

Sesion 1 - Introduccion a Git & GitHub

Nombre del instructor	Adolfo Centeno Tellez		
		Objetivos	Subtemas
Tema 1. Introducción al manejo de repositorios de código en la plataforma Github.	60m	1. Explica conceptos de repositorios de código y su importancia en la creación de productos de software 2. Crea una cuenta en la plataforma github, identifica los principales elementos y aprende como crear un repositorio de código 3. Describe los principales comandos git para el manejo de repositorios.	1.1 Introducción y conceptos de repositorios de código 1.2 Crear una cuenta en github y creación de repositorios de código. 1.3 Comandos básicos en git para el manejo de repositorios

TEMA 1: Introducción al manejo de repositorios de código en la plataforma Github

Introducción al tema

¿Sabías que las plataformas de internet como facebook, google, youtube están compuestas por millones de líneas de código ? En efecto, si en tu laptop o dispositivo móvil ejecutas algunas de estas aplicaciones estas usando código realizado por cientos o miles de ingenieros de software organizados y distribuidos por todo el mundo. Por ejemplo, el sistema operativo Android está compuesto de aproximadamente 12 millones de líneas de código, como es posible controlar tal cantidad de código, cuantos ingenieros colaborar al mismo tiempo, en cuantos países se encuentran y cómo manejan las distintas versiones sin crear conflictos

Como **Ingeniero de Software**, es importante conocer los fundamentos de herramientas para almacenar código de forma segura y eficiente, crear versiones del mismo, trabajar en equipos remotos y administrar tu proyecto de software, para generar valor tanto en tu vida personal como profesional.

Al terminar este tema tendrás una perspectiva muy clara de qué es una herramienta para almacenar código, serás capaz de crear tus repositorios de código, descargarlos de forma local, agregar cambios y subir versiones. Asimismo, tendrás un panorama más claro de las aplicaciones de las herramientas de versionamiento de código en la administración de proyectos con equipos remotos y su contribución en los proyectos de open source actuales. Entonces, ¿estás preparado para iniciar este interesante viaje en uno de los retos tecnológicos más relevantes en la ingeniería de software?

SUBTEMA 1. ¿Introducción y conceptos de repositorios de código?

Durante los últimos 3 años el uso de sitios web de comercio electrónico creció exponencialmente, así como las aplicaciones móviles para pedidos a domicilio, esto debido a la pandemia y a otros factores como la inminente llegada de la nueva red 5g, las nuevas características del ecommerce 3.0 como son el uso de criptomonedas, los metaversos, la realidad aumentada, entre otros.

Imagina que tu empresa/universidad necesita implementar una solución de software para renovar el portal de ventas en línea para agregar nuevas características que la hagan más competitivo, además de permitir entregar a domicilio y migrar la página de un hosting tradicional a una nube de cómputo, y tu eres contactado para liderar ese proyecto, luego entonces deberás resolver algunas interrogantes tales como:

- Escoger la tecnología que usarás para la creación del nuevo sitio web tomando en cuenta las tendencias actuales y herramientas existentes
- Seleccionar la base de datos donde se migrarán los datos del sistema actual
- Escoger la nube cómputo donde se desplegará el nuevo sitio, que sea escalable y permita agregar nuevas funcionalidades
- Investigar cómo almacenar de forma eficiente el código fuente del proyecto, que dicho sea de paso será elaborado por un equipo de 3 ingenieros y el líder de proyecto.
- Investigar cómo preparar ese código fuente para desplegarlo en un servidor de prueba con versiones del proyecto cada 2 semanas para así obtener retroalimentación de los usuarios.

Siendo tú la/el experto en ingeniería de software, los directivos de la empresa te piden asegurar el éxito del proyecto y cumplir con los presupuestos de tiempo, costo y calidad esperados.

El proyecto arranca con una sesión inicial con los accionistas de la empresa, donde lo primero que tienes que explicar es **cómo asegurar entregas frecuentes de proyecto cada 2 semanas** para obtener puntos de vista del diseño y funcionalidad en etapas tempranas del proyecto, además de asegurar como llevaras ese código a un sitio web real en modo producción.

De entrada inicias comentando que existen Repositorios de código que son plataformas en internet donde se almacena de forma segura el código fuente de un proyecto y que permite controlar las diferentes versiones de la aplicación, disponiendo de un historial con los cambios realizados sobre el original y sobre cada nueva versión. Además de que almacenando el código en una plataforma de este tipo llevarlo a un servidor real o una nube es sumamente sencillo.

Hasta este momento, todos los presentes en la reunión de arranque están de acuerdo y todo va viento en popa. Sin embargo, existe un momento de tensión al momento de proponer la solución, esto porque un miembro de la junta cuestiona cómo harás para administrar las tareas entre los 3 ingenieros, como asignaturas las tareas de cada uno de ellos y cómo se integrarán su trabajo en una sola versión en cada entrega quincenal.

Entonces comentas, que las herramientas de versionamiento de código, permiten crear ramas o “branches” de código base que puede ser distribuido a cada ingeniero y que cada uno de ellos puede crear sub-ramas donde pueden aportar su trabajo sin afectar el código principal. Posteriormente podrían integrar su trabajo en el código base sin crear conflictos.

¿Sabías que las herramientas de versionamiento de código más populares actualmente son Bitbucket, Gitlab y Github?

Nota: Una herramienta de versionamiento de código usa el sistema de control de versiones **Git** diseñado por Linus Torvalds el creador del sistema operativo Linux. En un sistema de gestión de versiones los desarrolladores pueden administrar su proyecto, ordenar el código de cada una de las nuevas versiones y así evitar confusiones. Así, al tener copias de cada una de las versiones de su aplicación, no se perderán los estados anteriores cuando se actualiza el código.

- Bitbucket (<https://bitbucket.org/>) es una herramienta de alojamiento de código y colaboración basada en Git diseñada para equipos. Se integra con las herramientas Jira y Trello y están concebidas para unir a todo el equipo de software con el fin de poner en práctica un proyecto. Ofrece un lugar en el que tu equipo puede colaborar con código desde el concepto hasta la nube, crear código de calidad mediante pruebas automatizadas e implementar código con total seguridad.

- Gitlab (<https://about.gitlab.com/>) Gitlab es un servicio web de control de versiones y desarrollo de software colaborativo basado en Git. Además de gestor de repositorios, el servicio ofrece también alojamiento de wikis y un sistema de seguimiento de errores, todo ello publicado bajo una Licencia de código abierto
- Github (<https://github.com>) es un sitio "social coding". Te permite subir repositorios de código **para** almacenarlo en el sistema de control de versiones **Git**.

Como dato curioso, GitHub creó un proyecto llamado Github Copilot (<https://copilot.github.com/>) es una herramienta de inteligencia artificial desarrollada por GitHub y OpenAI. Este sistema es capaz de generar código de forma autónoma y ayudar así a los desarrolladores a ahorrar tiempo y ser más eficientes. GitHub Copilot extrae el contexto de los comentarios, el código, y sugiere líneas individuales y funciones completas al instante.

Sin embargo otro directivo te cuestiona sobre la administración de los requerimientos del proyecto y cómo garantizamos el cumplimiento de cada uno de ellos, como organizaras prioridades, como establecemos las metas y fechas de entrega y que todo eso se vuelva código

Dada tu experiencia en Ciencia de Datos y proyectos de software, sabes que la mejor alternativa para abordar este proyecto es haciendo uso de un sistema de administración de proyectos basado en kanban, con automatización de tareas, issues, branches y para esto mencionas los siguientes puntos:

- La plataforma Github te permite administrar los proyectos de software en tableros tipo Kanban
- Además permite definir tareas, issues, pruebas y asignarlas a cada ingeniero de software
- Las tareas pueden categorizarse en Metas y en releases de código.

Tu argumentación fue convincente, y todos en la reunión, incluyendo los accionistas, están ahora convencidos de que la mejor forma de atacar este problema es utilizando **un sistema de control de versiones como Github**.

Sala de inspiración

Piensa en tu empresa u organización y analiza la siguiente pregunta:

- o ¿Existe alguna oportunidad de mejorar los procesos de desarrollo de software con el uso de repositorios de código? Por ejemplo, administrar los requerimientos de una manera más eficiente, o guardar las versiones de código para evitar pérdidas de información.

Durante los siguientes días, detente un momento para identificar actividades del proceso de desarrollo de software que podrían ser mejoradas o automatizadas usando un sistema de control de versiones. Esto te ayudará a relacionar los conocimientos adquiridos con situaciones reales y así se reafirmará tu aprendizaje.

Sala de pruebas

Dadas las conclusiones a las que han llegado, ahora usarás Github como herramienta para administrar el código de tus proyectos. Pensando en este modelo:

Pregunta 1: *¿Cuál de las siguientes opciones NO es una característica de un sistema de versionamiento de código?*

Opciones de respuesta	Retroalimentación
Opción a: Creación de branches	En efecto los branches son la base de los modelos de desarrollo dentro un sistema de versionamiento de código
Opción b: Manejo de tableros tipo kanban propios o integrados con otras herramientas	Las plataformas para versionar código van más allá de guardar código, actualmente permiten administrar requerimientos en tableros
Opción c: Permiten el trabajo en equipo administrando integrantes del equipo	Todos los sistemas para versionar código permiten el trabajo en equipo y administración de usuario, tareas, integración de código entre muchas otras
Opción d: IDE (Entorno de Desarrollo Integrado) con gran variedad de lenguajes de programación	En efecto, regularmente los gestores de versiones de código no poseen IDE's especializados

No obstante, que las plataformas de versionamiento de código, por el momento no tienen IDE's sofisticados para lenguajes de programación como Java,

Python, C++, etc. tienen grandes aportaciones en el proceso de ingeniería de software.

Pregunta 2: ¿Cuál de los siguientes productos NO es un software de versionamiento de código?

Opciones de respuesta	Retroalimentación
Opción a: https://github.com/	Github es la plataforma para almacenar código más popular en el mundo
Opción b: https://trello.com/	Trello no es una plataforma para almacenar código, solo administra proyectos con tablero tipo kanban
Opción c: https://gitlab.com/	Gitlab es un software versionador de código muy popular
Opción d: https://bitbucket.org/	bitbucket es también una plataforma de versionamiento de código muy popular con herramientas de administración de proyectos muy usadas

Definitivamente los modelos de desarrollo de software usando un software de versionamiento de código son muy versátiles

Pregunta 3: Github te permite organizar tus requerimientos y darles seguimiento para volverlos código

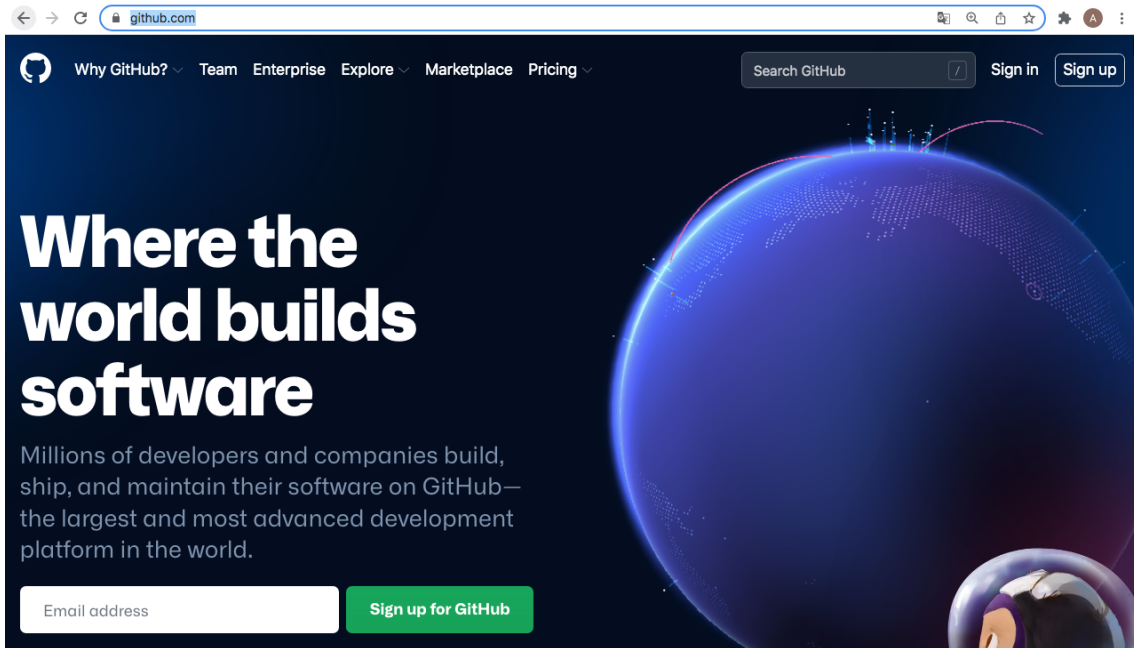
Opciones de respuesta	Retroalimentación
Opción a: Verdadero.	Github tiene características para crear tableros kanban con requerimientos, crear issues, metas, integrar código de miembros del equipo, entre otros.
Opción b: Falso	Github si permite administrar la construcción de un producto de software además de guardar el código

Ahora que tienes mayor familiaridad con los conceptos de software para manejo de versiones y administración de proyectos es importante tener presente retos importantes que debes enfrentar con modelos de desarrollo usando estas herramientas.

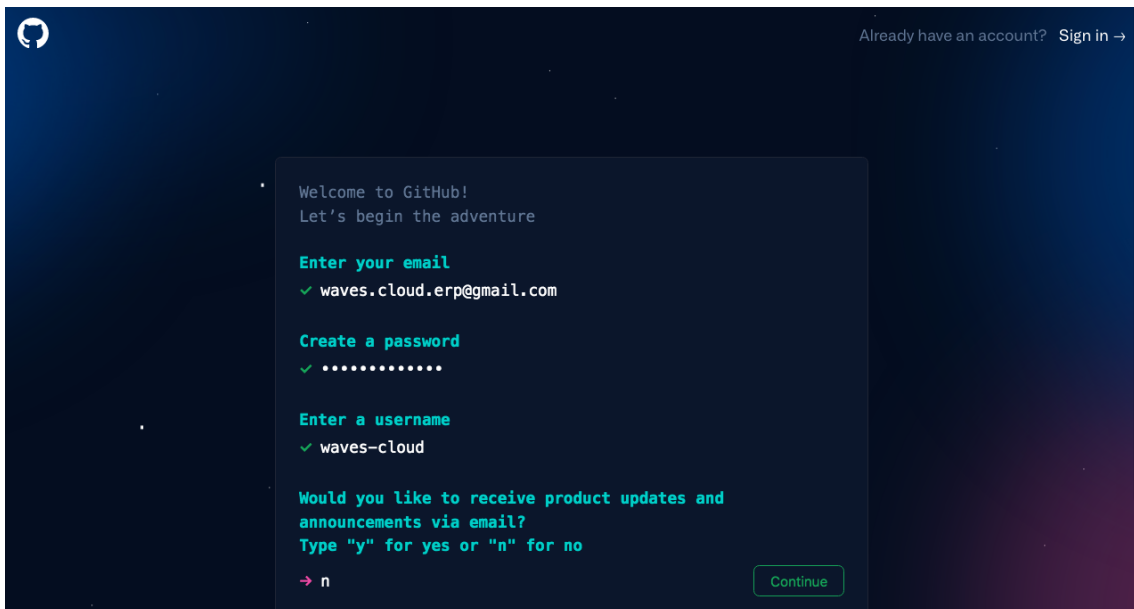
SUBTEMA 2. Crear una cuenta en github y creación de repositorios de código.

Todos los accionistas de la junta directiva están convencidos por tus argumentos en la reunión de arranque, y todos quieren saber más sobre el tema, por lo tanto, los días siguientes están llenos de preguntas acerca de **Github**, como *¿cuáles son las aplicaciones más exitosas guardadas en Github?*, *¿como se crea una cuenta para usar Github?*, *¿Cual es el costo?*, *¿qué se necesita para hacer crear un repositorio donde iniciar el proyecto?*, entre muchas otras. Por lo tanto, decides que lo mejor será redactar un documento que pueda ser circulado por email por toda la empresa, en donde respondas las preguntas que consideres más relevantes. Dicho documento para resolver esas interrogantes se presenta a continuación:

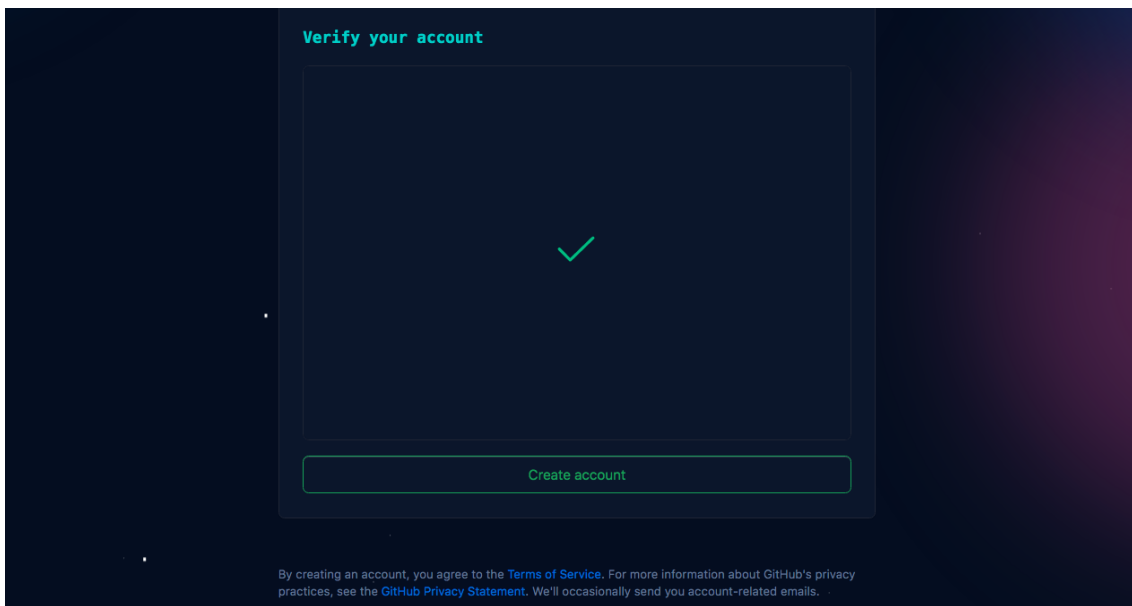
- **Github alberga billones de líneas de código en millones de proyectos alrededor del mundo**, solo por mencionar algunos de los proyectos populares hospedados en esta plataforma:
 - o Tensor flow
 - o React.
 - o Vue.
 - o Kubernetes.
 - o Swift.
 - o Visual Studio Code.
- Para usar Github, primero debemos crear una cuenta así que ingresa a: <https://github.com/> y presiona el botón **Sign up** que aparece en la esquina superior derecha.



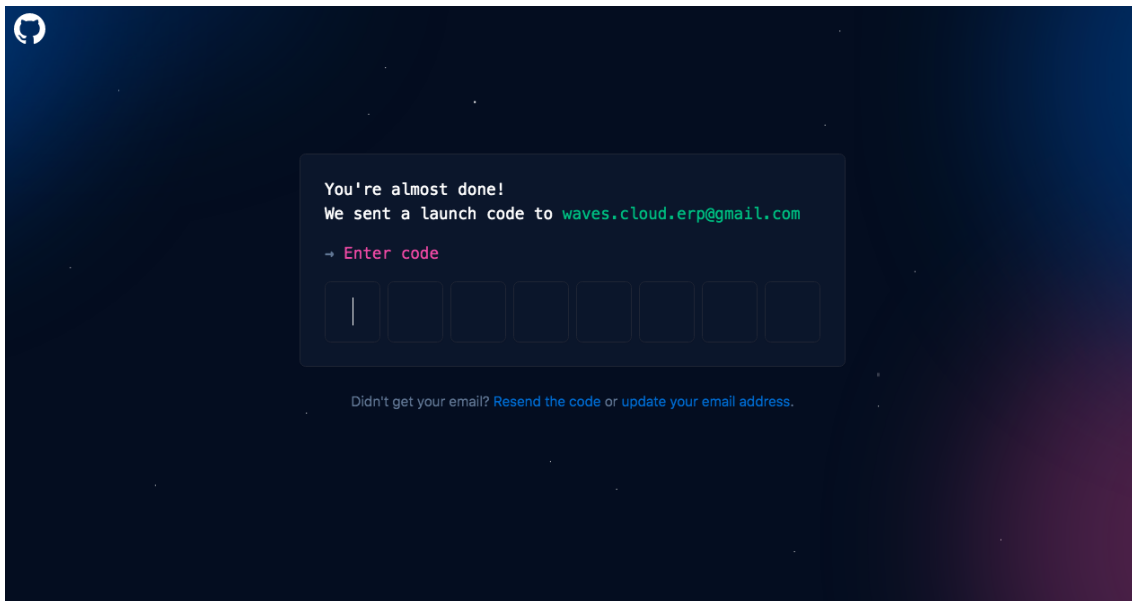
- Posteriormente se te pedirá ingreses tu email (puedes usar cualquiera que ya tengas creado de gmail, outlook, institucional), password y nombre de usuario git. El nombre de usuario git es muy importante ya que todos los repositorios nuevos contienen tu nombre de usuario.



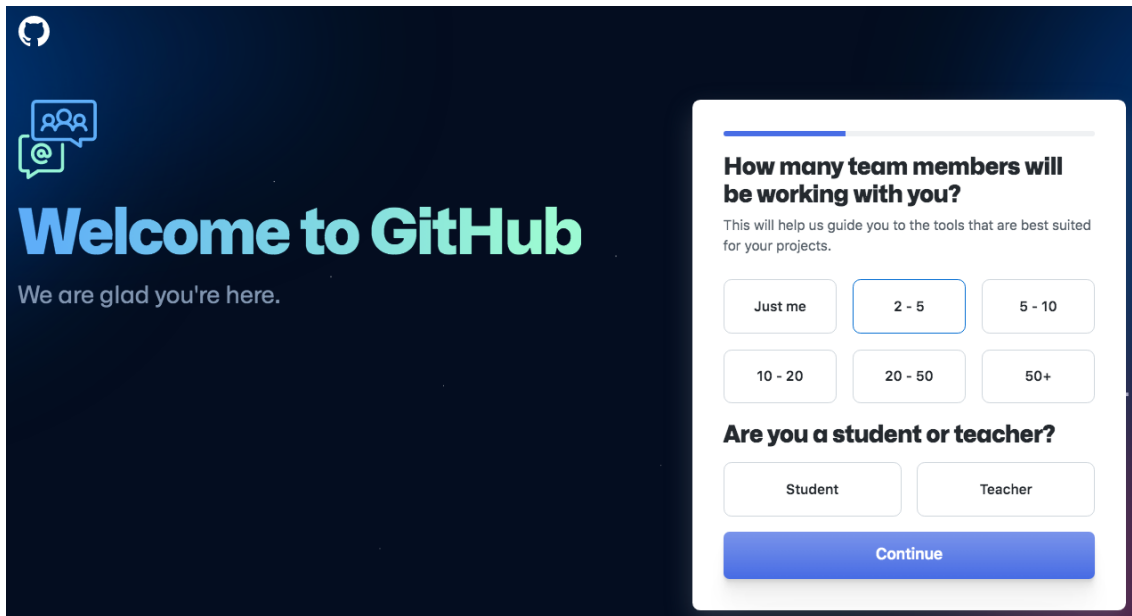
- Finalmente para asegurar que eres un humano y no un robot se te pedirá resuelvas un **Puzzle** que se genera de forma automática y aleatoria, una vez resuelto te confirmara si deseas crear tu cuenta.



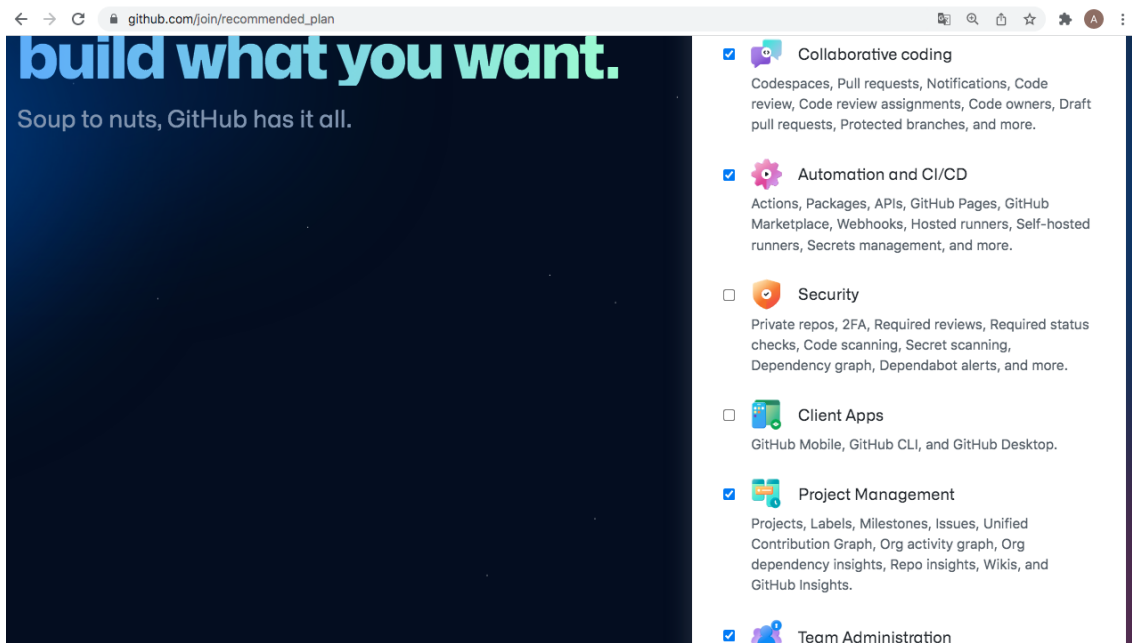
- Posteriormente se te pedirá verificar con un código enviado a tu email proporcionado en los pasos anteriores



- Otro punto importante es escoger el tamaño de tu equipo de ingenieros, teniendo en cuenta que los equipos de 4 ingenieros pueden tener repositorios privados sin costo. Asimismo podemos tener repositorios públicos con cualquier cantidad de ingenieros. Elegimos equipos de 2-5 integrantes.

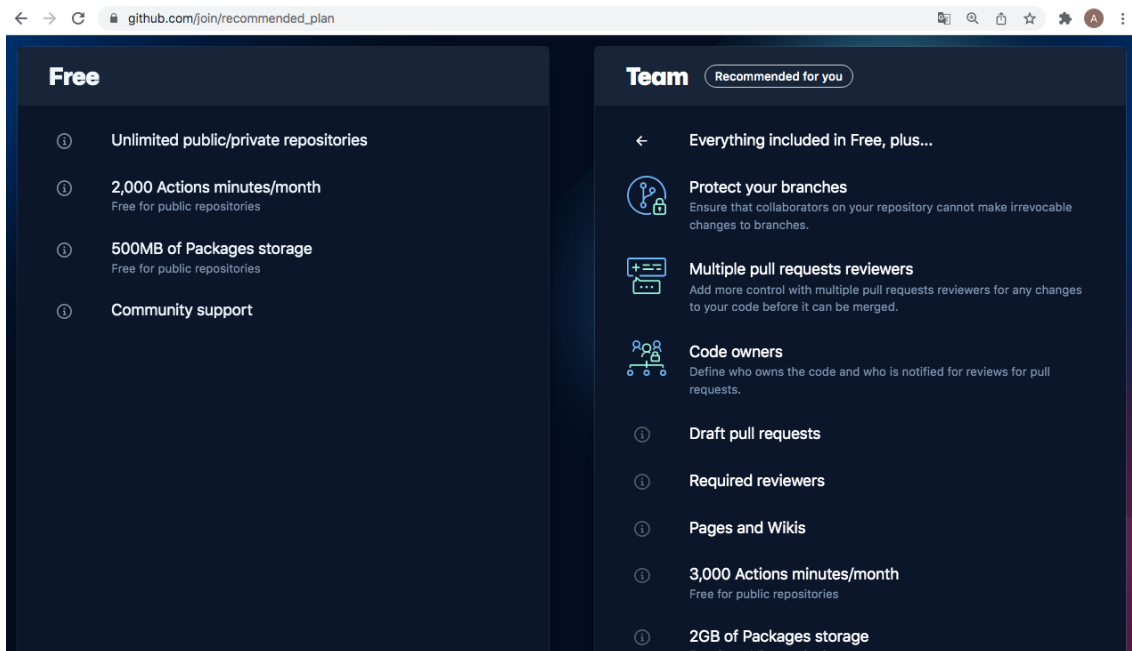


- Después debemos seleccionar las características que usaremos de Github, podemos seleccionar las 4 básicas relacionadas con la administración de proyectos como son: Trabajo de código colaborativo, automatización para integración y despliegue continuo, administración de de proyectos y finalmente administración de equipos.

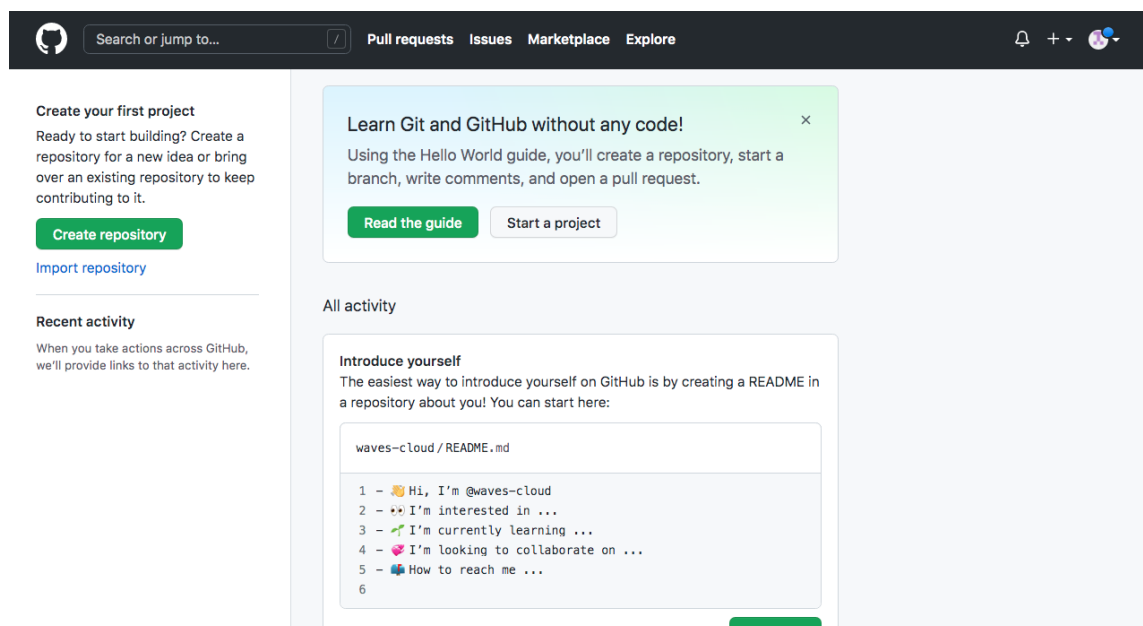


- El costo de usar Github es gratuito con las siguientes características: Repositorios públicos ilimitados sin restricción en tamaño del equipo, un

repositorio público es visible para todos los usuarios de Github, cualquier persona puede clonarlo pero no pueden escribir en él, es la opción ideal para proyectos de open source o con fines didácticos y/o entrenamiento. Repositorios privados con máximo 4 integrantes esta es nuestra mejor opción para el proyecto, solo es visible para los ingenieros que sean invitados a colaborar y no es visible para nadie más. Por tanto escogemos la opción **Free**.



- Entonces, para empezar con el proyecto.. crearemos nuestro primer repositorio de código, dar click en Botón Create Repository





- o **Configurar repositorio:** En la pantalla de creación de repositorio nos preguntará el nombre de nuestro repositorio, ponemos el nombre **mytienda-online**. Nota que el nombre está precedido por el nombre del usuario escogido en los pasos anteriores, por que el nombre de este repositorio seria: **waves-cloud/mytienda-online** y el nombre completo es: <https://github.com/waves-cloud/mytienda-online> (formado por 3 elementos: <https://github.com/> , nombre usuario y nombre del repositorio).
- o Después nos pide seleccionar si el repositorio es **Público** (visible para todo mundo) o **Privado** (visible para los miembros del equipo)
- o También nos pregunta si deseamos inicializar nuestro repositorio con algunos archivos de configuración como:
 - **README.-** Usado para documentar el propósito de nuestro repositorio.
 - **.gitignore.-** usado para configurar que archivos no deben ser subidos al repositorio es decir deben ser ignorados.
 - **Choose a license.-** Escoger el tipo de licencia que aplicará para nuestro proyecto (MIT license, Apache License, entre otros)

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner *



Repository name *

 waves-cloud ▾ / mytienda-online 

Great repository names are short and memorable. Need inspiration? How about [stunning-system?](#)

Description (optional)

Proyecto de tienda en línea

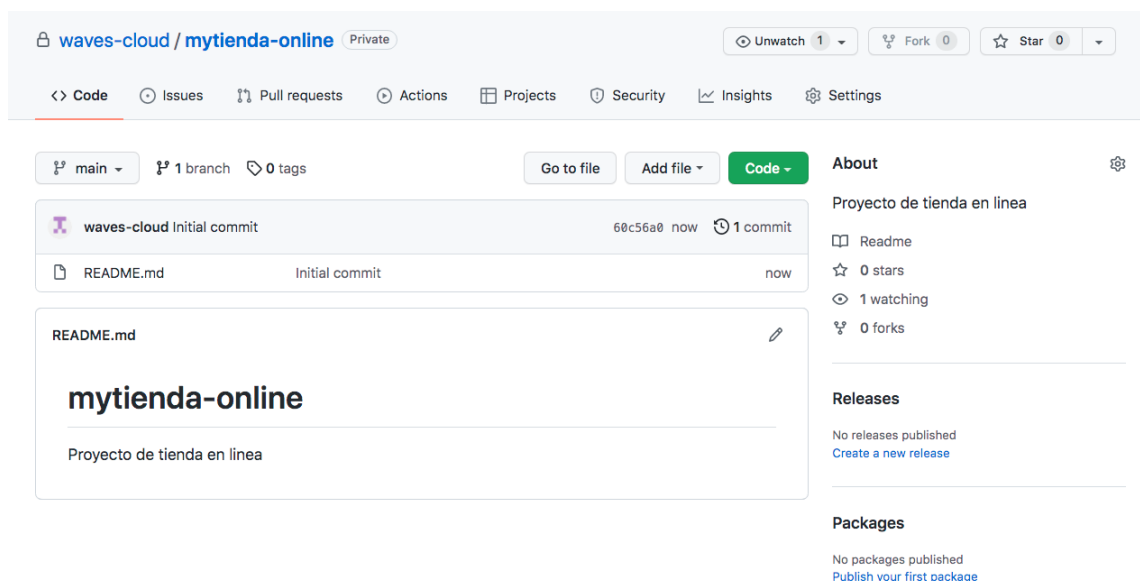
- ☐  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☒  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

- ☒ **Add a README file**
This is where you can write a long description for your project. [Learn more](#).
- ☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more](#).
- ☐ **Choose a license**

- o Finalmente el repositorio listo para usar se vera asi



Finalmente, es importante aclarar que existen muchas opciones dentro de un repositorio además del Código, permite controlar issues (con tareas, bugs, documentos, nuevos requerimientos, entre otros), Pull requests que son procesos de solicitud para integrar código por parte de algún integrante, Actions que son scripts de automatización de tareas, Projects donde se crean los tableros tipo kanban para administrar los proyectos, Security donde se establecen los permisos e integrantes del equipo de desarrollo, Insights donde se obtienen reportes y datos sobre la actividad de nuestro proyecto y Settings con opciones de configuración del repositorio.

Así, con este documento, todos en la empresa quedaron satisfechos y están convencidos que **Github** es la mejor opción para abordar este proyecto, además todos están muy emocionados por iniciar este proyecto.

Piensa en tu empresa u organización y analiza la siguiente pregunta:

- o Además del código fuente de los proyectos de software ¿Existe algún otro tipo de información que podrías guardar en repositorios? Por ejemplo, información con empleados para nomina quincenal, datos de ventas mensuales, información contable, documentos de word, hojas de cálculo

Durante los siguientes días, detente un momento para identificar qué información te gustaría respaldar de una forma segura y con versiones de tus actividades diarias.

Sala de pruebas

Dadas las conclusiones a las que han llegado, y de cómo el uso efectivo de Github puede potencializar la buena administración de un proyecto de software. Responde las siguientes preguntas:

Pregunta 1: *¿Cuál de las siguientes opciones NO forma parte del nombre completo de un repositorio de código en la plataforma Github)?*

Opciones de respuesta	Retroalimentación
Opción a: nombre del usuario	El nombre de usuario es parte fundamental del nombre de un repositorio ya que indica quien es el propietario del mismo, recuerda nuestro ejemplo. https://github.com/waves-cloud/mytienda-online
Opción b: https://github.com/	La dirección de internet de la plataforma Github es parte esencial del nombre ya que indica donde está hospedado el código. https://github.com/waves-cloud/mytienda-online
Opción c: nombre del branch principal	El nombre del branch NO forma parte del nombre base de un repositorio, aunque podemos aclarar que si tenemos permisos si podemos acceder a los diferentes branches.
Opción d: nombre del repositorio	En efecto, el nombre del repositorio es el tercer elemento de un repositorio https://github.com/waves-cloud/mytienda-online

--	--

Hasta ahora ya sabemos como como crear una cuenta en Github, crear repositorios a manera de repaso contesta lo siguiente.

Pregunta 2: ¿Cuál de las siguientes opciones NO pertenece a las permitidas en la inicialización de un repositorio?

Opciones de respuesta	Retroalimentación
Opción a: Branch inicial	El branch inicial NO es requerido ya que por default se asigna el branch main o master como el branch principal
Opción b: .gitignore	Si se puede elegir crear un archivo .gitignore para configurar qué archivos deben ser ignorados
Opción c: Choose License	El tipo de licencia si es una opción en la creacion de un repositorio
Opción d: README	Si es posible inicializar nuestro repositorio con README.md para documentar su propósito.

Como un caso práctico cuando inicias un proyecto, debes decidir cuando usar repositorios públicos y cuando uno privado, responde la siguiente pregunta:

Pregunta 3: Pensemos, si tu empresa desea iniciar un proyecto para administrar su contabilidad, es correcto guardar el código de este proyecto en un repositorio público ?

Opciones de respuesta	Retroalimentación
Opción a: Verdadero.	NO, debido a que manejas información contable y es confidencial, no debe ser visible públicamente.
Opción b: Falso	Correcto, la información sensible es preferible se almacene en repositorios privados, visibles solo para los miembros del equipo.

Ahora ya tienes mayor familiaridad con los conceptos de repositorios, hemos creado una cuenta en Github, ya sabemos como crear un repositorio, conocemos la nomenclatura de su nombre y las opciones de inicialización.

SUBTEMA 3. Comandos básicos en git para el manejo de repositorios

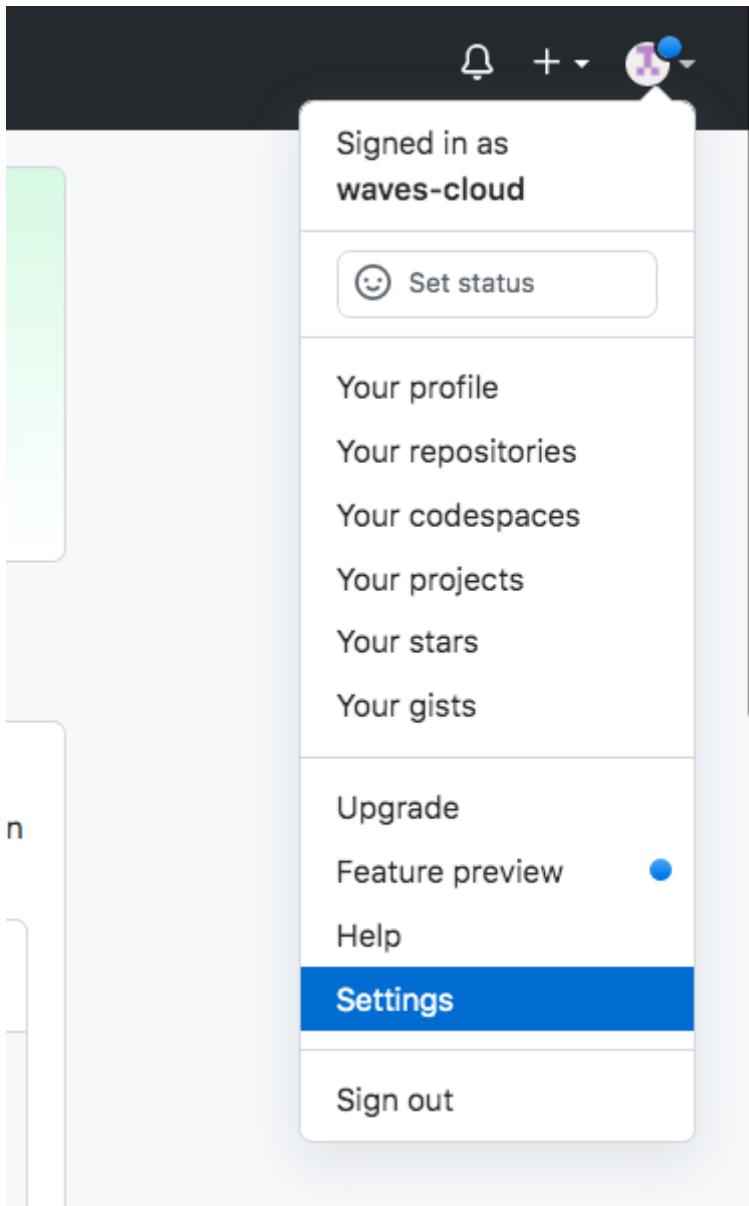
Afortunadamente todo marcha muy bien hasta ahora, tanto la junta directiva como ingenieros están convencidos en usar Github para administrar el

proyecto y almacenar el código fuente, sin embargo los ingenieros no tienen mucha experiencia en **git**, y tienen dudas puntuales acerca de **Github**, como *¿Una vez creado un repositorio en Github como realizamos la configuración en nuestra workstation para usarlo?, ¿cómo configuramos la rama principal de código? ¿Que comandos git se usan para subir a Github el código local ? ¿cómo creamos un branch nuevo?, ¿cómo integramos el código ?, entre muchas otras.* Por lo tanto, dado que los integrantes del equipo tienen experiencia en Python entonces decides crear un Notebook en la plataforma Google Colab para entrenar a los ingenieros.

Dado que uno de los ingenieros no ha usado Google Colab (<https://colab.research.google.com/>) explicas que es una plataforma que permite a cualquier usuario escribir y ejecutar código en Python directamente en el navegador, evitando así instalar y configurar Python localmente y que es especialmente adecuado para tareas de aprendizaje automático, análisis de datos y entrenamiento. Desde un punto de vista más técnico, Colab es un servicio en la nube que no requiere configuración y que ofrece acceso gratuito a recursos informáticos, como GPUs.





Entonces pones manos a la obra ejecutando las siguientes acciones para crear tu proyecto Colab para el entrenamiento de los ingenieros del equipo, siguiendo los pasos:

- Nuestro primer paso es crear un mecanismo de seguridad para trabajar con nuestro repositorio. Crearemos un Token para comprobar que somos los propietarios del repositorio. Ingresamos nuevamente a <https://github.com> con nuestro usuario y contraseña. Seleccionamos el menú de nuestro perfil en la esquina superior derecha y la opción **Settings**.




- Una vez seleccionado **Settings**, ingresar a la opción **Developer Settings**.



Code, planning, and automation

-  Repositories
-  Packages
-  Pages
-  Saved replies



Security

-  Code security and analysis

Integrations

-  Applications
-  Scheduled reminders

Archives

-  Security log
-  Sponsorship log

<> Developer settings

- Dentro de la opción de **Developer settings**, ingresar a **Personal access tokens**.

[Settings](#) / Developer settings

GitHub Apps

OAuth Apps

Personal access tokens

Personal access tokens

Generate new token Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

personal token — repo

Last used within the last week Delete

Expires on Wed, Apr 27 2022.

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

- En **Personal access tokens**, dar click en **Generate new token**, dentro de la pantalla **New personal access token**, ingresar un Nota o

descripción para nuestro token nuevo, poner el tiempo expiracion de nuestro token, para nuestro ejemplo podemos ponerlo a 90 dias, finalmente establecemos los permisos otorgados a nuestro token, para este caso seleccionamos los permisos relacionados a **repo**, y presionamos el botón de **Create Token**.

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

personal token

What's this token for?

Expiration *

90 days The token will expire on Fri, Apr 29 2022

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows

- Copiamos en un lugar seguro nuestro token, en un bloc de notas o en cualquier otro editor de texto plano (Ejemplo de token: **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**)

Personal access tokens Generate new token Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

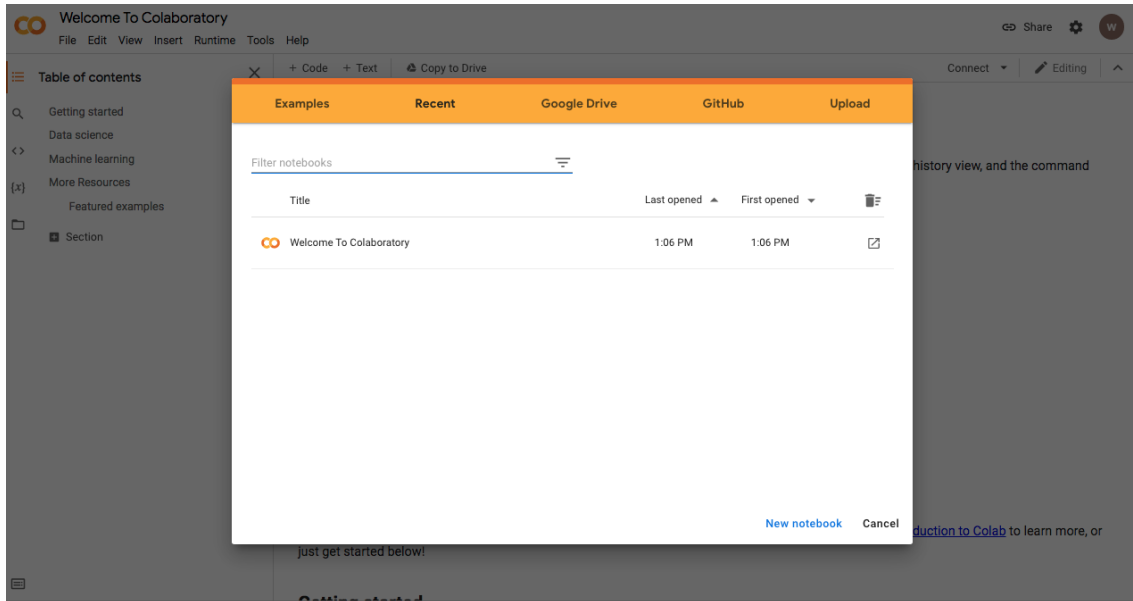
✓ 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 Delete

personal token — repo Last used within the last week Delete

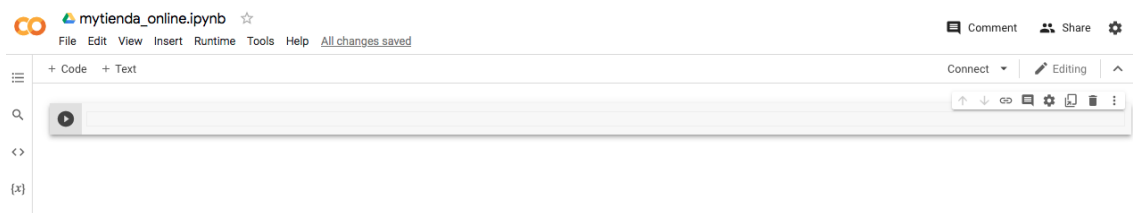
Expires on Wed, Apr 27 2022.

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

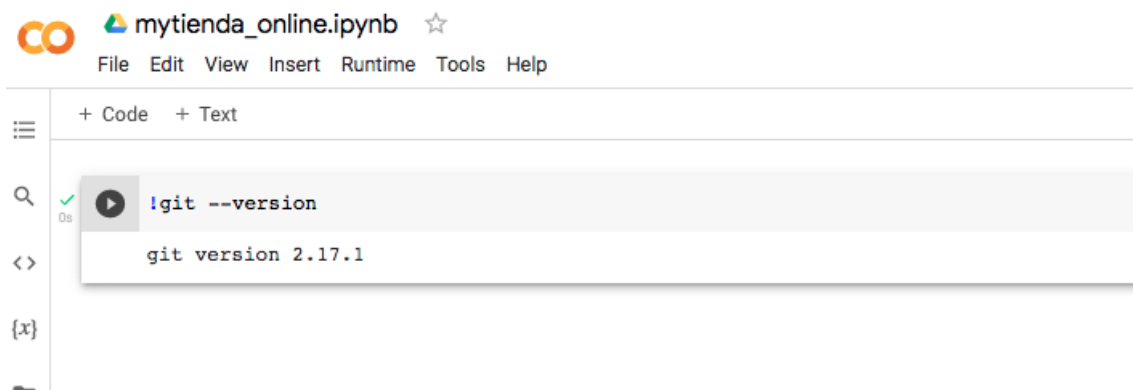
- Una vez creado y almacenado nuestro token de **Github**, Creamos un proyecto nuevo (llamado Notebook) en Google Colab, ingresamos a <https://colab.research.google.com/> y presionamos New notebook.



- De inmediato se crea nuestro proyecto y podemos asignarle un nombre, y está listo para ingresar Python en las celdas de código..



- Nuestro segundo paso es verificar que nuestro Notebook tenga el software de **git** instalado, para eso usas el comando: **!git --version**



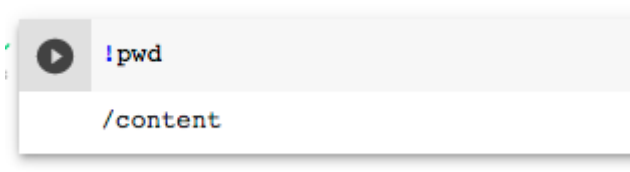
- Entonces, una vez confirmado que git está instalado, explicas que el primer paso es ejecutar un comando que nos permita descargar el

repositorio creado previamente, por lo que debemos copiar el url o dirección del repositorio y usar otro comando (**git clone**) para descargarlo a nuestro Notebook. Note que usamos nuestro token **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** como parte de la sintaxis del comando: `https://<token>@github.com/usuario/repositorio`

```
[ ] !git clone https://ghp_ccob3v6gRF0jeIAkP9rtNUIWB4aGaJ3SN0VZ@github.com/waves-cloud/mytienda-online.git

Cloning into 'mytienda-online'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
```

- Les debes recordar que Colab es en realidad una máquina virtual con Linux Ubuntu con una interfaz amigable para ejecutar código, por tanto podemos ejecutar comandos linux, comandos git y código python precedidos por el símbolo **!**. Colab tiene una carpeta por default donde guarda toda la información, esta es llamada **/content** y la obtenemos con el comando **!pwd** que nos retorna la ruta del directorio actual de trabajo.



```
!pwd

/content
```

- Por lo tanto nuestro repositorio descargado estará en **/content/mytienda-online** y podemos movernos a nuestro proyecto usando el comando linux `cd`, recuerda que el comando `cd` no lleva el símbolo **!**.

```
[ ] cd /content/mytienda-online/

/content/mytienda-online
```

- Ahora podemos comprobar si estamos dentro de nuestro repositorio con el comando **!git remote -v**

```
!git remote -v
origin https://ghp_ccob3v6gRF0jeIAkP9rtNUIWB4aGaJ3SNOVZ@github.com:waves-cloud/mytienda-online.git (fetch)
origin https://ghp_ccob3v6gRF0jeIAkP9rtNUIWB4aGaJ3SNOVZ@github.com:waves-cloud/mytienda-online.git (push)
```

- Luego podemos ejecutar el como linux **!ls -la** para ver el contenido actual de nuestro repositorio

```
!ls -la
total 16
drwxr-xr-x 3 root root 4096 Jan 29 07:30 .
drwxr-xr-x 1 root root 4096 Jan 29 06:52 ..
drwxr-xr-x 8 root root 4096 Jan 29 06:52 .git
-rw-r--r-- 1 root root  46 Jan 29 06:52 README.md
```

En este punto debes recordarles que existe una carpeta oculta llamada **.git** donde se guarda automáticamente información del repositorio, los branches, versiones y configuraciones necesarias para que git funcione.

- Análogamente debes recordar que git guarda el código en espacios de trabajo denominados **branches**, por tanto podemos usar el comando **!git branch** para saber como se llama nuestro branch por default o principal

```
!git branch
* main
```

- Debemos indicar a Github cual es nuestro usuario actual

```
[ ] !git config --global user.email "waves-cloud-erp@gmail.com"
```

- Para demostrar cómo crear una versión nueva del proyecto podemos crear una celda para crear un archivos de prueba, haremos un archivo llamado **index.html**


```
%%writefile index.html
<html>
<body>
tienda online demo..
</body>
</html>
```

Writing index.html

- Ahora podemos usar el comando **git status** para comparar el estado local de los archivos contra el repositorio en la nube de Github

```
!git status

On branch main
Your branch is up to date with 'origin/main'.

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

    index.html

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

- En este punto git nos está indicando que tenemos un archivo local, pendiente de subir a nuestro repositorio en la nube de Github, entonces debemos agregar estos cambios a la nueva versión de nuestro repositorio con el comando **git add**

```
!git add index.html
```

- Una vez hecho esto, crearemos una versión de nuestro proyecto usando el comando **git commit**.

```
!git commit -m "agregamos pagina de prueba index.html"

[main lccc29b] agregamos pagina de prueba index.html
1 file changed, 5 insertions(+)
create mode 100644 index.html
```

- Finalmente subimos los cambios a github usando el comando git push

```
!git push -u origin main

Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 338 bytes | 338.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/waves-cloud/mytienda-online.git
0514ee3..lccc29b main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Finalmente, es importante aclarar que el motor de git tiene cientos de comandos, el propósito de esta sección es demostrar los comandos básicos para clonar y subir cambios en la rama principal, en la siguiente tabla mostramos un resumen de los comandos git usados en este sección.

Comando git	Descripción
git --version	Regresa la versión actual de git, y nos ayuda a saber si git está instalado
git clone	Permite clonar nuestro repositorio a la estación de trabajo local o Colab
git remote -v	Obtiene el url de nuestro repositorio, siempre y cuando estemos dentro de la carpeta donde se clono
git config --global user.email "gituser@email"	Permite configurar nuestro usuario actual en la carpeta del repositorio activo
git status	Nos regresa que archivos han cambiado o cuales estan pendientes por subir.

git add <file>	Permite agregar un archivo, para ser subido al repositorio, NOTA git add . agrega todos los archivos cambiados o modificados.
git commit -m "mensaje "	git commit permite hacer un "checkpoint" o versión del proyecto que podamos subir al repositorio, es importante poner un mensaje que describa los cambios realizados
git push origin <branch>	git push sube los cambios marcados con git commit al repositorio, en el branch indicado.

Así, con este documento y el Notebook de Colab (<https://colab.research.google.com/drive/1DxxpoN6i3SqMU9hfXwxGh6Cn6mUP8Zv?usp=sharing>) los ingenieros de software se sienten más confiados del éxito del proyecto.

Sala de inspiración

Piensa en tu empresa, organización, e incluso en tu vida cotidiana y considera las siguientes preguntas:

- o ¿Qué tipos de proyectos de la universidad podrías haber guardado en un repositorio si hubieras sabido de git?
- o ¿Qué datos personales en la actualidad podrías almacenar en un repositorio privado, previniendo alguna falla de tu laptop o computadora de escritorio ?

Piensa en posibles consecuencias que tendría el no tener versiones actuales de tu proyecto respaldadas en un repositorio de código, y cómo podría tu organización recuperarse de un hecho desafortunado con el hardware.

Sala de pruebas

Una vez que todos estén de acuerdo en el tipo de modelo de Github a utilizar para la aplicación web de tienda en línea.

Pregunta 1: *Cuál de los siguientes es la forma correcta de clonar un repositorio privado en Google Colab usando tu token*

Opciones de respuesta	Retroalimentación
------------------------------	--------------------------

Opción a: git clone https://<token>@github.com/<usuario/<repositorio>	Esta opción sería correcta pero si es en Google Colab falta el símbolo ! antes del comando git
Opción b: !git clone https://github.com/<usuario/<repositorio>	Esta opción esta es incorrecta porque le falta el token en la url, al ser repositorio privado nos negaría el acceso.
Opción c: !git clone https://<token>@github.com/<usuario/<repositorio>	Esta es la opción correcta tiene el simbolo ! propio de Colab, el token, usuario y repositorio
Opción d: !git clone -b develop https://github.com/<usuario/<repositorio>	Esta opción tiene el símbolo !, pero le falta token y ademas el parámetro -b develop, indica que clone un repositorio específico llamado develop

Es importante notar, todos los comandos git al ser ejecutados dentro de Colab deben llevar el símbolo !, si usas git en tu laptop este símbolo no es necesario.

Pregunta 2: *¿Cuál de los siguientes comandos NO pertenece a git ?*

Opciones de respuesta	Retroalimentación
Opción a: git config	git config, es el comando git que nos permite configurar opciones como el email del usuario propietario del repositorio
Opción b: git remote -v	git remote -v, nos regresa el url del repositorio al que apuntamos desde nuestro directorio
Opción c: pwd	pwd no es un comando git, es un comando de linux que nos regresa la ruta de trabajo actual.
Opción d: git status	git status nos muestra los archivos pendientes por actualizar en el repositorio.

Para que puedas subir cambios a tu repositorio, regularmente se ocupan 3 comandos git en cierto orden, en ese sentido.

Pregunta 3: ¿Cuál de las siguientes comandos git NO es usado para subir cambios ?

Opciones de respuesta	Retroalimentación
Opción a: git push	git push si es usado para subir los commits realizados con el comando git commit
Opción b: git commit	git commit si es usado para crear versiones de código.
Opción c: git clone	git clone NO es usado para subir cambios, sino para descargar un repositorio a una máquina local
Opción d: git add	git add permite agregar archivos a una versión o commit, para después ser subidos con el comando git push

Hemos estudiado una perspectiva global que nos ha permitido poner en contexto qué es la creación de productos de software bien administrados y controlados por plataformas de versionamiento de código ayudan en el aseguramiento de la calidad del producto. El contenido cubierto brinda un panorama general del uso de la plataforma Github para almacenar el código de un proyecto. Además, te permitirá cuestionar qué tipo de información podrías almacenar de forma segura en un repositorio para evitar pérdidas importantes de información de tu empresa en caso de un fallo de hardware.

Ideas para llevar

Como parte de tu formación como científico de datos ahora tienes más claro que:

- A diferencia de los modelos de almacenamiento tradicionales como Dropbox, Drive, entre otros un software de versionamiento de código es más eficiente para administrar un proyecto y sus versiones.
- El motor Git para manejar versiones de código, es la base de plataformas como Gitlab, Bitbucket o Github.
- Github es gratuito para proyectos de open source con un número ilimitado de ingenieros
- Github permite repositorios privados para equipos de máximo 4 integrantes.

- Las plataformas actuales basadas en Git, tiene muchas funcionalidades adicionales para administrar proyecto como tableros kanban, administración de issues, tareas, scripts para automatización de tareas, entre otros.
- Un reto muy importante al diseñar e implementar modelos basados en Git en equipos de trabajo es la correcta distribución de tareas entres los ingenieros, tal que no haya dependencias que atrasen el proyecto.

Pasos a seguir.

<https://thecodingtrain.com/beginners/git-and-github/>

<https://www.udemy.com/course/git-desde-cero/>

<https://www.udacity.com/course/version-control-with-git--ud123>