|  |  |
| --- | --- |
| **UNIDAD ACADEMICA:** | INGENIERIA DE SISTEMAS |
| **CURSO:** | INGENIERÍA DE SOFTWARE II |
| **PRACTICA N.º 1:** | INTRODUCCIÓN A LOS SISTEMAS DE CONTROL DE VERSIONES |

# OBJETIVOS

* + Aprender a instalar e implementar Git
  + Evidenciar el control de versiones utilizando GitHub

# FUNDAMENTO TEORICO

# Un Sistema de Control de Versiones (VCS) se refiere al método utilizado para guardar las versiones de un archivo para referencia futura.

# De manera intuitiva muchas personas ya utilizan control de versiones en sus proyectos al renombrar las distintas versiones de un mismo archivo de diferentes formas como blogScript.js, blogScript\_v2.js, blogScript\_v3.js, blogScript\_final.js, etcétera. Sin embargo, esta manera de abordarlo es susceptible a errores y resulta inefectiva para proyectos grupales, pues rastrear qué cambió, quién lo cambió y porqué se cambió, se convierte en una tarea tediosa. Esto resalta la importancia de un sistema de control de versiones confiable y colaborativo como Git.

# ¿Qué es Git?

# Git es un Sistema de Control de Versiones Distribuido (DVCS). El término "distribuido" significa que cuando se le instruye que comparta el directorio de un proyecto, no sólo comparte la última versión del archivo, sino que distribuye cada versión que ha registrado para ese proyecto. Esta herramienta facilita el versionado, administración y distribución de archivos, permitiendo un acceso rápido y disponible en cualquier momento.

# ¿Qué es GitHub?

# GitHub es una plataforma basada en la web donde los usuarios pueden alojar repositorios Git. Facilita compartir y colaborar en proyectos con cualquier persona en cualquier momento. También fomenta una participación más amplia en proyectos Código Abierto al proporcionar una manera segura de editar archivos en repositorios de otros usuarios.

# Git y GitHub son dos entidades diferentes pero complementarias. Git sirve para controlar las versiones de los archivos mientras que GitHub es una plataforma para alojar los repositorios Git. Su diferencia radica en la plataforma de operación. En el primer caso, los usuarios instalan y ejecutan Git en sus equipos locales, esto significa que la mayoría de las de operaciones de Git se pueden lograr sin una conexión a internet. Contrario a GitHub, que trabaja como un servicio basado en la web y opera solamente en línea [1].

# EQUIPOS, MATERIALES Y REACTIVOS

|  |  |  |
| --- | --- | --- |
| **Equipos** | **Materiales** | **Sustancias y/o Reactivos** |
| Computadores |  |  |

# PROCEDIMIENTO

# Creación de una cuenta en GitHub

# En primer lugar, será necesario dirigirse a la página oficial de GitHub para crear una cuenta. En caso de ya estar registrado, se podrá omitir este paso. Puede utilizar el siguiente link para acceder directamente al formulario de registro <https://github.com/join> donde tendrá que ingresar un nombre de usuario, correo (debe ser la cuenta institucional) y contraseña.

# Recibirá un código de confirmación al correo ingresado y a continuación podrá configurar el numero de miembros del equipo y seleccionar su rol como estudiante. Despues vamos a elegir las características de GitHub, en este caso solo será Codificación colaborativa y podremos dar click en siguiente.

# 

# Finalmente dando click en la opción de versión libre, la cuenta habrá sido creada, y se podrá observar una ventana como la siguiente.

# 

# Instalación y configuración de Git

# Para instalar Git nos vamos a dirigir al siguiente link <https://git-scm.com/downloads> .En este caso se utilizará el instalador para Windows.

# Una vez descargado el archivo .exe, lo ejecutaremos y nos saldrá la siguiente ventana.

# 

# Vamos a dar click en siguiente a los acuerdos de licencia, al directorio y componentes por defecto hasta llegar a la siguiente ventana donde seleccionaremos el editor Nano.

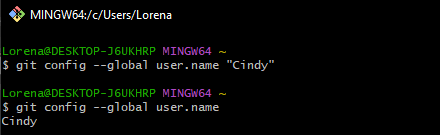
# 

A partir de este punto daremos siguiente a las demás opciones por defecto y finalmente iniciará la descarga de Git. Una vez terminada podremos buscar ‘**Git Bash’** en nuestro equipo.

Desde la consola que sale en pantalla vamos a configurar el nombre de usuario con el siguiente comando, cambiando solamente el dato que se encuentra en comillas.

**git config --global user.name "NombreDeUsuario"**

Podemos confirmar que el proceso se realizó correctamente escribiendo **git config --global user.name**



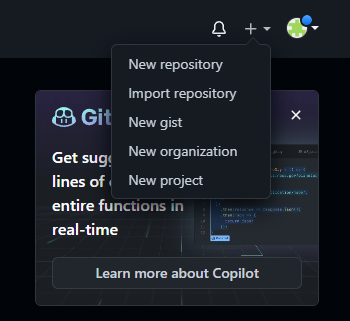
Ahora vamos a configurar el correo electrónico, es importante que **sea el mismo email con el que se creo la cuenta en GitHub.** Para ello usaremos el siguiente comando, nuevamente cambiando los datos dentro de las comillas.

**git config --global user.email "email@example.com"**

Podemos confirmar que el proceso se realizó correctamente escribiendo **git config --global user.email**

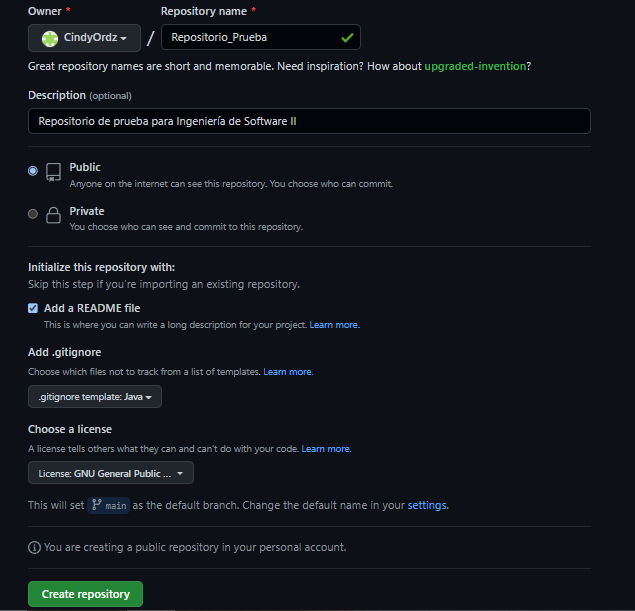
**Creación de un repositorio en GitHub**

Ingresamos a GitHub, presionamos el botón **+** y después la opción New repository.

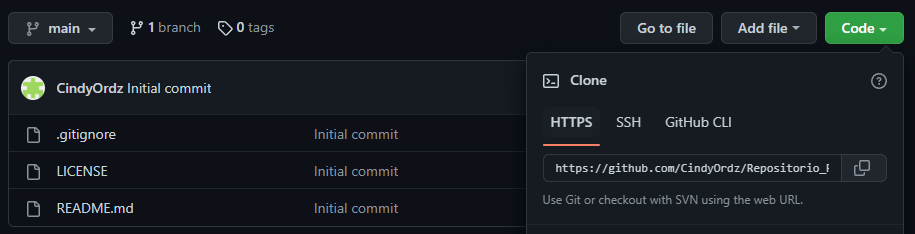


A continuación, le asignamos un nombre y una descripción al repositorio. En este caso se llamará ‘Repositorio\_Prueba’ y la descripción será ‘Repositorio de prueba para Ingeniería de Software II’. Además, lo dejáramos público y con un archivo README.

El *gitignore template* será Java para indicarle a Git qué archivos y directorios ignorar cuando realiza una confirmación, y la licencia será GNU. Estos parámetros son solo de prueba, varían de acuerdo al tipo de proyecto. La creación del repositorio se observa en la siguiente imagen.



De esta forma quedará creado el repositorio y ahora podremos, por ejemplo, clonarlo para trabajarlo en remoto. Para ello damos click en *Code* y copiamos el enlace.

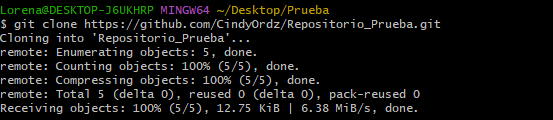


**Clonar un repositorio**

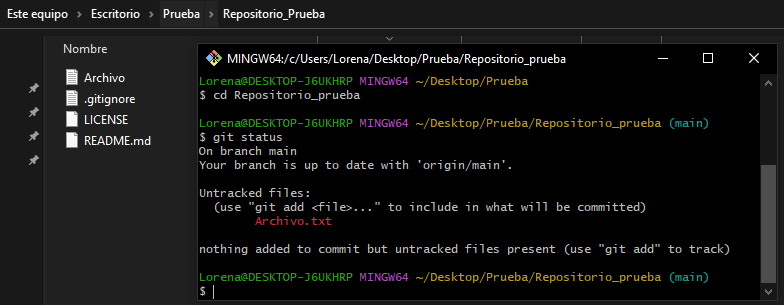
A continuación, creamos una carpeta en el escritorio llamada *Prueba*, y estando **dentro de la carpeta** presionamos click derecho y la opción ***Git Bash Here****.* De esta forma se abriría una consola Git donde escribiremos el comando **git init** Esto crea un nuevo subdirectorio llamado .git que contiene todos los archivos de repositorio necesarios: un esqueleto de repositorio de Git.



Ahora escribimos **git clone** seguido del enlace que copiamos desde GitHub.



Despues creamos un archivo dentro de la carpeta del repositorio que hemos clonado, por ejemplo, un txt llamado *Archivo* con el contenido *Hola mundo.*  Luego, desde la consola nos ubicamos en la carpeta usando: **cd Repositorio\_Prueba** y ejecutamos el comando **git status,** de esta forma veremos el estado del directorio de trabajo y los cambios que se han ejecutado.



**Añadir y confirmar cambios**

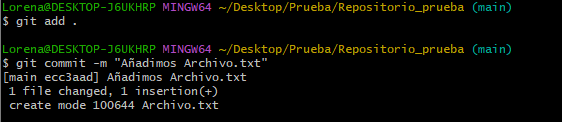
Para añadir un cambio del directorio de trabajo en el entorno de ensayo, en este caso usamos el comando: **git add Archivo.txt**

Otra opción es usar **git add .** que trabaja de forma mas general añadiendo todos los archivos en los que ha ocurrido algún cambio.

Para confirmar una instantánea del directorio del entorno de ensayo en el historial de confirmaciones de los repositorios usamos **git commit.** Este comando **se debe ejecutar añadiendo un** **-m** al final para

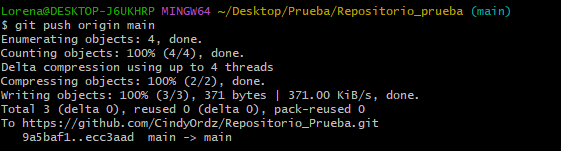
agregar un mensaje que describa el commit, por ejemplo:

**git commit -m "Añadimos Archivo.txt"**

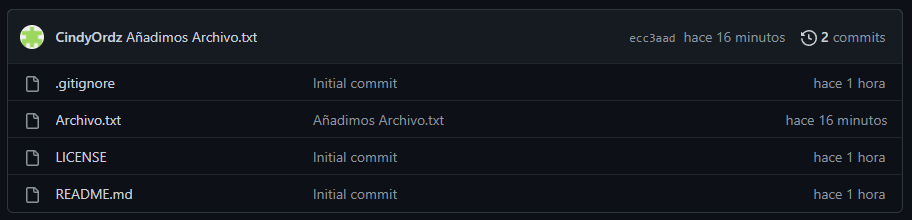


**Cargar cambios del repositorio local a un repositorio remoto**

Usamos el comando **git push origin main.** En este paso nos puede solicitar el correo o iniciar sesión desde el navegador.



Si volvemos a GitHub y recargamos la página, podremos ver los cambios realizados.



Si bien hemos creado un repositorio y hemos cargado cambios de forma remota, Git provee otros comandos que vale la pena conocer, se recomienda dirigirse a la documentación oficial <https://git-scm.com/doc>

**Creación de la clave SSH**

Una llave SSH es una clave que permite autenticarse ante un servicio, en este caso, GitHub. Por lo general vienen 2 llaves: una pública y una privada. Con la pública se puede enviar y recibir información cifrada de la web, mientras que con la privada se puede descifrar esa información para poder ver su

contenido. Es importante **nunca compartir la llave privada.**

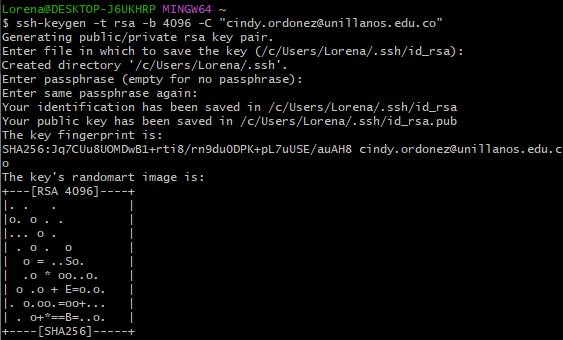
Para crear la llave SSH usamos el siguiente comando con nuestra dirección de correo

**ssh-keygen -t rsa -b 4096 -C "your\_email@example.com"**

Una vez que se ejecute el comando se presentarán dos preguntas:

1. Si desea que la llave SSH se guarde en el directorio por defecto. En este caso esa será la opción deseada así que solo daremos enter.
2. Luego de seleccionar el directorio, pedirá una contraseña extra (passphrase). Esta será solicitada cada vez que se quiera usar la llave. En este caso se dejará vacío, así que nuevamente se presiona enter dos veces para confirmar la clave [2].

De esta forma se habrá creado la clave SSH, como se observa a continuación. Esta llave nos permitirá conectarnos a GitHub sin proporcionar nombre de usuario y token de acceso personal en cada visita.



“your\_email@example.com”

your\_email@example.com

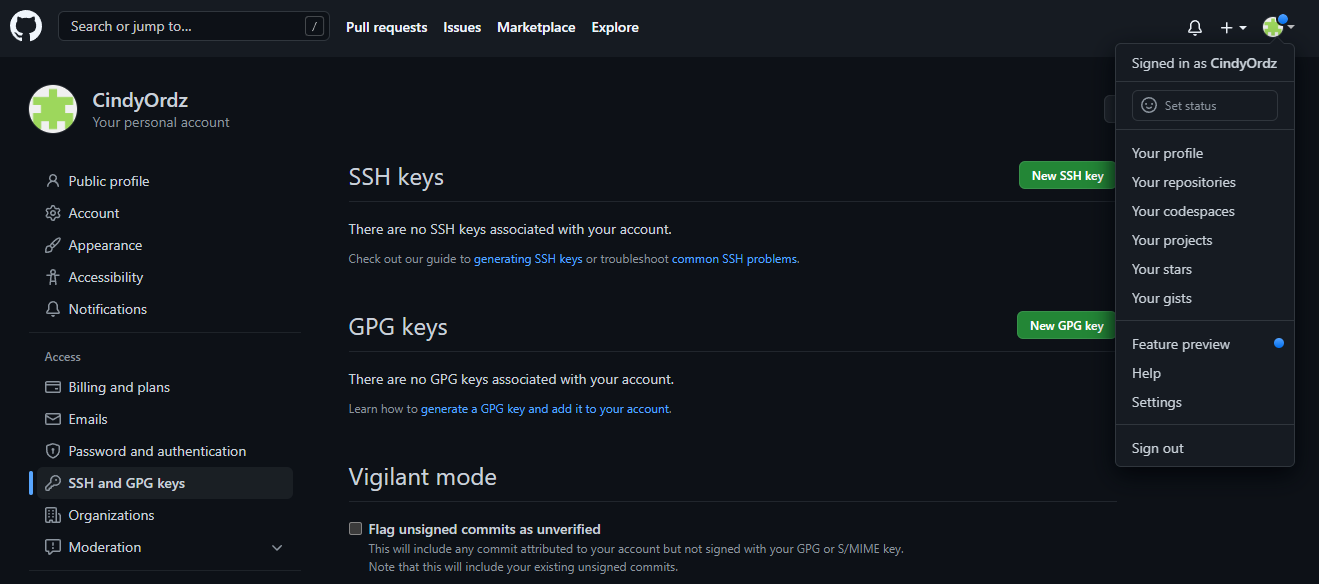
A continuación, con el comando: **cat ~/.ssh/id\_rsa.pub** podremos ver la llave creada desde la consola. Otra opción es seguir la ruta **C:\Users\TuNombreDeUsuario\.ssh** y abrir con un editor de texto el archivo .pub

Después de copiar la llave vamos a GitHub en la opción *Settings*, presionamos *SSH and GPC keys* y luego la opción *New SSH key*. Esto se realiza así:

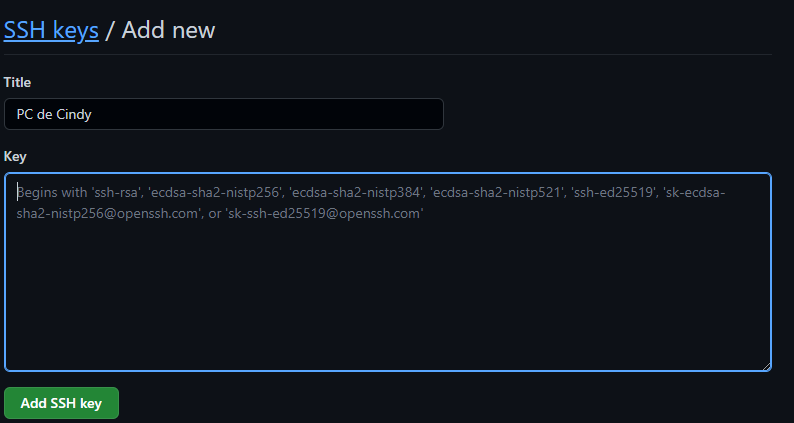
- Dar click en el logo del usuario en GitHub y elegir la opción “Settings”

- Dar click en “SSH and GPC keys”

- Dar click en la opción “New SSH key”



Finalmente copiamos la llave en el recuadro *key* y le asignamos un nombre en el campo *title.* Este proceso puede necesitar que recargue la pagina o inicie sesión nuevamente.



# RESULTADOS

# Crear un repositorio en GitHub y cargar un proyecto propio. Despues realizar al menos un cambio sobre el proyecto usando Git. Se debe subir a la plataforma el enlace del repositorio creado donde se evidencie que los procedimientos realizados son recientes.

# REFERENCIAS

[1] Castellanos, E. (2021, 14 febrero). Git vs GitHub – ¿Qué es el Control de Versiones y Cómo Funciona? freeCodeCamp.org. <https://www.freecodecamp.org/espanol/news/git-vs-github-what-is-version-control-and-how-does-it-work/>

[2] J. (2020, 23 noviembre). Cómo crear y vincular una llave SSH con GitHub. DEV Community. <https://dev.to/johncardenasp/como-crear-y-vincular-una-llave-ssh-con-github-42i4#:%7E:text=Una%20llave%20SSH%20es%20una,informaci%C3%B3n%20para%20poder%20su%20contenido>.