

Programación de Aplicaciones Telemáticas

Practica 1: Instalación de Herramientas



Rim TYOUSS

3°B GITT+BA

Tabla de contenido

Objetivos de la practica.....	3
GitHub	3
Prerrequisitos	3
Desarrollo de la práctica.....	3
• git clone.....	3
• git status	4
• git add.....	5
• git commit -m.....	5
• git push.....	6
• Git checkout -b feature/1	6
• Git checkout main	8
Entorno de desarrollo JVM	8

Objetivos de la practica

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

GitHub

GitHub es la plataforma de desarrollo de software más grande del mundo. Es una plataforma colaborativa que permite a los desarrolladores guardar y compartir código, controlar versiones y colaborar en proyectos.

Prerrequisitos

El alumno tendrá que tener cuenta en GitHub y con su cuenta de usuario, hará un fork sobre el repositorio: <https://github.com/gitt-3-pat/hello-world>



Hacemos un fork porque nos permite experimentar libremente con un repositorio de otro usuario con los cambios que queramos, sin afectar el proyecto original. Haciendo un fork, creamos una copia de un repositorio en nuestra propia cuenta, con el fin de poder crear una versión personalizada de algo que ya existe, sin tener los permisos de escritura en el repositorio original. Además, se pueden enviar solicitudes de “pull” al usuario del repositorio original para que los cambios que hayamos hecho se integren también en el suyo.

Desarrollo de la práctica

Tenemos que probar diferentes comandos, en el entorno de desarrollo en línea (IDE) de GitHub “Codespaces”.

- **git clone**

El comando “git clone” nos permite descargar un proyecto, un repositorio completo desde un entorno remoto a su propio ordenador (localmente). Al hacer un clone, todos los cambios hechos en el repositorio original se verán también en la copia local. A cambio del fork que se guarda en la nube, el clone se guarda localmente. Además, si

el clone se ejecuta sin hacer un fork, y se quiere subir a GitHub, estará denegado si no tenemos los permisos.

Hago un clone del repositorio “PAT” en la carpeta temporal “tmp”.

```
● @rimtyouss → /tmp $ git clone https://github.com/rimtyouss/PAT
Cloning into 'PAT'...
remote: Enumerating objects: 40, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 40 (delta 1), reused 2 (delta 1), pack-reused 34
Unpacking objects: 100% (40/40), 59.75 KiB | 1.87 MiB/s, done.
```

Podemos verificar si esta correctamente descargado haciendo un “ls”:

```
● @rimtyouss → /tmp $ ls
buildscriptGenerator  hsperfdata_codespace  vscode-git-a496dcffe5.sock
codespaces.log        SA                     vscode-ipc-b26a5464-20a9-4b2a-9fcf-da23603ecf5f.sock
dockerd.log           _ sshd.log             vscode-ipc-e806b1b9-8d23-4375-bc80-0a3bca704755.sock
```

Efectivamente, podemos ver que el repositorio “PAT” esta descargado.

- **git status**

Este comando sirve para ver el estado actual de un repositorio y al ejecutarlo, nos mostrara información sobre los archivos modificados, nuevos o eliminados en comparación con el ultimo “commit”, además de cualquier modificación que no está guardada todavía.

Si ahora, añado un nuevo fichero a mi repositorio y vuelvo a ejecutar el comando, me aparece este mensaje:

```
● @rimtyouss → /workspaces/PAT (main X) $ 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)
  hello-world/
```

Observamos que nos muestra el mensaje “*use git add <file>...*” to include in what will be committed”.

- **git add**

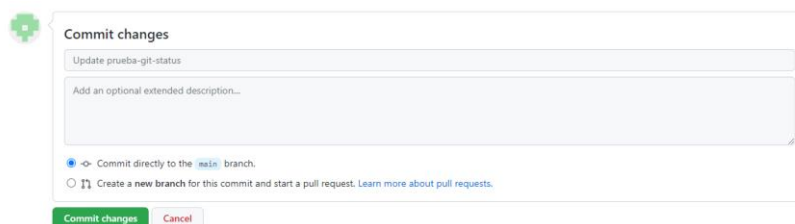
Es un comando para agregar archivos o cambios al “staging area” del repositorio. Como se puede ver la captura anterior del ejemplo git status, al final del codigo podemos ver la frase “changes to be committed : ...” : cuando hacemos cambios en el repositorio, estos cambios todavia no son parte de un commit, ya que deben ser agregados al staging area mediante el comando “git add”.

Ahora podemos ver que “new file : hello-world” esta en verde, significando que ya se puede guardar.

```
● @rimtyouss → /workspaces/PAT (main) $ git add .
● @rimtyouss → /workspaces/PAT (main) $ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   hello-world
```

Se podria hacer desde GitHub directamente, para cuando cambiamos un codigo localmente y que se suba el cambio al repositorio, haciendo un commit :



- **git commit -m**

El comando “commit” se utiliza para confirmar los cambios añadidos previamente al staging area, en nuestro caso, del repositorio “PAT”. Con el mensaje añadido, podemos dejar una descripcion breve del cambio que hemos realizado y que nos facilite el trabajo en un futuro.

```
● @rimtyouss → /workspaces/PAT (main) $ git commit -m "HOLA RIM"
[main 2afa444] HOLA RIM
 1 file changed, 1 insertion(+)
 create mode 160000 PAT
○ @rimtyouss → /workspaces/PAT (main) $ █
```

Al ejecutarlo, podemos ver que nos dice que un fichero ha sido modificado.

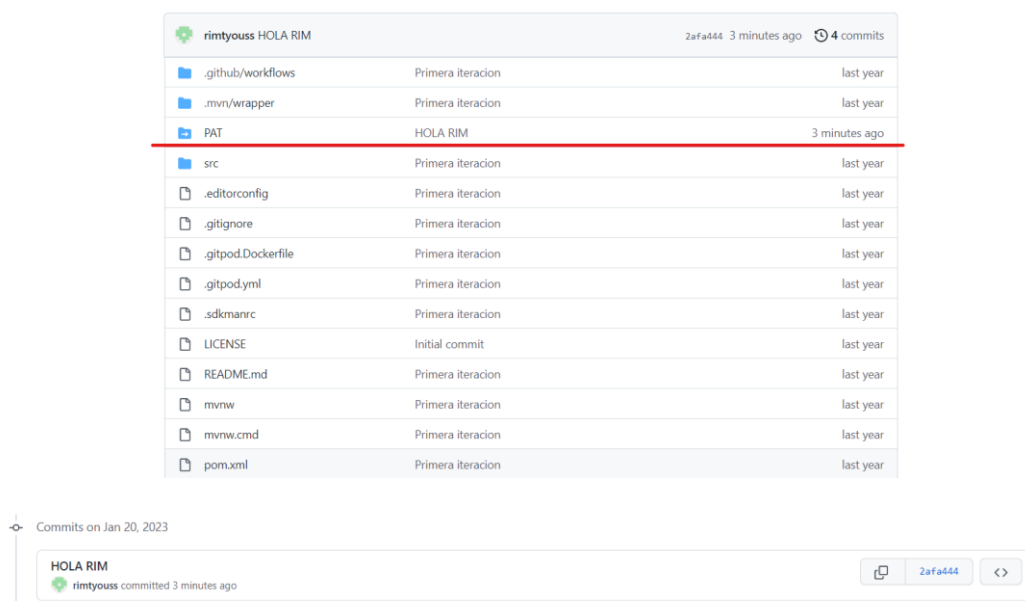
Es importante saber que, antes de ejecutar el comando, primero añadir los archivos a seguir con el commit, con el comando “git add” visto anteriormente.

- **git push**

Después de realizar un commit, los cambios están registrados en la copia del repositorio local, y no en el repositorio en línea. Para subir los cambios al repositorio en línea, se utiliza el comando “git push”.

```
● @rimtyouss → /workspaces/PAT (main) $ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 274 bytes | 274.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/rimtyouss/PAT
48fe276..2afa444 main -> main
```

Efectivamente, ahora el commit es visible en nuestro repositorio en línea:



The screenshot shows the GitHub interface for the repository 'rimtyouss HOLA RIM'. The top bar indicates the commit hash '2afa444' was made '3 minutes ago' and shows '4 commits'. Below this is a table of files in the repository:

File	Commit Message	Commit Time
.github/workflows	Primera iteracion	last year
.mvnw/wrapper	Primera iteracion	last year
PAT	HOLA RIM	3 minutes ago
src	Primera iteracion	last year
.editorconfig	Primera iteracion	last year
.gitignore	Primera iteracion	last year
.gitpod.Dockerfile	Primera iteracion	last year
.gitpod.yml	Primera iteracion	last year
.sdkmanrc	Primera iteracion	last year
LICENSE	Initial commit	last year
README.md	Primera iteracion	last year
mvnw	Primera iteracion	last year
mvnw.cmd	Primera iteracion	last year
pom.xml	Primera iteracion	last year

Below the file list, the 'Commits on Jan 20, 2023' section shows a single commit:

- HOLA RIM**
rimtyouss committed 3 minutes ago

Buttons for copying the commit hash '2afa444' and viewing the commit details are visible.

- **Git checkout -b feature/1**

Este comando nos permite crear una nueva rama (Branch) llamada “feature/1” y así cambiar el directorio donde estamos trabajando a esa nueva rama. Ahora ya no

estaríamos en el (main) pero en la nueva rama creada, pudiendo hacer cambios/commits sin afectar la rama principal (el main).

```
● @rimtyouss →/workspaces/PAT (main) $ git checkout -b feature/1
  Switched to a new branch 'feature/1'
○ @rimtyouss →/workspaces/PAT (feature/1) $
```

Al ejecutar el comando “git status”, podemos ver que estamos en la nueva rama:

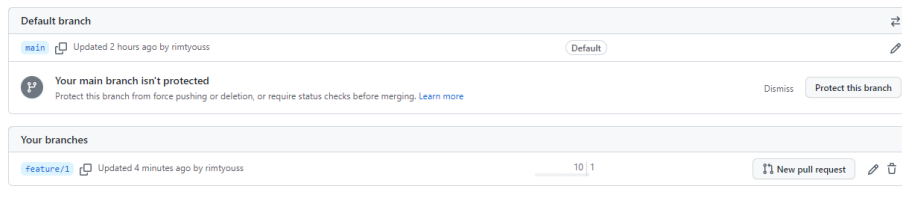
```
● @rimtyouss →/workspaces/PAT (feature/1) $ git status
On branch feature/1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   hello-world
```

De nuevo, para subir esta nueva rama al repositorio en línea, habría que hacer un commit y un push de la rama “feature/1”. Creo un fichero en esa nueva rama y lo subo, añadiendo el fichero al staging área del repositorio, confirmando los cambios y finalmente enviar los cambios al repositorio original:

```
● @rimtyouss →/workspaces/PAT (feature/1 X) $ touch prueba.md
● @rimtyouss →/workspaces/PAT (feature/1 X) $ git add prueba.md
● @rimtyouss →/workspaces/PAT (feature/1) $ git commit -m "prueba nueva rama"
[feature/1 aa4a45b] prueba nueva rama
 2 files changed, 1 insertion(+)
 create mode 160000 hello-world
 create mode 100644 prueba.md

● @rimtyouss →/workspaces/PAT (feature/1) $ git push origin feature/1
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 2 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 588 bytes | 588.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
remote:
remote: Create a pull request for 'feature/1' on GitHub by visiting:
remote:   https://github.com/rimtyouss/PAT/pull/new/feature/1
remote:
To https://github.com/rimtyouss/PAT
 * [new branch]   feature/1 -> feature/1
```

Ahora podemos comprobar que la nueva rama ya existe en nuestro repositorio original, además del fichero creado “prueba.md”



- **Git checkout main**

Finalmente, como estábamos en una rama secundaria, para volver a la rama principal (main) desde la rama (feature/1), hay que ejecutar el comando “git checkout main”.

```
● @rimtyouss → /workspaces/PAT (feature/1) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Entorno de desarrollo JVM

Evidencias de la instalación de:

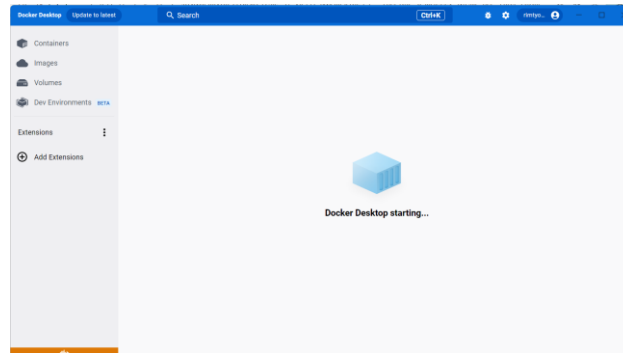
- Java 17

```
C:\Users\tyous>java --version
java 17.0.6 2023-01-17 LTS
Java(TM) SE Runtime Environment (build 17.0.6+9-LTS-190)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.6+9-LTS-190, mixed mode, sharing)
```

- Maven


```
C:\Users\tyous>mvn --version
Apache Maven 3.8.7 (b89d5959fcde851dcb1c8946a785a163f14e1e29)
Maven home: C:\Program Files\apache-maven-3.8.7-bin\apache-maven-3.8.7
Java version: 17.0.6, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-17
Default locale: es_ES, platform encoding: Cp1252
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"
```

- Docker



- Visual Studio Code

