparado para el siguiente commit.

A screenshot of a Windows PowerShell terminal window. The title bar at the top says "Windows PowerShell". The terminal text includes the standard PowerShell startup messages: "Windows PowerShell", "Copyright (C) Microsoft Corporation. Todos los derechos reservados.", and a promotional message "Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6". The prompt "PS C:\Users\RafaelAlamanac>" is followed by the command "git add *".

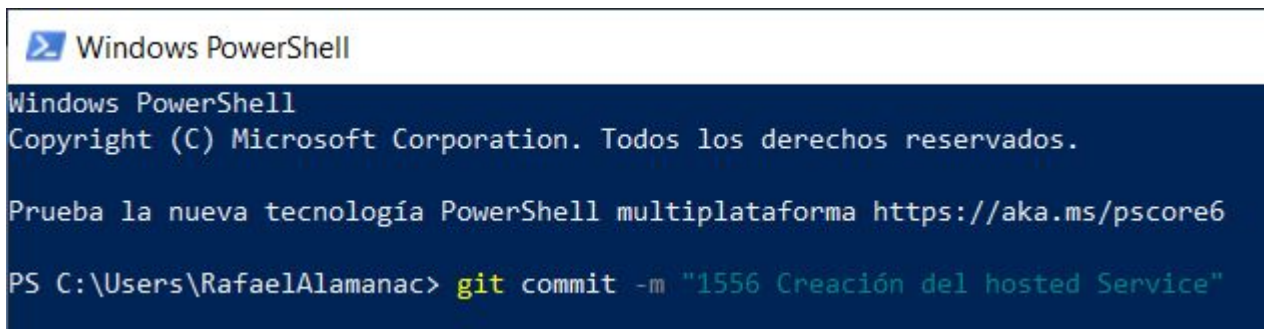
```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\RafaelAlamanac> git add *
```

Comandos locales: Commit

El comando "commit" se utiliza para guardar los cambios en el repositorio local.



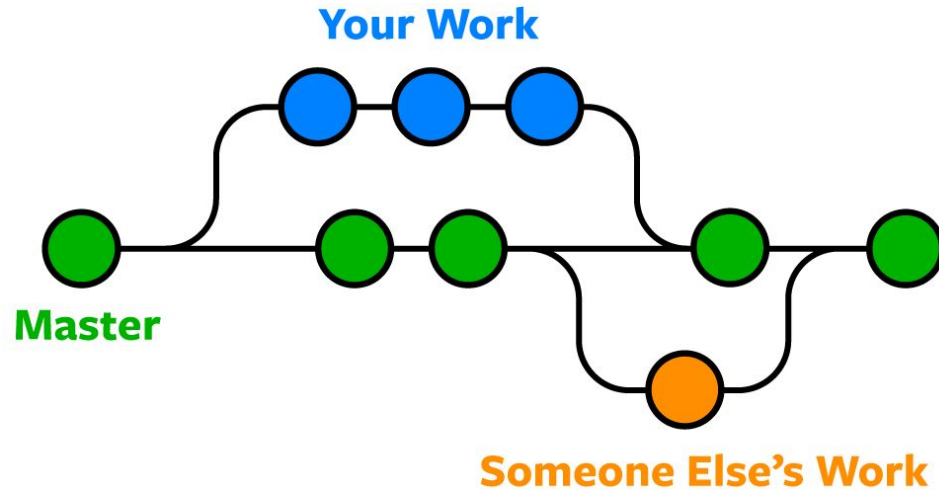
```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\RafaelAlamanac> git commit -m "1556 Creación del hosted Service"
```

Comandos locales: Branch

Permite crear, mostrar o borrar ramas.



Comandos locales: Checkout

Actualiza los archivos en el árbol de trabajo para que coincidan con la versión en el índice o el árbol especificado.

Si no se da ninguna especificación de ruta, git checkout también actualizará HEAD para establecer la rama especificada como la rama actual.

Comandos locales: Merge

Incorpora los cambios de los commits nombrados (desde el momento en que sus historias divergen de la rama actual) a la rama actual. (desde el momento en que sus historias divergen de la rama actual) a la rama actual.

Comandos remoto: Clone

Este comando incorpora todos los cambios hechos en un repositorio remoto en local.

Lo que haces es crear una copia en local de toda la historia de commits que se encuentra en el remoto. Esta copia es necesario actualizarla a través de los comandos **pull** y **push**.

Comandos remoto: Pull

Incorpora los cambios de un repositorio remoto a la rama actual. En su modo por defecto, `git pull` es la abreviatura de `git fetch` seguido de `git merge FETCH_HEAD`.

Comandos remoto: Push

El comando `git push` se utiliza para subir el contenido del repositorio local a un repositorio remoto. Pushing es la forma de transferir las confirmaciones de su repositorio local a un repositorio remoto.

Comandos remoto: Remote

El comando `git remote` te permite crear, ver y eliminar conexiones a otros repositorios. Las conexiones remotas son más como marcadores que como enlaces directos a otros repositorios. En lugar de proporcionar acceso en tiempo real a otro repositorio, sirven como nombres convenientes que pueden ser utilizados para hacer referencia a una URL.

Uso de GitHub

Uso de GitHub

- Primero creamos un repositorio en GitHub.
- Clonamos el repositorio en nuestro local.
- Creamos una rama nueva de trabajo o trabajamos en la principal.
- Modificamos los archivos en nuestro local y hacemos commit.
- Hacemos un pull de la rama actual.
- Hacemos un push de la rama actual.
- Los cambios ya están en el repositorio.

Uso avanzado de GitHub

- Primero creamos un repositorio en GitHub.
- Clonamos el repositorio en nuestro local.
- Creamos una rama nueva de trabajo
- Modificamos los archivos en nuestro local y hacemos commit.
- Hacemos un push de la rama nueva al repositorio remoto, esto hará que se cree la rama en GitHub.
- Desde GitHub creamos un Pull Request.