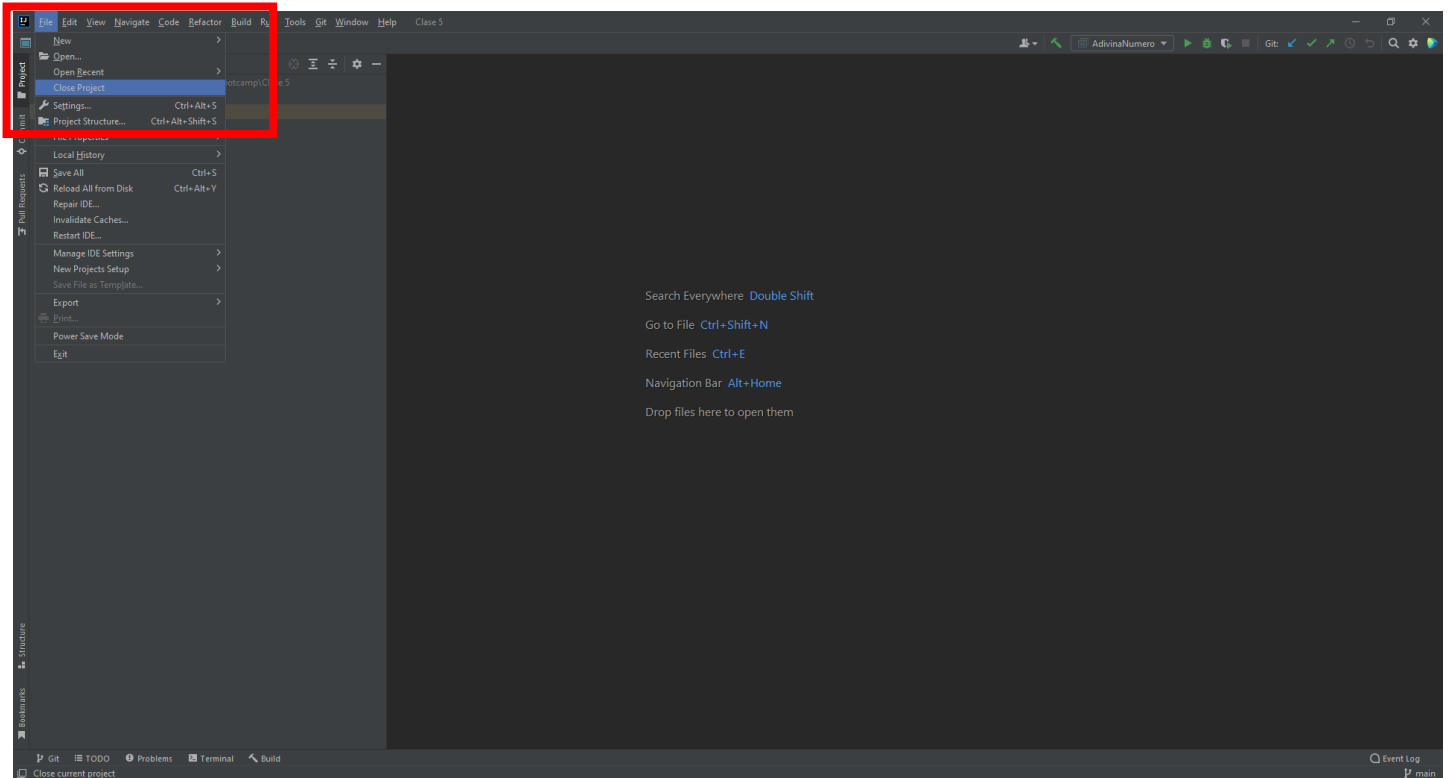


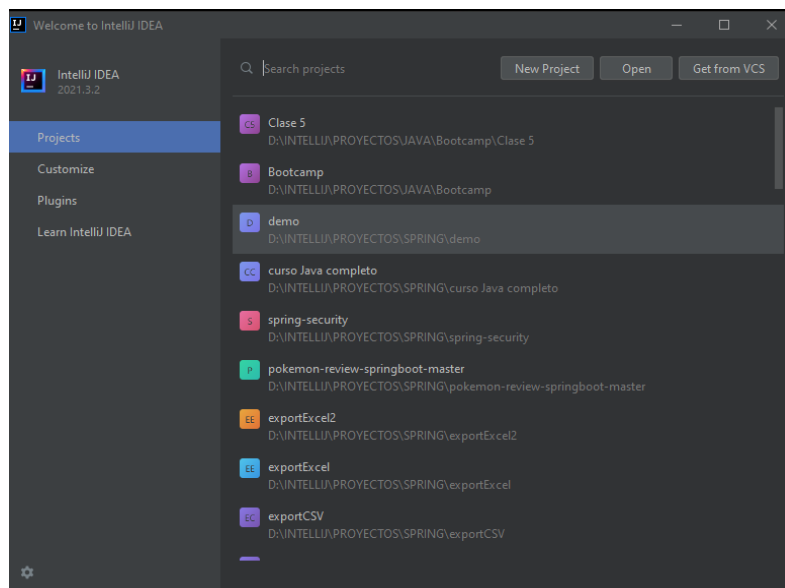
# CREAR PROYECTO EN INTELLIJ

*Hay muchas formas de hacerlo, pero esta es como lo hago yo.*

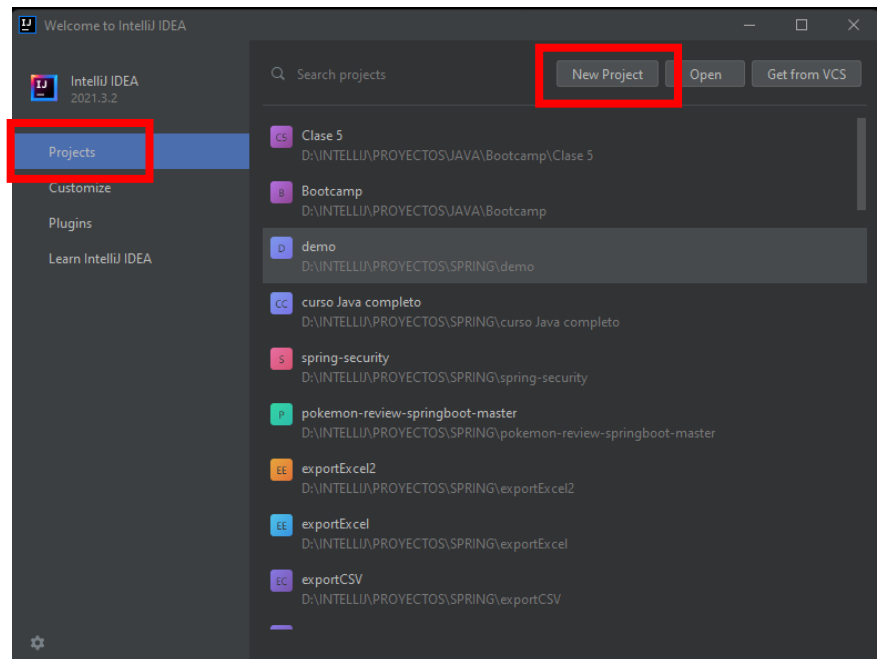
1. Lo primero que tenemos que hacer es tener instalado IntelliJ. <https://www.jetbrains.com/idea/idea/download/#section=windows> descargar la versión gratis CommunityEdition.
2.
  - a. Vamos a crear un proyecto desde 0 en IntelliJ, para eso si ya tenemos un proyecto abierto nos dirigimos donde dice **FILE** seguido **CLOSE PROJECT**.



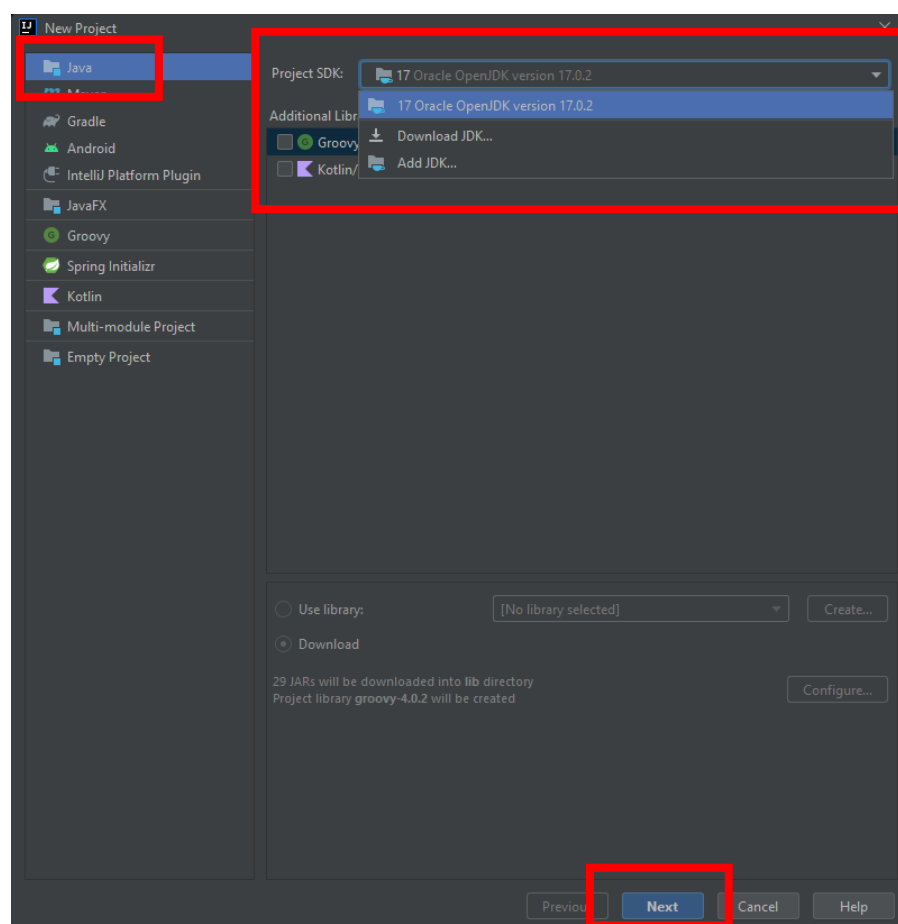
- b. En caso contrario nos aparecerá la pagina de inicio de IntelliJ. Es decir, aparecerá lo siguiente.



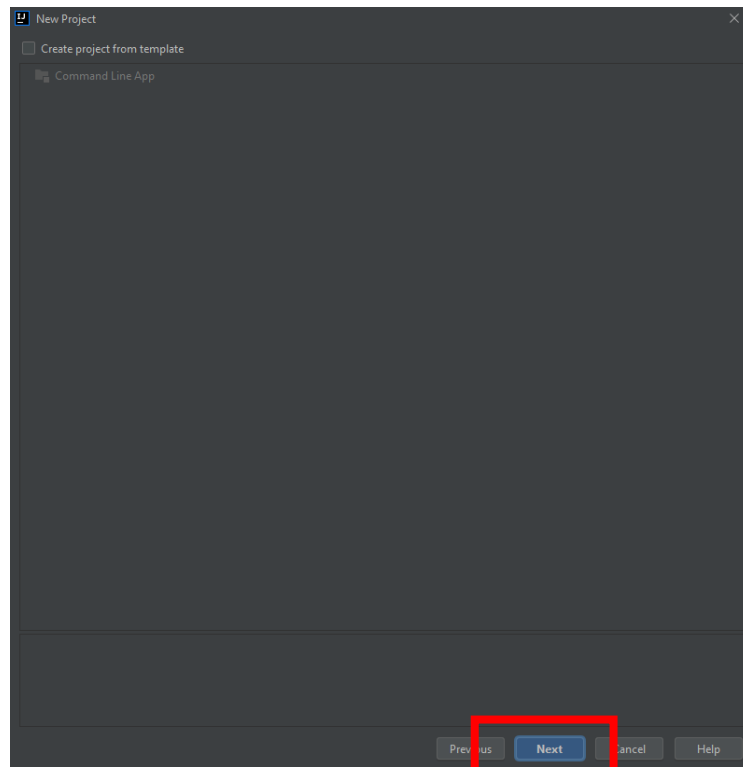
3. Creamos nuestro proyecto apretando el botón **Projects** y luego **New Project**.



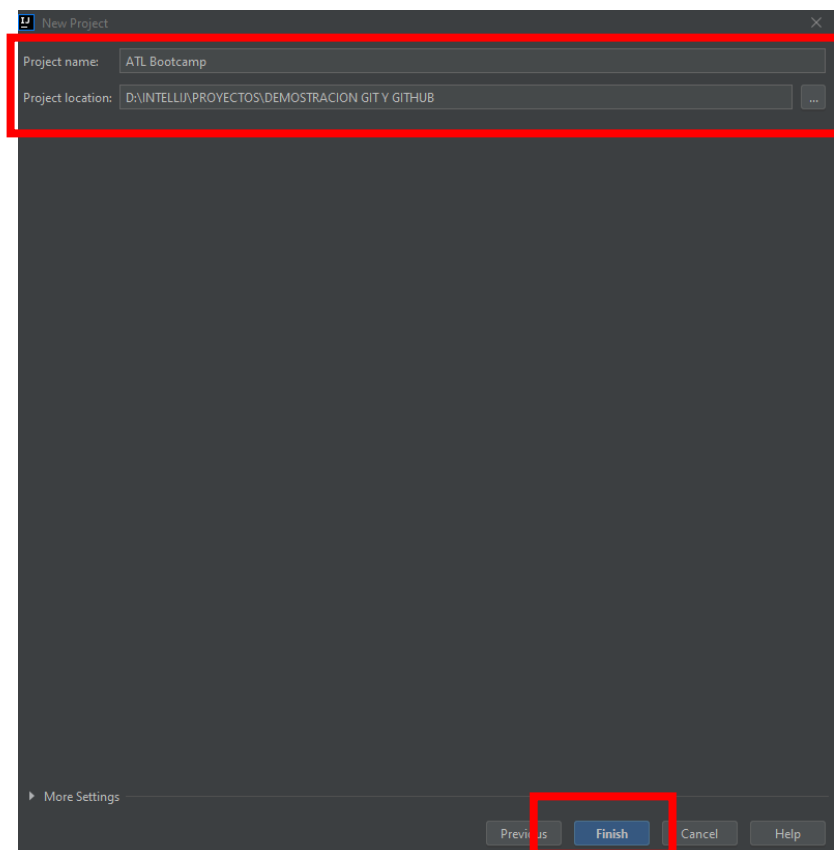
4. Se les abrirá otra pestaña, ahí tienen que crear un proyecto **JAVA**, y donde dice **Project SDK** utilizar la versión que quieran, como se puede ver yo tengo 17 Oracle OpenJDK versión 17.0.2 (o si no tienen descargado el JDK, apretar la opción **download JDK...**). Luego apretar **Next**.



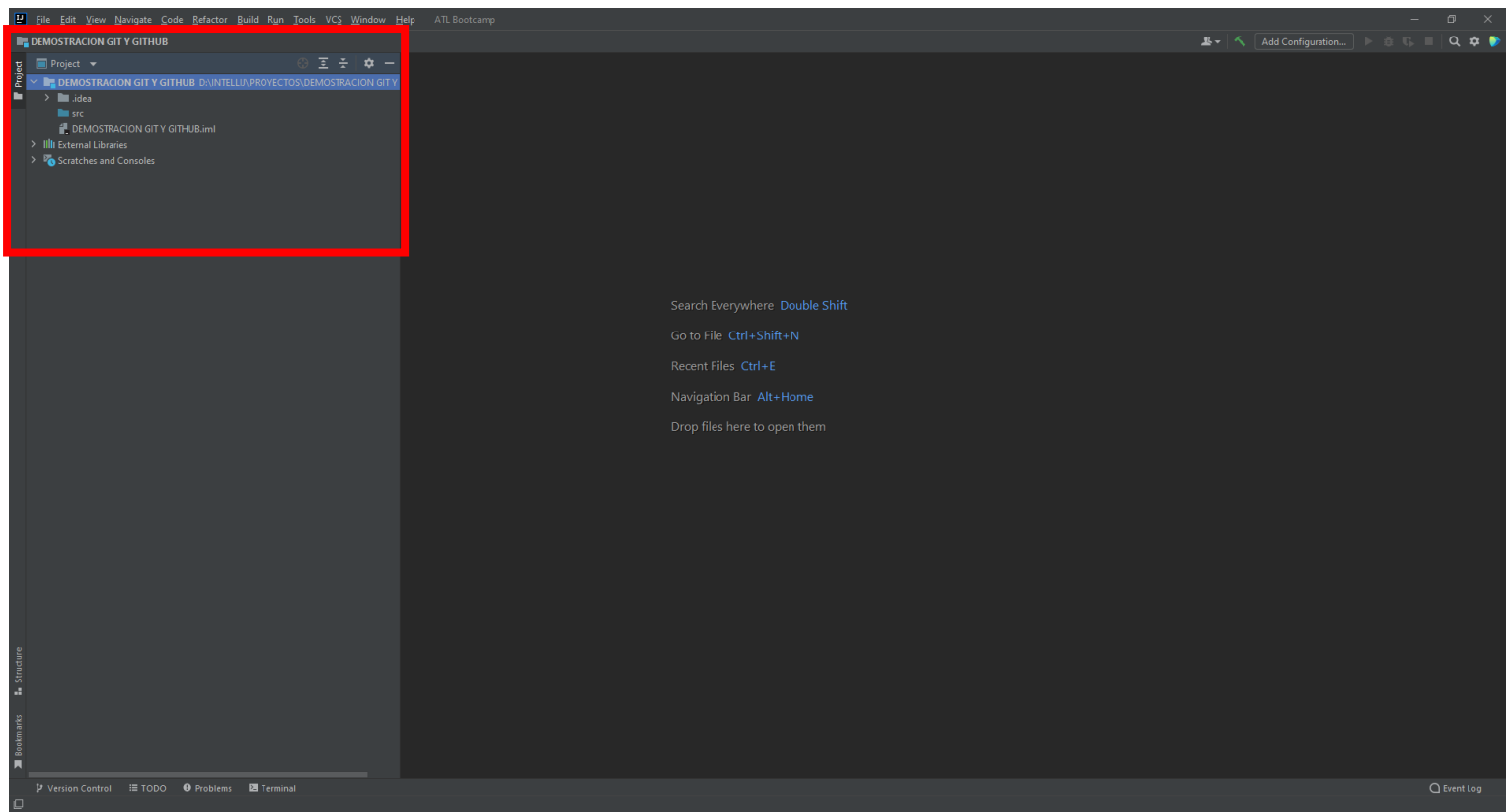
5. En la siguiente ventana le dan **Next**, sin modificar nada.



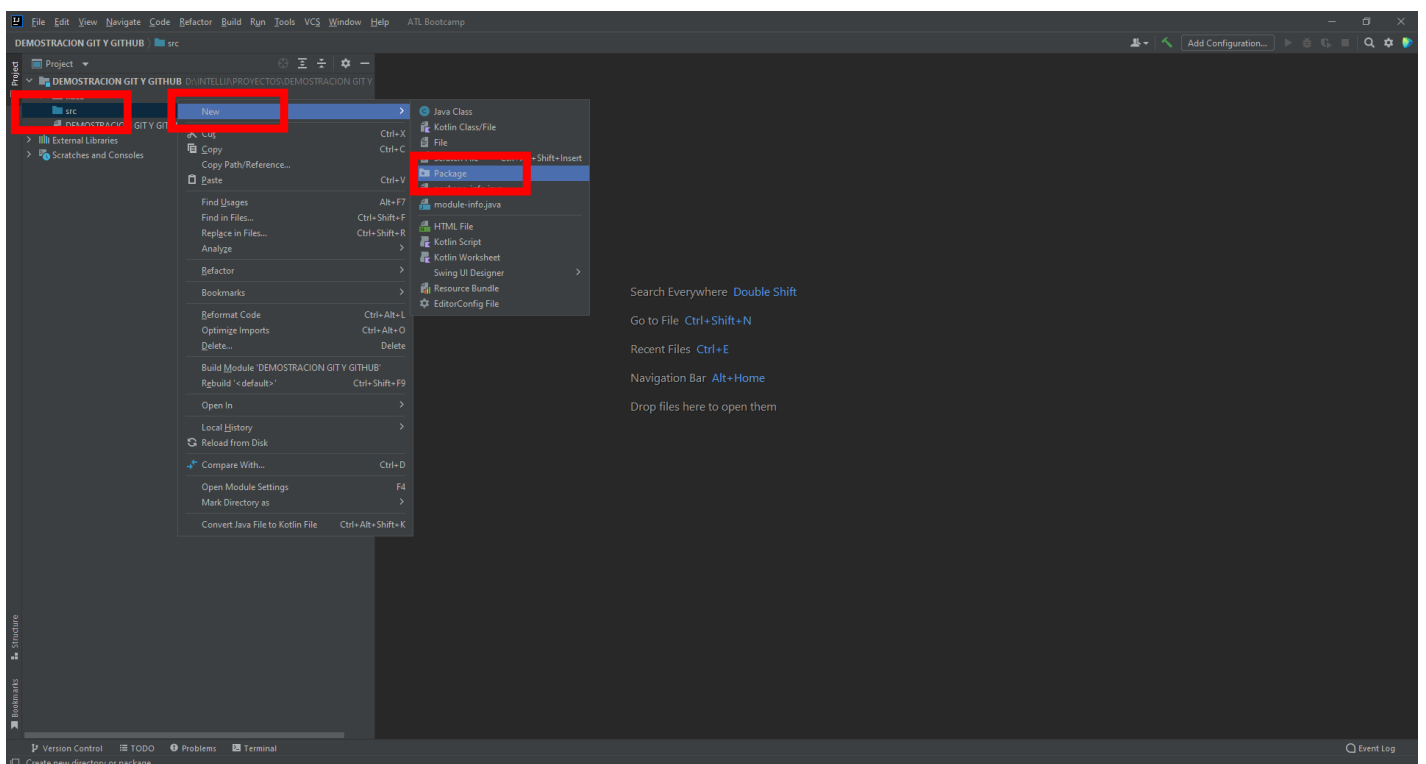
6. Nuevamente en la siguiente ventana, le asignan un nombre al proyecto **ATL Bootcamp** y le dan **Finish**. (**Project Location** es la dirección de su PC donde se les guardará el proyecto, acuérdense así cuando quieran entrar para algo en específico saben dónde está alojado el proyecto). En mi caso esta localizado en **D:/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB**.



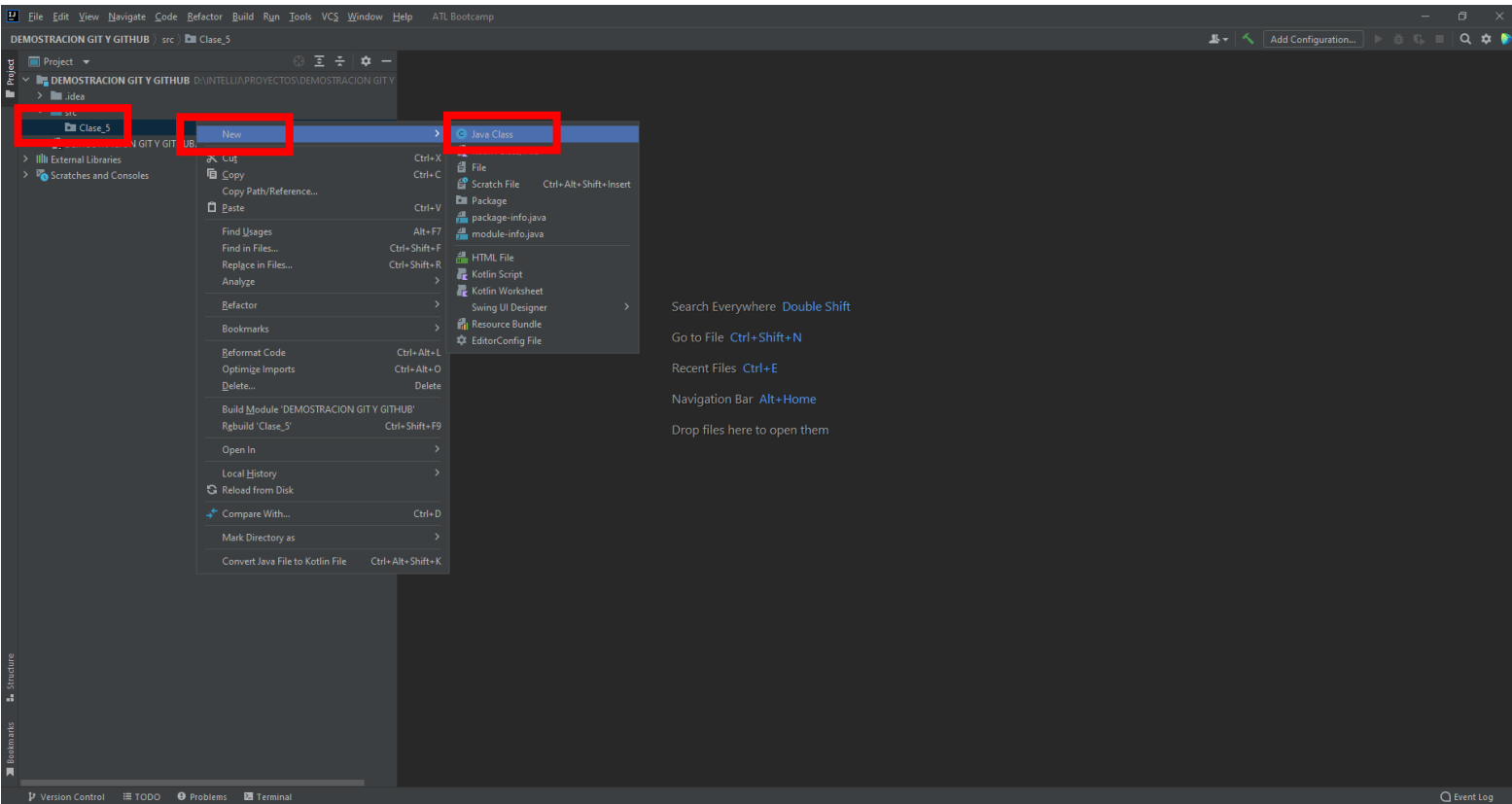
7. Si siguieron estos pasos, le saldrá todo el proyecto con sus carpetas



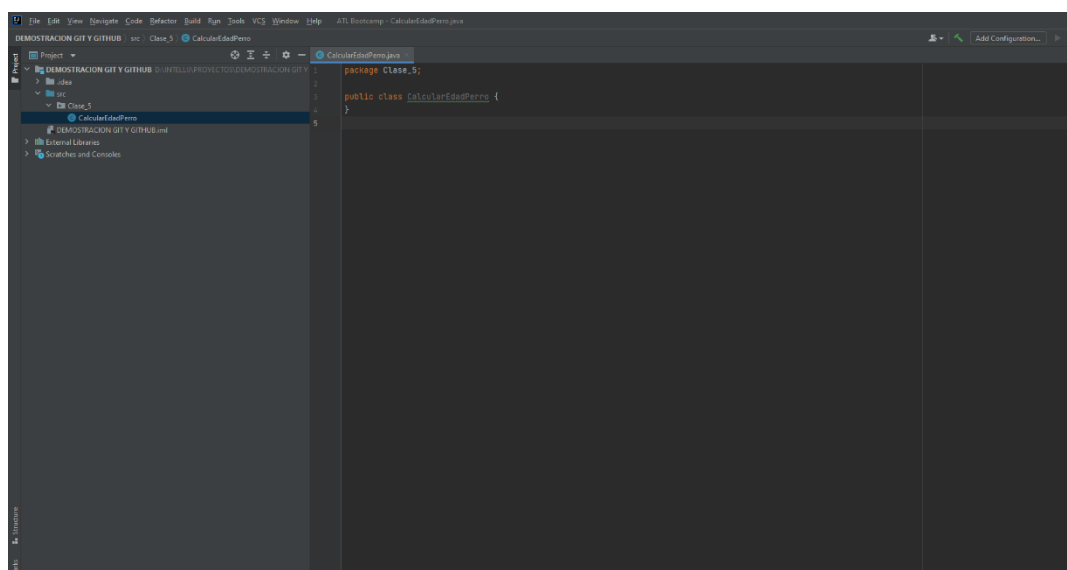
8. Donde dice **src** le dan click derecho con el mouse, **New**, **Package** y click botón izquierdo. Le pondremos de nombre Clase5 o Clase\_5. Pero no puede estar separados con espacios en blanco.



9. En el paquete (package) llamado Clase5 o Clase\_5 agregaremos clases, que representarán las actividades de la clase desde la actividad 1 a la 5. Yo las llamaré por un nombre convencional así se entiende. Por ejemplo, la actividad 1 la llamaré CalcularEdadPerro. Así sucesivamente... Para crear una clase voy al paquete Clase5 click botón derecho del mouse, New, Java Class le doy click botón izquierdo, y la nombro CalcularEdadPerro. Por convención el nombre de la clase empieza con mayúscula y por cada palabra diferente agrego mayúsculas.



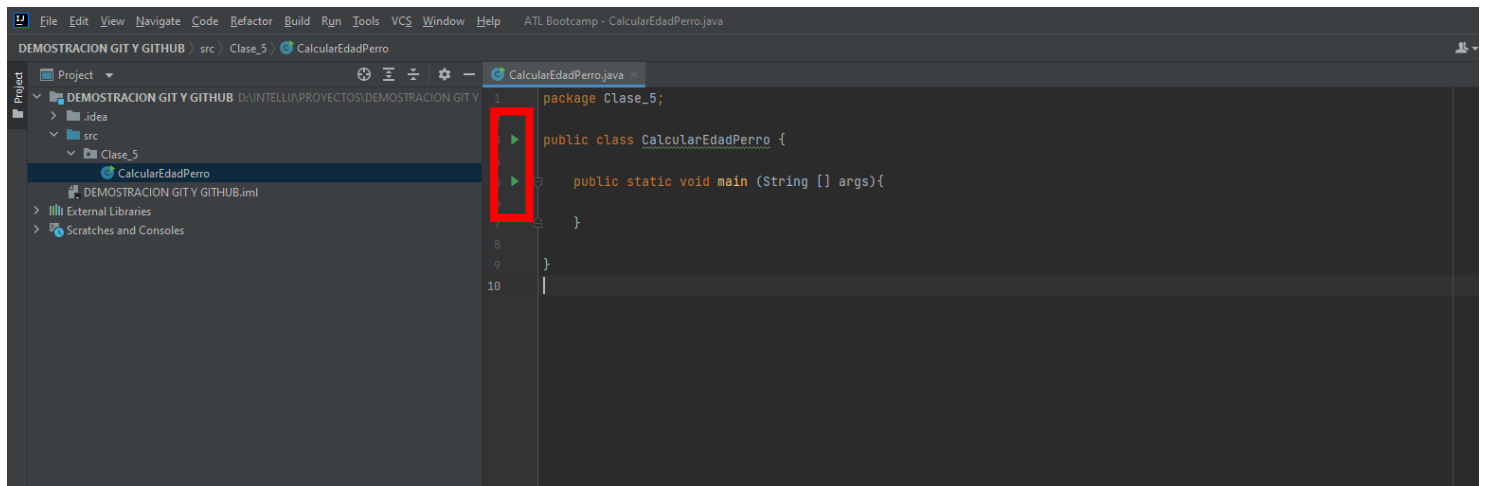
10. En este punto tenemos creado un proyecto Java, con un paquete llamado Clase5 y una clase CalcularEdadPerro que representa la actividad 1.



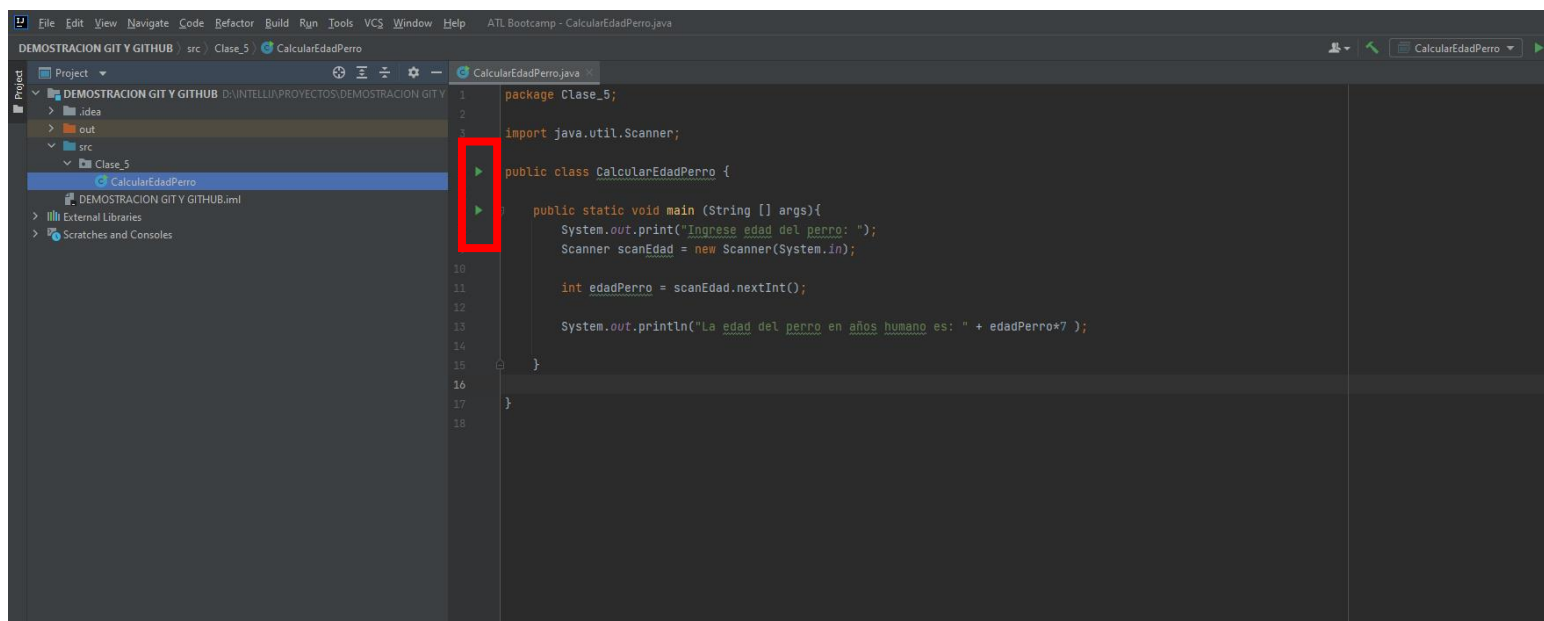
11. Como se puede ver tenemos creado la clase: **public class** *CalcularEdadPerro* {}, dentro de las llaves siempre (por ahora para nivel básico), debemos crear la función

**public static void** main (String [] args) {} y dentro de las llaves crear nuestro código. Es decir, para cada Java Class que creamos (para las 5 actividades de la clase 5 que vayamos a crear) debemos agregarle **public static void** main (String [] args) {} para que nuestro proyecto pueda funcionar.

Para que se entienda, debe quedar como sigue: y se observa como aparecieron 2 triángulos verdes.

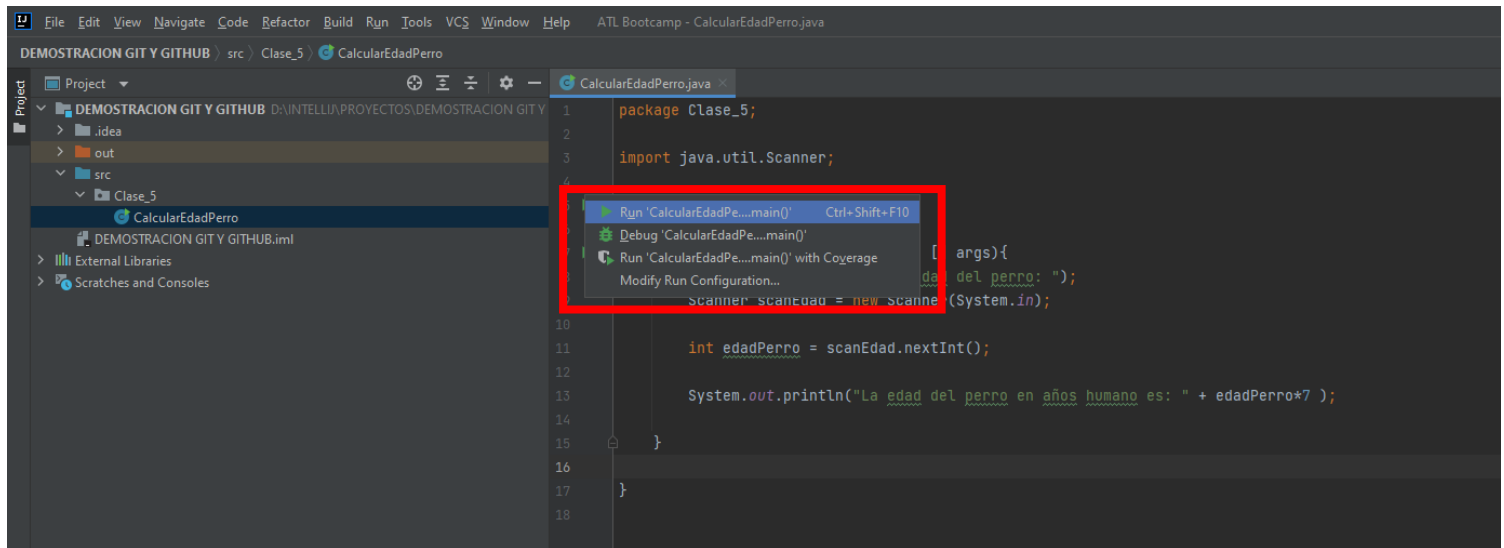


12. Una vez que tenemos esto hecho completamos la función, les muestro como lo hice yo.



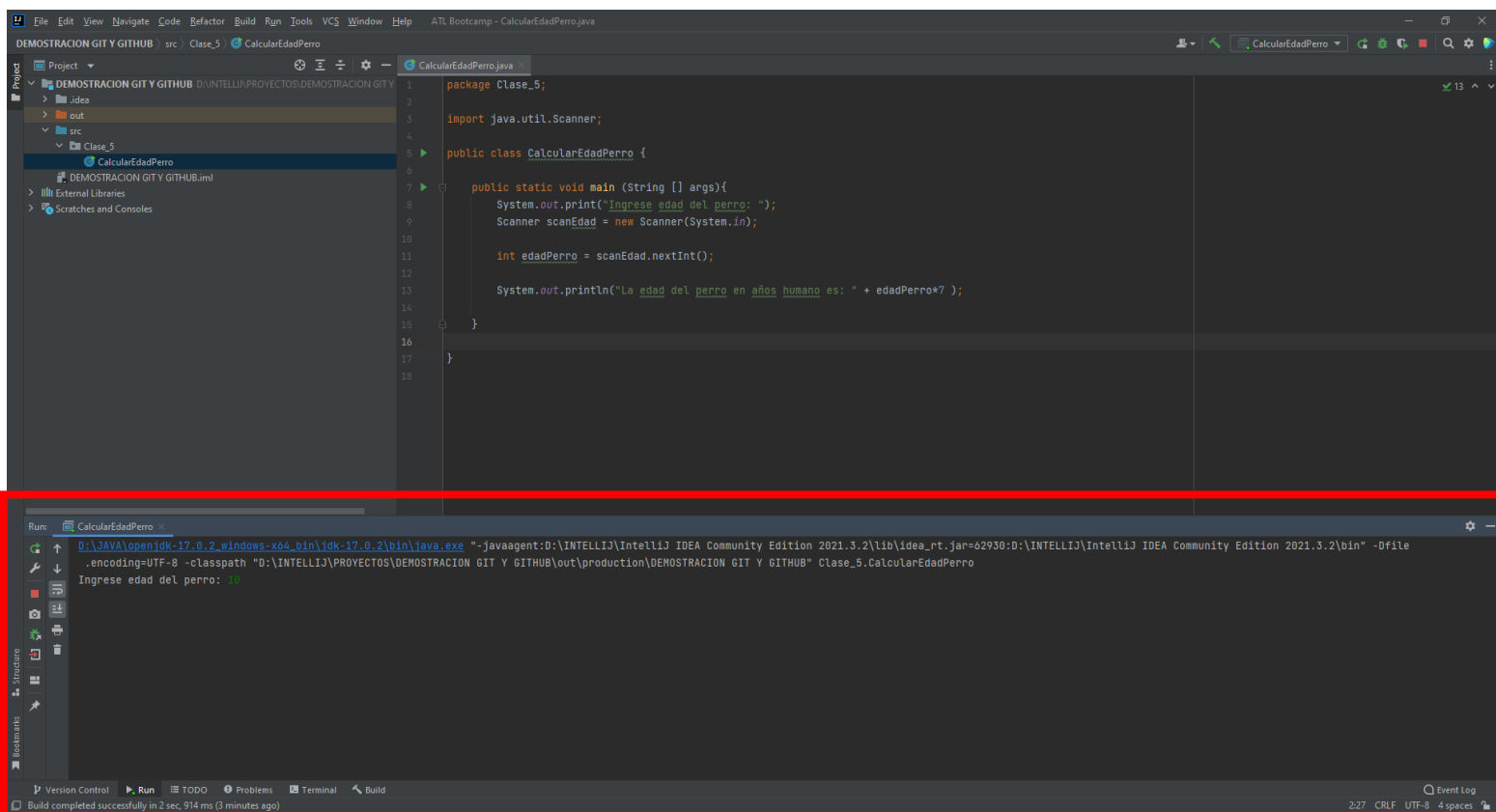
*Prestar atención a los punto y coma ";" y las llaves "{}". Son muy importantes en JAVA respetar el orden de abrir y cerrar llaves y poner punto y coma cada vez que terminamos una línea de comando. Con la práctica uno se termina acostumbrando. Al principio es normal olvidarse.*

13. Una vez que tenemos la función la vamos a correr y probar. Para ello vamos a utilizar los **2 triangulitos verdes**. Allí es donde le daremos click izquierdo del mouse para ejecutar el proyecto. Apretamos cualquiera de los botones verdes y luego le damos a Run.

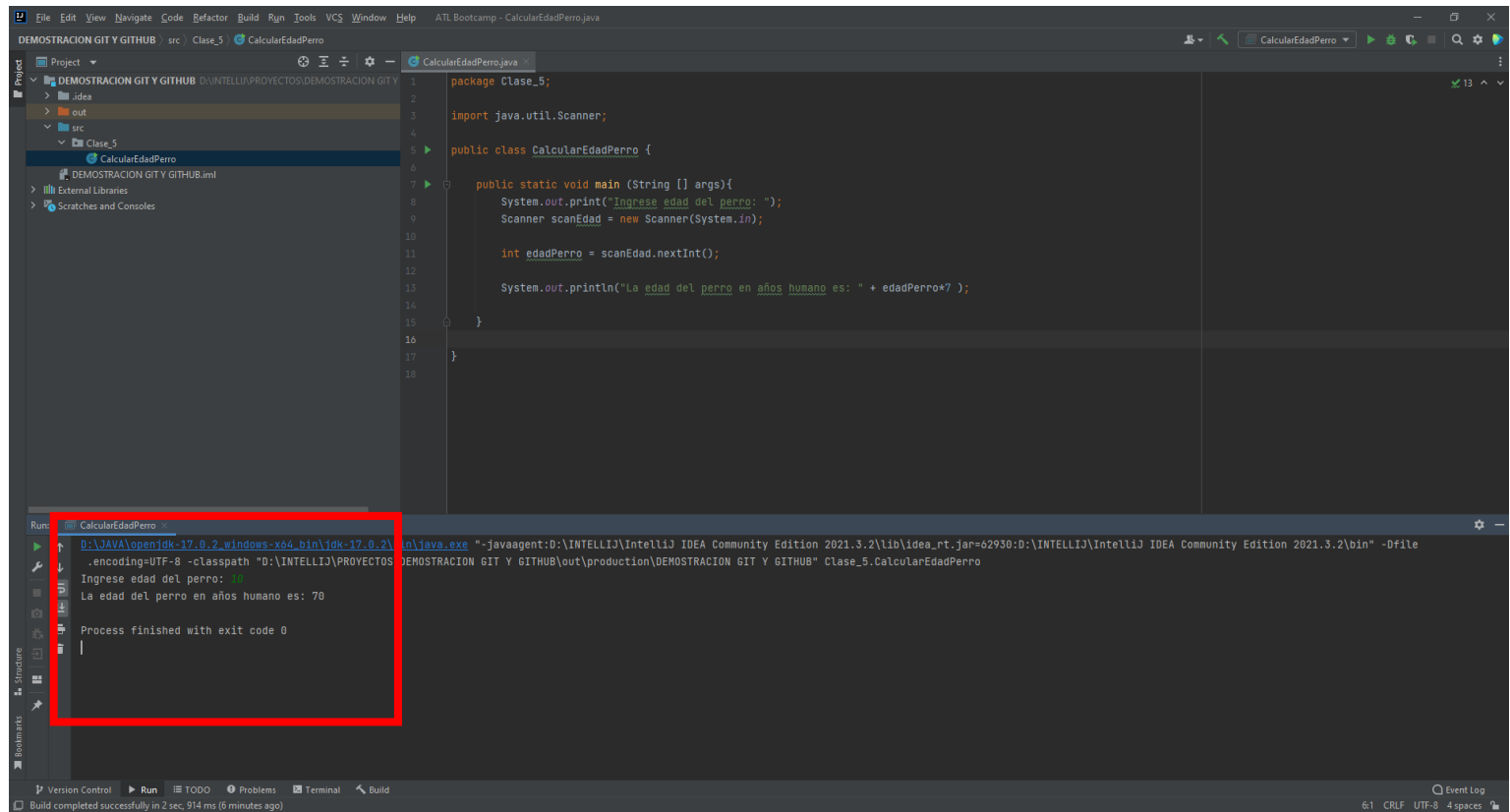


14. Una vez que se ejecuta el programa vemos abajo en la consola ejecutándose el proyecto.

Donde dice Ingrese edad del perro: yo escribí 10 desde teclado, el programa no avanzará hasta que no se ingrese un numero entero y le demos Enter.



15. Una vez que le dimos Enter nos mostrará el resultado a mí me dio 70 años Por lo que funciona correctamente. Si todo esta bien podemos proceder a guardar en Git o GitHub. nuestro repositorio.



```
package Clase_5;

import java.util.Scanner;

public class CalcularEdadPerro {

    public static void main (String [] args){
        System.out.print("Ingrese edad del perro: ");
        Scanner scanEdad = new Scanner(System.in);

        int edadPerro = scanEdad.nextInt();

        System.out.println("La edad del perro en años humano es: " + edadPerro*7 );
    }
}
```

Run: `D:\JAVA\openjdk-17.0.2_windows-x64_bin\jdk-17.0.2\bin\java.exe -javaagent:D:\IntelliJ\IntelliJ IDEA Community Edition 2021.3.2\lib\idea_rt.jar=62930:D:\IntelliJ\IntelliJ IDEA Community Edition 2021.3.2\bin" -Dfile.encoding=UTF-8 -classpath "D:\IntelliJ\PROYECTOS\DEMOSTRACION GIT Y GITHUB\out\production\DEMOSTRACION GIT Y GITHUB" Clase_5.CalcularEdadPerro`

Ingrese edad del perro: 70

La edad del perro en años humano es: 70

Process finished with exit code 0

Hasta este punto hemos creado un **proyecto en IntelliJ**, donde creamos un **paquete que representa la Clase\_5** de ATL Bootcamp, y dentro de la clase 5 creamos una **clase Java que representa nuestra actividad 1**, es decir la de calcular edad de perros. Los demás ejercicios lo deberían completar ustedes.

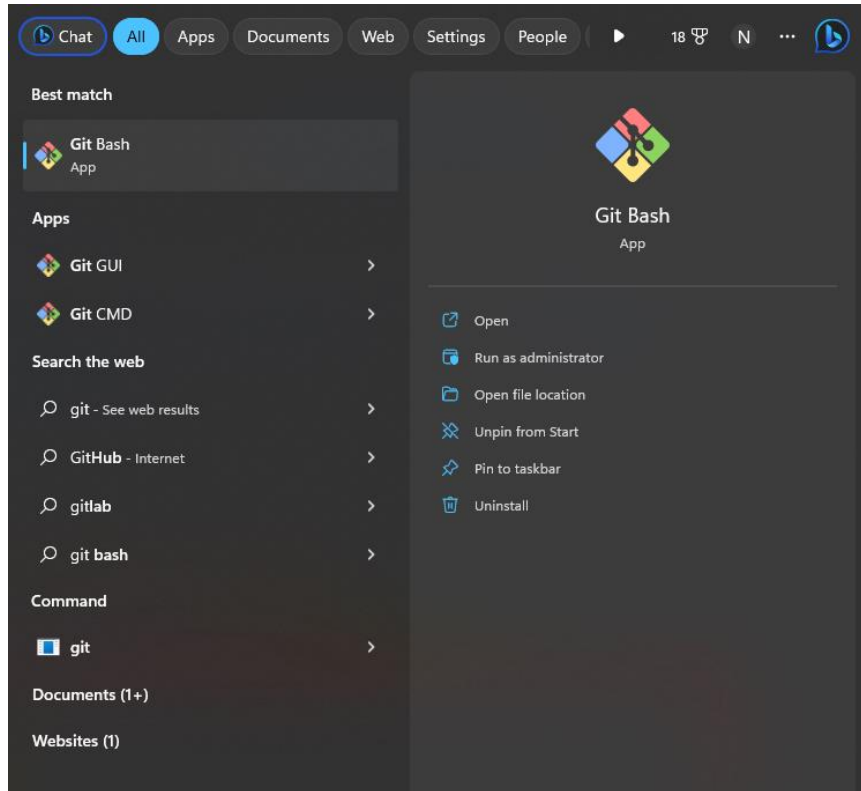
*Las demás clases las crearé como paquetes y dentro de cada paquete creo una clase Java que Representará a cada actividad.*

Una vez que completen los 5 ejercicios procedemos a guardar nuestro proyecto en un repositorio local en Git desde la terminal y en remoto, es decir en GitHub.



## GUARDAR PROYECTO DE INTELLIJ EN GIT DESDE LA TERMINAL

Vamos a instalar Git. Primero lo descargamos <https://git-scm.com/downloads>. Según qué sistema operativo estén usando. Dato importante para Windows, cuando instalen Git deben ver que se haya instalado Git Bash ya que NUNCA USAREMOS LA TERMINAL CM DE WINDOWS. Ya que maneja otros comandos y no vamos a entender nada. Siempre abrir la terminal GIT BASH.

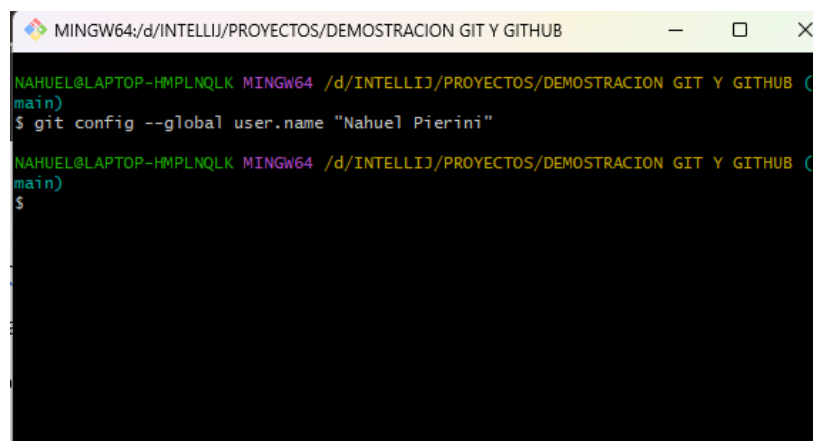


## CONFIGURACION DE GIT CON USUARIO E EMAIL

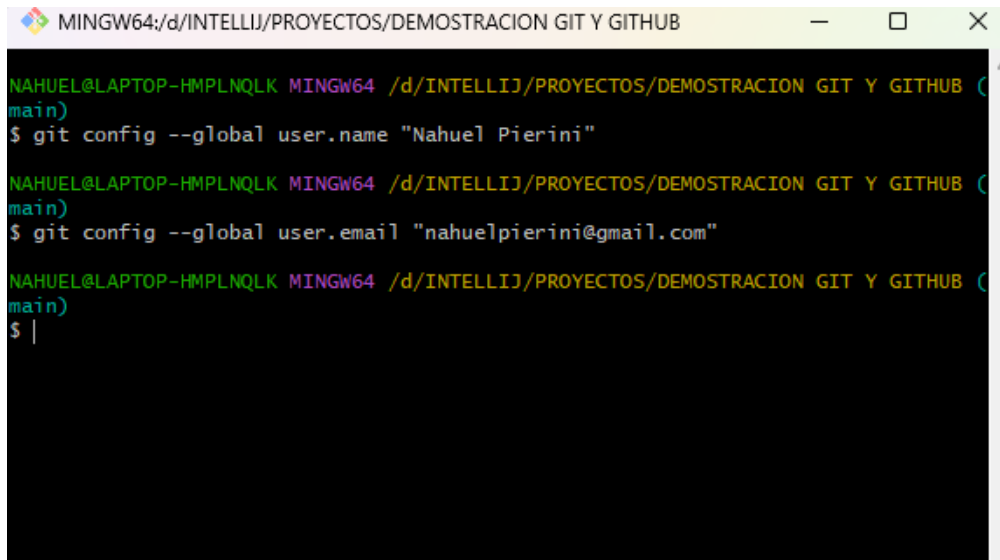
En el terminal primero configuraremos nuestro email y nombre.

Para ello al abrir la terminal escribo:

- Git config --global user.name "Nahuel Pierini" y le doy enter. Con esto asigno mi nombre a Git. (En -- global son dos guiones medio)



- b. `git config --global user.email "nahuelpierini@gmail.com"` : con esto asigno mi email a Git es el mismo que uso en GitHub para que no haya problemas mas tarde al usar GitHub.



```
MINGW64;D:/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB
NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (
main)
$ git config --global user.name "Nahuel Pierini"
NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (
main)
$ git config --global user.email "nahuelpierini@gmail.com"
NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (
main)
$ |
```

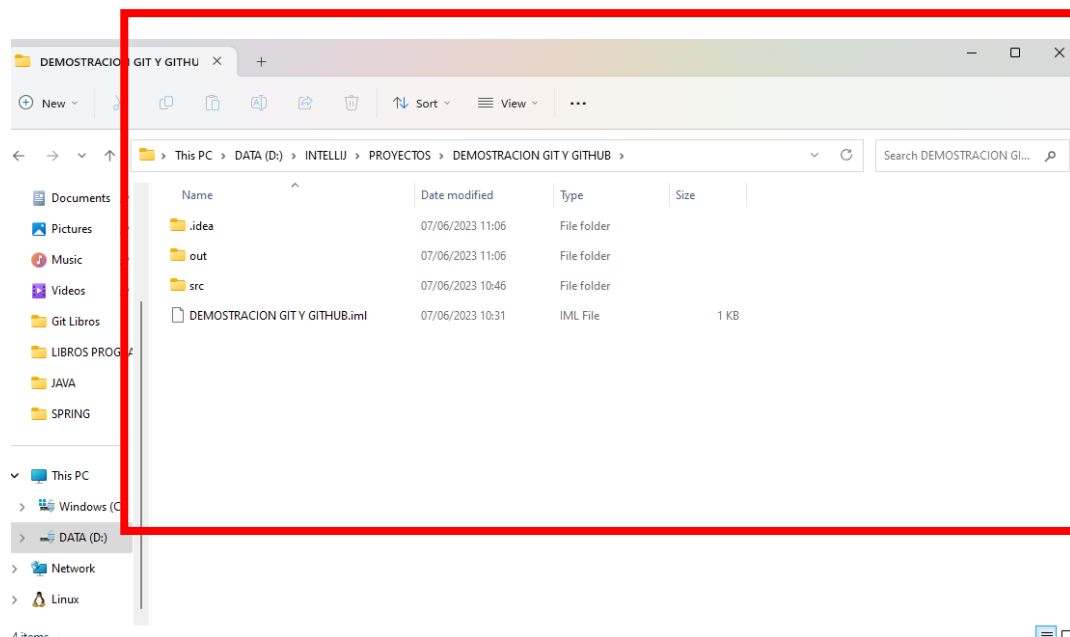
Con estos dos pasos tengo configurado mi usuario y email en git.

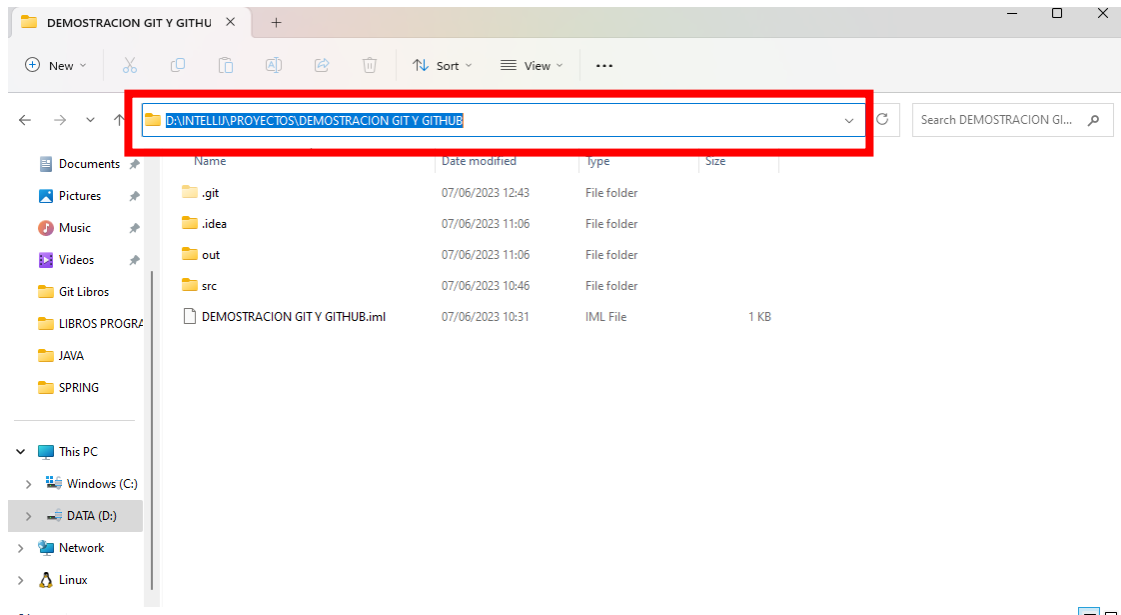
## VOLVEMOS A NUESTRO PROYECTO DE INTELLIJ

Vamos a entender esto. Nuestro proyecto que en el inciso 6 lo llamamos “ATL Bootcamp” quedó guardado en una carpeta en nuestra PC, en mi caso quedó en la siguiente ruta

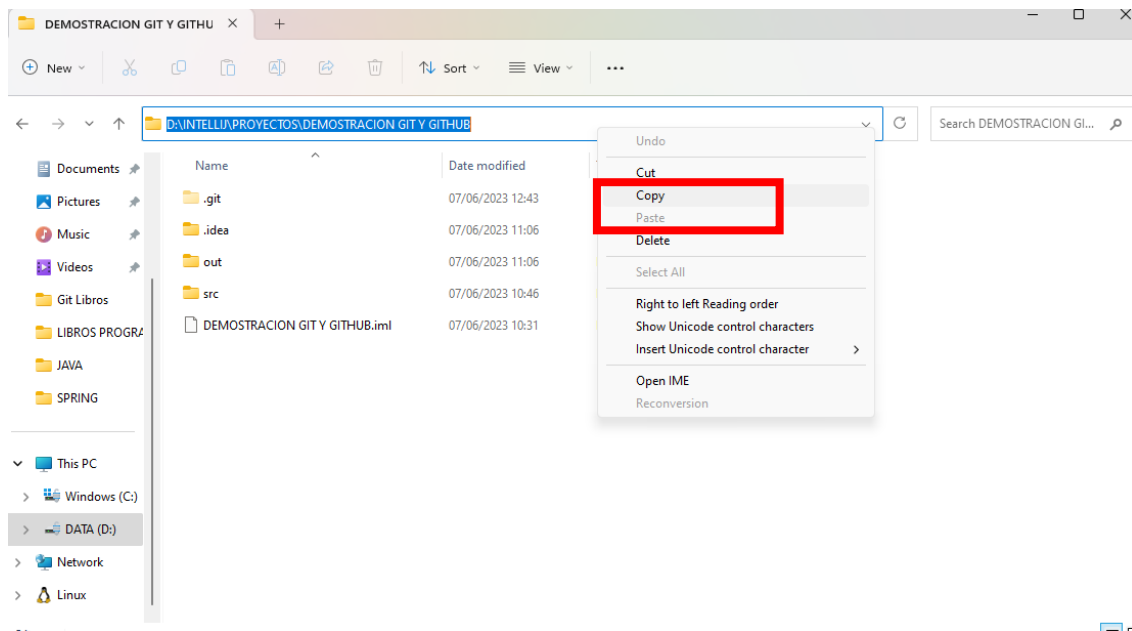
`D:/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB`.

Para copiar la ruta donde tengo mi proyecto, lo voy a buscar en mi PC donde lo guardé y simplemente con el mouse doy click izquierdo en la ruta:





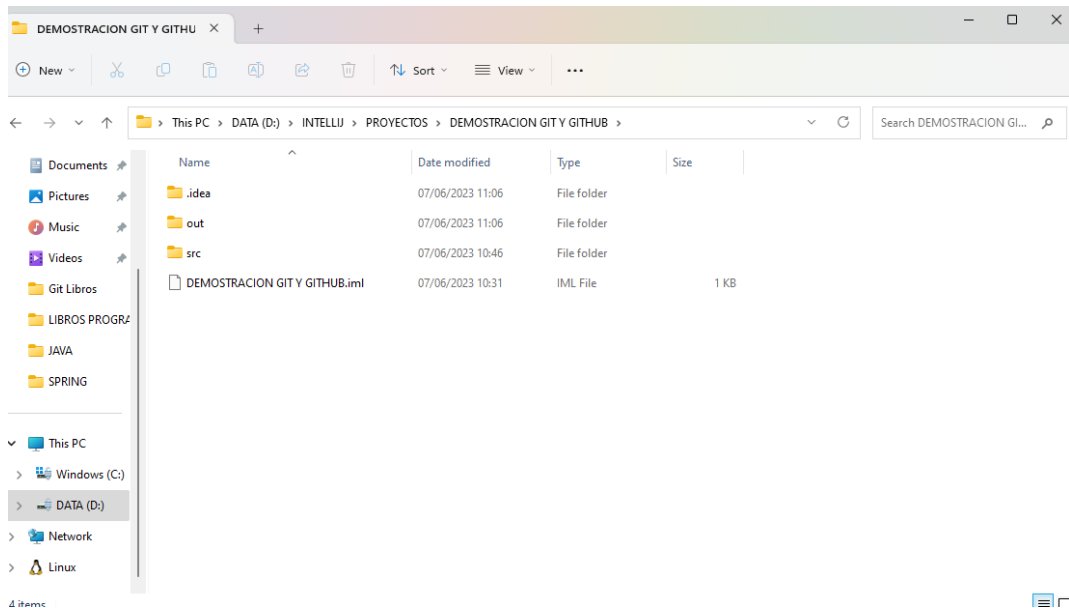
Luego click botón derecho con el mouse y copiar (Copy), **LISTO TENGO MI RUTA COPIADA:**



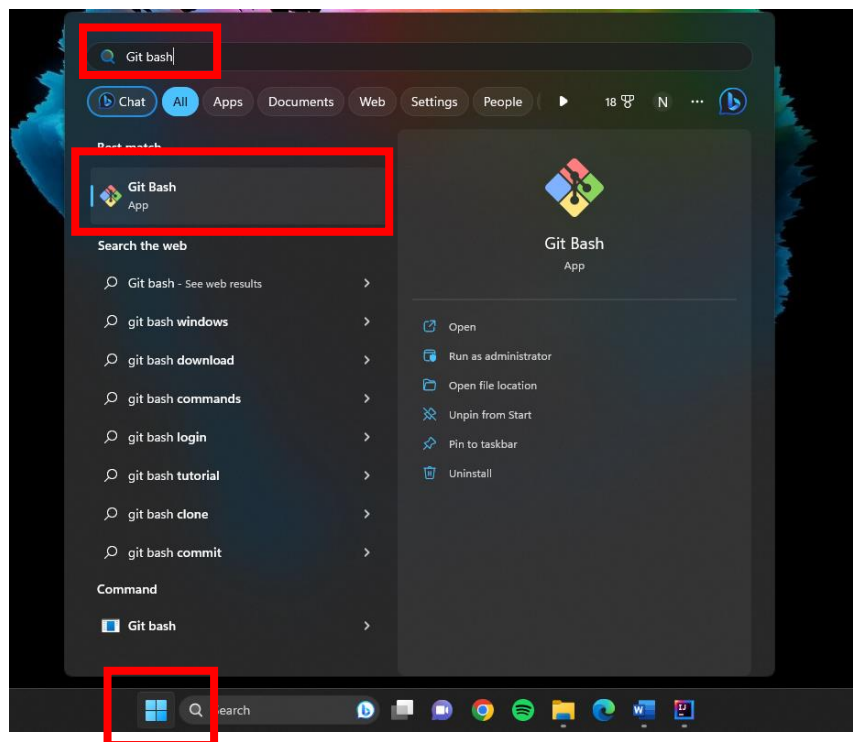
Ahora bien. Dentro de esa carpeta vamos a crear un repositorio GIT, que nos va a permitir controlar las versiones de nuestro proyecto. Cuando se crea esta carpeta de Git, es invisible, no la podremos ver (salvo que hagamos ver carpetas ocultas configurando Windows o el sistema operativo que tengan). Ahora vamos a ver de que trata esto.

1. Me dirijo a la carpeta donde cree el proyecto en mi caso:

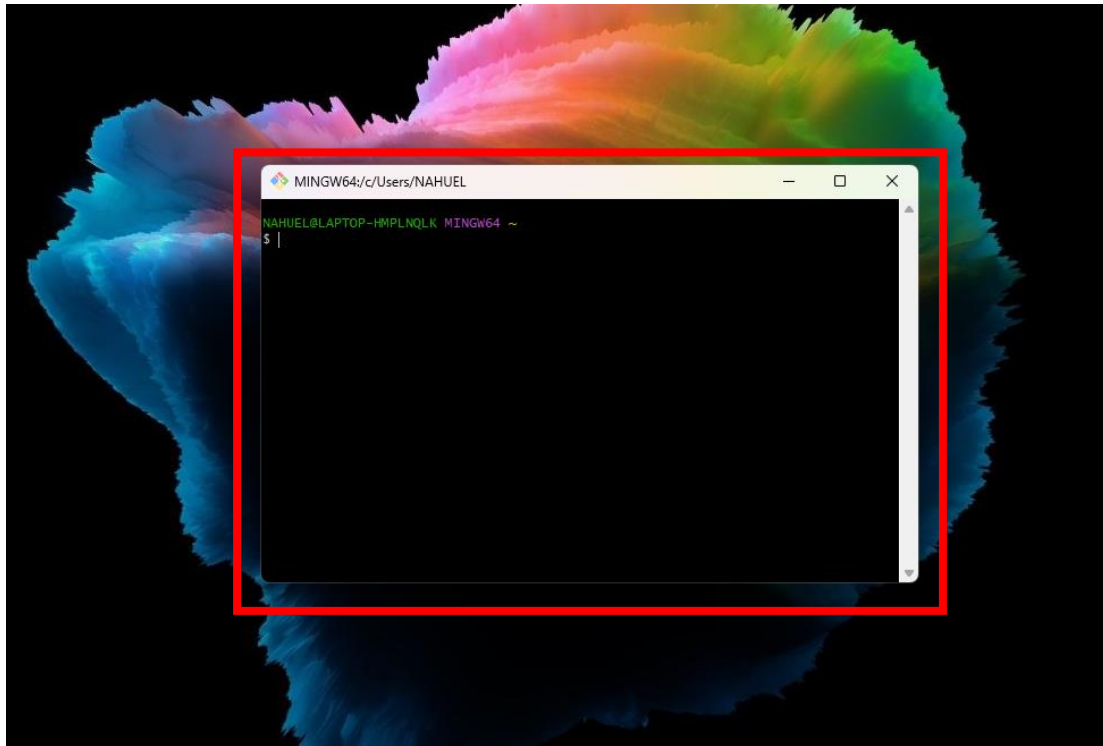
D:/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB



2. Voy a abrir Git Bash, como yo uso Windows lo abro desde el Inicio, en buscar escribo Git Bash y le doy click. Tengo la terminal abierta y lista para trabajar.



3. Con la terminal abierta, se verá así:



4. Lo primero es ir a la ruta de mi carpeta donde cree el proyecto. Para ello escribo en la terminal `cd 'Ruta donde guarde mi proyecto'`. En mi caso es:

*`cd D:/INTELLIJ/PROYECTOS/'DEMOSTRACION GIT Y GITHUB'`*

así tal cual y le doy enter. SOLO TENER EN CUENTA DOS COSAS, las dejo remarcadas abajo:

como la ruta:

`/DEMOSTRACION GIT Y GITHUB`

contiene espacios en blanco **SI O SI** debo escribir todo eso

entre comillas simples:

`'DEMOSTRACION GIT Y GITHUB'`.

Esto es cuando en la ruta de acceso tengo espacios en blanco. De lo contrario

no nos ubicaremos en nuestro proyecto

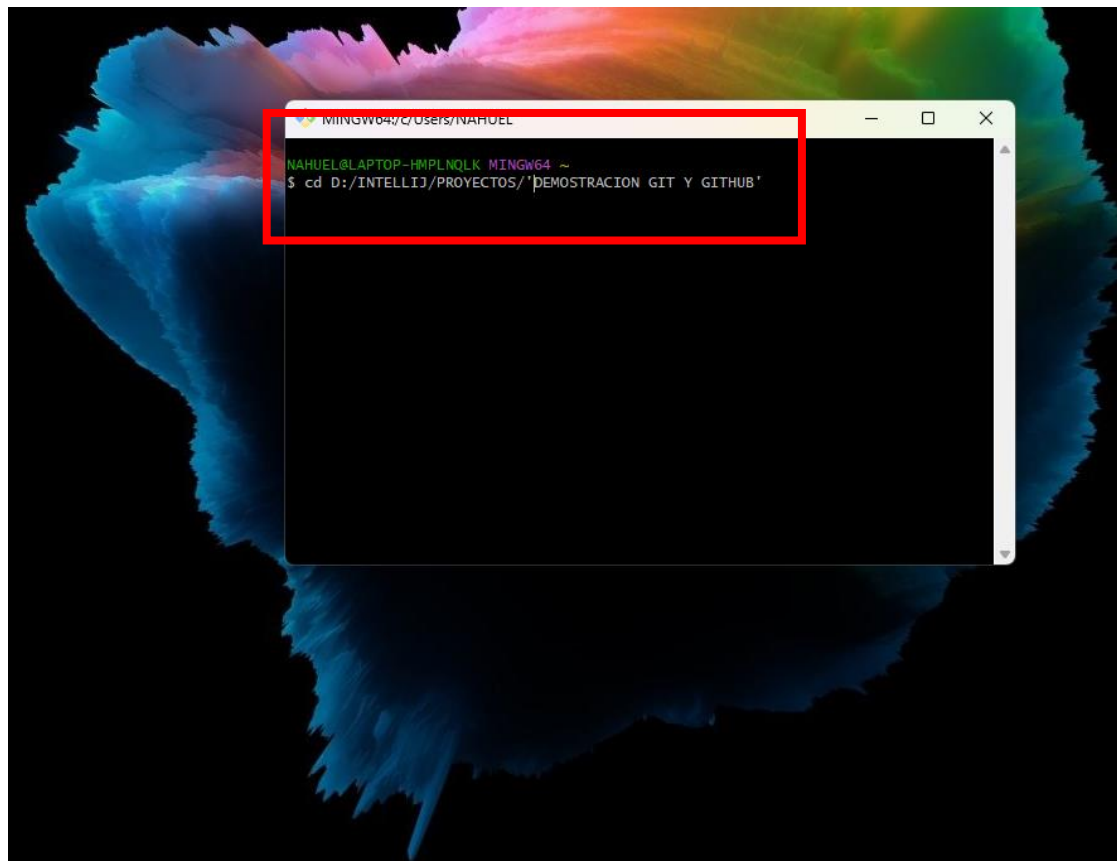
#### NOTA:

Cuando copio la ruta original desde me quedan las barras diagonales invertidas, las debo cambiar sino Git Bash puede que se trabe. Es decir, al copiar y pegar la ruta de mi proyecto se vería así:

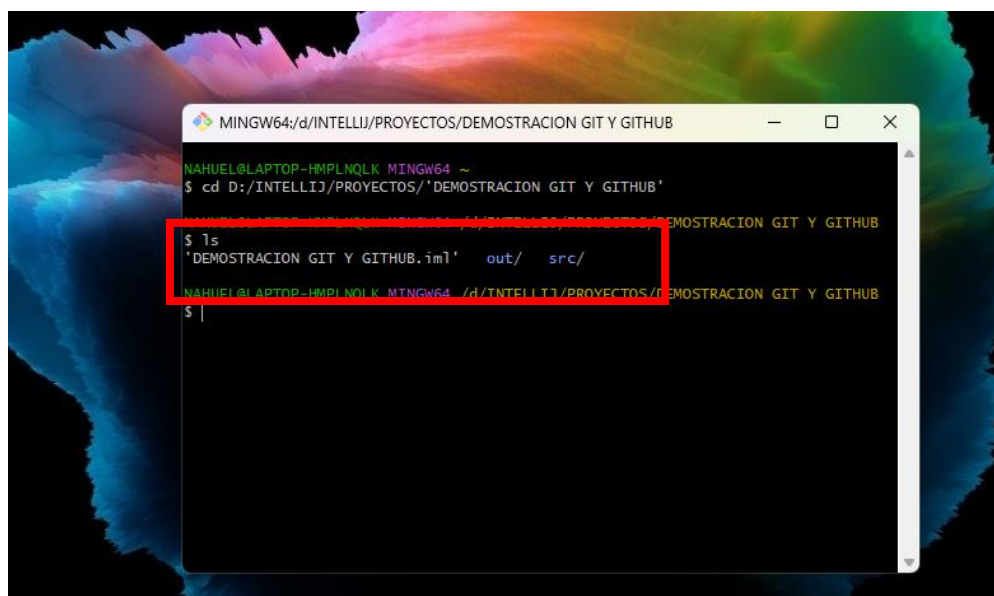
`D:\INTELLIJ\PROYECTOS\DEMOSTRACION GIT Y GITHUB`

Pero en Git Bash debo alterar las barras `"\"` por estas `"/` de la siguiente forma:

`D:/INTELLIJ/PROYECTOS/'DEMOSTRACION GIT Y GITHUB'` En la siguiente imagen se puede ver como quedó:



5. Escribo el comando “ls” y le doy enter para ver los archivos que contiene la carpeta.



6. En este punto me encuentro ubicado en la carpeta del proyecto y de ahora en más Git me permite subir repositorios de código para almacenarlo en el sistema de control de versiones Git. Para ello vamos a usar los comandos de Git para crear nuestro primer repositorio. Ellos son

- a. **Git init** (Inicializa mi repositorio Git del proyecto)

```
MINGW64:/d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB
NAHUEL@LAPTOP-HMPLNQLK MINGW64 ~
$ cd D:/INTELLIJ/PROYECTOS/'DEMOSTRACION GIT Y GITHUB'

NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB
$ ls
'DEMOSTRACION GIT Y GITHUB.iml'  out/  src/

NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB
$ git init
Initialized empty Git repository in D:/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB/.git/

NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (master)
$ |
```

- b. **Git add .** (Es la primera forma de agregar los archivos de nuestro proyecto al stage, el punto significa que agregará todas las carpetas, este proceso no guarda en nuestro repositorio simplemente los agrega para luego con git commit poder ser guardados). *Importante agregar el punto "." En el comando "git add ."*

```
MINGW64:/d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB
NAHUEL@LAPTOP-HMPLNQLK MINGW64 ~
$ cd D:/INTELLIJ/PROYECTOS/'DEMOSTRACION GIT Y GITHUB'

NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB
$ ls
'DEMOSTRACION GIT Y GITHUB.iml'  out/  src/

NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB
$ git init
Initialized empty Git repository in D:/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB/.git/

NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (master)
$ git add .

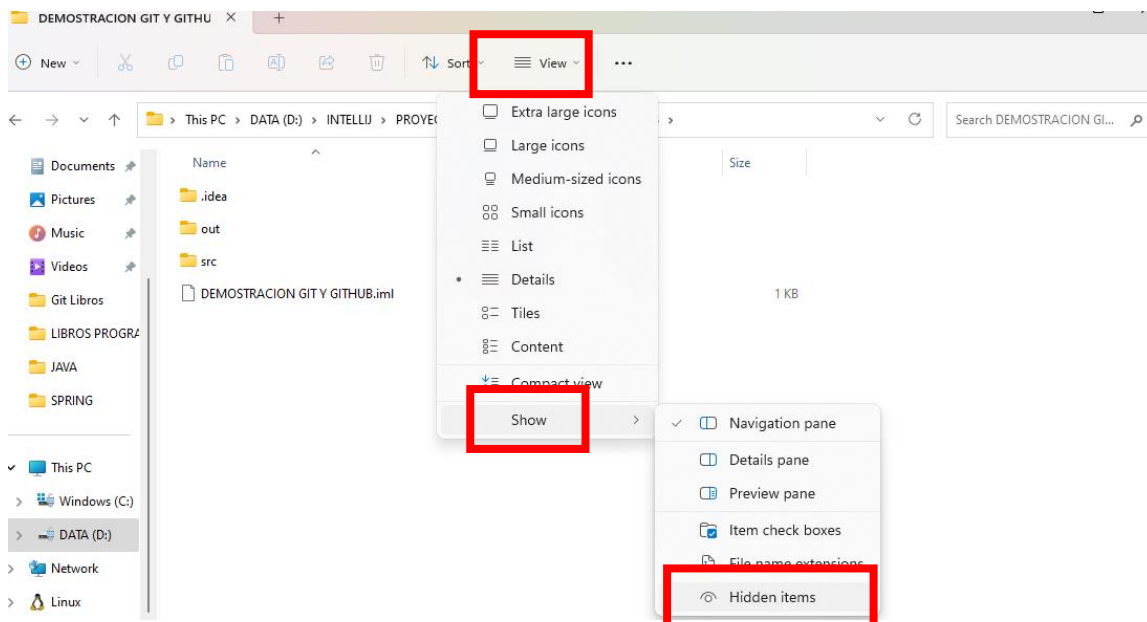
NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (master)
$ |
```

- c. **Git commit -m "mensaje"** (Vamos a comitear nuestro proyecto y en el siempre escribir un mensaje para saber que estamos haciendo, en este punto guardamos todo lo que hayamos agregado a nuestro repositorio con el comando anterior)

```
MINGW64:/d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB
HUB/.git/
NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (
master)
$ git add .
NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (
master)
$ git commit -m "Creo clase 5 Actividad 1(CalcularEdadPerro)"
[master (root-commit) 417a4bf] Creo clase 5 Actividad 1(CalcularEdadPerro)
8 files changed, 173 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/.name
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/uiDesigner.xml
create mode 100644 DEMOSTRACION GIT Y GITHUB.iml
create mode 100644 out/production/DEMOSTRACION GIT Y GITHUB/Clase_5/CalcularEda
dPerro.class
create mode 100644 src/Clase_5/CalcularEdadPerro.java
NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (
master)
$
```

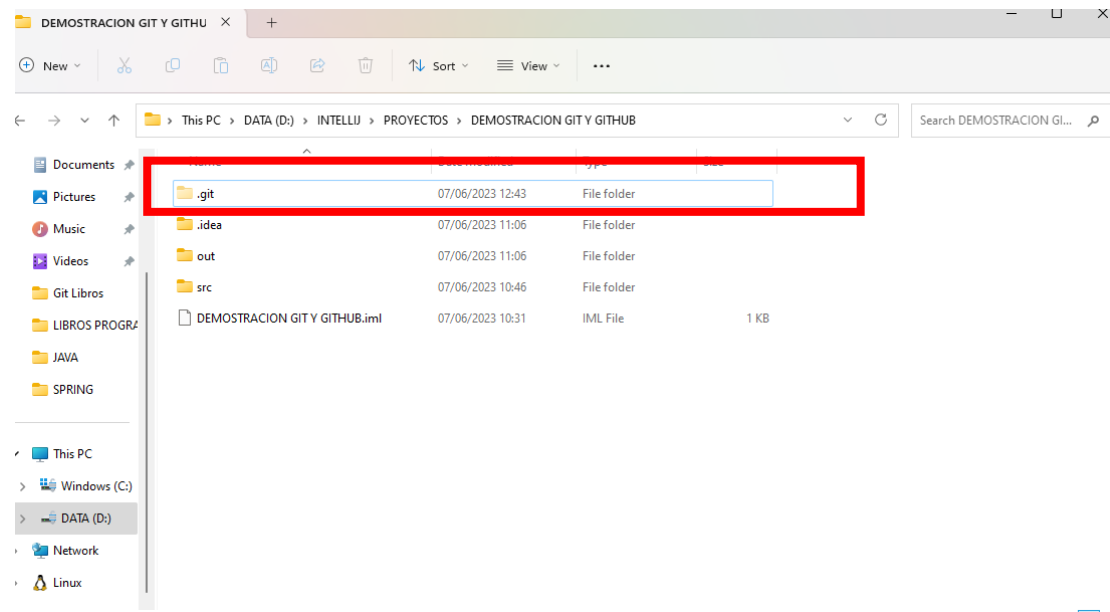
7. Si todo salió correctamente se les debería ver de esa forma. Ahora para entender vamos a ver que acabamos de crear. En nuestra ruta del proyecto, en mi caso: [D:/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB](#). Vamos a ver carpetas ocultas. Desde Windows lo hago de esta forma (En mi caso tengo la PC en inglés):

- a. Ver (View)
- b. Mostrar (Show)
- c. items ocultos (Hidden items)





8. Veremos como apareció una carpeta oculta que creamos con todos los comandos de Git. *Ese es nuestro repositorio local de Git.* Dentro hay muchas cosas raras. Mejor no tocar nada.



Con estos pasos ya tenemos creado un repositorio de Git en nuestro proyecto, con el cual podremos trabajar. Con Git.

## PASOS A SEGUIR PARA TRABAJAR CON LA CONSOLA GIT BASH

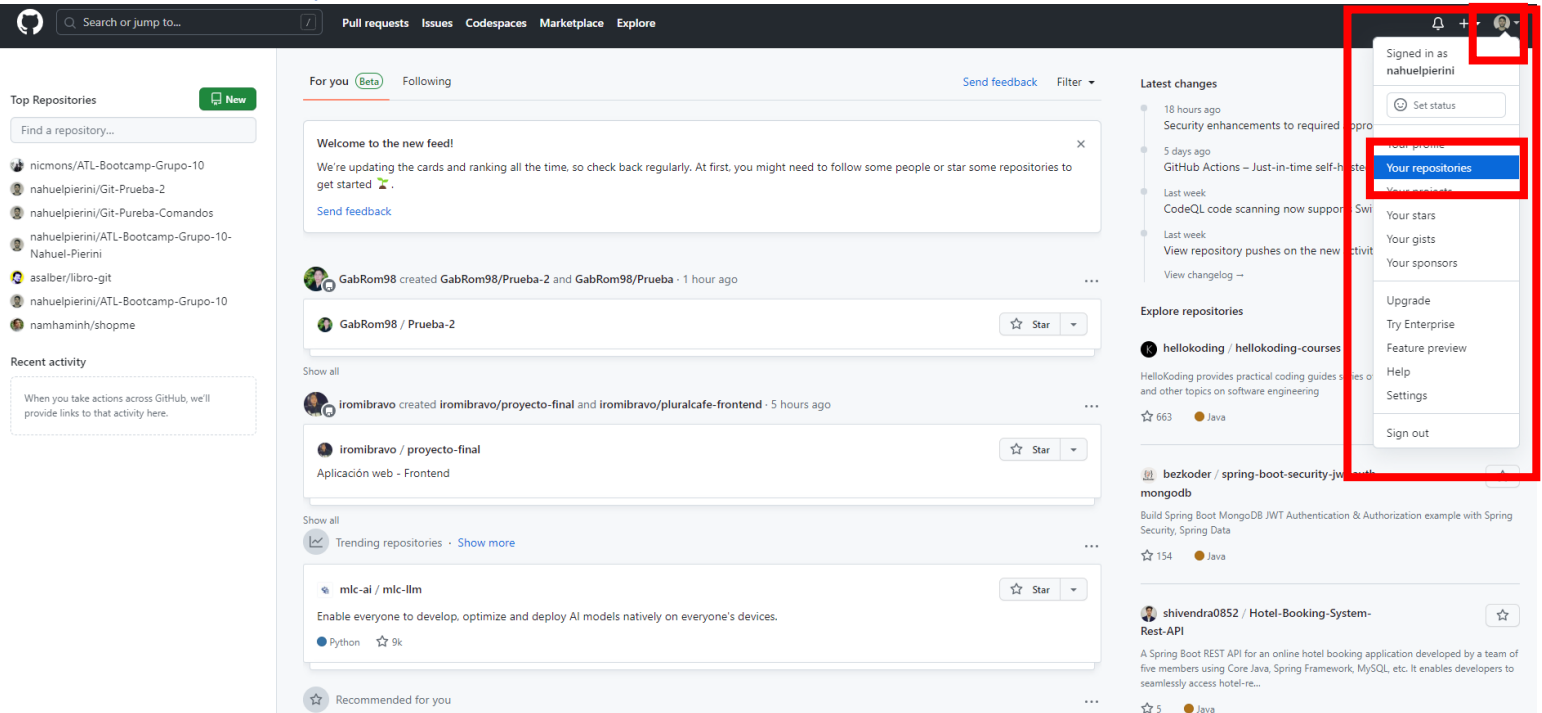
Pasos	PASOS	COMANDOS GIT
1	Dirigirse a la ruta donde tengo el proyecto guardado	<code>cd &lt;ruta de mi carpeta exacta&gt;</code> tener en cuenta que los espacios en blanco para git no serán leído correctamente, para ello uso comillas simples
2	Creo mi repositorio local	<code>Git init</code>
3	Agrego mis archivos al stage	<code>Git add .</code>
4	Guardo los archivos en el stage	<code>Git commit -m "mensaje"</code>

# GUARDAR PROYECTO DE INTELIJ EN GITHUB DESDE LA TERMINAL

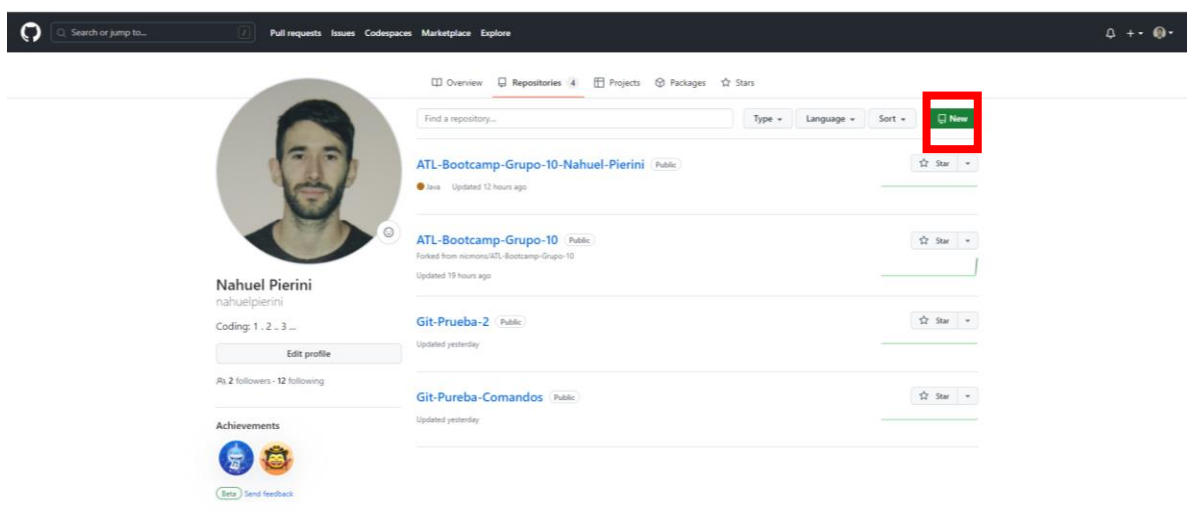
En esta sección lo que haremos es guardar nuestro repositorio de manera remota, para nuestro caso, en GitHub.

Voy a crear un proyecto desde 0 en GitHub.

1. Voy a la derecha donde [está la foto o sin foto de mi usuario](#) le doy click y luego a [Your repositories](#).



2. Luego en mi repositorio le doy a **New** que está en verde.



9. Le asigno un nombre, en este caso le puse Bootcamp y luego doy click a **Create Repository**.

Issues Codespaces Marketplace Explore

## Create a new repository

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

Owner \*  
nahuelpierini

Repository name \*  
Bootcamp

Great repository names are short and memorable. Need inspiration? How about [improved-octo-dollop?](#)

Description (optional)

☒ **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:

☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: **None**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: **None**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

☐ You are creating a public repository in your personal account.

Create repository

10. En este punto veremos muchos comandos raros. Pero nos daremos cuenta que nos dice que es lo que hay que hacer.

Search or jump to... Pull requests Issues Codespaces Marketplace Explore

nahuelpierini / Bootcamp Public

Pin Unwatch Fork 0 Star 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

**Set up GitHub Copilot**  
Use GitHub's AI pair programmer to autocomplete suggestions as you code.

**Invite collaborators**  
Find people using their GitHub username or email address.

**Quick setup — if you've done this kind of thing before**

Set up in Desktop or **HTTPS** SSH `https://github.com/nahuelpierini/Bootcamp.git`

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

**...or create a new repository on the command line**

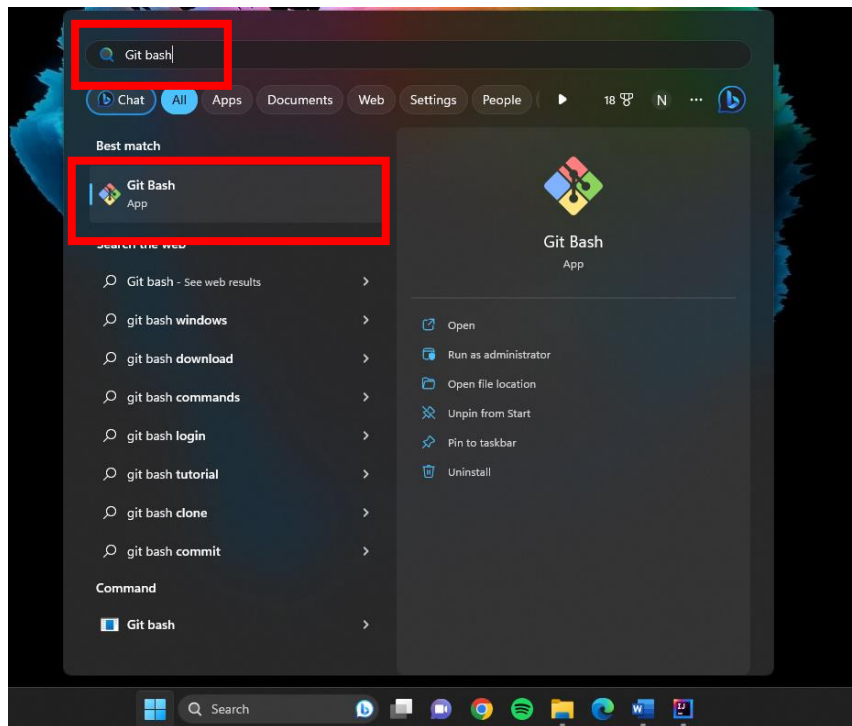
```
echo "# Bootcamp" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/nahuelpierini/Bootcamp.git
git push -u origin main
```

**...or push an existing repository from the command line**

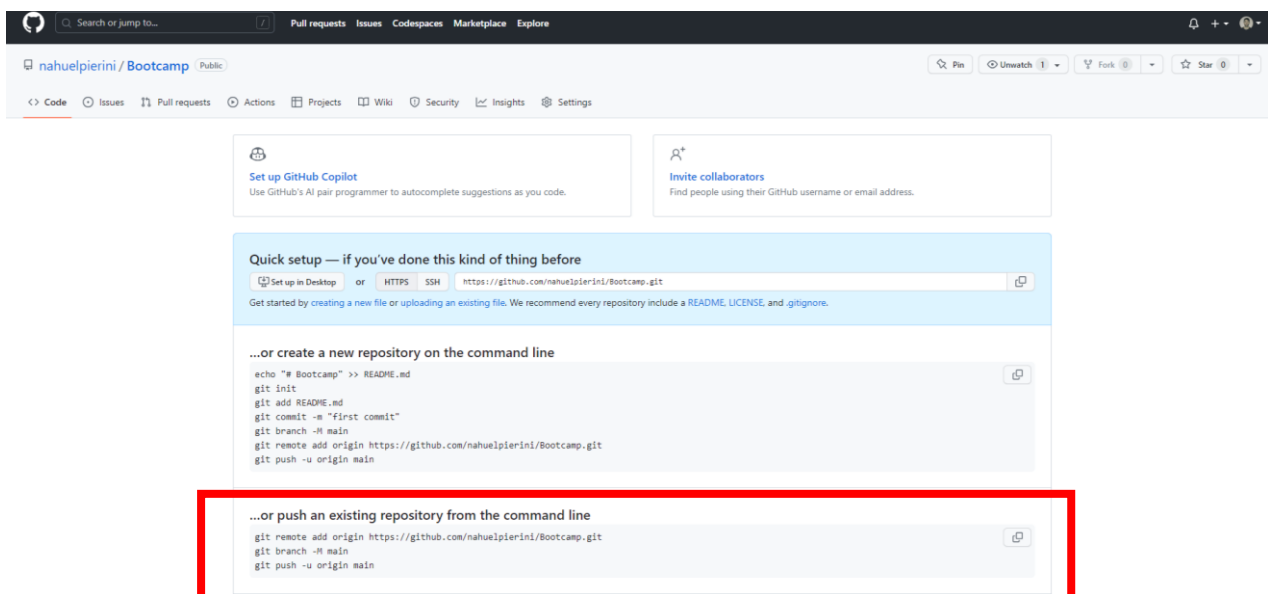
```
git remote add origin https://github.com/nahuelpierini/Bootcamp.git
git branch -M main
git push -u origin main
```

11. Si nos fijamos bien, en la imagen anterior esos comandos son ayuda de pasos a seguir para subir un proyecto a nuestro repositorio remoto de GitHub. *Una opción es crear un nuevo repositorio desde nuestra línea de comando (Git Bash). Pero como se ve, crea todo desde 0, git init... git add ... git commit... (otros pasos que no nos interesan por ahora). Como ya creamos nuestro repositorio local, vamos a seguir los pasos de la segunda opción es decir subir un repositorio existente desde la línea de comando.*

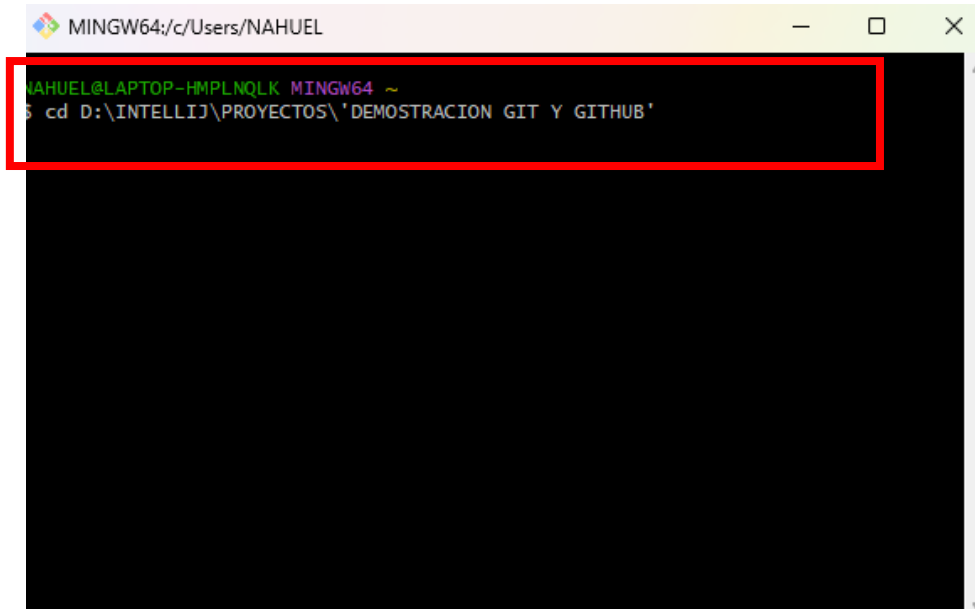
12. Volvemos a la terminal de Git Bash desde nuestra PC y la abrimos.



13. Seguimos los pasos que dice nuestro GitHub escribiéndolos en nuestra terminal.



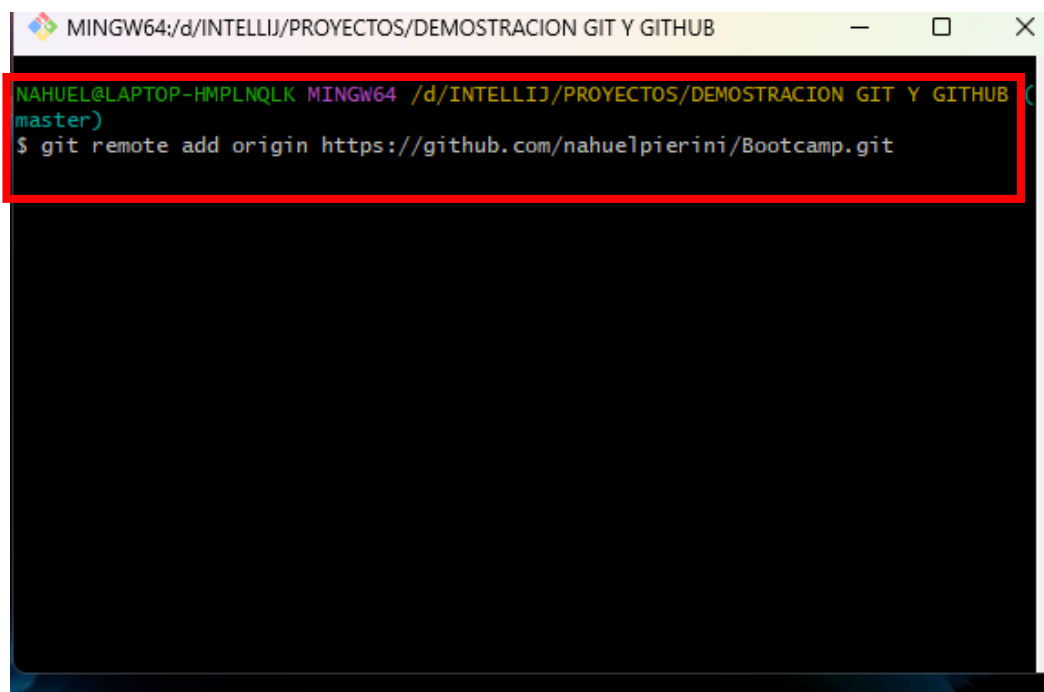
- a. Con la terminal abierta primero me tengo que encontrar en la ruta de mi carpeta. Ya lo hicimos en la explicación de Git. `cd D:\INTELLIJ\PROYECTOS\DEMOSTRACION GIT Y GITHUB` y le doy `enter`. Recordar que DEMOSTRACION GIT Y GITHUB TIENE ESPACIOS EN BLANCO PARA ELLO ESTA PARTE DE LA RUTA VA ENTRE COMILLAS SIMPLES. DE LO CONTRARIO NOS SALE UN ERROR:



```
MINGW64:/c/Users/NAHUEL
NAHUEL@LAPTOP-HMPLNQLK MINGW64 ~
$ cd D:\INTELLIJ\PROYECTOS\DEMOSTRACION GIT Y GITHUB
```

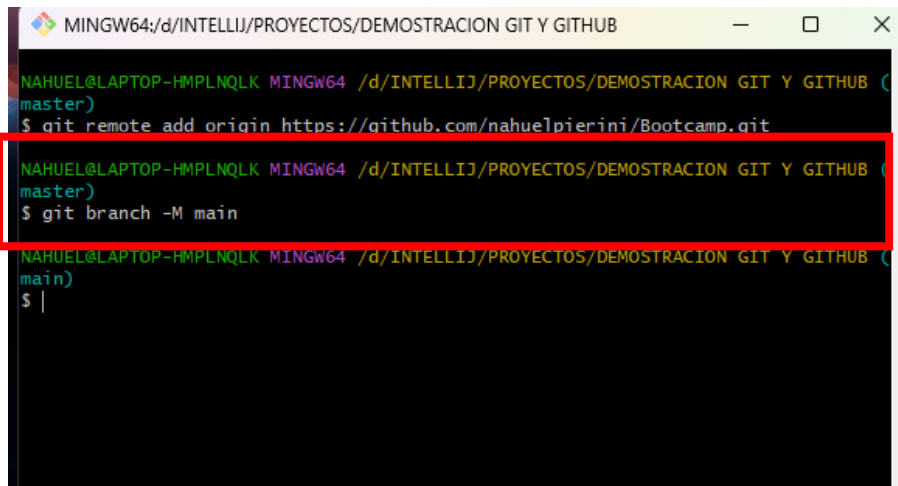
- b. Una vez ubicados en nuestro proyecto seguimos los pasos de GitHub:

- `git remote add origin`  
<https://github.com/nahuelpierini/Bootcamp.git> y le damos `enter`: Asignamos nuestro repositorio remoto con esa URL.



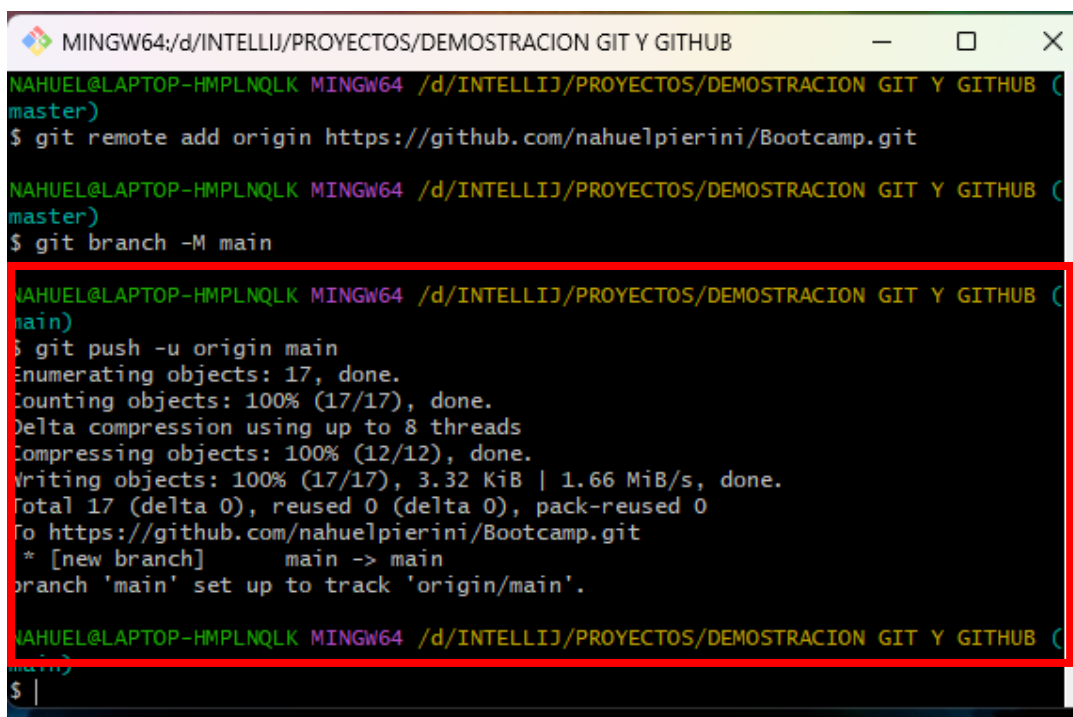
```
MINGW64:/d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB
NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (master)
$ git remote add origin https://github.com/nahuelpierini/Bootcamp.git
```

- `git branch -M main` y le damos enter: cambiamos el nombre de la rama master por main en nuestro repositorio.

A terminal window titled 'MINGW64:/d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB'. The prompt is 'NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (master)'. The user enters '\$ git remote add origin https://github.com/nahuelpierini/Bootcamp.git'. The prompt changes to '(master)'. The user enters '\$ git branch -M main'. The prompt changes to '(main)'. The user enters '\$ |'.

```
NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (
master)
$ git remote add origin https://github.com/nahuelpierini/Bootcamp.git
NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (
master)
$ git branch -M main
NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (
main)
$ |
```

- `git push -u origin main`: finalmente agregamos nuestro proyecto a GitHub.

A terminal window titled 'MINGW64:/d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB'. The prompt is 'NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (master)'. The user enters '\$ git remote add origin https://github.com/nahuelpierini/Bootcamp.git'. The prompt changes to '(master)'. The user enters '\$ git branch -M main'. The prompt changes to '(main)'. The user enters '\$ git push -u origin main'. The terminal shows the output of the push command: 'Enumerating objects: 17, done.', 'Counting objects: 100% (17/17), done.', 'Delta compression using up to 8 threads', 'Compressing objects: 100% (12/12), done.', 'Writing objects: 100% (17/17), 3.32 KiB | 1.66 MiB/s, done.', 'Total 17 (delta 0), reused 0 (delta 0), pack-reused 0', 'To https://github.com/nahuelpierini/Bootcamp.git', '\* [new branch] main -> main', 'branch 'main' set up to track 'origin/main''. The prompt changes to '(main)'. The user enters '\$ |'.

```
NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (
master)
$ git remote add origin https://github.com/nahuelpierini/Bootcamp.git
NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (
master)
$ git branch -M main
NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (
main)
$ git push -u origin main
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 8 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (17/17), 3.32 KiB | 1.66 MiB/s, done.
Total 17 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/nahuelpierini/Bootcamp.git
* [new branch] main -> main
branch 'main' set up to track 'origin/main'.
NAHUEL@LAPTOP-HMPLNQLK MINGW64 /d/INTELLIJ/PROYECTOS/DEMOSTRACION GIT Y GITHUB (
main)
$ |
```

14. Finalmente refrescamos la pagina de nuestro repositorio y veremos nuestros archivos guardados en GitHub.

Search or jump to...

Pull requests Issues Codespaces Marketplace Explore

nahuel pierini / Bootcamp Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

Go to file Add file + Create new file

**nahuel pierini** Creo clase 5 Actividad 1(CalcularEdadPerro) 4174bf 1 hour ago 1 commit

.idea	Creo clase 5 Actividad 1(CalcularEdadPerro)	1 hour ago
out/production/DEMOSTRACION GIT...	Creo clase 5 Actividad 1(CalcularEdadPerro)	1 hour ago
src/Clase_5	Creo clase 5 Actividad 1(CalcularEdadPerro)	1 hour ago
DEMOSTRACION GIT Y GITHUB.iml	Creo clase 5 Actividad 1(CalcularEdadPerro)	1 hour ago

Help people interested in this repository understand your project by adding a README. [Add a README](#)

**About**

No description, website, or topics provided.

Activity

0 stars

1 watching

0 forks

**Releases**

No releases published

[Create a new release](#)

**Packages**

No packages published

[Publish your first package](#)

**Languages**

Java 100.0%

**Suggested Workflows**

**Con esto tenemos nuestro nuevo repositorio en GitHub**