Práctica 1. Instalación de herramientas

Programación de Aplicaciones Telemáticas

Eva Calvo-Sotelo Piera

3ºB GITT+BA

1. Objetivo de la práctica

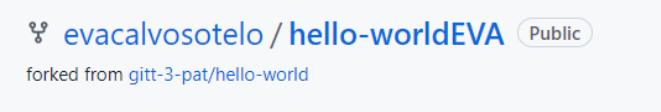
Tener unas nociones de como usar Github como plataforma de desarrollo, Git sistema de control de versiones de código fuente de Software.

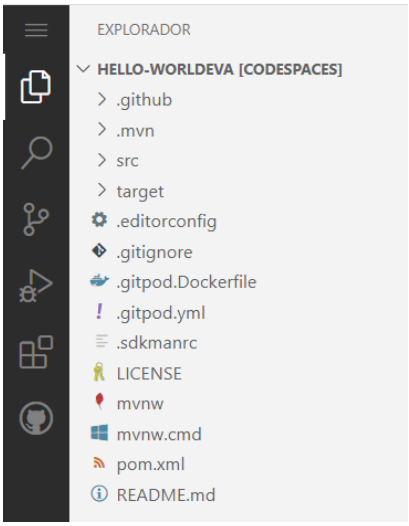
2. Desarrollo de la práctica

Uno de los requisitos antes de empezar es tener cuenta en Github, con mi propio usuario. Desde esa cuenta se hace un fork sobre el repositorio:

https://github.com/gitt-3-pat/hello-world

Creo un fork: Fork > New Fork > hello-worldEVA





Una vez hecho el fork, tendremos el repositorio en nuestra cuenta. Para poder empezar a editar los archivos y utilizar los comandos de git, vamos a clonar el repositorio en nuestro ordenador. Esto sirve para que podamos trabajar sobre dicho repositorio, sin necesidad de tener el permiso de su dueño.

Git clone

Para ello, tenemos que cambiarnos al directorio “tmp”, ya que estamos trabajando en local, y escribir el comando “git clone”.

```
@evacalvosotelo →/ $ cd tmp
@evacalvosotelo →/tmp $ git clone https://github.com/gitt-3-pat/hello-world
Cloning into 'hello-world'...
remote: Enumerating objects: 38, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 38 (delta 0), reused 0 (delta 0), pack-reused 34
Unpacking objects: 100% (38/38), 59.54 KiB | 1.45 MiB/s, done.
```

Una vez clonado, ya puedo abrir y trabajar el repositorio en mi entorno de desarrollo, a pesar de que no se me conceda acceso, gracias a que tengo una copia del repositorio en mi ordenador.

```
@evacalvosotelo →/tmp $ cd hello-world
@evacalvosotelo →/tmp/hello-world (main) $ ls
LICENSE  mvnw  mvnw.cmd  pom.xml  README.md  src
@evacalvosotelo →/tmp/hello-world (main) $ cd ..
@evacalvosotelo →/tmp $ cd ..
```

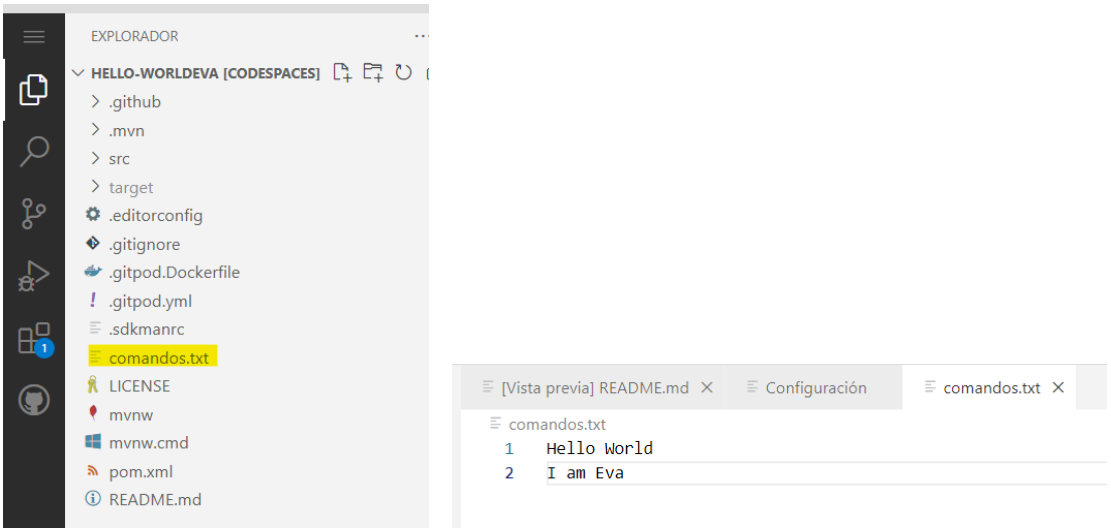
Git status:

Sirve para comprobar la rama en la que estoy situada y ver los cambios, si los hay, de mi repositorio. Primero, debemos volver a cambiar el directroio de vuelta a workspaces, que es donde está nuestro fork.

```
@evacalvosotelo →/ $ cd workspaces/
@evacalvosotelo →/workspaces $ cd hello-worldEVA/
@evacalvosotelo →/workspaces/hello-worldEVA (main) $ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Como se ve en la imagen de arriba, nos encontramos en la rama “main” y no hemos hecho cambios a ningún archivo. Como por ahora no ha pasado nada, creo un nuevo documento .txt, y vuelvo a hacer git status.



Ahora si se ha detectado el cambio que hemos hecho: la creación de un documento.

```
@evacalvosotelo →/workspaces/hello-worldEVA (main) $ 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)
        comandos.txt

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

Git add, git commit, git push

El comando “git add” sirve para seleccionar los archivos que hemos modificado y cuyos cambios queremos que se guarden en la nube. Con el comando “git add .” añadimos TODOS los archivos que han sufrido cambios, pero si sólo queremos añadir un archivo concreto, como por ejemplo, el documento comandos.txt que acabamos de crear, escribimos el siguiente comando:

$ git add comandos.txt

Ya seleccionado el archivo que queremos guardar en nuestro repo, hay que hacer un “git commit”, ya que este comando crea una instantánea de los cambios que se han realizado y la guarda en el directorio git.

Para terminar hacemos “git push”, que sirve para poner la dirección de rama a la que queremos enviar los cambios guardados en la instantánea. Si no le pasas ningún argumento a “git push”, se entiende que lo vas a enviar a la rama principal y si quieres hacer push a otra rama, debes utilizar “git push origin ramal”.

```
@evacalvosotelo ➜/workspaces/hello-worldEVA (main X) $ git add comandos.txt
@evacalvosotelo ➜/workspaces/hello-worldEVA (main) $ git commit -m "Hola Mundo Soy Eva"
[main 0c18c44] Hola Mundo Soy Eva
 1 file changed, 2 insertions(+), 1 deletion(-)
@evacalvosotelo ➜/workspaces/hello-worldEVA (main) $ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 306 bytes | 306.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/evacalvosotelo/hello-worldEVA
   4eb3b6d..0c18c44  main -> main
```

Git heckout

Este comando sirve para crear una rama (branch), además de ayudarnos a navegar por ellas. Para crear una rama utilizamos el comando “git checkout -b <Branch_Name>”.

Este comando nos mueve automáticamente a la rama creada. Para movernos de una rama a otra, utilizamos simplemente git checkout.

```
@evacalvosotelo ➜/workspaces/hello-worldEVA (main) $ git checkout -b branch2
Switched to a new branch 'branch2'
@evacalvosotelo ➜/workspaces/hello-worldEVA (branch2) $ git checkout
branch2       FETCH_HEAD    HEAD          main          origin/HEAD   origin/main
```

Este último comando, con la ayuda del tabulador, nos muestra todas las ramas posibles a las que nos podemos mover.

3. Instalación de herramientas

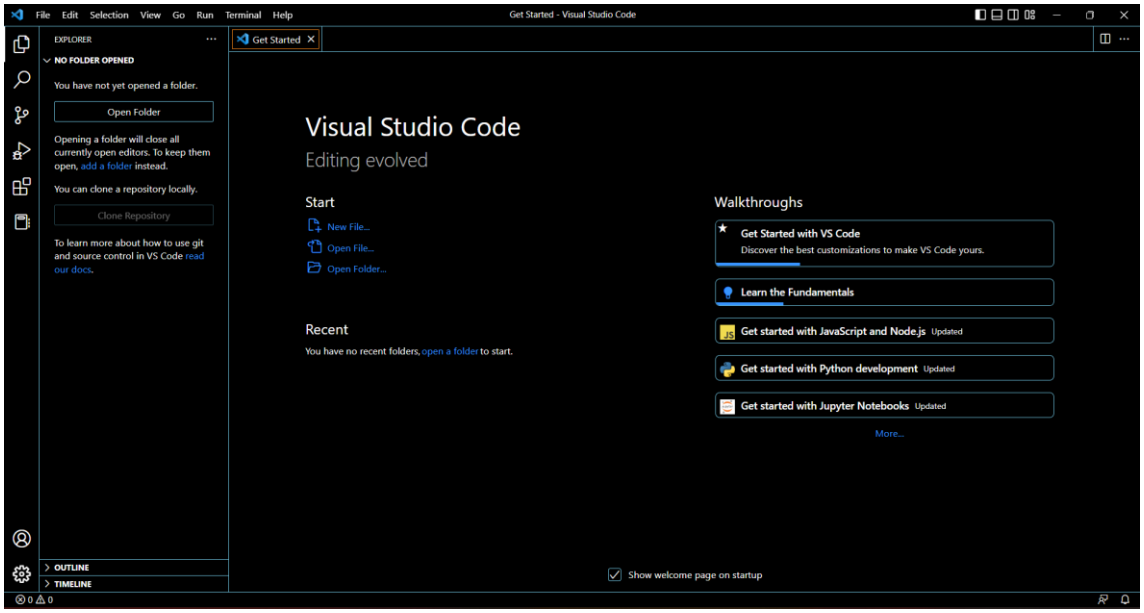
Java

```
C:\Users\evaca>java -version
java version "15.0.2" 2021-01-19
Java(TM) SE Runtime Environment (build 15.0.2+7-27)
Java HotSpot(TM) 64-Bit Server VM (build 15.0.2+7-27, mixed mode, sharing)
```

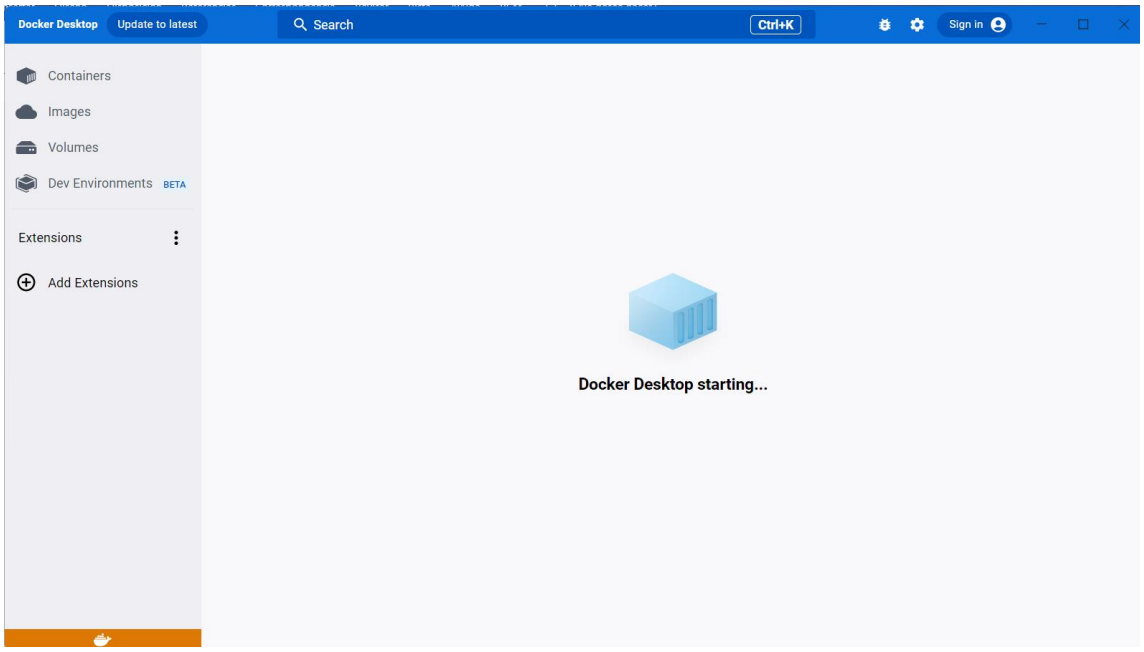
Maven

```
C:\Users\evaca>mvn --version
Apache Maven 3.8.7 (b89d5959fcde851dcb1c8946a785a163f14e1e29)
Maven home: C:\Program Files\apache-maven-3.8.7
Java version: 15.0.2, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-15.0.2
Default locale: es_ES, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

VisualCode



Docker Desktop


Docker Desktop
Update to latest
Search
Ctrl+K
Sign in
Containers
Images
Volumes
Dev Environments
BETA
Extensions
Add Extensions
Docker Desktop starting...