

# Programación para ingenieros

Docente: Oscar Stiven Morales Zapata - Sebastian Durango Idarraga

Correo electrónico: [oscars.morales@autónoma.edu.co](mailto:oscars.morales@autónoma.edu.co)

[sebastiandi@autonoma.edu.co](mailto:sebastiandi@autonoma.edu.co)



## Unidad 1

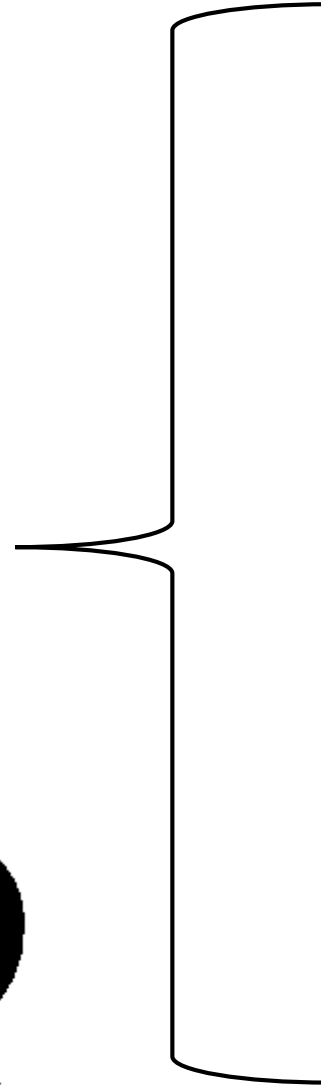
# ¿Existe alguna forma de controlar las diferentes versiones de código de forma eficiente y confiable?

Docente: Oscar Stiven Morales Zapata - Sebastian Durango Idarraga

Correo electrónico: [oscars.morales@autónoma.edu.co](mailto:oscars.morales@autónoma.edu.co)

[sebastiandi@autonoma.edu.co](mailto:sebastiandi@autonoma.edu.co)







Final



Final\_1



Final\_2



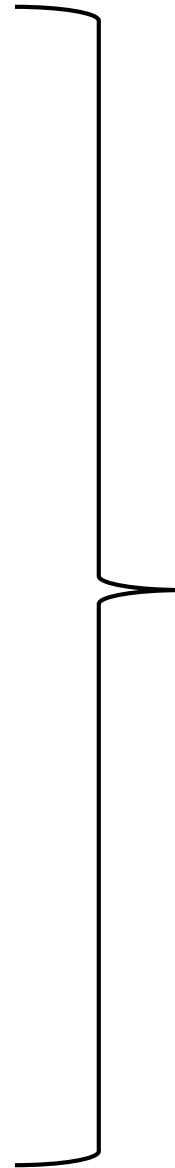
Final\_3

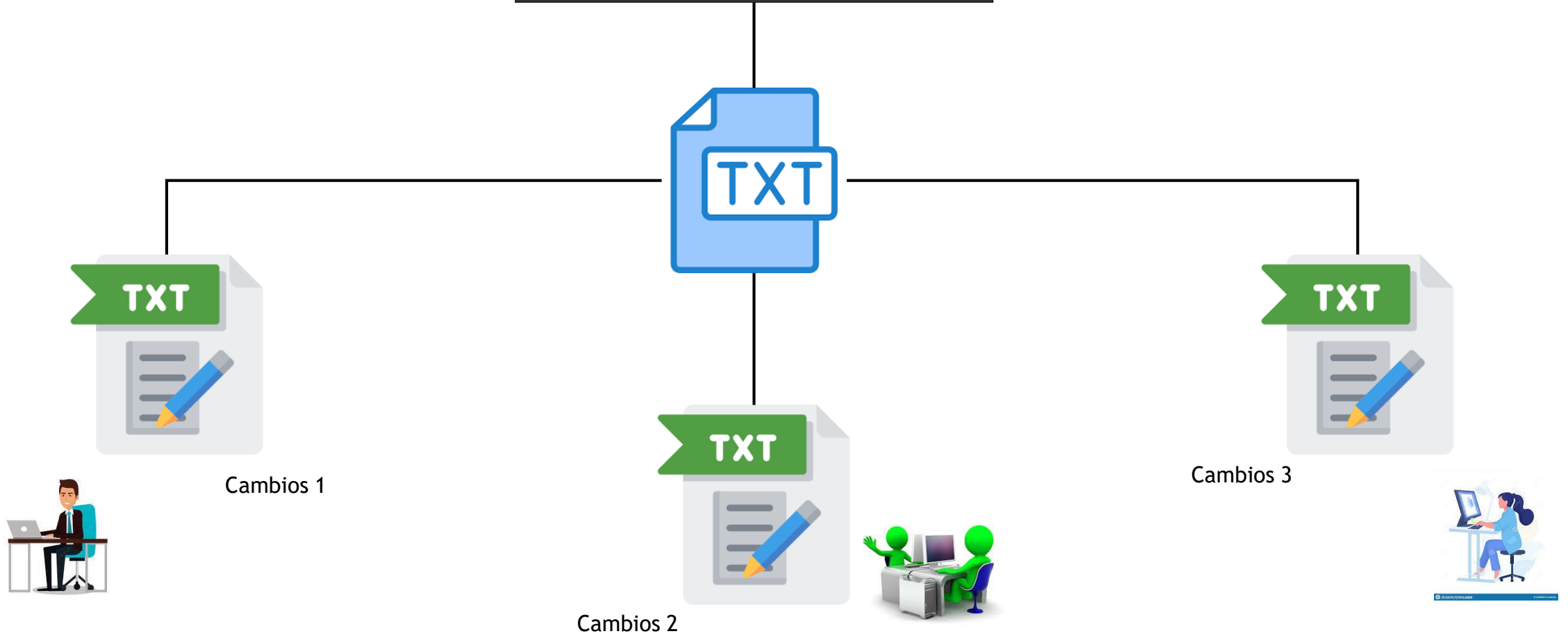


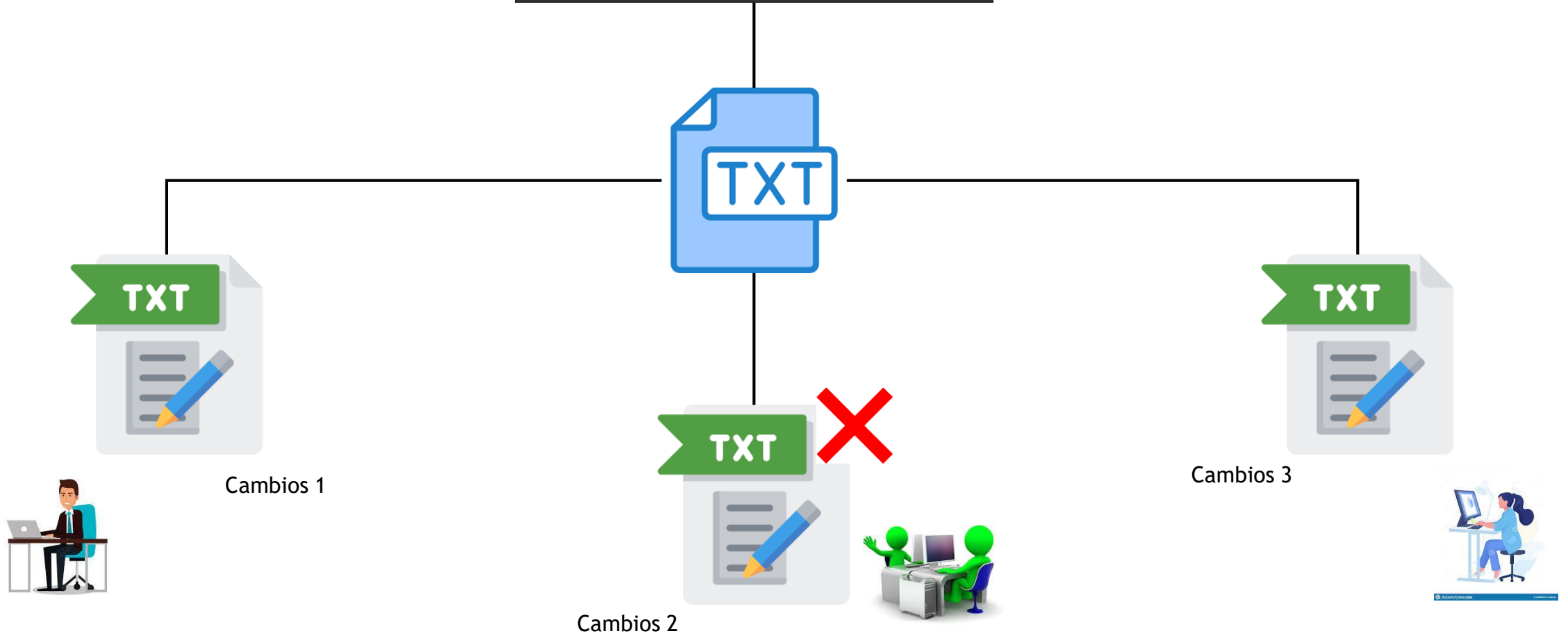
Final\_4

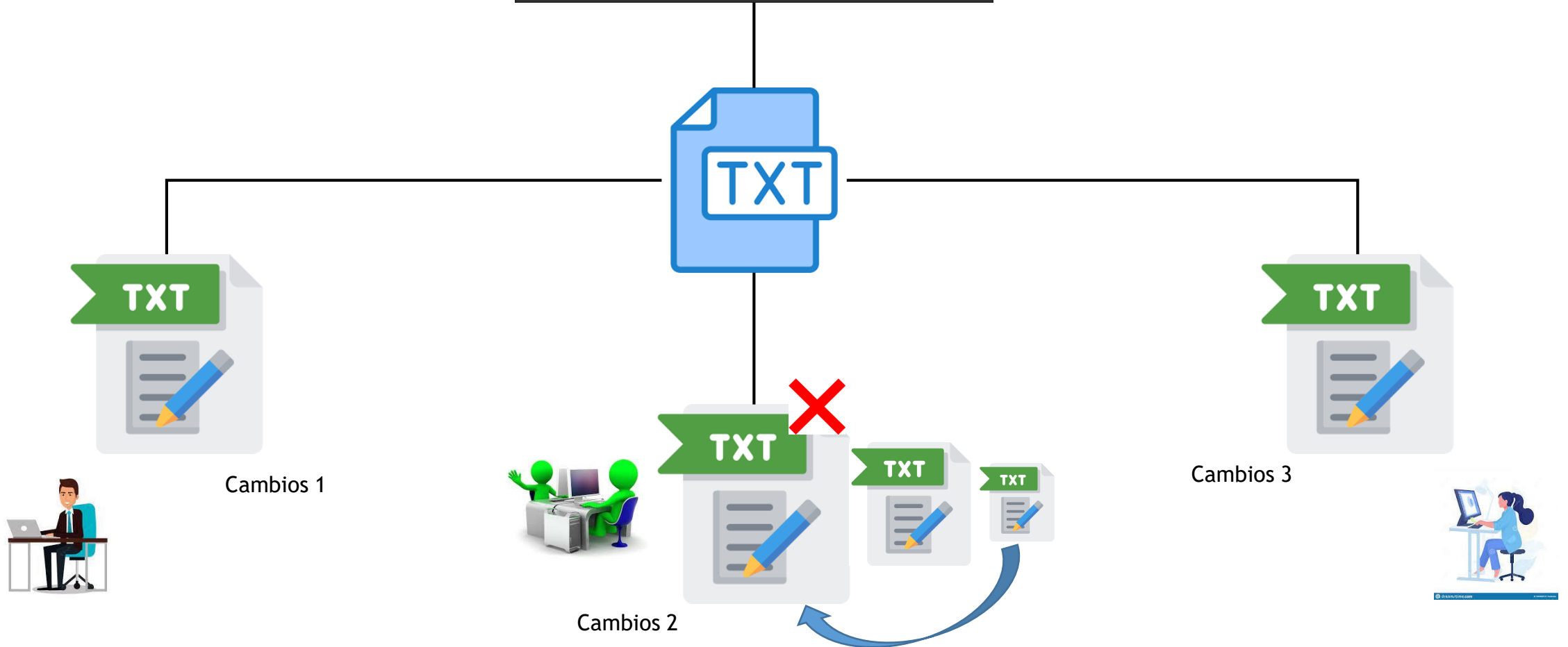


Final\_5











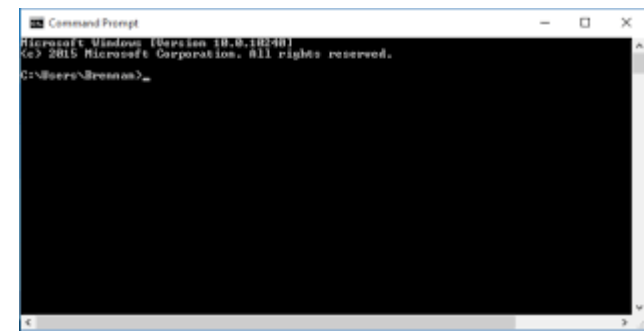


```
$ gh pr status

Current branch
  There is no pull request associated with [develop]

Created by you
#1011 Update readme [readme-fix]
  - Checks pending - Review required

Requesting a code review from you
#1015 Improve error handling [better-error-handling]
  ✓ Checks passing  ✓ Changes requested
```







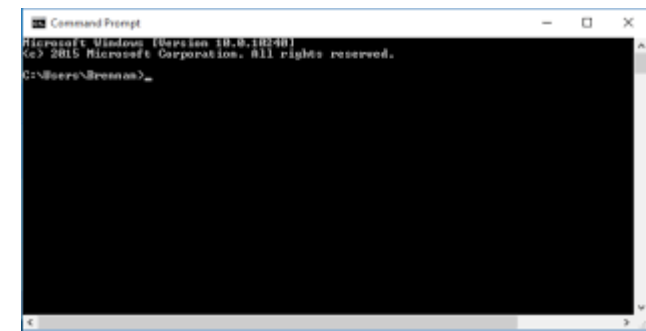
merge  
pull  
add  
commit

```
$ gh pr status

Current branch
  There is no pull request associated with [develop]

Created by you
  #1011 Update readme [readme-fix]
  - Checks pending - Review required

Requesting a code review from you
  #1015 Improve error handling [better-error-handling]
  ✓ Checks passing - Changes requested
```



Colaborar con  
otros

Compartir tu  
código



Colaborar con  
otros

Compartir tu  
código



RED SOCIAL

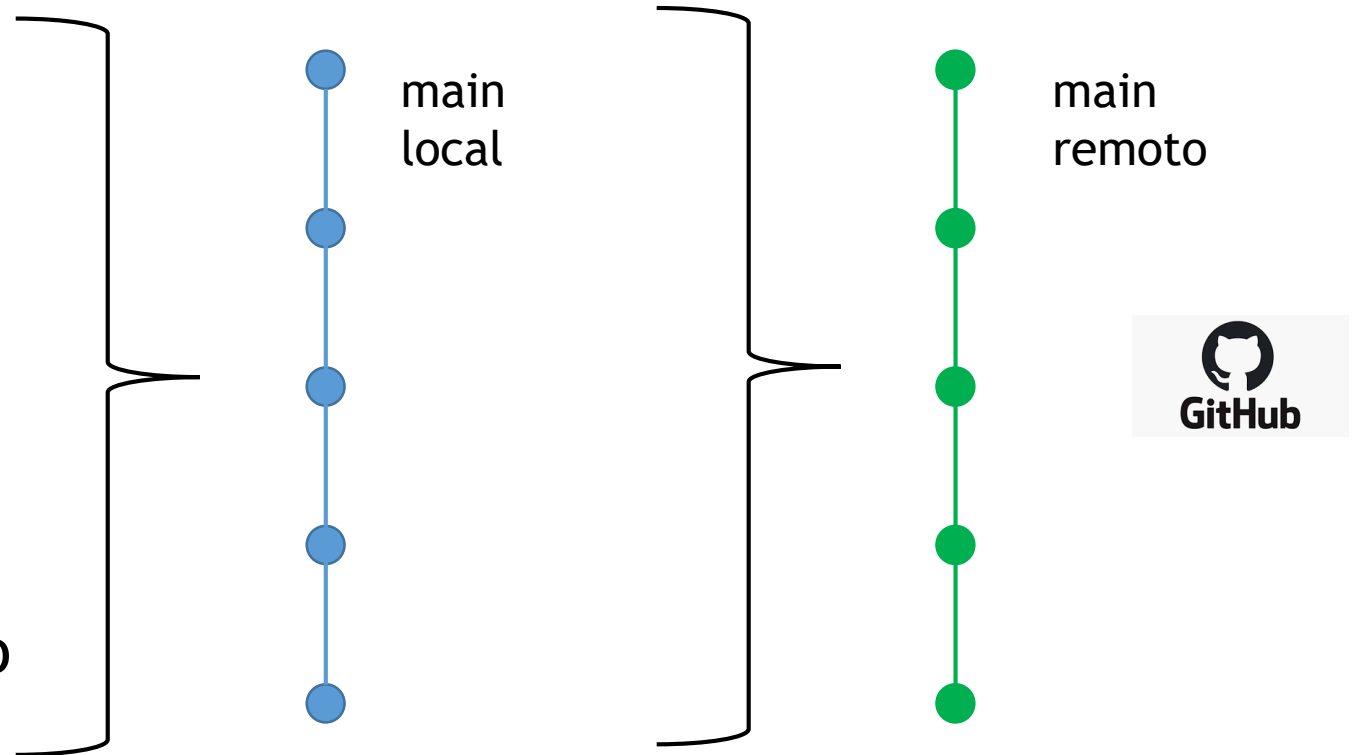
Sistema como Facebook  
o Twitter.

Guarda tu proyecto  
Guarda tus cambios  
Guarda cada versión

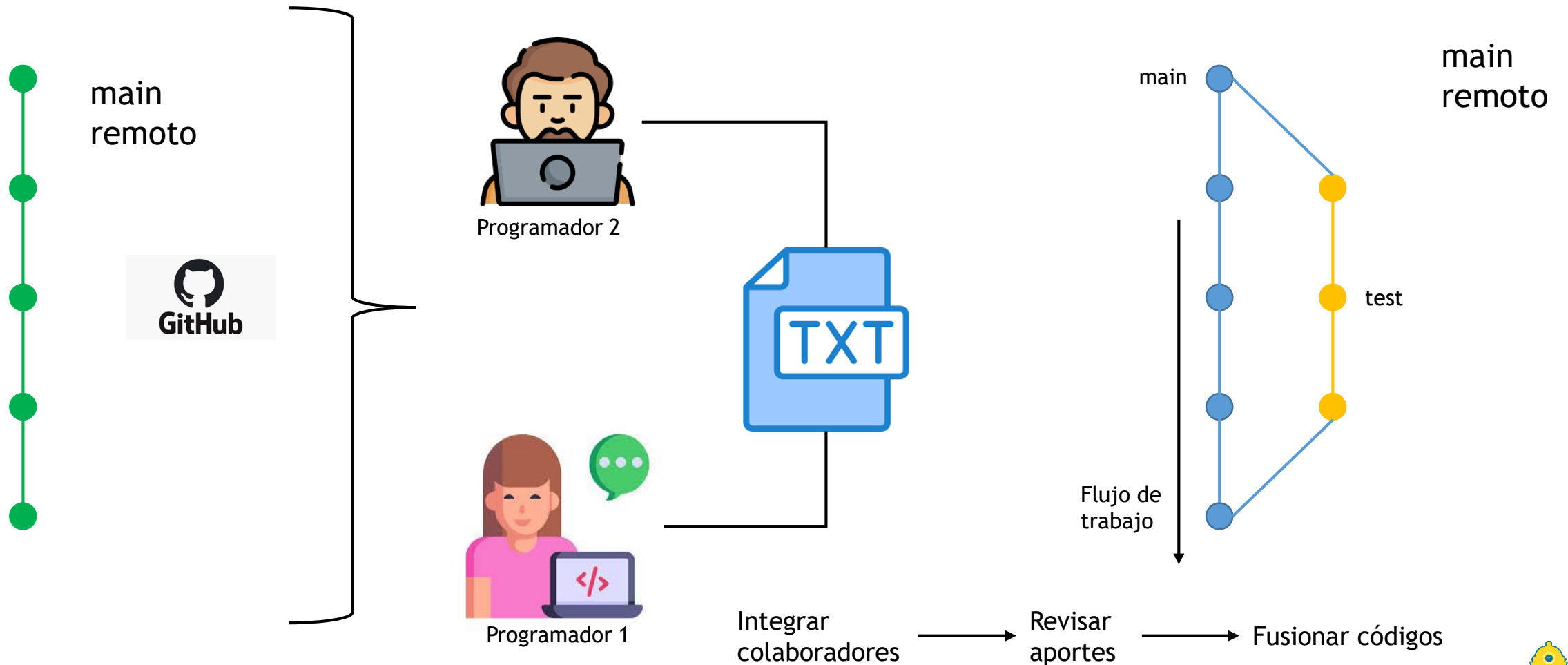


# Al finalizar el curso

- Instalar git
- Crear un repositorio local
  - Cambiarlo
  - Dañarlo
  - Crear ramas
  - Fusionarlas
  - Arreglar cambios
  - Dominar el flujo de trabajo



# Al finalizar el curso

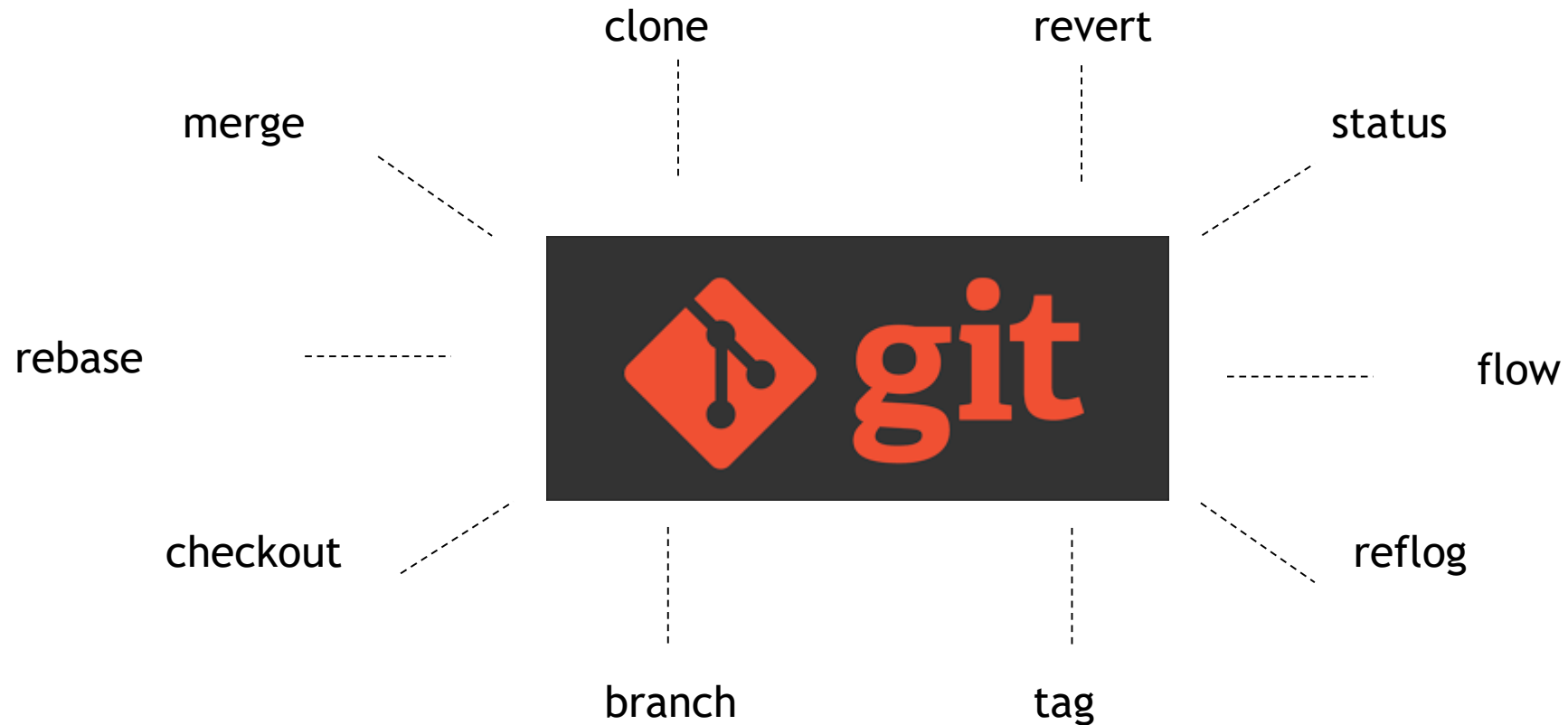


# ¿Cual es el comienzo de todo?

- Crear un directorio
  - Volverlo un repositorio de git
  - Añadir tus archivos
  - Confirmar los cambios en el repositorio
  - Enviar los cambios a la nube
- mkdir proyecto  
cd proyecto
  - git init
  - git add archivo.m
  - git commit -m “Crear archivo.m”
  - git push -u origin main



# Recomendación



# ¿Por qué usar un sistema de control de versiones como Git?

Mi nombre es Oscar Morales.

Soy el CTO de DSVcode y en mis tiempos libres juego futbol, leo libros y realizo algunos tutoriales.

V1

Mi nombre es Oscar Morales.

Soy el CEO de DSVcode y en mis tiempos libres corro carreras, leo libros y realizo algunos tutoriales en youtube.

V2

# ¿Por qué usar un sistema de control de versiones como Git?

Mi nombre es Oscar Morales.

Soy el CTO de DSVcode y en mis tiempos libres juego fútbol, leo libros y realizo algunos tutoriales.

V1

Mi nombre es Oscar Morales.

Soy el CEO de DSVcode y en mis tiempos libres corro carreras, leo libros y realizo algunos tutoriales en youtube.

V2

¿Existirá alguna forma de como guardar aquellos cambios y no todo el archivo?

# ¿Por qué usar un sistema de control de versiones como Git?

- Los sistemas de control de versiones, guardan apenas los cambios generados y no todo el documento, dejando claro:
  - Donde ocurrieron
  - Cuando ocurrieron
  - Quien los hizo
  - Viajar en el pasado
- Git fue creado por la fundación Linux, particularmente por Linus Torvalds.
- El control de código fuente, permite rastrear y gestionar los cambios en el código de software.

# Git

- Sistema de control de versiones distribuido, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.
  - Es optimizado para guardar cambios de forma incremental.
  - Permite contar con un historial, regresar a una versión anterior y agregar funcionalidades.
  - Lleva un registro de los cambios que otras personas realicen en los archivos.

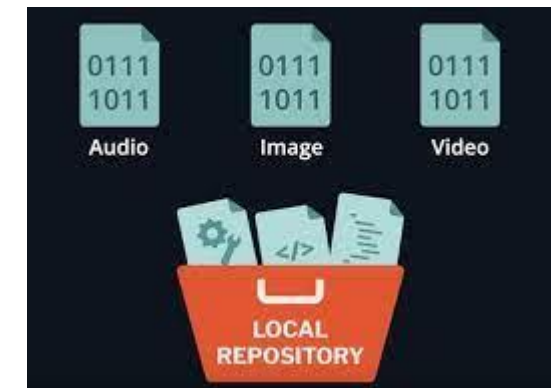
# Git

- Git fue diseñado para operar en un entorno Linux.
- Actualmente, es multiplataforma, es decir, es compatible con Linux, MacOS y Windows.
- En la máquina local se encuentra Git, se utiliza bajo la terminal o línea de comandos y tiene comandos como *merge*, *pull*, *add*, *commit* y *rebase*, entre otros.



# ¿Para qué proyectos sirve Git?

- Con Git se obtiene una mayor eficiencia usando archivos de texto plano, ya que con archivos binarios no puede guardar solo los cambios, sino que debe volver a grabar el archivo completo ante cada modificación, por mínima que sea, lo que hace que incremente demasiado el tamaño del repositorio.



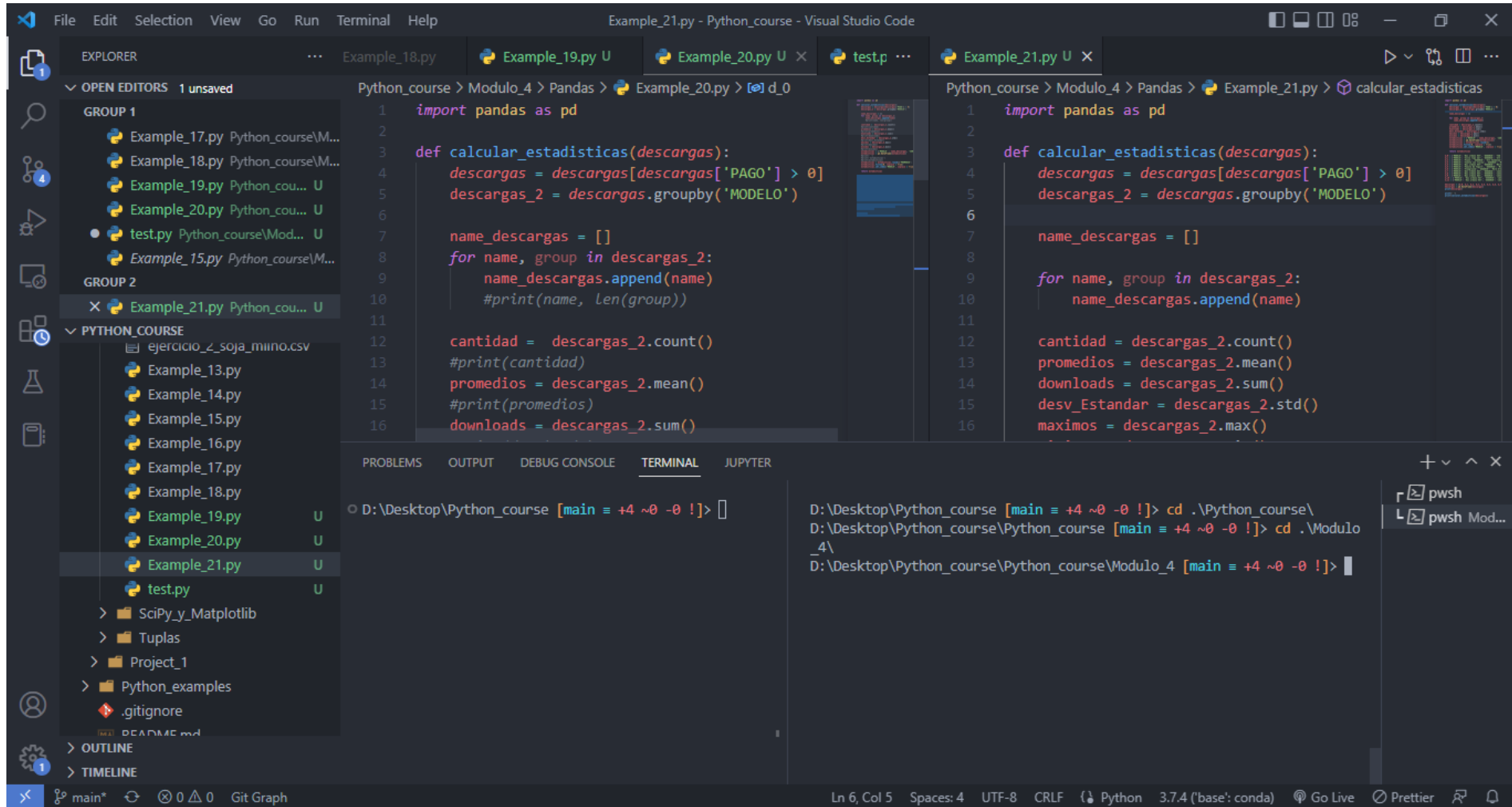
# Editor de código - IDE



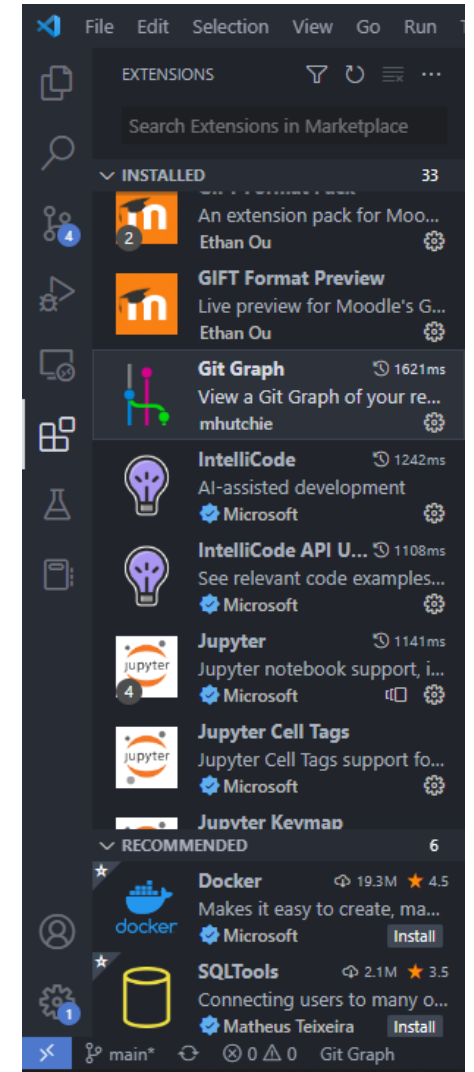
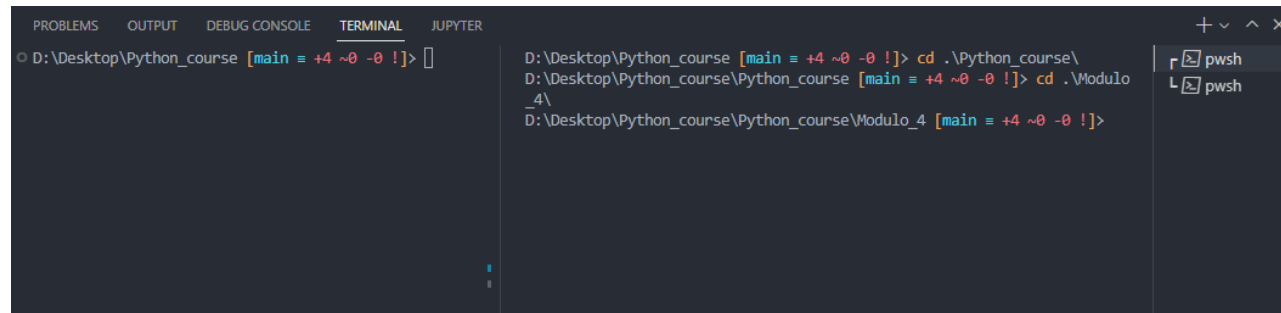
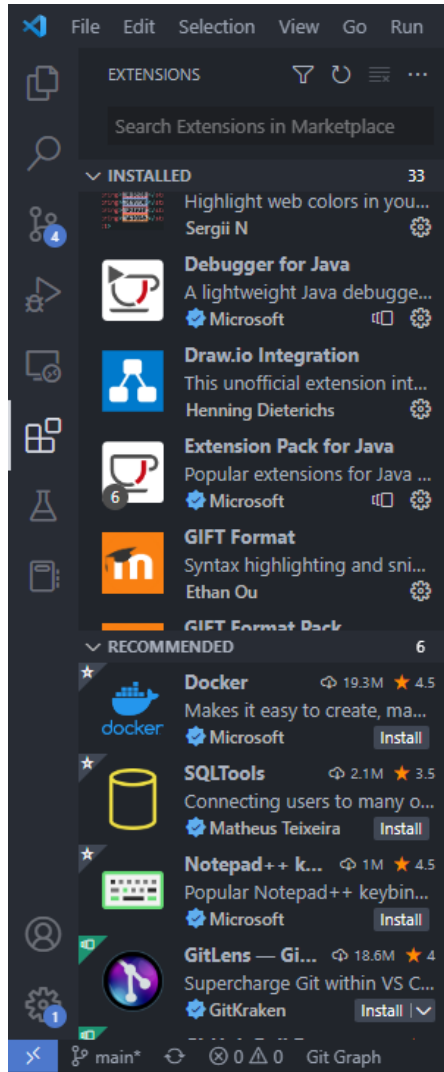
Un entorno de desarrollo integrado o entorno de desarrollo interactivo, es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.



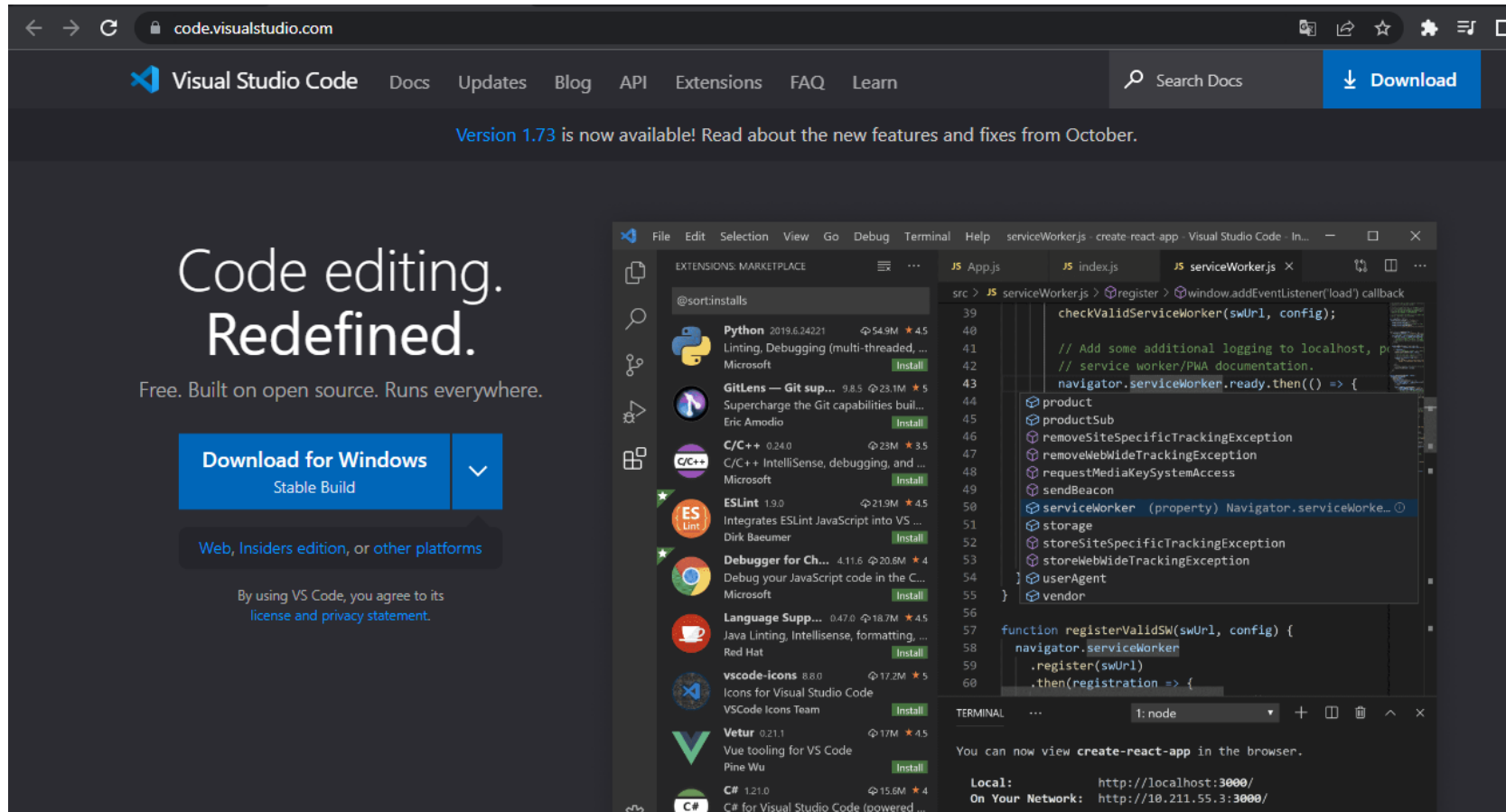
# Editor de código - IDE



# Editor de código - IDE



# Instalación Visual Studio Code



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

# Instalación PowerShell v7.3.0

```
Admin: Python_course [main] X + v
D:\Desktop\Python_course [main ≡ +4 ~0 -0 !]> ls

Directory: D:\Desktop\Python_course

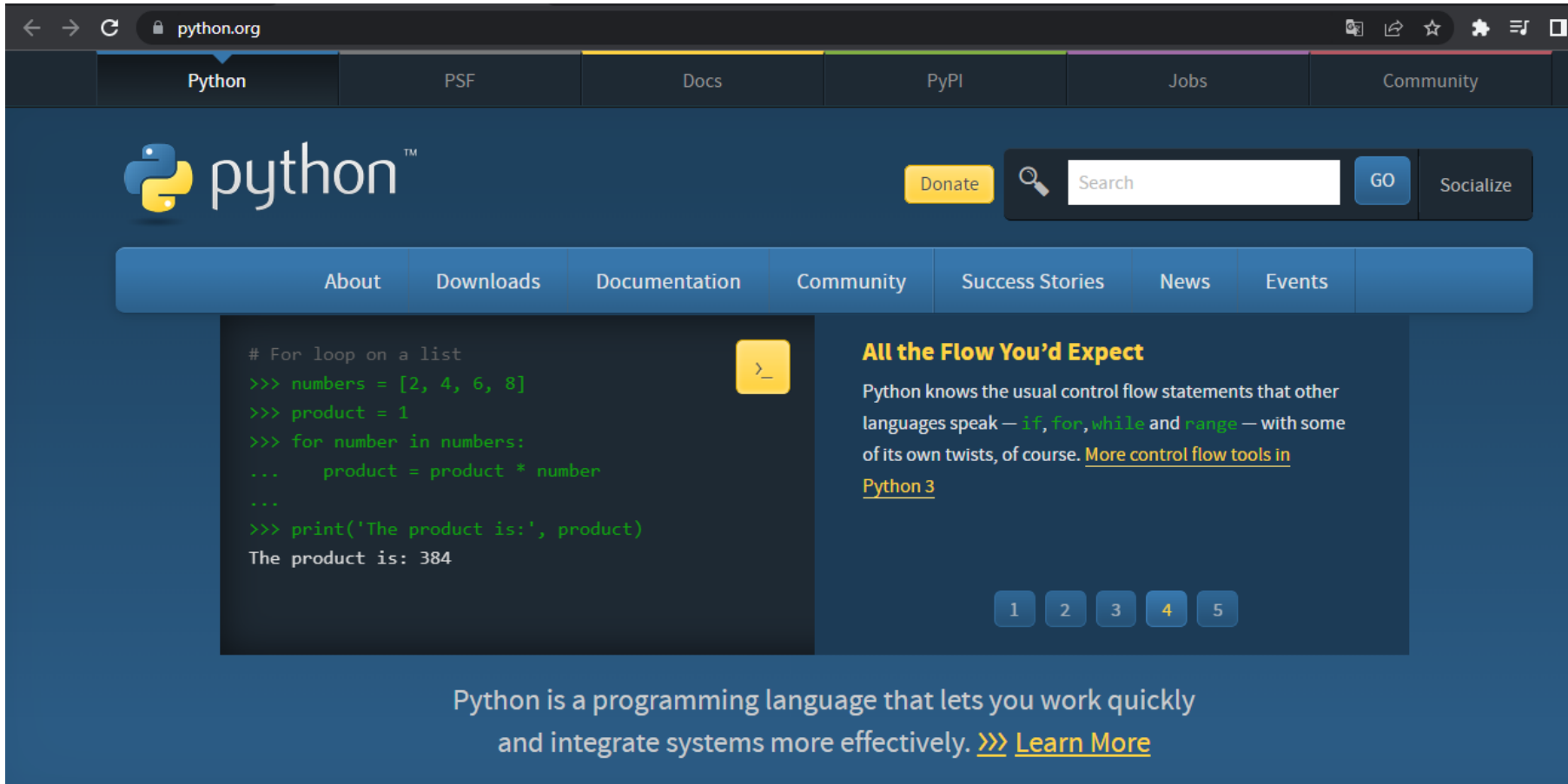
Mode                LastWriteTime         Length Name
----                -
d----             27/10/2022    07:52         Python_course
d----             03/10/2022    17:18         Python_examples
-a---             05/10/2022    16:00            11 .gitignore
-a---             29/10/2022    10:23            82 README.md

D:\Desktop\Python_course [main ≡ +4 ~0 -0 !]> |
```

<https://github.com/PowerShell/PowerShell/releases/tag/v7.3.0>



# Instalación Python



The screenshot shows the Python.org website. At the top, there's a navigation bar with links to Python, PSF, Docs, PyPI, Jobs, and Community. Below this is a dark blue header with the Python logo, a 'Donate' button, a search bar, and a 'Socialize' button. A secondary navigation bar contains links to About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area features a code snippet on the left, a yellow terminal icon, and an article titled 'All the Flow You'd Expect' on the right. The code snippet demonstrates a for loop that calculates the product of numbers in a list. The article text explains that Python supports standard control flow statements like if, for, while, and range, along with some unique features. At the bottom of the main content area, there are five numbered buttons (1-5). Below the main content, a footer states: 'Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)'.

```
# For loop on a list
>>> numbers = [2, 4, 6, 8]
>>> product = 1
>>> for number in numbers:
...     product = product * number
...
>>> print('The product is:', product)
The product is: 384
```

**All the Flow You'd Expect**

Python knows the usual control flow statements that other languages speak — `if`, `for`, `while` and `range` — with some of its own twists, of course. [More control flow tools in Python 3](#)

1 2 3 4 5

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)

<https://www.python.org/>



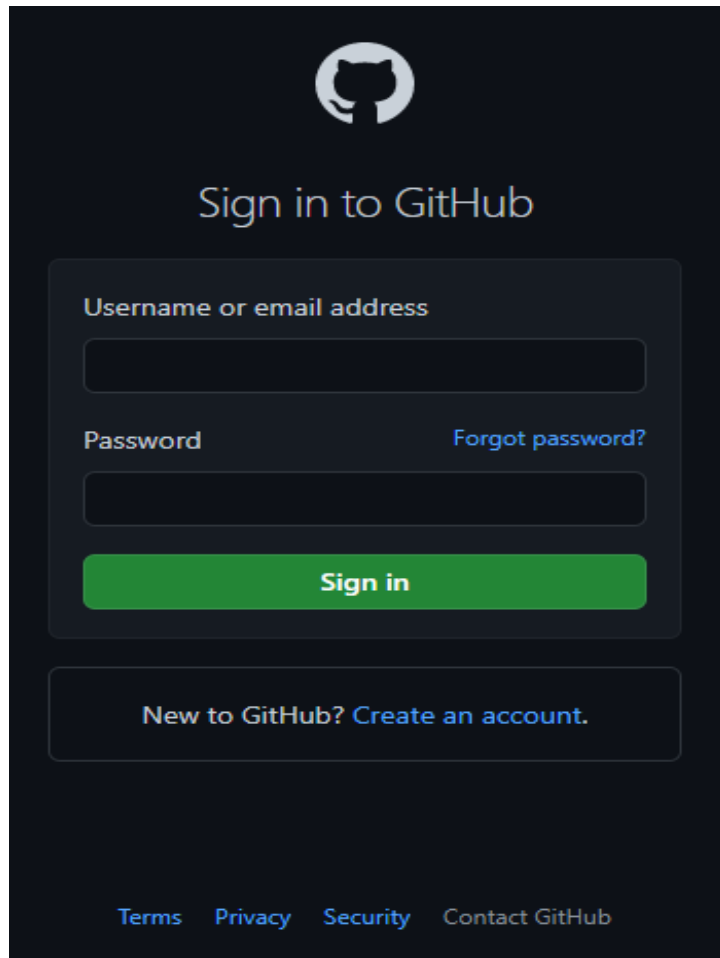
<https://www.python.org/>

# Instalación Matlab

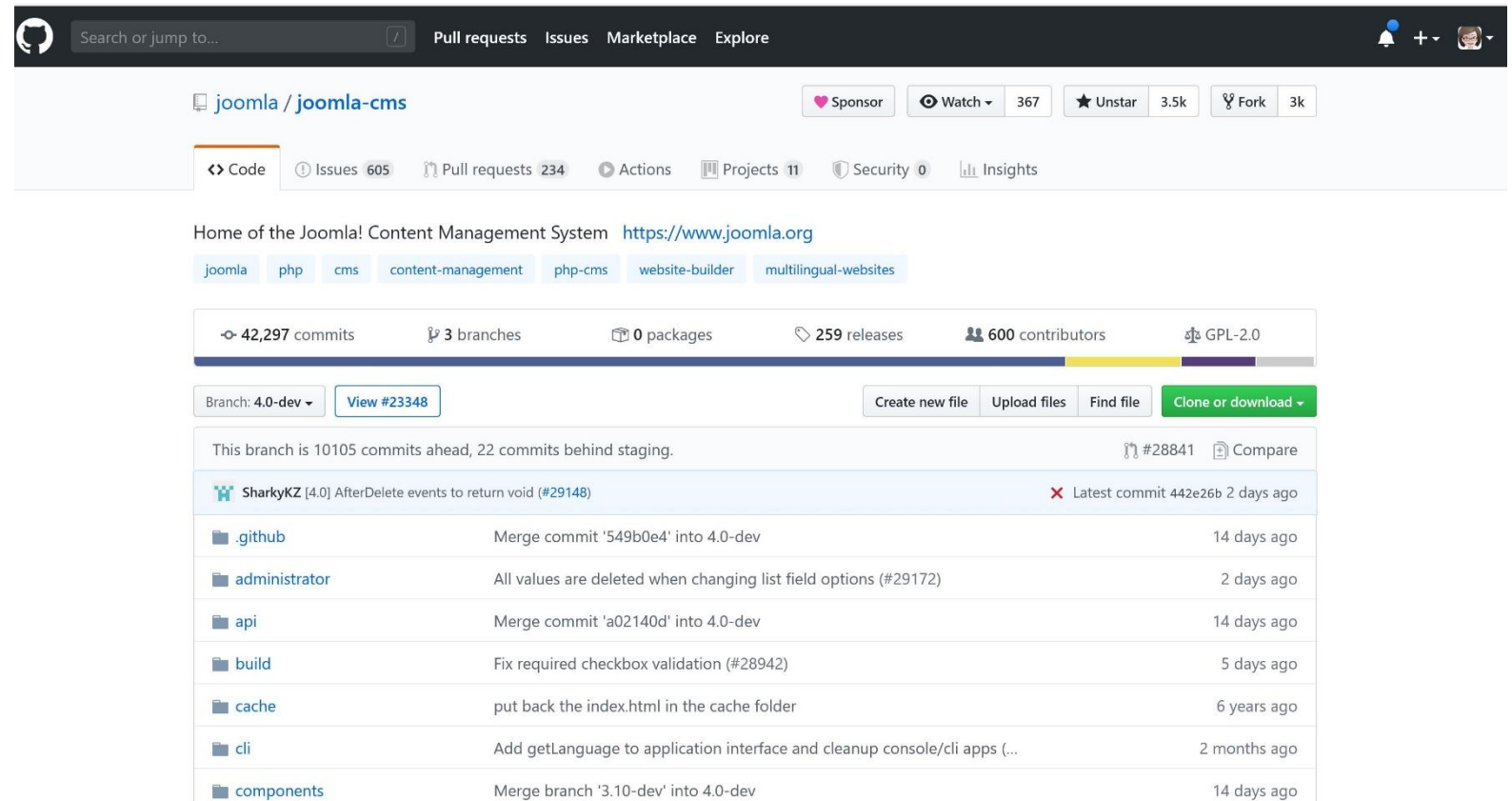


<https://la.mathworks.com/>

# Crear cuenta GitHub



The image shows the GitHub sign-in page. At the top is the GitHub logo. Below it, the text "Sign in to GitHub" is displayed. There are two input fields: "Username or email address" and "Password". To the right of the password field is a link "Forgot password?". Below the input fields is a green "Sign in" button. At the bottom, there is a link "New to GitHub? Create an account." and a footer with links for "Terms", "Privacy", "Security", and "Contact GitHub".



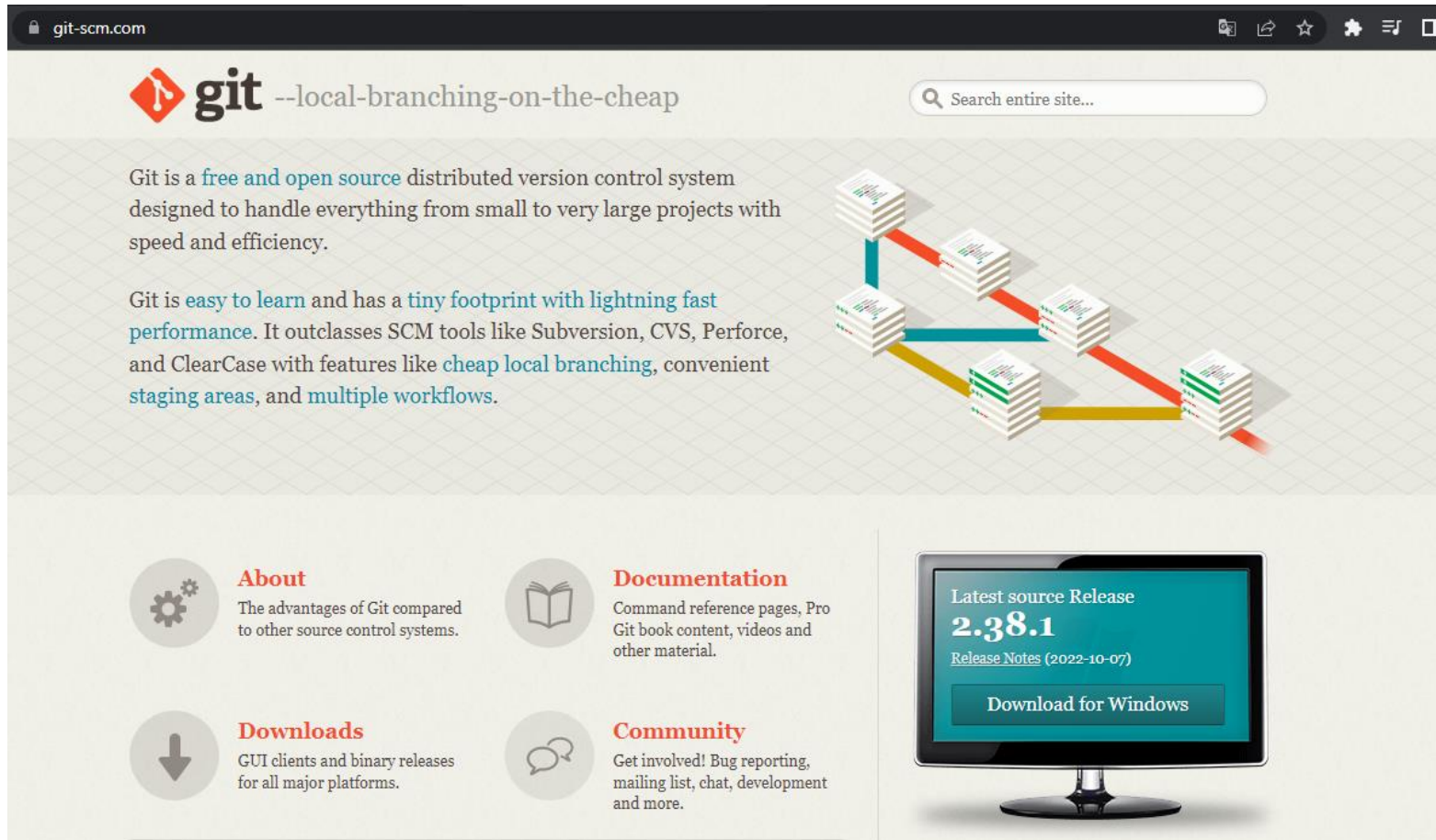
The image shows the GitHub repository page for Joomla! CMS. The repository is named "joomla / joomla-cms". It has 42,297 commits, 3 branches, 0 packages, 259 releases, 600 contributors, and is licensed under GPL-2.0. The page shows the "4.0-dev" branch, which is 10105 commits ahead and 22 commits behind staging. A table of recent commits is displayed, including a merge commit for ".github" and a commit for "administrator".

Commit	Message	Time
#28841	Merge commit '549b0e4' into 4.0-dev	14 days ago
#29172	All values are deleted when changing list field options	2 days ago
#28942	Merge commit 'a02140d' into 4.0-dev	14 days ago
#28942	Fix required checkbox validation	5 days ago
#28942	put back the index.html in the cache folder	6 years ago
#28942	Add getLanguage to application interface and cleanup console/cli apps (...)	2 months ago
#28942	Merge branch '3.10-dev' into 4.0-dev	14 days ago

<https://github.com/login>



# Instalación Git



The screenshot shows the Git website homepage. At the top, the browser address bar displays "git-scm.com". The main header features the Git logo (a red diamond with a white 'g') and the tagline "--local-branching-on-the-cheap". A search bar is located to the right of the header. The main content area has a light gray background with a subtle grid pattern. It contains two paragraphs of text describing Git as a free and open source distributed version control system, highlighting its speed, efficiency, ease of learning, and tiny footprint. To the right of the text is a 3D illustration of several stacks of books connected by colored lines (red, yellow, blue) representing a branching model. Below the text, there are four circular icons with corresponding text: "About" (gears icon), "Documentation" (book icon), "Downloads" (downward arrow icon), and "Community" (speech bubbles icon). On the right side of the page, there is a monitor displaying the "Latest source Release 2.38.1" and a button labeled "Download for Windows".

git-scm.com

git --local-branching-on-the-cheap

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.

**About**  
The advantages of Git compared to other source control systems.

**Documentation**  
Command reference pages, Pro Git book content, videos and other material.

**Downloads**  
GUI clients and binary releases for all major platforms.

**Community**  
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release  
**2.38.1**  
[Release Notes \(2022-10-07\)](#)  
[Download for Windows](#)

<https://git-scm.com/>

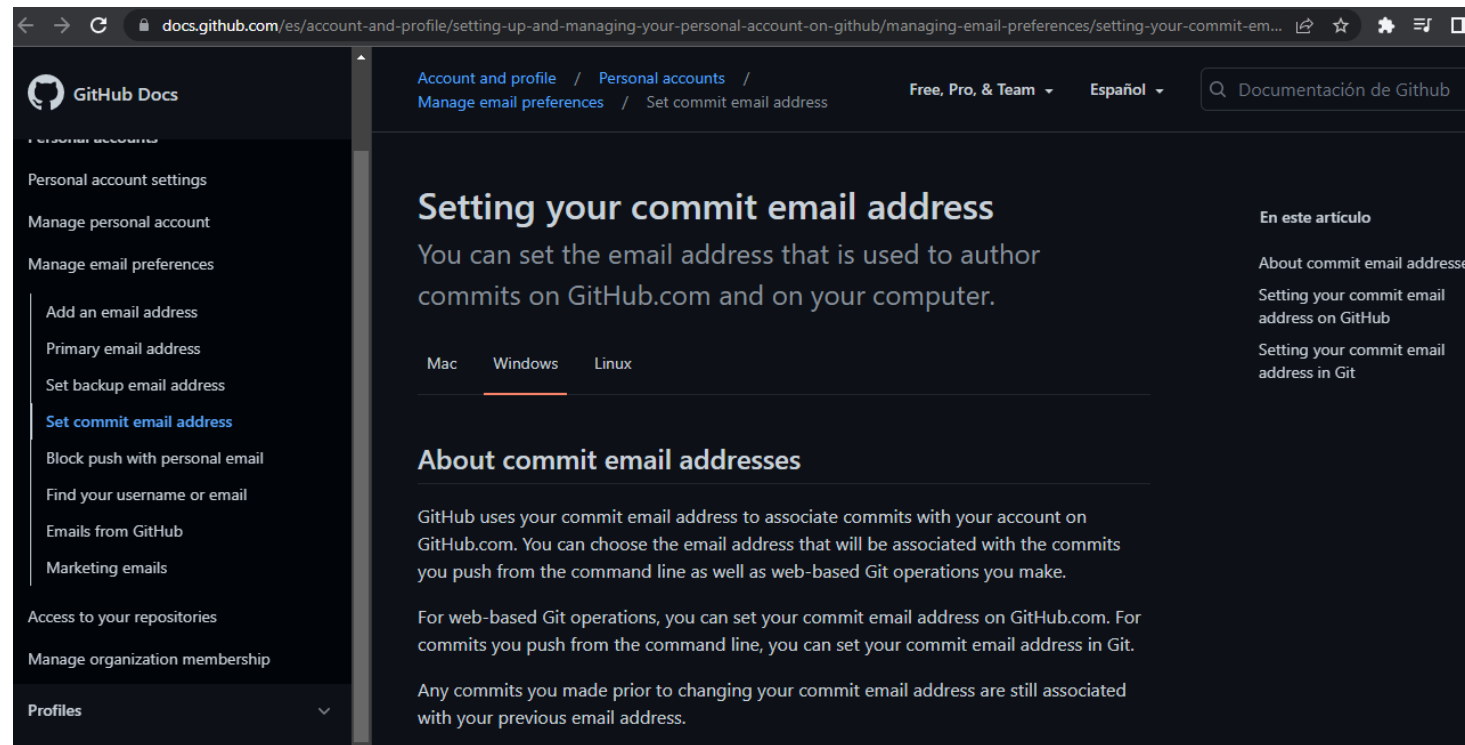
# Configurar nombre de usuario en Git



<https://docs.github.com/es/get-started/getting-started-with-git/setting-your-username-in-git>

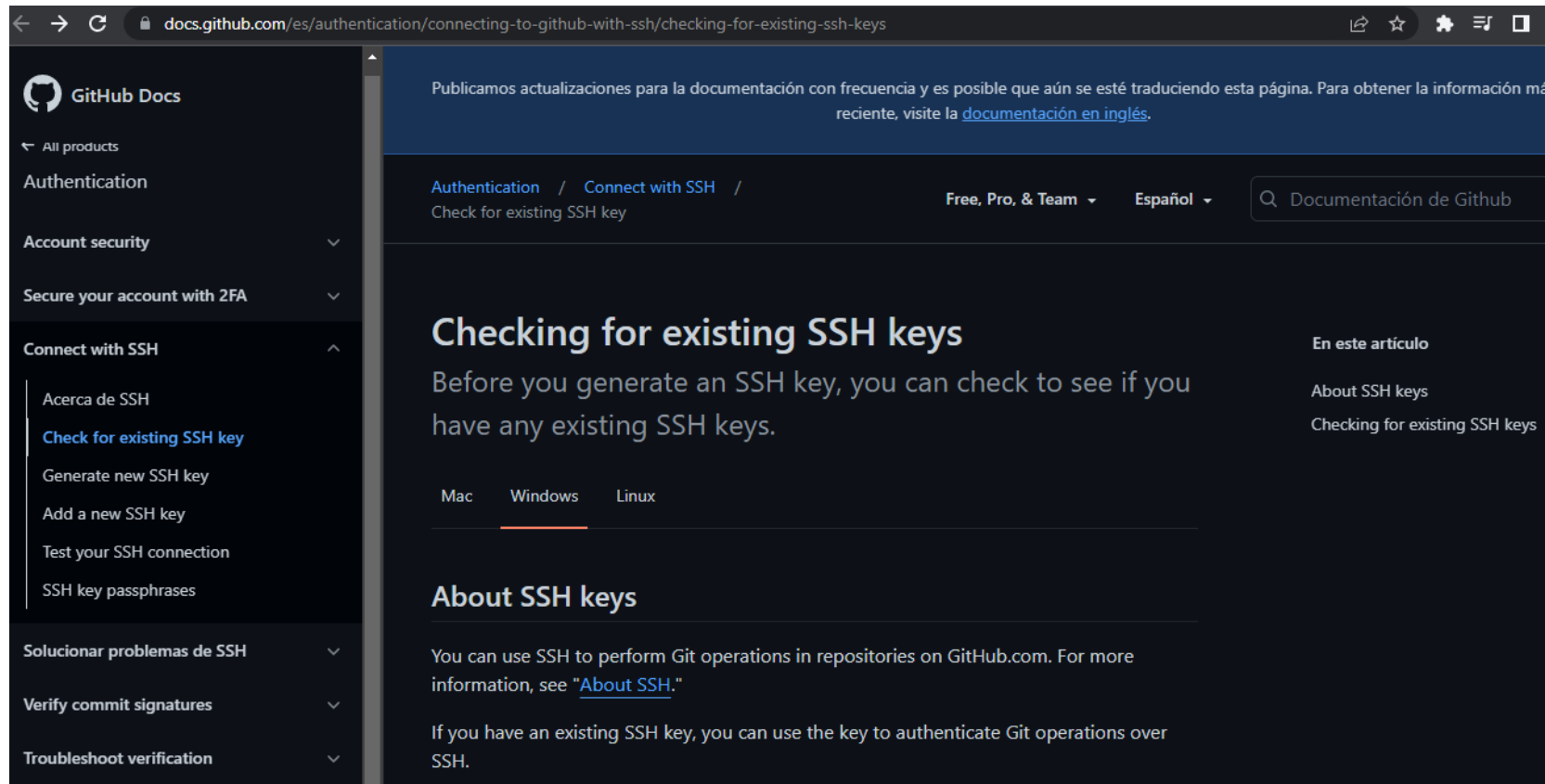


# Configurar dirección de correo electrónico en Git



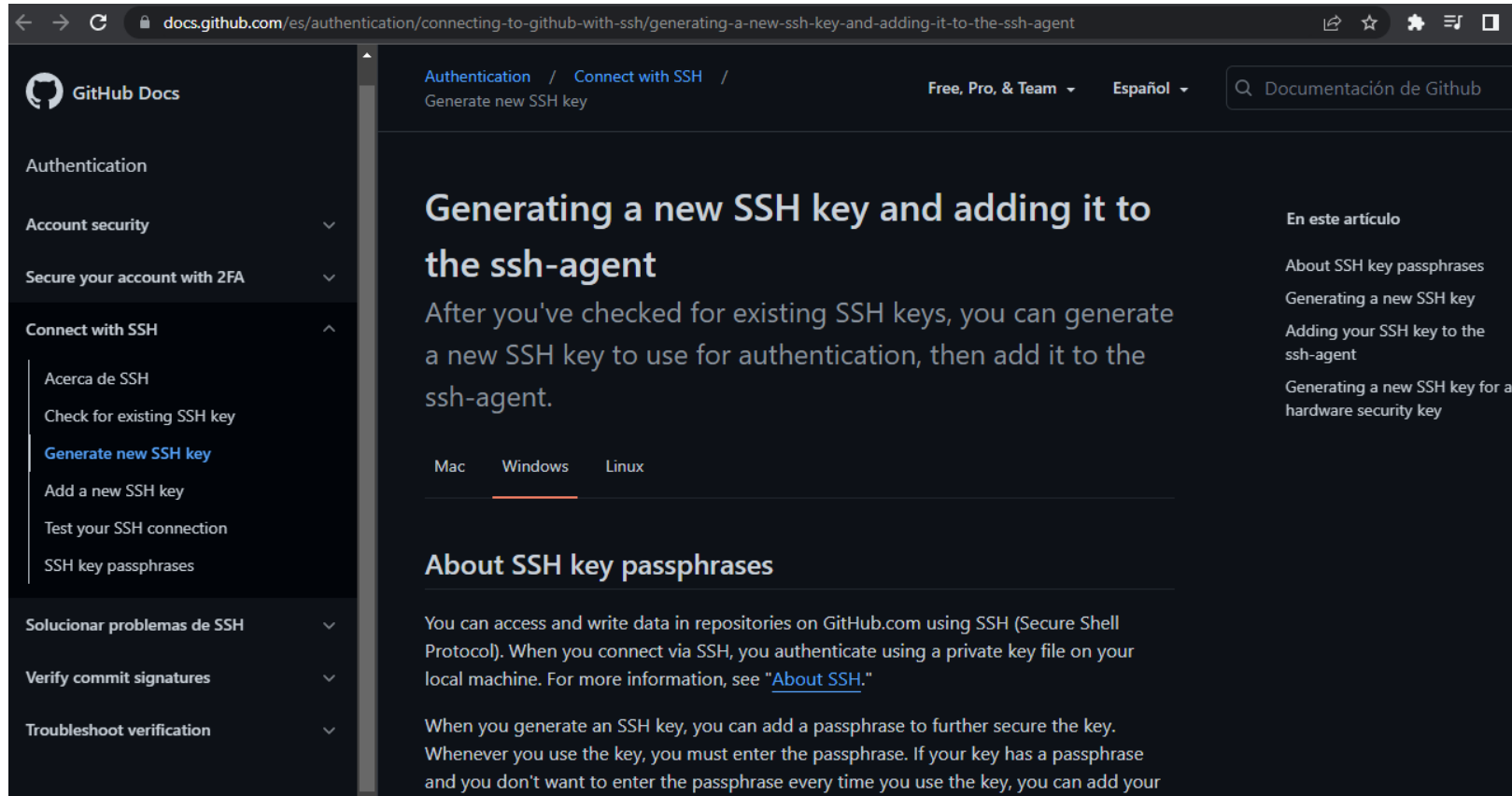
<https://docs.github.com/es/account-and-profile/setting-up-and-managing-your-personal-account-on-github/managing-email-preferences/setting-your-commit-email-address>

# Comprobar tus llaves SSH existentes



<https://docs.github.com/es/authentication/connecting-to-github-with-ssh/checking-for-existing-ssh-keys>

# Generar una nueva clave SSH



The screenshot shows the GitHub Docs website in Spanish. The left sidebar contains a navigation menu with categories like 'Authentication', 'Account security', 'Secure your account with 2FA', 'Connect with SSH', 'Solucionar problemas de SSH', 'Verify commit signatures', and 'Troubleshoot verification'. The 'Connect with SSH' section is expanded, showing sub-links: 'Acerca de SSH', 'Check for existing SSH key', 'Generate new SSH key' (which is highlighted), 'Add a new SSH key', 'Test your SSH connection', and 'SSH key passphrases'. The main content area is titled 'Generating a new SSH key and adding it to the ssh-agent'. It includes a breadcrumb trail 'Authentication / Connect with SSH / Generate new SSH key', a language selector set to 'Español', and a search bar. The article text explains that after checking for existing SSH keys, a new one can be generated and added to the ssh-agent. Below the text are tabs for 'Mac', 'Windows' (which is selected), and 'Linux'. A section titled 'About SSH key passphrases' follows, explaining that a passphrase can be added to the key for extra security.

docs.github.com/es/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent

Authentication / Connect with SSH / Generate new SSH key

Free, Pro, & Team Español Documentación de Github

## Generating a new SSH key and adding it to the ssh-agent

After you've checked for existing SSH keys, you can generate a new SSH key to use for authentication, then add it to the ssh-agent.

Mac Windows Linux

### About SSH key passphrases

You can access and write data in repositories on GitHub.com using SSH (Secure Shell Protocol). When you connect via SSH, you authenticate using a private key file on your local machine. For more information, see ["About SSH."](#)

When you generate an SSH key, you can add a passphrase to further secure the key. Whenever you use the key, you must enter the passphrase. If your key has a passphrase and you don't want to enter the passphrase every time you use the key, you can add your

En este artículo

- About SSH key passphrases
- Generating a new SSH key
- Adding your SSH key to the ssh-agent
- Generating a new SSH key for a hardware security key

<https://docs.github.com/es/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

# Posh-git

- ABCD representam o índice; | ( DelimStatus ); EFGH representa o diretório de trabalho
  - + = Arquivos adicionados
  - ~ = arquivos modificados
  - - = arquivos removidos
  - ! = Arquivos em conflito
  - Assim como na `git status` saída, o status do índice é exibido em verde escuro e o status do diretório de trabalho em vermelho escuro
- W representa o status geral do diretório de trabalho
  - ! = Existem alterações não preparadas na árvore de trabalho ( `LocalWorkingStatusSymbol` )
  - ~ = Há alterações não confirmadas, ou seja, alterações em estágio na árvore de trabalho aguardando para serem confirmadas ( `LocalStagedStatusSymbol` )
  - Nenhum = Não há alterações não confirmadas ou não confirmadas na árvore de trabalho ( `LocalDefaultStatusSymbol` )
- ] ( `AfterStatus` )

Os símbolos e o texto ao redor podem ser personalizados pelas propriedades correspondentes em `$GitPromptSettings`.

Por exemplo, um status de `[main ≡ +0 ~2 -1 | +1 ~1 -0]` corresponde ao seguinte `git status`:

```
# On branch main
#
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   this-changed.txt
```

posh-git é um módulo do PowerShell que integra o Git e o PowerShell, fornecendo informações resumidas do status do Git que podem ser exibidas no prompt do PowerShell, por exemplo:

```
~\GitHub\posh-git [master ≡ +0 ~1 -0 | +0 ~1 -0 !]>
```

```
$ PowerShellGet\Install-Module posh-git -Scope CurrentUser -Force
$ Import-Module posh-git
$ Add-PoshGitToProfile
```

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

# Referencias

- Chacon, S., & Straub, B. *Pro git*. Springer Nature, 2014.
- Platzi.com
- [https://www.youtube.com/watch?v=cYLapo1FFmA&ab\\_channel=doneberDev](https://www.youtube.com/watch?v=cYLapo1FFmA&ab_channel=doneberDev)
- [https://www.youtube.com/watch?v=wHh3IgJvXcE&ab\\_channel=FalconMasters](https://www.youtube.com/watch?v=wHh3IgJvXcE&ab_channel=FalconMasters)
- <https://platzi.com/clases/1557-git-github/20215-que-es-git/>

# Comandos básicos de git

- `git init`: inicializa un repositorio de GIT en la carpeta donde se ejecute el comando.
- `git add`: añade los archivos especificados al área de preparación (staging).
- `git commit -m "commit description"`: confirma los archivos que se encuentran en el área de preparación y los agrega al repositorio.
- `git commit -am "commit description"`: añade al staging area y hace un commit mediante un solo comando. (No funciona con archivos nuevos)
- `git status`: ofrece una descripción del estado de los archivos (untracked, ready to commit, nothing to commit).
- `git rm (. -r, filename) (-cached)`: remueve los archivos del index.
- `git config --global user.email tu@email.com`: configura un email.
- `git config --global user.name <Nombre como se verá en los commits>`: configura un nombre.
- `git config --list`: lista las configuraciones.

# Analizar cambios en los archivos de un proyecto Git

- `git log`: lista de manera descendente los commits realizados.
- `git log --stat`: además de listar los commits, muestra la cantidad de bytes añadidos y eliminados en cada uno de los archivos modificados.
- `git log --all --graph --decorate --oneline`: muestra de manera comprimida toda la historia del repositorio de manera gráfica y embellecida.
- `git show filename`: permite ver la historia de los cambios en un archivo.
- `git diff <commit1> <commit2>`: compara diferencias entre en cambios confirmados.

# Volver en el tiempo con branches y checkout

- `git reset <commit> --soft/hard`: regresa al commit especificado, eliminando todos los cambios que se hicieron después de ese commit.
- `git checkout <commit/branch> <filename>`: permite regresar al estado en el cual se realizó un commit o branch especificado, pero no elimina lo que está en el staging area.
- `git checkout - <filePath>`: deshacer cambios en un archivo en estado modified (que ni fue agregado a staging)



# git rm y git reset

## git rm:

- Este comando nos ayuda a eliminar archivos de Git sin eliminar su historial del sistema de versiones. Esto quiere decir que si necesitamos recuperar el archivo solo debemos “viajar en el tiempo” y recuperar el último commit antes de borrar el archivo en cuestión.
- git rm no puede usarse por sí solo, así nomás. Se debe utilizar uno de los flags para indicar a Git cómo eliminar los archivos que ya no se necesitan en la última versión del proyecto:
  - git rm --cached <archivo/s>: elimina los archivos del área de Staging y del próximo commit, pero los mantiene en nuestro disco duro.
  - git rm --force <archivo/s>: elimina los archivos de Git y del disco duro. Git siempre guarda todo, por lo que podemos acceder al registro de la existencia de los archivos, de modo que podremos recuperarlos si es necesario (pero debemos aplicar comandos más avanzados).

# git rm y git reset

## git reset

- Con git reset volvemos al pasado sin la posibilidad de volver al futuro. Borramos la historia y la debemos sobrescribir.
- git reset --soft: Vuelve el branch al estado del commit especificado, manteniendo los archivos en el directorio de trabajo y lo que haya en staging considerando todo como nuevos cambios. Así podemos aplicar las últimas actualizaciones a un nuevo commit.
- git reset --hard: Borra absolutamente todo. Toda la información de los commits y del área de staging se borra del historial.
- git reset HEAD: No borra los archivos ni sus modificaciones, solo los saca del área de staging, de forma que los últimos cambios de estos archivos no se envíen al último commit. Si se cambia de opinión se los puede incluir nuevamente con git add.

# Ramas o Branches en git

- Al crear una nueva rama se copia el último commit en esta nueva rama. Todos los cambios hechos en esta rama no se reflejarán en la rama master hasta que hagamos un merge.
  - `git branch <new branch>`: crea una nueva rama.
  - `git checkout <branch name>`: se mueve a la rama especificada.
  - `git merge <branch name>`: fusiona la rama actual con la rama especificada y produce un nuevo commit de esta fusión.
  - `git branch`: lista las ramas generadas.

# Programación para ingenieros

Docente: Oscar Stiven Morales Zapata - Sebastian Durango Idarraga

Correo electrónico: [oscars.morales@autonoma.edu.co](mailto:oscars.morales@autonoma.edu.co)

[sebastiandi@autonoma.edu.co](mailto:sebastiandi@autonoma.edu.co)

