COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI ICADE CIHS

Programación de Aplicaciones Telemáticas

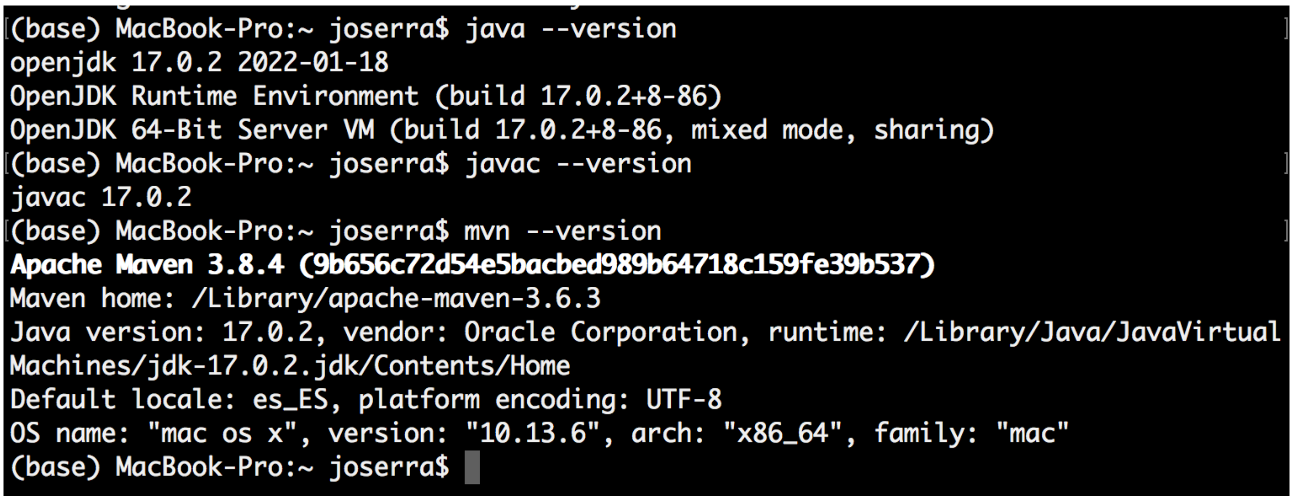
Práctica 1: Uso de git

José Ramón Porro Nieves
201800035 3ºA GITT

27 de enero del 2022

Entorno de desarrollo

En primer lugar, se instalaron todas las herramientas necesarias para el desarrollo de software. En este caso Java JDK 17, Maven y IntelliJ.



```
(base) MacBook-Pro:~ joserra$ java --version
openjdk 17.0.2 2022-01-18
OpenJDK Runtime Environment (build 17.0.2+8-86)
OpenJDK 64-Bit Server VM (build 17.0.2+8-86, mixed mode, sharing)
(base) MacBook-Pro:~ joserra$ javac --version
javac 17.0.2
(base) MacBook-Pro:~ joserra$ mvn --version
Apache Maven 3.8.4 (9b656c72d54e5bacbed989b64718c159fe39b537)
Maven home: /Library/apache-maven-3.6.3
Java version: 17.0.2, vendor: Oracle Corporation, runtime: /Library/Java/JavaVirtual
Machines/jdk-17.0.2.jdk/Contents/Home
Default locale: es_ES, platform encoding: UTF-8
OS name: "mac os x", version: "10.13.6", arch: "x86_64", family: "mac"
(base) MacBook-Pro:~ joserra$
```

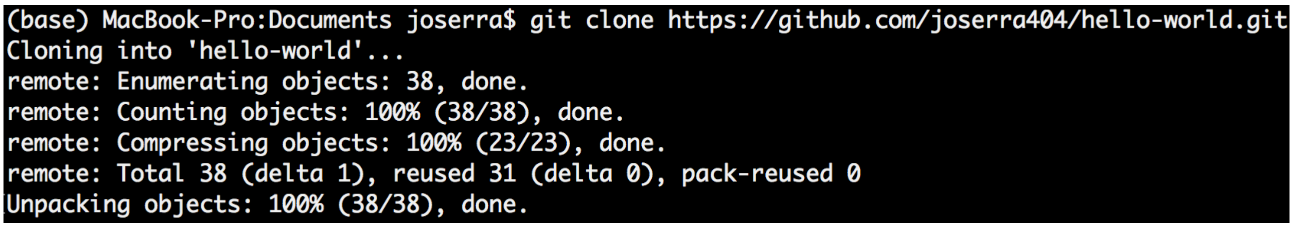
En esta captura se puede comprobar que todas las herramientas requeridas están instaladas correctamente.

Uso de git

Git es el software por excelencia de control de versiones y es ampliamente utilizado en el desarrollo de software.
En primer lugar, se ha hecho un “fork” a partir del repositorio hello-world del curso. Al hacer un “fork”, se crea una copia exacta del repositorio original en el propio GitHub del usuario, de forma que esta forma se pueden realizar modificaciones sobre el repositorio sin problemas.

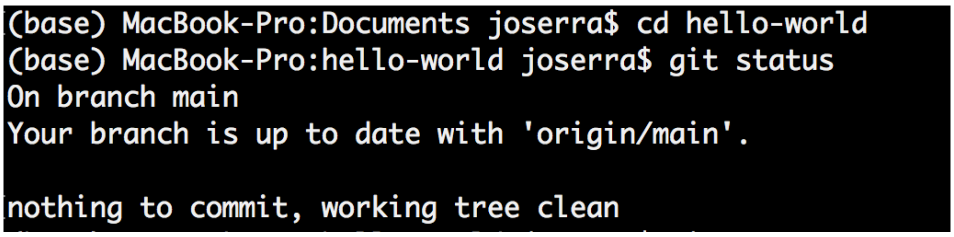
A continuación, se va a detallar el uso de los comandos principales de git.
El repositorio se encuentra en este enlace: https://github.com/joserra20/hello-world

`git clone`



```
(base) MacBook-Pro:Documents joserra$ git clone https://github.com/joserra404/hello-world.git
Cloning into 'hello-world'...
remote: Enumerating objects: 38, done.
remote: Counting objects: 100% (38/38), done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 38 (delta 1), reused 31 (delta 0), pack-reused 0
Unpacking objects: 100% (38/38), done.
```

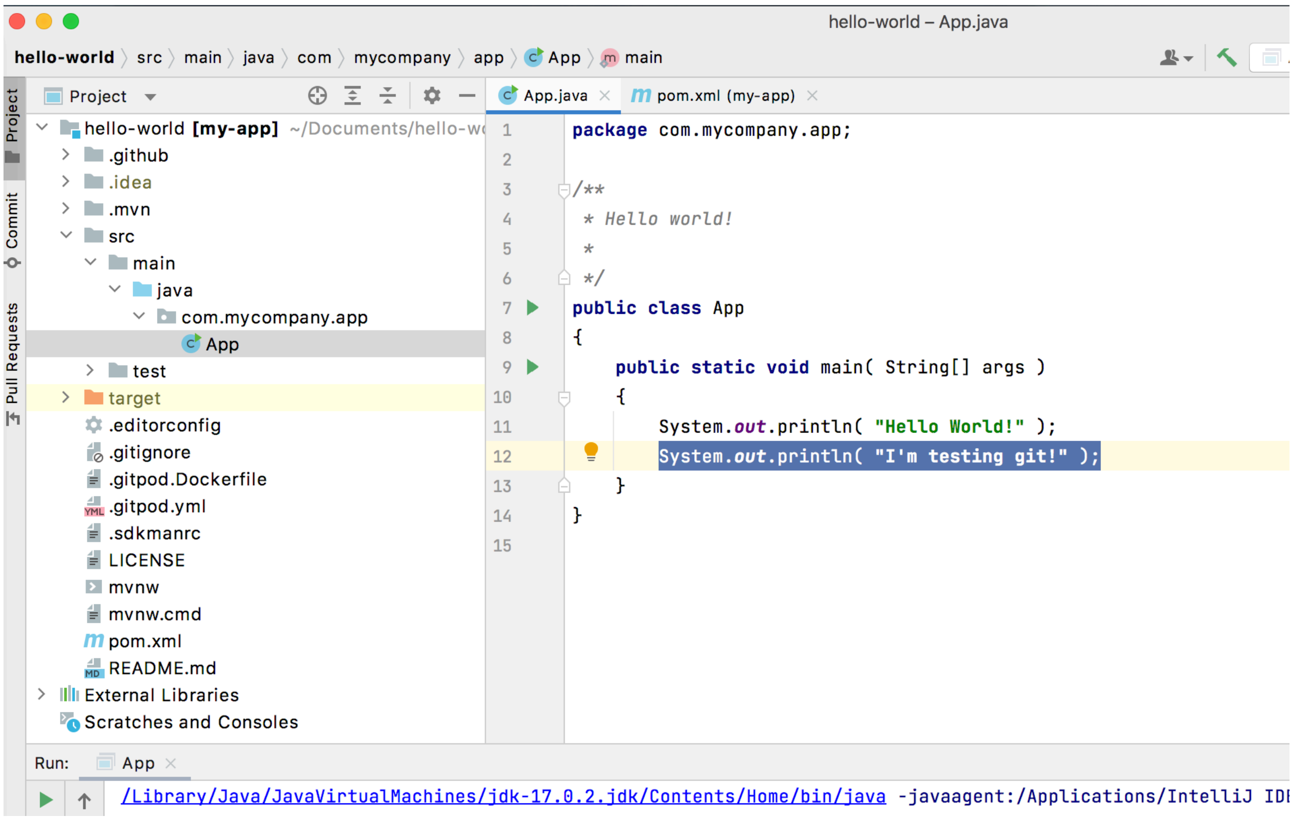
Clonar el repositorio objetivo en un nuevo directorio con el nombre del repositorio. Además crea branches remotas para controlar cada branch clonada del repositorio (visible con git branch –remotes), y crea una branch inicial a la que se realiza un ”fork” desde la branch activa del repositorio clonado.



```
(base) MacBook-Pro:Documents joserra$ cd hello-world
(base) MacBook-Pro:hello-world joserra$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

git status

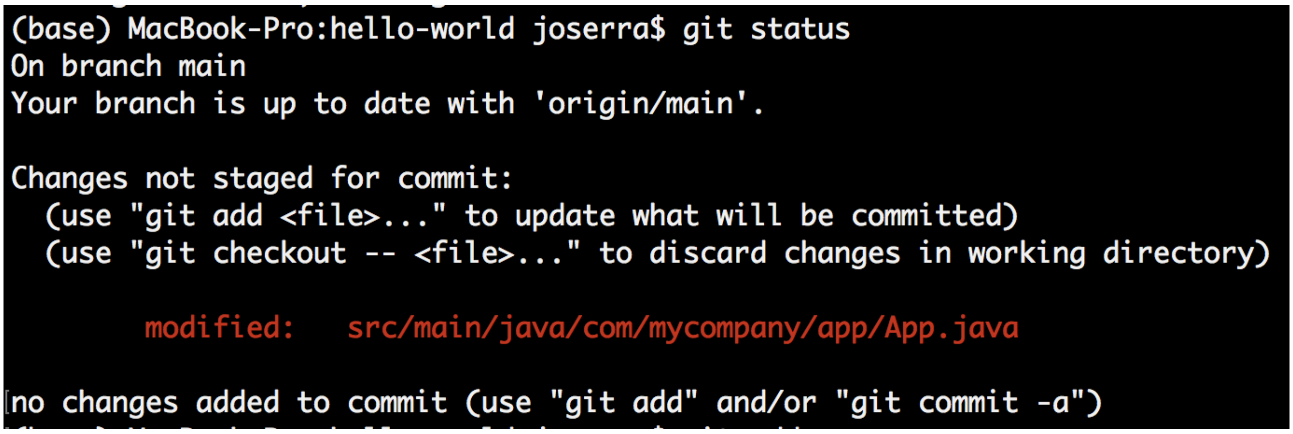
Muestra directorios y archivos donde se hayan encontrado modificaciones entre el repositorio local y el commit activo y directorios donde haya modificaciones entre el árbol y los archivos.
Se puede observar en la captura que se ha añadido una línea de código al proyecto en IntelliJ. Para reflejar este cambio en el repositorio de GitHub se ejecutan los siguientes comandos.



```java
package com.mycompany.app;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
        System.out.println( "I'm testing git!" );
    }
}
```

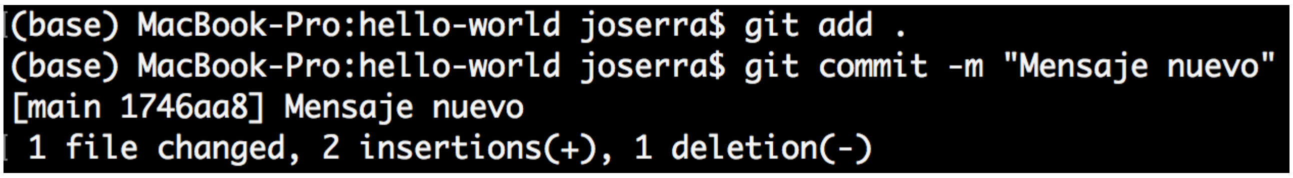
```
/Library/Java/JavaVirtualMachines/jdk-17.0.2.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDE
```

Si se vuelve a ejecutar `git status` tras la modificación se muestra lo siguiente.



```
(base) MacBook-Pro:hello-world joserra$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   src/main/java/com/mycompany/app/App.java

no changes added to commit (use "git add" and/or "git commit -a")
```

Se está indicando que se ha modificado el archivo App.java y que no ha sido reflejado en el repositorio. Para añadir este cambio emplea el comando `gitt add .`



```
(base) MacBook-Pro:hello-world joserra$ git add .
(base) MacBook-Pro:hello-world joserra$ git commit -m "Mensaje nuevo"
[main 1746aa8] Mensaje nuevo
 1 file changed, 2 insertions(+), 1 deletion(-)
```

`git add`

El comando `git add` actualiza el index usando el contenido del árbol de trabajo para preparar los contenidos elegidos para el próximo commit.

`git commit -m "Mensaje"`

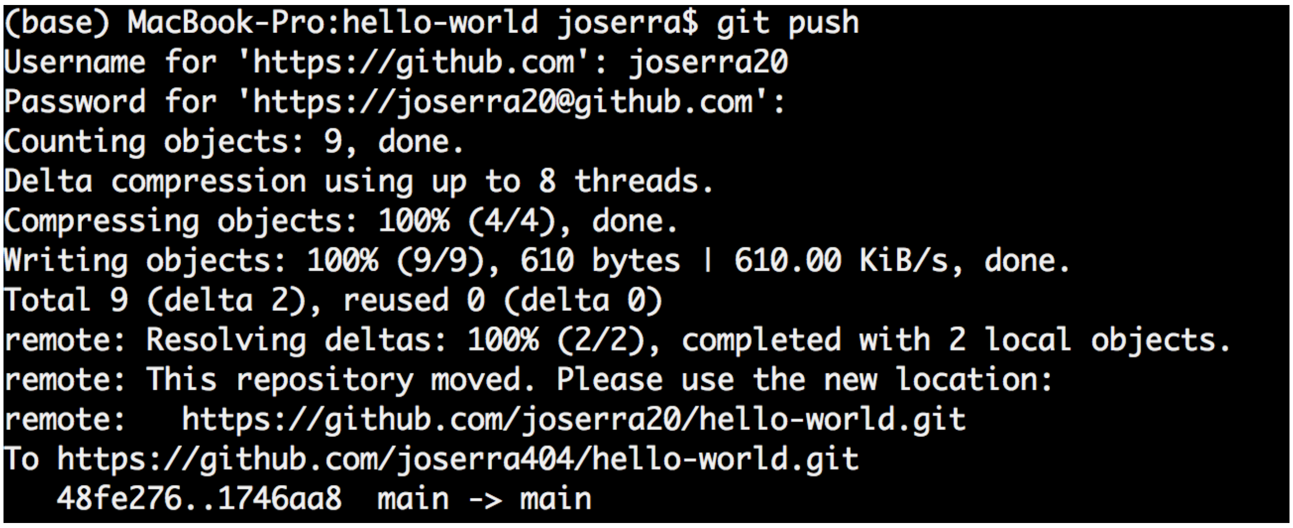
Crea una confirmación con un mensaje de confirmación usado. Por defecto, si se omite el flag -m "Mensaje" se muestra un editor de texto y se solicita que introduzcas un mensaje. Siendo más cómodo y rápido emplear -m "Mensaje".

`git push`

Se usa para cargar contenido del repositorio local a un repositorio remoto, es necesario haber realizado previamente el commit. La sintaxis es la siguiente:

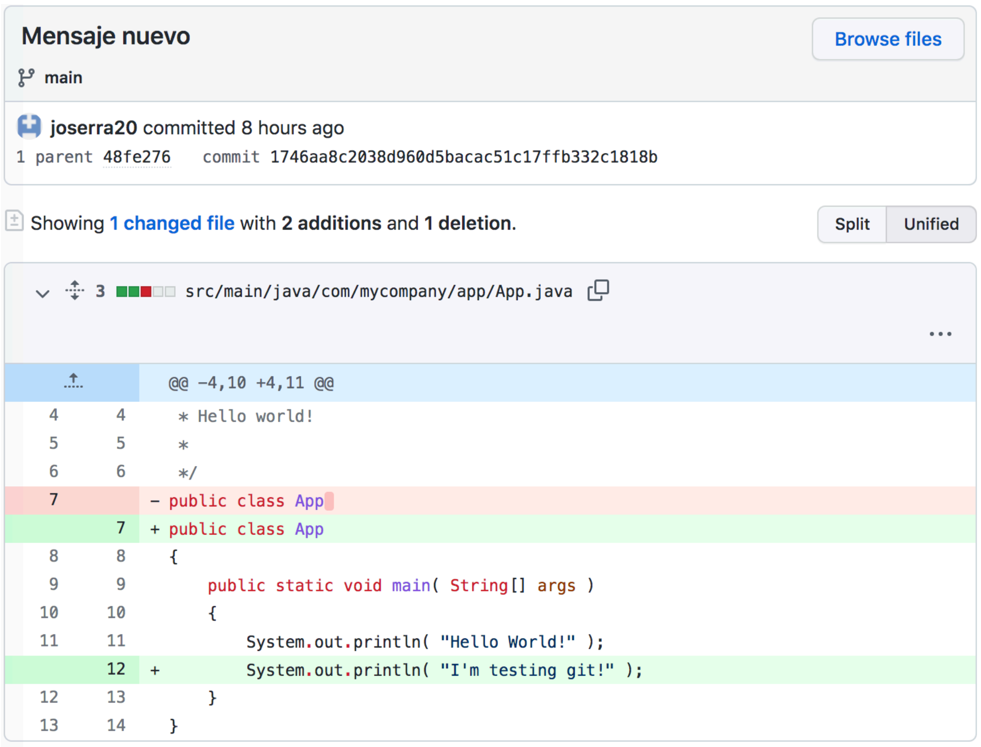
```
git push <nombre del repositorio> <nombre de la rama>
```

Si no se especifica ninguna rama, se cargará en la rama por defecto.



```
(base) MacBook-Pro:hello-world joserra$ git push
Username for 'https://github.com': joserra20
Password for 'https://joserra20@github.com':
Counting objects: 9, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (9/9), 610 bytes | 610.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote: This repository moved. Please use the new location:
remote:   https://github.com/joserra20/hello-world.git
To https://github.com/joserra404/hello-world.git
   48fe276..1746aa8  main -> main
```

Si nos dirigimos a GitHub, se puede observar con claridad los cambios del commit realizado.



Mensaje nuevo

main

joserra20 committed 8 hours ago

1 parent 48fe276   commit 1746aa8c2038d960d5bacac51c17ffb332c1818b

Showing 1 changed file with 2 additions and 1 deletion.

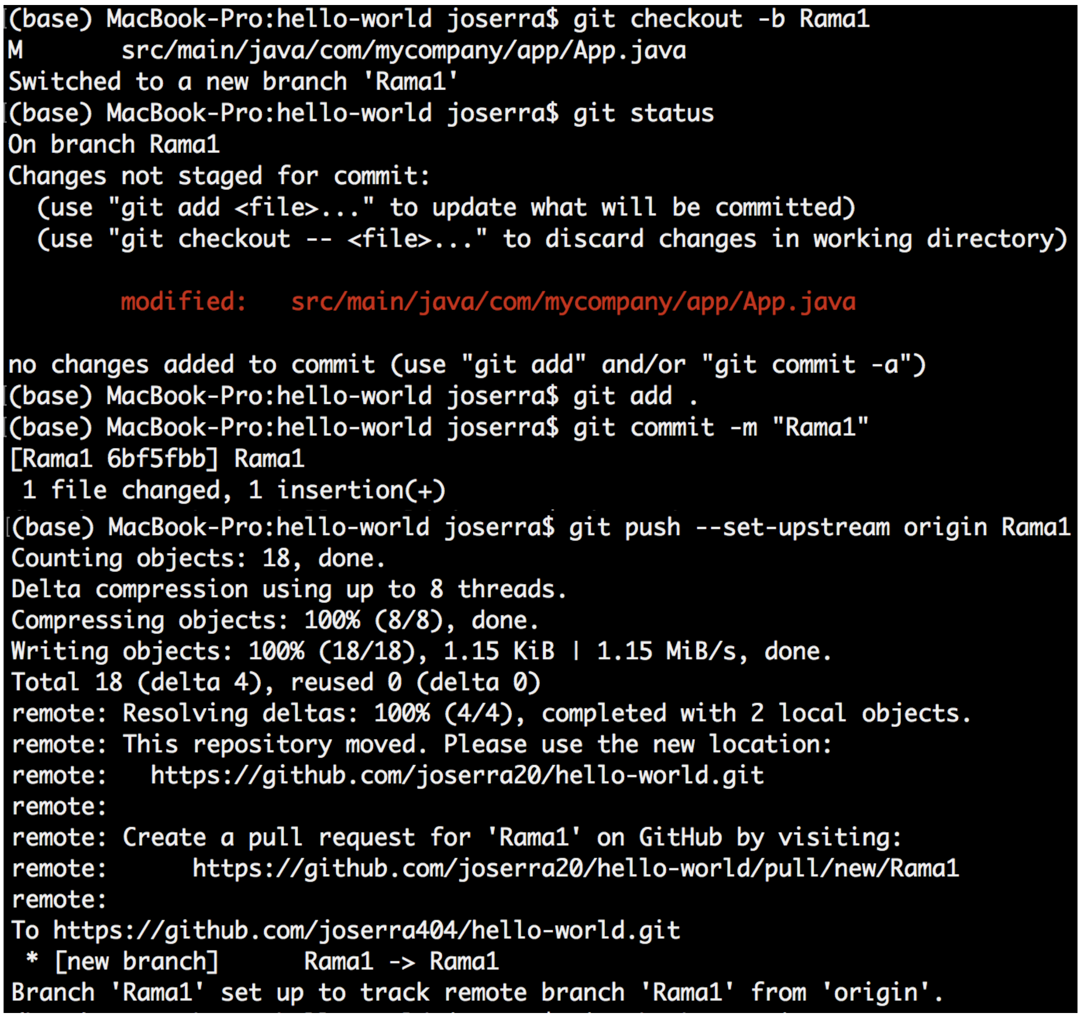
```
3  src/main/java/com/mycompany/app/App.java

        @@ -4,10 +4,11 @@
4    4      * Hello world!
5    5      *
6    6      */
7        - public class App 
     7   + public class App
8    8     {
9    9         public static void main( String[] args )
10   10        {
11   11            System.out.println( "Hello World!" );
     12  +          System.out.println( "I'm testing git!" );
12   13        }
13   14    }
```

git checkout

Se utiliza para cambiar entre ramas y seleccionar cual se va a utilizar.

```
git checkout -b <nombre de la rama>
```

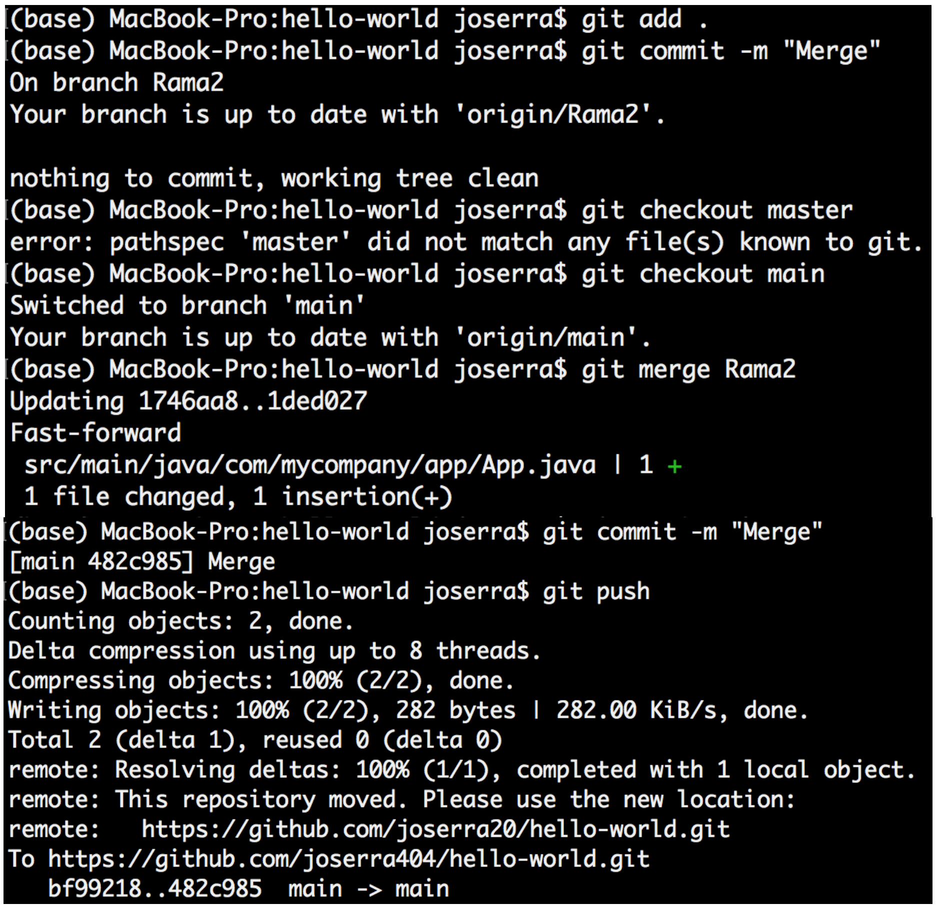
A continuación se muestra la creación de una nueva rama llamada "Rama1" y los pasos necesarios para hacer push a la nueva rama.



```
(base) MacBook-Pro:hello-world joserra$ git checkout -b Rama1
M       src/main/java/com/mycompany/app/App.java
Switched to a new branch 'Rama1'
(base) MacBook-Pro:hello-world joserra$ git status
On branch Rama1
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   src/main/java/com/mycompany/app/App.java

no changes added to commit (use "git add" and/or "git commit -a")
(base) MacBook-Pro:hello-world joserra$ git add .
(base) MacBook-Pro:hello-world joserra$ git commit -m "Rama1"
[Rama1 6bf5fbb] Rama1
 1 file changed, 1 insertion(+)
(base) MacBook-Pro:hello-world joserra$ git push --set-upstream origin Rama1
Counting objects: 18, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (18/18), 1.15 KiB | 1.15 MiB/s, done.
Total 18 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.
remote: This repository moved. Please use the new location:
remote:   https://github.com/joserra20/hello-world.git
remote:
remote: Create a pull request for 'Rama1' on GitHub by visiting:
remote:      https://github.com/joserra20/hello-world/pull/new/Rama1
remote:
To https://github.com/joserra404/hello-world.git
 * [new branch]      Rama1 -> Rama1
Branch 'Rama1' set up to track remote branch 'Rama1' from 'origin'.
```

git merge

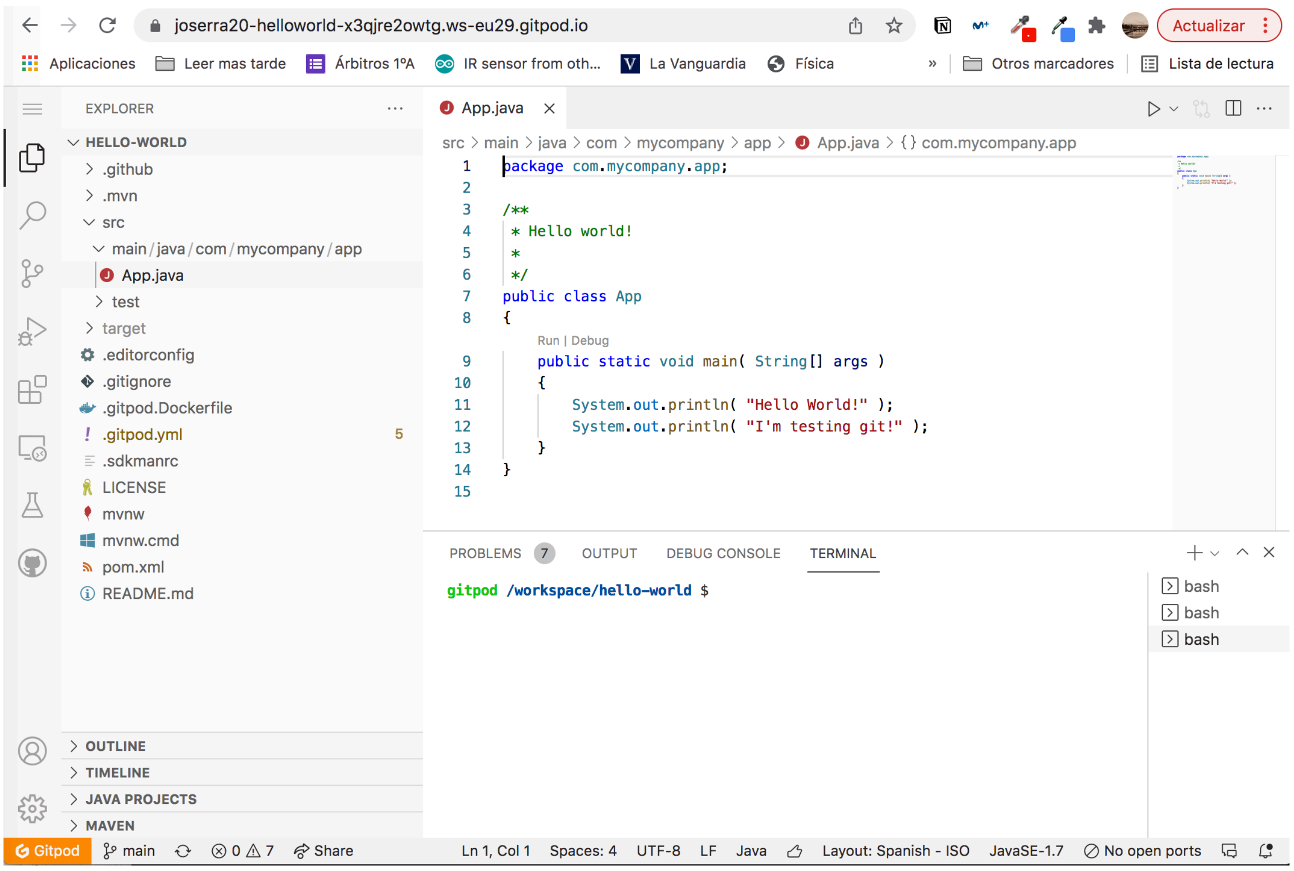
Si deseamos incorporar los cambios o nuevas funcionalidades incluidas previamente en un branch a la rama principal, se emplea `git merge`. En este ejemplo, se va a hacer merge desde el branch2 a la rama principal. En ocasiones esta operación puede producir conflictos.



```
(base) MacBook-Pro:hello-world joserra$ git add .
(base) MacBook-Pro:hello-world joserra$ git commit -m "Merge"
On branch Rama2
Your branch is up to date with 'origin/Rama2'.

nothing to commit, working tree clean
(base) MacBook-Pro:hello-world joserra$ git checkout master
error: pathspec 'master' did not match any file(s) known to git.
(base) MacBook-Pro:hello-world joserra$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
(base) MacBook-Pro:hello-world joserra$ git merge Rama2
Updating 1746aa8..1ded027
Fast-forward
 src/main/java/com/mycompany/app/App.java | 1 +
 1 file changed, 1 insertion(+)
(base) MacBook-Pro:hello-world joserra$ git commit -m "Merge"
[main 482c985] Merge
(base) MacBook-Pro:hello-world joserra$ git push
Counting objects: 2, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 282 bytes | 282.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote: This repository moved. Please use the new location:
remote:   https://github.com/joserra20/hello-world.git
To https://github.com/joserra404/hello-world.git
   bf99218..482c985  main -> main
```

Gitpod

GitPod es un plataforma en línea que incorpora un editor muy similar a la versión de escritorio de Visual Studio Code. Es muy útil pues permite hacer uso de un entorno de desarrollo online, esto te permite realizar modificaciones del código con control de versiones sin la necesidad de tener tu equipo de desarrollo habitual. En la captura que se muestra a continuación se ve la interfaz de GitPod con el repositorio de la práctica abierto.



joserra20-helloworld-x3qjre2owtg.ws-eu29.gitpod.io

```java
package com.mycompany.app;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
        System.out.println( "I'm testing git!" );
    }
}
```

```
gitpod /workspace/hello-world $
```