Práctica 1

1. Ejecución y explicación de comandos

Git clone

La principal función de git clone consiste en copiar un repositorio existente de Git a un nuevo directorio local. Lo que está haciendo en realidad es “clonar” o descargar todo el historial de versiones, ramas y archivos del repositorio a nuestra máquina local. Cabe destacar, que después de clonar, Git configura automáticamente un enlace remoto llamado “origin” que apunta al repositorio remoto desde cual hemos clonado.

```
@alvaromoraga ➜ /workspaces/p1 (main) $ git clone https://github.com/gitt-3-pat/p1
Cloning into 'p1'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 5
Receiving objects: 100% (6/6), done.
```

Git status

El comando git status se utiliza principalmente para mostrar el estado actual de un repositorio. Git status proporciona información sobre cambios que por ejemplo no han sido confirmados, archivos que han sido modificados o nuevos, archivos eliminados y también diferentes aspectos del estado actual de la rama en la que estamos trabajando. Por lo tanto, este comando es útil para obtener una visión general del estado actual de nuestros repositorio local y nos puede ayudar a entender qué cambios hemos realizado y qué acciones debemos realizar para modificar nuestro repositorio.

```
@alvaromoraga ➜ /workspaces/p1 (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)
        p1/

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

Git add

El comando git add . se utiliza para añadir todos los archivos modificados y nuevos en el directorio de trabajo al área de preparación. Esta operación prepara los cambios para ser incluidos en el próximo commit.

```
@alvaromoraga ➜ /workspaces/p1 (main) $ git add .
warning: adding embedded git repository: p1
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> p1
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached p1
hint:
hint: See "git help submodule" for more information.
```

Git commit -m

Este comando se utiliza para registrar los cambios realizados en nuestro área de preparación en la historia del repositorio Git. Este mensaje debería de ser informativo y expresar el propósito del commit. En mi caso, como simplemente era una prueba, he puesto el mensaje “HOLA SOY ÁLVARO”. Este comando podría ser útil ya que podrían facilitar la comprensión de los cambios y la colaboración entre los miembros del equipo si se trabaja en grupo.

```
@alvaromoraga ➜ /workspaces/p1 (main) $ git commit -m "HOLA SOY ALVARO"
[main a9641f8] HOLA SOY ALVARO
 1 file changed, 1 insertion(+)
 create mode 160000 p1
```

Git push

Git push sirve para enviar los commits locales a un repositorio remoto. Por lo tanto, una vez hayas realizado cambios puedes usar git push para subir esos cambios al repositorio remoto.

```
@alvaromoraga ➜ /workspaces/p1 (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), 289 bytes | 289.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/alvaromoraga/p1
   07720b5..a9641f8  main -> main
```

Git checkout -b

La función de este comando es sencilla; básicamente se crea una rama y te cambias a esa rama en un solo paso. Podría decirse que es una forma abreviada y conveniente de realizar dos operaciones: la creación de una nueva rama y el cambio a esa rama.

```
@alvaromoraga ➜ /workspaces/p1 (main) $ git checkout -b feature/1
Switched to a new branch 'feature/1'
```

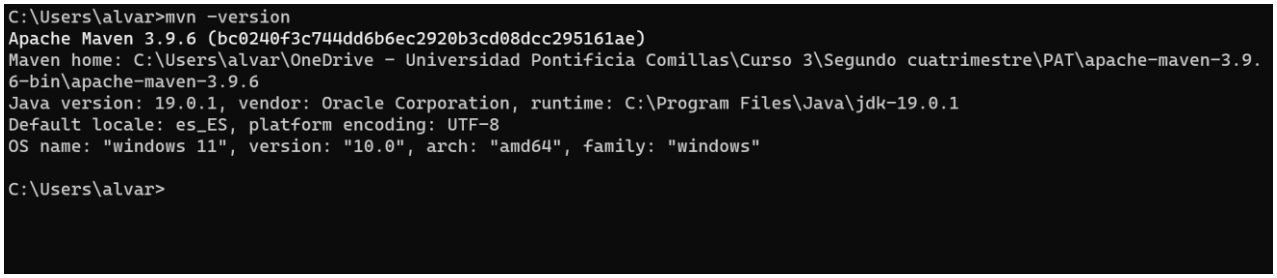
Git checkout main

Este comando se utiliza para cambiar a la rama llamada main. Al cambiar a una rama específica, nuestro directorio de trabajo y la cabecera de la rama actual se actualizan para reflejar el estado de esa rama. Por ello, es de utilidad cuando se desea volver a la rama principal de nuestro proyecto después de haber trabajado en otra rama secundaria.

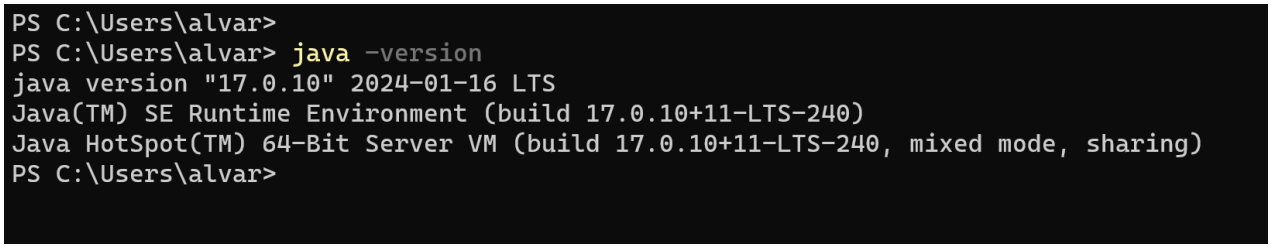
```
@alvaromoraga ➜ /workspaces/p1 (feature/1) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

2. Entorno de desarrollo Java

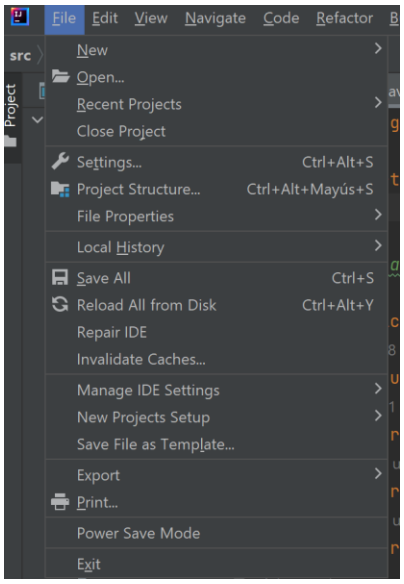
En este apartado, hemos tenido que descargar diferentes versiones y entornos. A continuación, se verifica que todo se ha descargado e implantado correctamente:



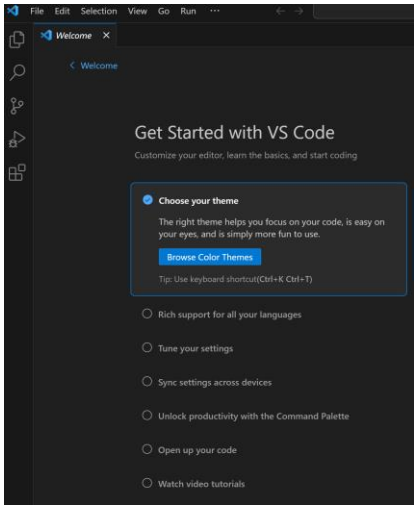
Descarga Maven



Descarga versión 17 Java



Descarga IntelliJ



Descarga VScode

Álvaro Moraga