El concepto de Git Branch es uno de los conceptos más comunes dentro de Git como sistema de control de versiones. Normalmente cuando estamos trabajando con nuestro código vamos realizando varios commits en el respositorio con los cambios que consideramos los adecuados. Por ejemplo supongamos que tenemos el siguiente bloque de código:

```
package com.arquitecturajava;

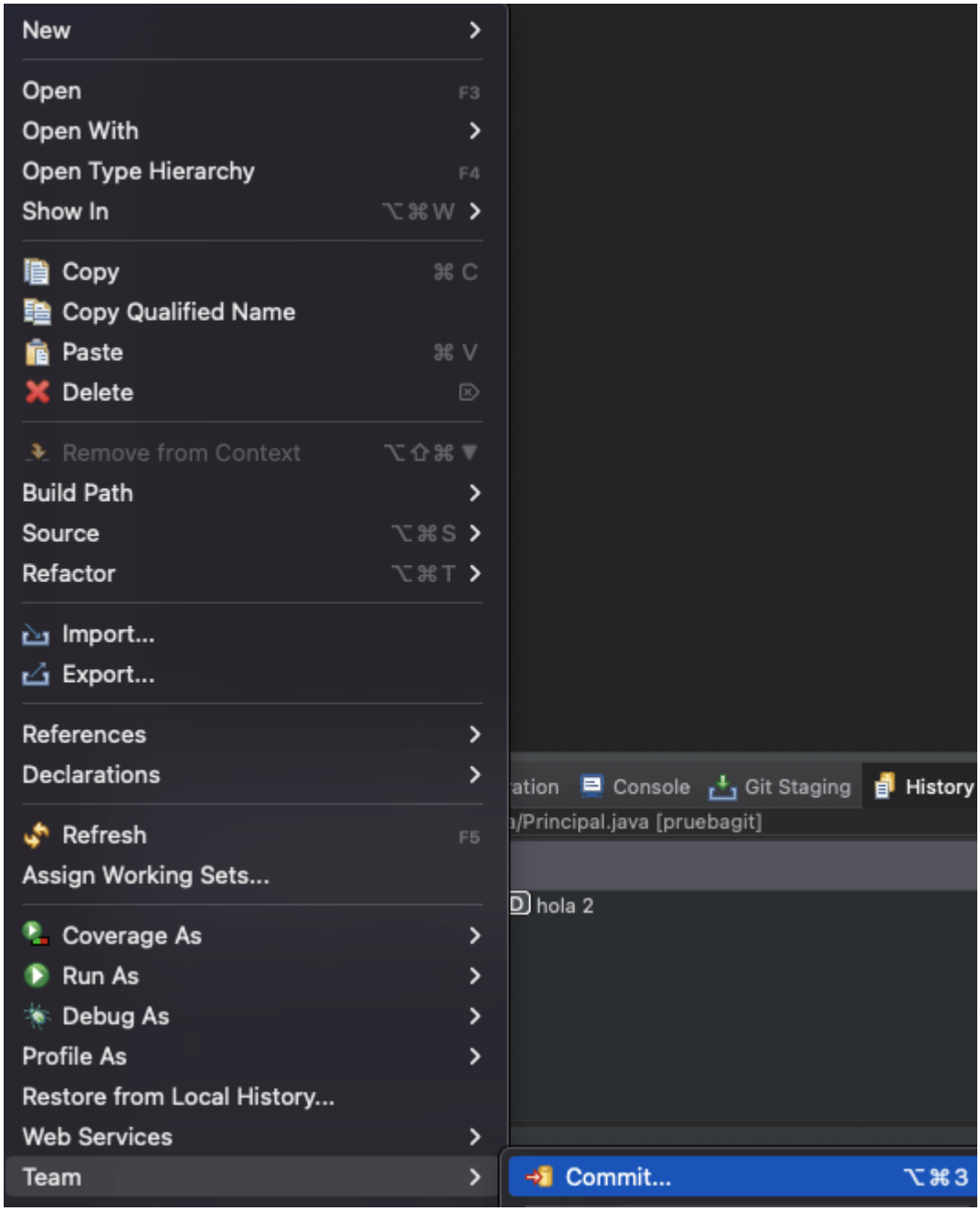
public class Principal {

	public static void main(String[] args) {
		System.out.println("hola");

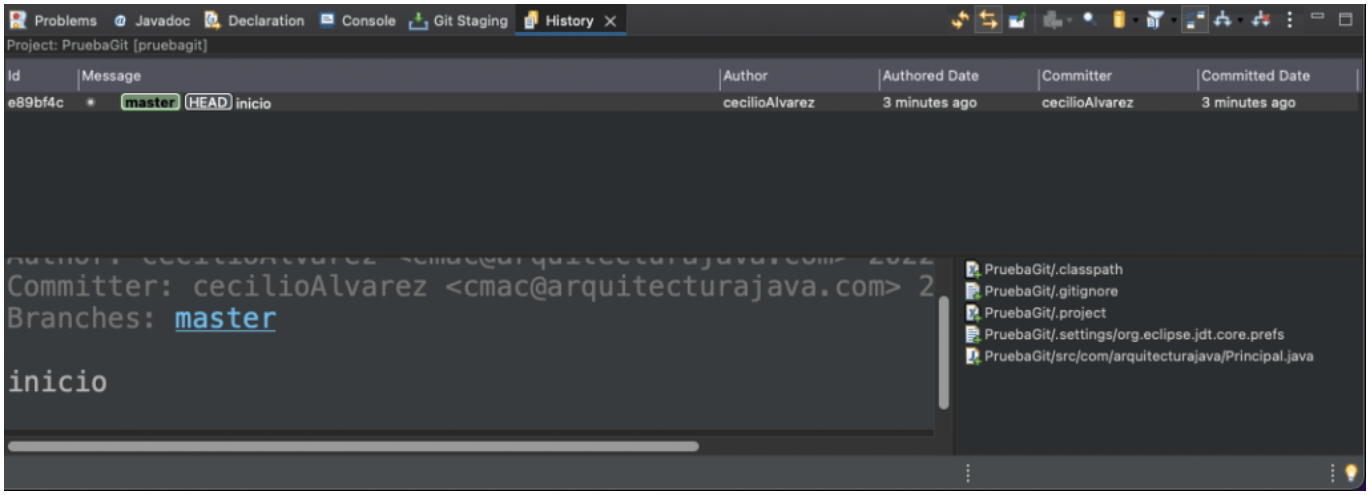
	}

}
```

Este código puedo haberlo salvado con Eclipse en un repositorio GIT y haber realizado un commit inicial pulsando boton derecho sobre el proyecto Team->Commit.




En nuestro caso ya hemos hecho una operación de commit y podemos ver el mensaje de “inicio” en el que se añaden los ficheros principales del proyecto.




En la pestaña de history de Eclipse tenemos un único commit en el repositorio con los ficheros iniciales . Es momento de añadir otra linea de código y realizar otro commit adicional.

```
package com.arquitecturajava;

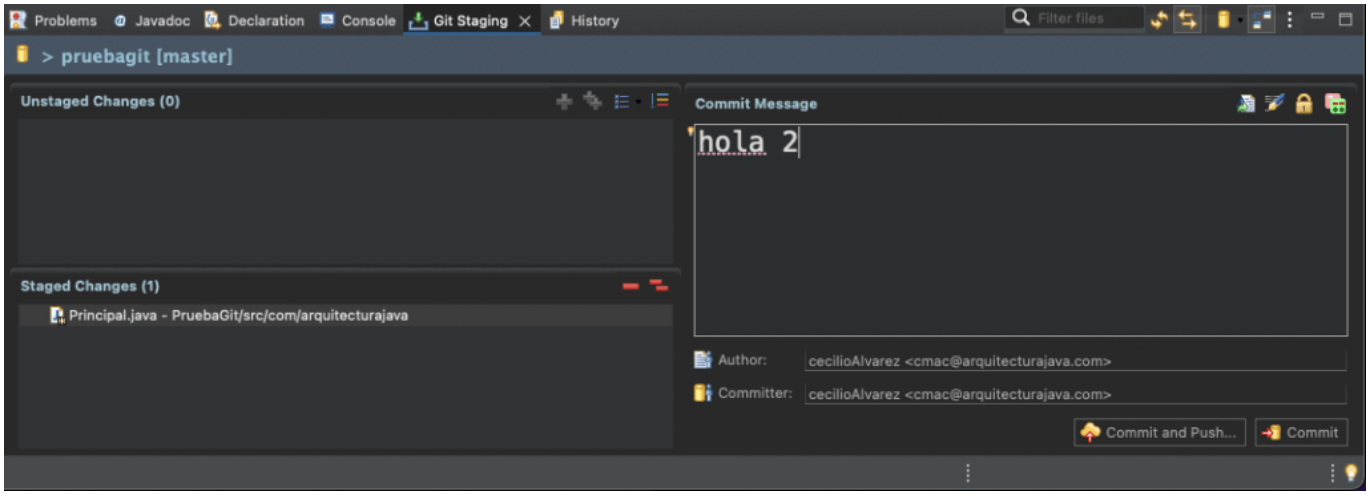
public class Principal {

	public static void main(String[] args) {
		System.out.println("hola");
		System.out.println("hola2");

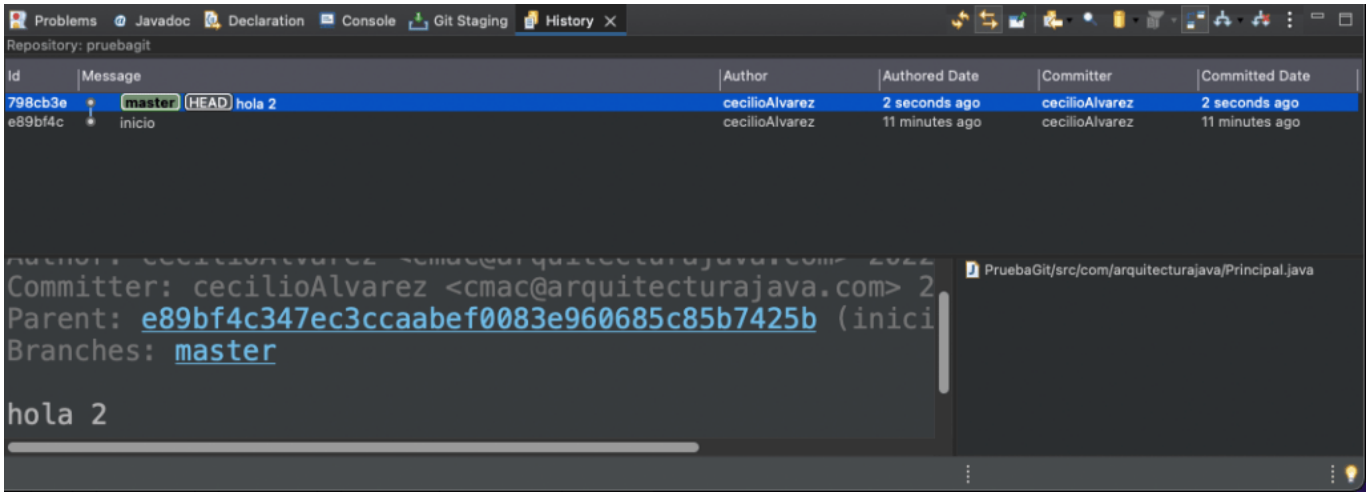
	}

}
```

El mensaje es muy sencillo al hacer el commit:

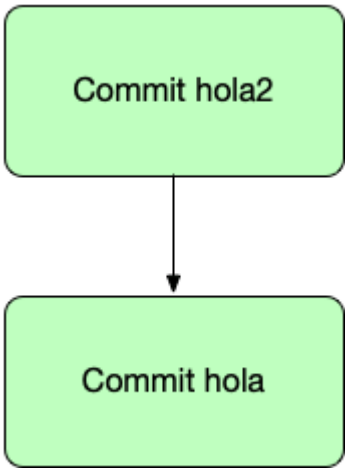



Ya tenemos en el historial 2 commits y todo funciona correctamente:

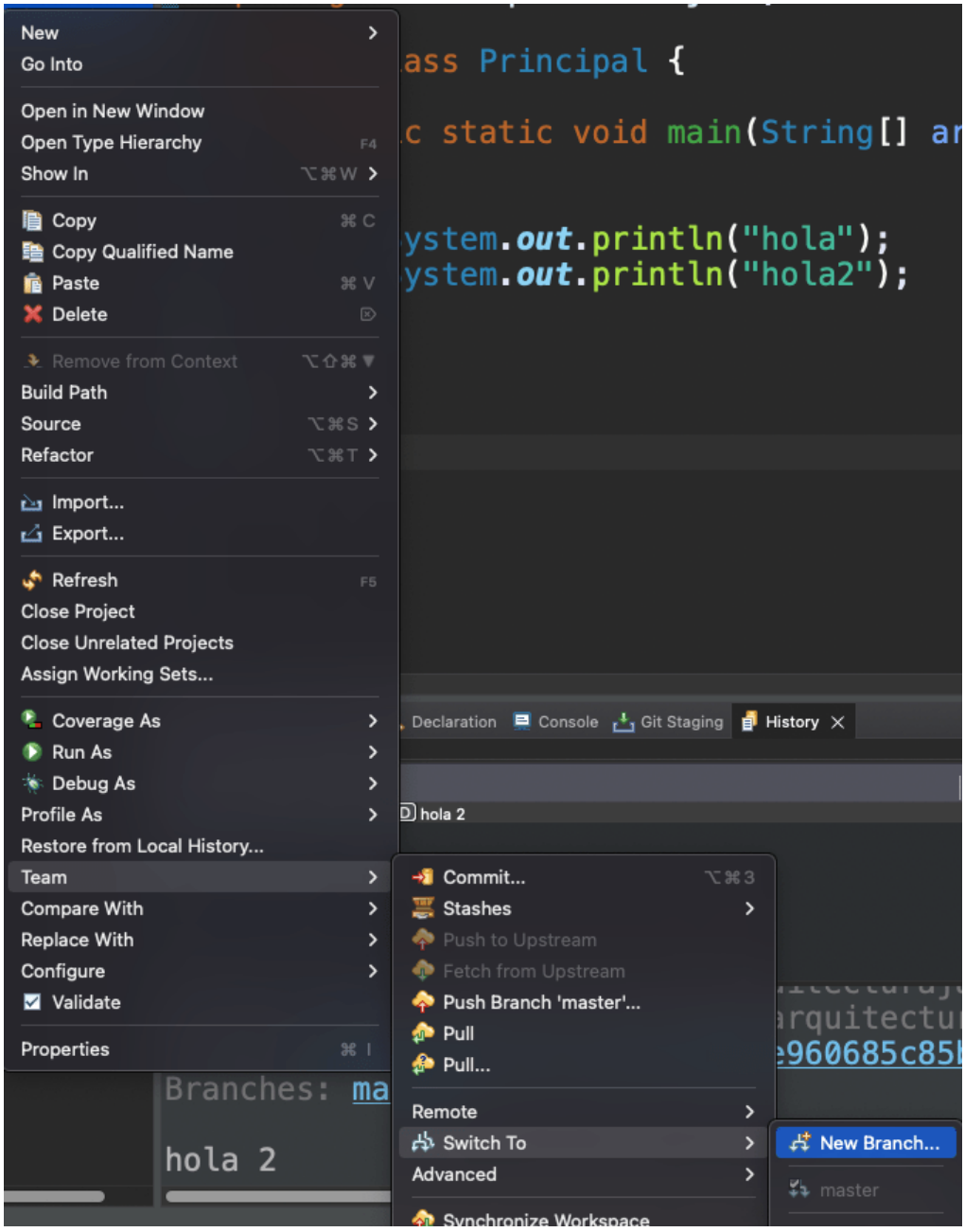



# Git Branch (Ramas)

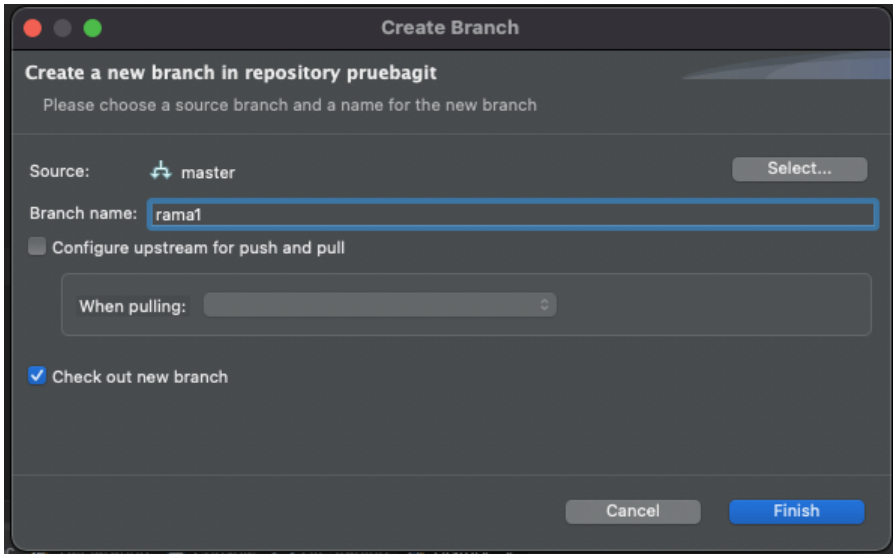
Nos puede parecer un commit válido y podemos seguir trabajando de la misma manera . Sin embargo las cosas no son siempre tan sencillas y sí queremos trabajar de una forma más cómoda y flexible nos podemos apoyar en Git Branch o Ramas de Git. ¿Cómo funcionan las ramas? . Las ramas es una característica de Git que nos permite separarnos del repositorio principal que dispone de una rama master con los commits que acabamos de realizar.



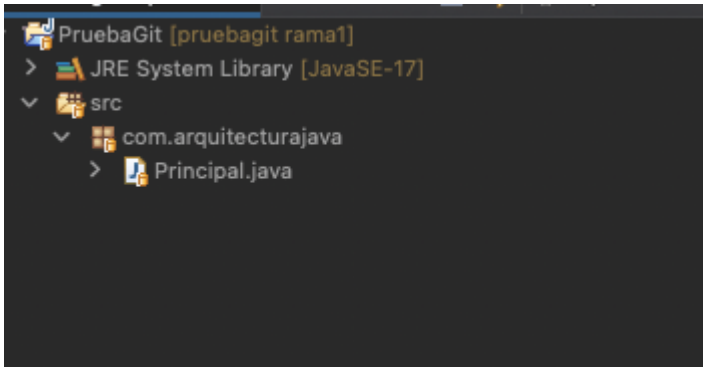

Para ello normalmente ejecutamos el comando git branch rama1 o usamos el propio Eclipse y le decimos que genere una rama nueva como se muestra a continuación.



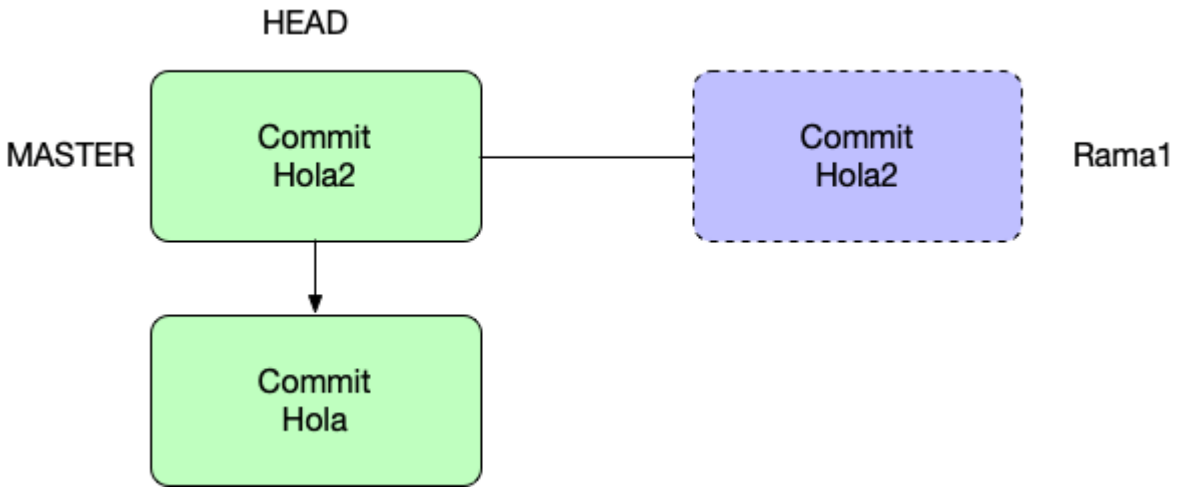
Creamos la rama1




Pulsamos sobre finalizar y el repositorio se encontrará sobre la rama1




Así pues ahora mismo tenemos un repositorio de Git en el cual hay dos ramas



Una vez nosotros tenemos el código sobre la Rama1 podemos ir añadiendo commits adicionales . En este caso modifico dos veces el código y realizo dos operaciones de commit para añadir dos mensajes adicionales.

```
package com.arquitecturajava;

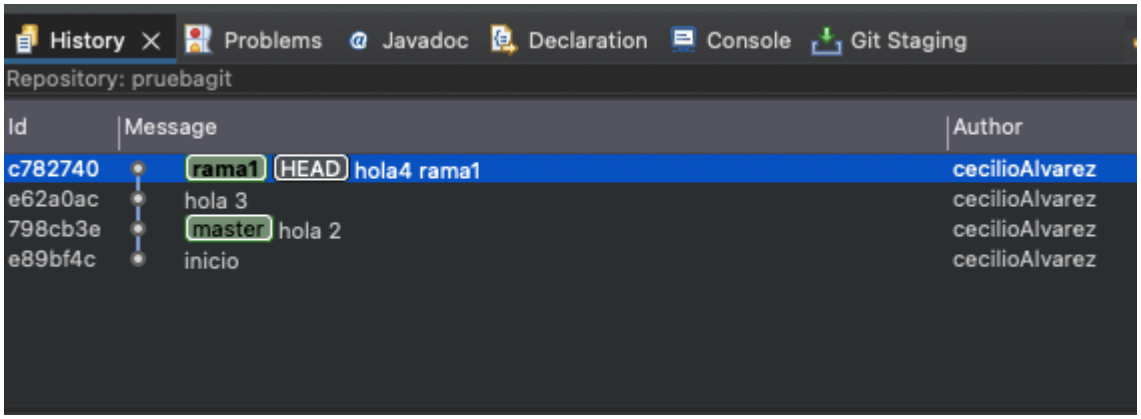
public class Principal {

	public static void main(String[] args) {
		System.out.println("hola");
		System.out.println("hola2");
		System.out.println("hola3 rama1");
		System.out.println("hola4 rama1");

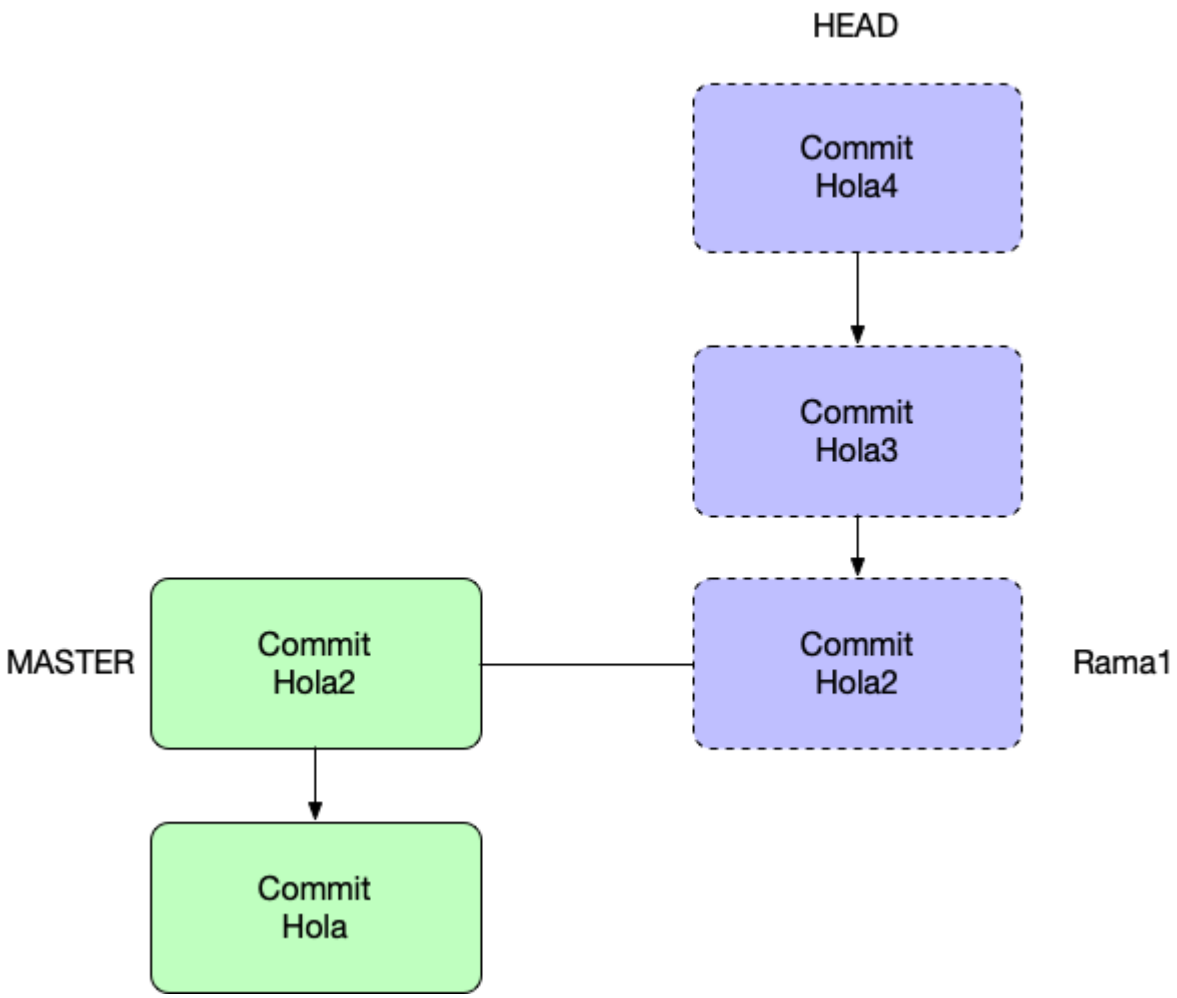
	}

}
```

A nivel del historial del repositorio veremos :

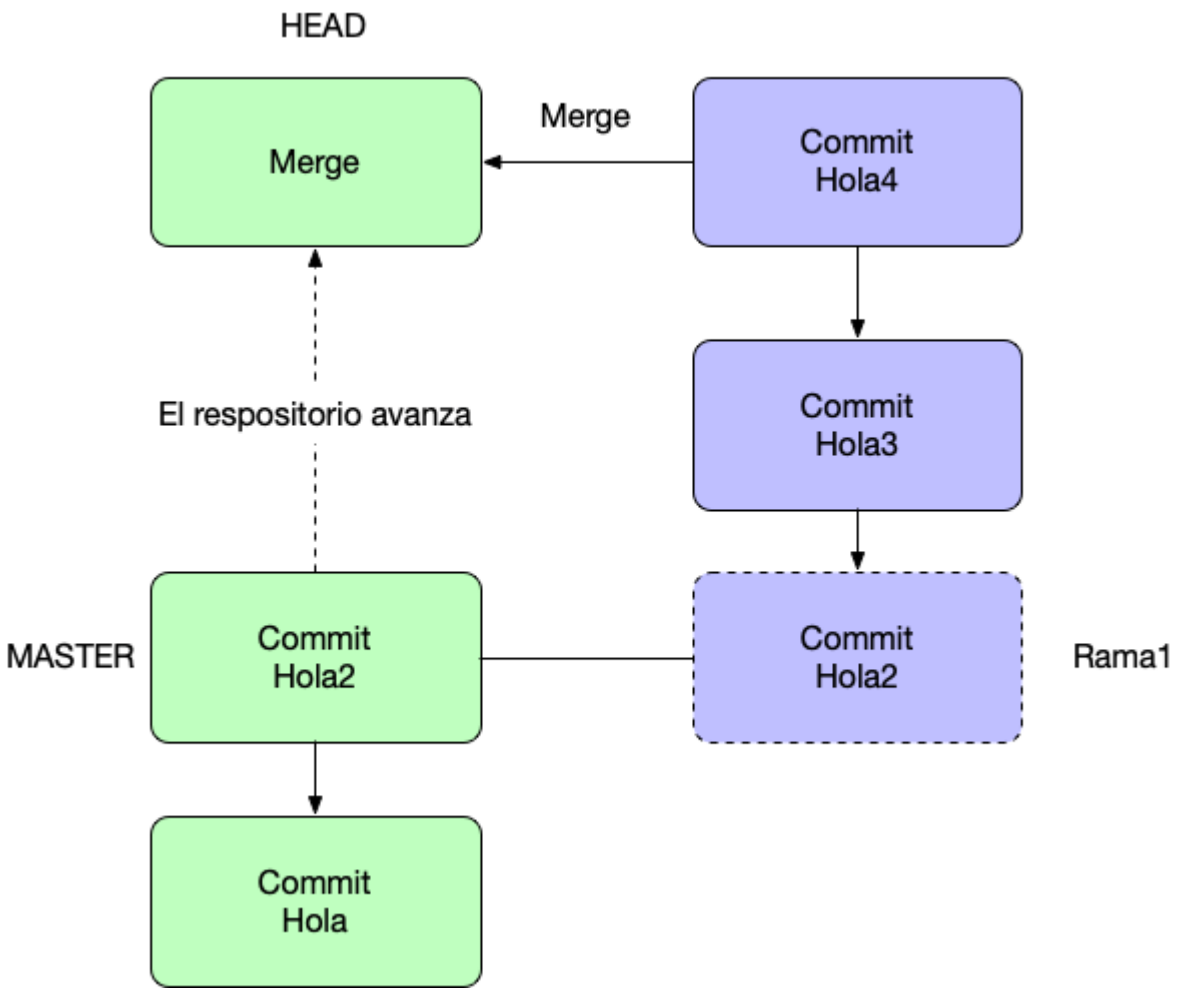



Ya disponemos de 4 commits pero se encuentran divididos en dos ramas:

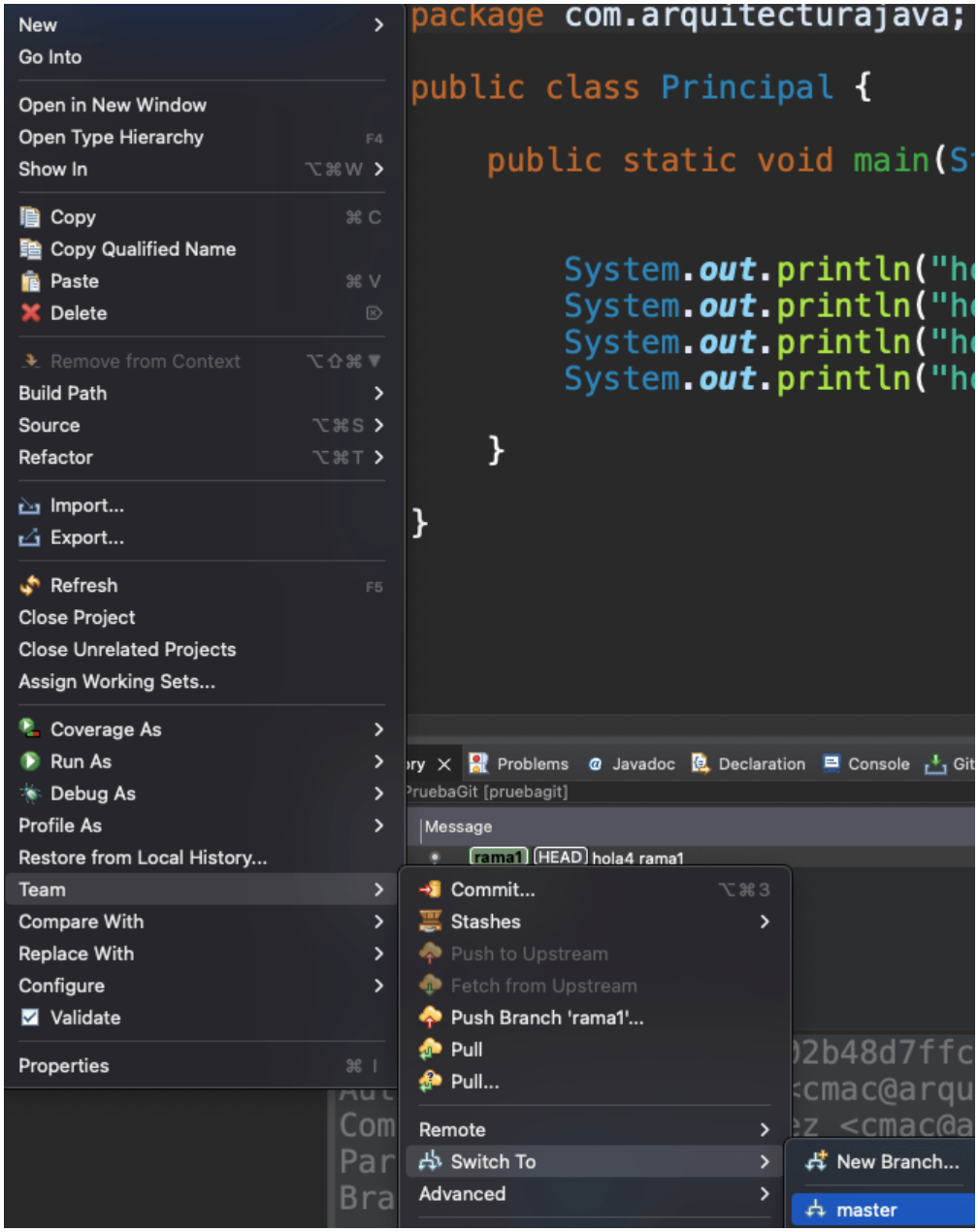



# Git Merge

Es momento de usar las capacidad de Git Merge para fusionar (unir) la Rama1 con la rama Master dentro de nuestro repositorio.




Vamos a hacerlo realizando una operación de switch a la rama Master:




Volvemos a encontrarnos con el código que solo tiene dos commits.

```
package com.arquitecturajava;

public class Principal {

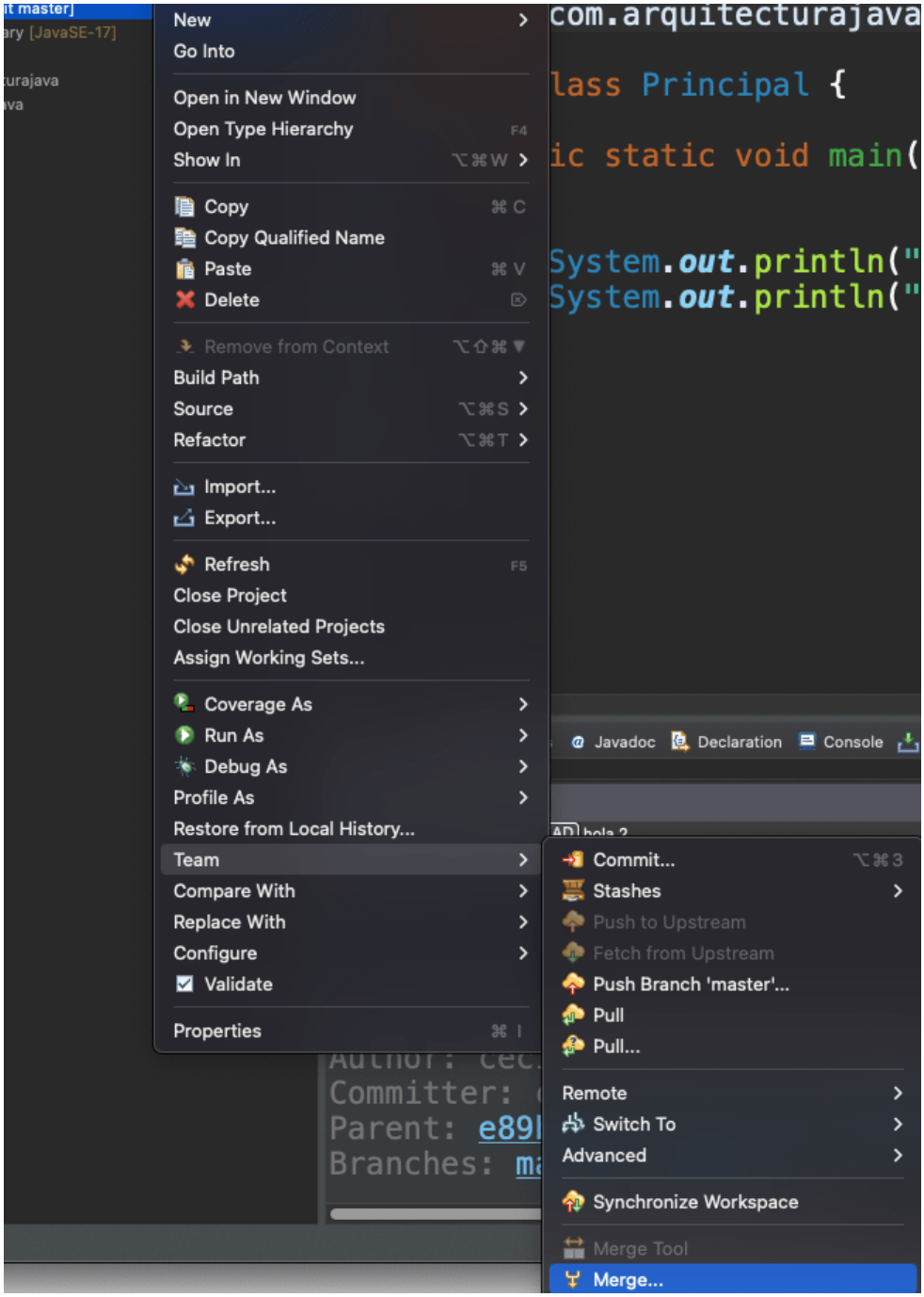
        public static void main(String[] args) {
```

```
                System.out.println("hola");
                System.out.println("hola2");

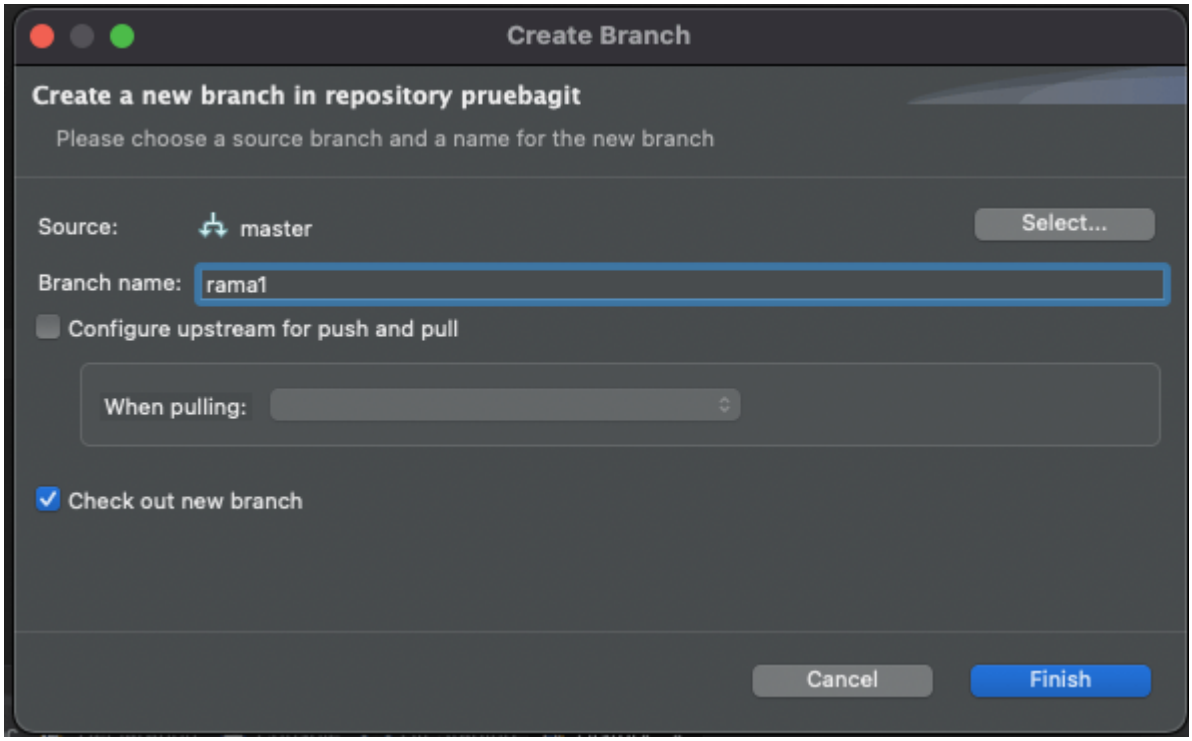
        }

}
```

Es momento de realizar una operación de Git Merge y fusionar la Rama1 con la Master




Una vez elegida la operación nos solicitará elegir la rama que queremos fusionar o unir . En este caso es la Rama1:




La rama Master acaba de realizar una fusión y nos encontramos con el código que teníamos en la Rama1 dentro del Master.

```
package com.arquitecturajava;

public class Principal {

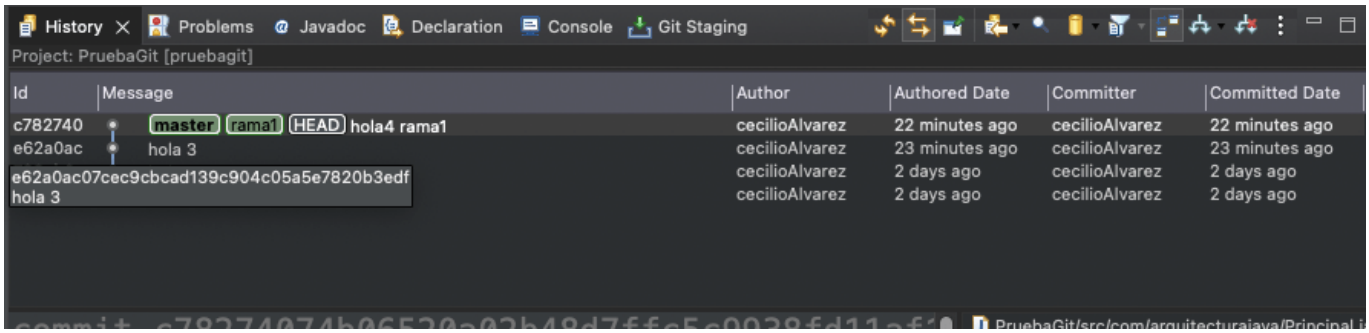
	public static void main(String[] args) {
		System.out.println("hola");
		System.out.println("hola2");
		System.out.println("hola3 rama1");
		System.out.println("hola4 rama1");

	}

}
```

Si miramos al historial del repositorio nos podremos dar cuenta de que todos los commits se

han incorporado a la rama master.




# Git Merge y Buenas Prácticas

En principio la operación que hemos realizado es muy practica ya que hemos fusionado el código que tenemos en la rama con el código que tenemos en el Master. Sin embargo puede haber situaciones en las que tengamos problemas al fusionar la rama Master con cualquier otra rama . Vamos a hablar de ello un poco creando una nueva rama (Rama2) que contenga el siguiente código:

```
package com.arquitecturajava;

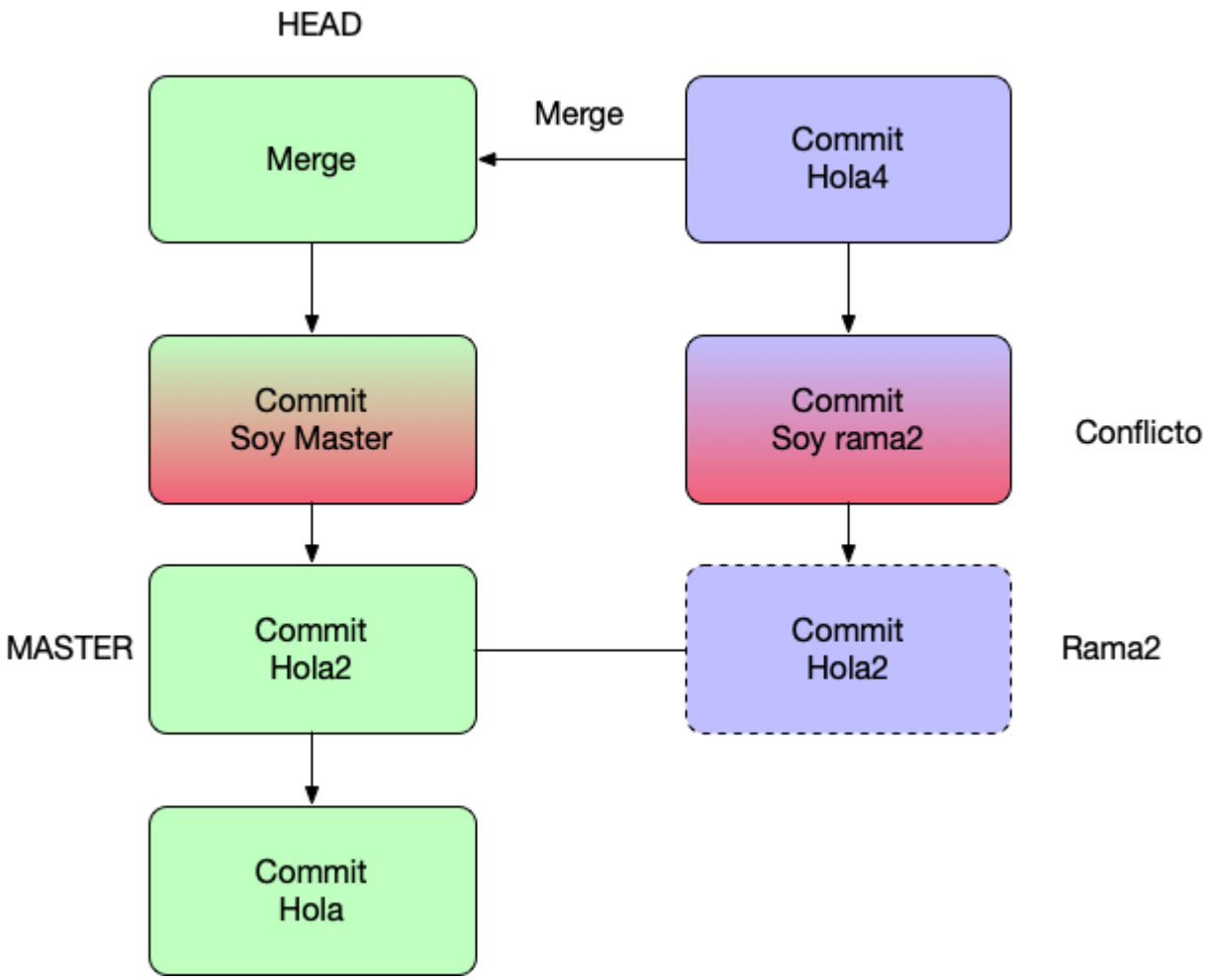
public class Principal {

	public static void main(String[] args) {
		System.out.println("hola");
		System.out.println("hola2");
		System.out.println("soy la rama2");
		System.out.println("hola3 rama1");
		System.out.println("hola4 rama1");

	}

}
```

En este caso hemos añadido una linea que pone “soy la rama2” . Los problemas vienen cuando alguien en la rama Master ha realizado una operación similar y ha escrito lo siguiente:




En este caso el intentar realizar una operación de Merge generará un problema ya que hay lineas de código en conflicto entre las dos Ramas y nuestro código fuente quedará como sigue:

```
package com.arquitecturajava;

public class Principal {
```

```
        public static void main(String[] args) {
                System.out.println("hola");
                System.out.println("hola2");
<<<<<<< HEAD
                System.out.println("soy el master");
=======
                System.out.println("soy la rama2");
>>>>>>> refs/heads/rama2
                System.out.println("hola3 rama1");
                System.out.println("hola4 rama1");

        }

}
```

Eclipse nos permite definir con que bloque de código nos quedamos. En este caso podemos quedarnos con los dos y salvar el fichero.

```
package com.arquitecturajava;

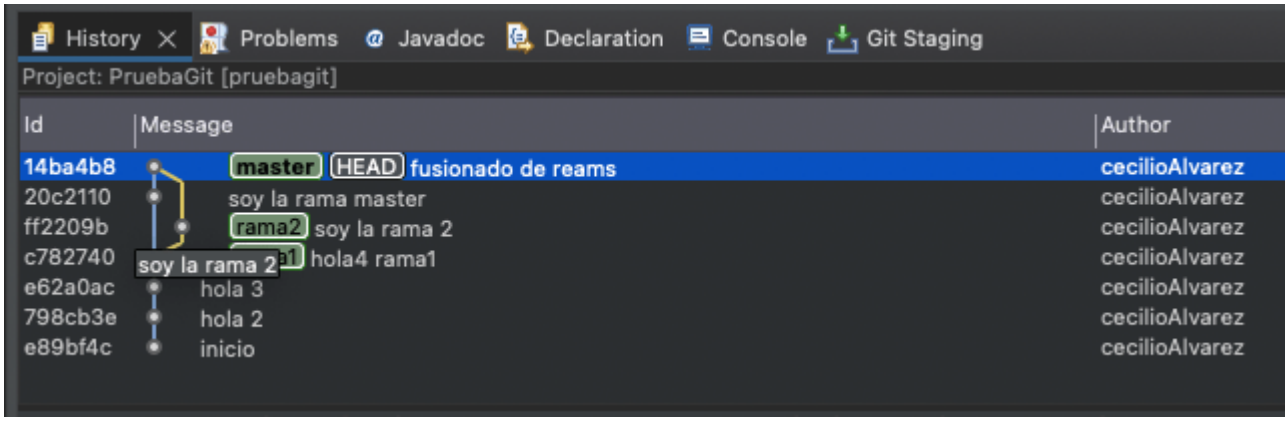
public class Principal {

        public static void main(String[] args) {
                System.out.println("hola");
                System.out.println("hola2");
                System.out.println("soy el master");
                System.out.println("soy la rama2");
                System.out.println("hola3 rama1");
                System.out.println("hola4 rama1");

        }
```

```
}
```

Una vez salvados los cambios simplemente realizamos una operación de commit




Se puede ver perfectamente como las ramas se fusionan una vez realizados los cambios en los ficheros y como Eclipse muestra el gráfico.

# Git Branch ¿Podemos mejorar?

- [ihc-hide-content ihc_mb_type=”show” ihc_mb_who=”4″ ihc_mb_template=”1″ ]
  Esta operación es muy habitual y la podemos dar por correcta . Ahora bien tiene un problema importante y es que fusionamos la rama Rama2 con la rama Master nuestra rama principal si a la hora de fusionar se producen muchos problemas nos encontraremos que nuestros Master esta lleno de errores y no nos pondremos nerviosos. ¿Cómo podemos evitar este caso’? . Vamos a verlo , para ello crearemos una nueva Rama3 que contendrá otra pequeña modificación del código.

```
package com.arquitecturajava;

public class Principal {

	public static void main(String[] args) {
		System.out.println("hola");
		System.out.println("hola2");
```

```
                System.out.println("soy el master");
                System.out.println("soy la rama2");
                System.out.println("hola3 rama1");
                System.out.println("soy la rama3");
                System.out.println("hola4 rama1");

        }

}
```

Una vez hecha modificación y realizado un commit haremos algo similar en la rama master.

```
package com.arquitecturajava;

public class Principal {

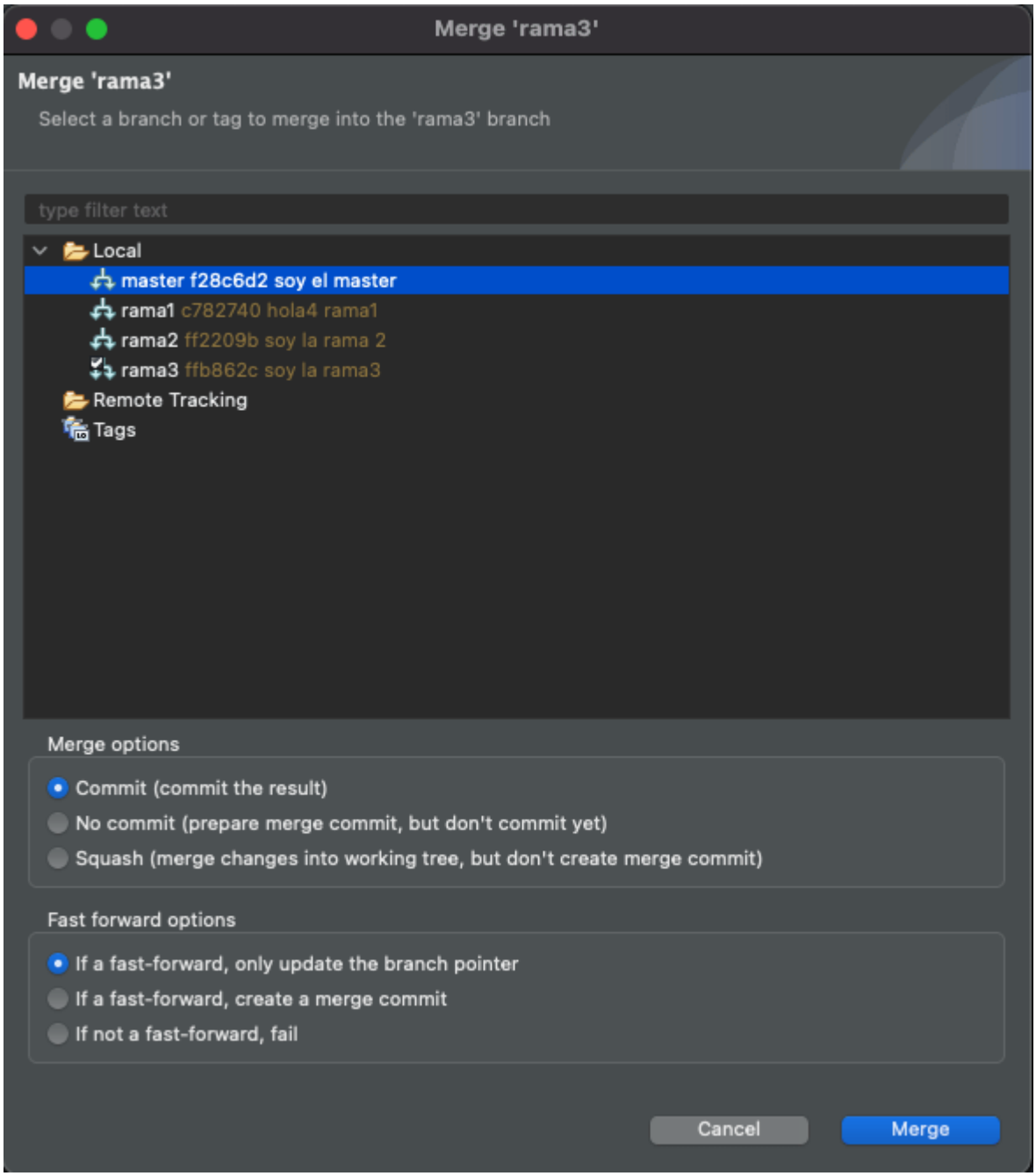
        public static void main(String[] args) {
                System.out.println("hola");
                System.out.println("hola2");
                System.out.println("soy el master");
                System.out.println("soy la rama2");
                System.out.println("hola3 rama1");
                System.out.println("soy el master");
                System.out.println("hola4 rama1");

        }

}
```

Ahora tenemos dos ramas que deseamos fusionar . Pero en vez de realizar la fusión desde Master para añadir los cambios de la Rama3 lo haremos al Revés . Nos posicionaremos en la Rama3 y fusionaremos el Master de esta Manera si tenemos algún error estos se ubicaran el

la Rama3 , los solventaremos allí y la Rama Master no se verá afectada




Esto nos generará un aviso de que tenemos un problema en el código y no se puede realizar una operación de Merge de forma automática.

```
package com.arquitecturajava;
```

```
public class Principal {

	public static void main(String[] args) {
		System.out.println("hola");
		System.out.println("hola2");
		System.out.println("soy el master");
		System.out.println("soy la rama2");
		System.out.println("hola3 rama1");
<<<<<<< HEAD
		System.out.println("soy la rama3");
=======
		System.out.println("soy el master");
>>>>>>> refs/heads/master
		System.out.println("hola4 rama1");

	}

}
```

Corregimos los errores y asumimos ambos códigos como válidos:

```
package com.arquitecturajava;

public class Principal {

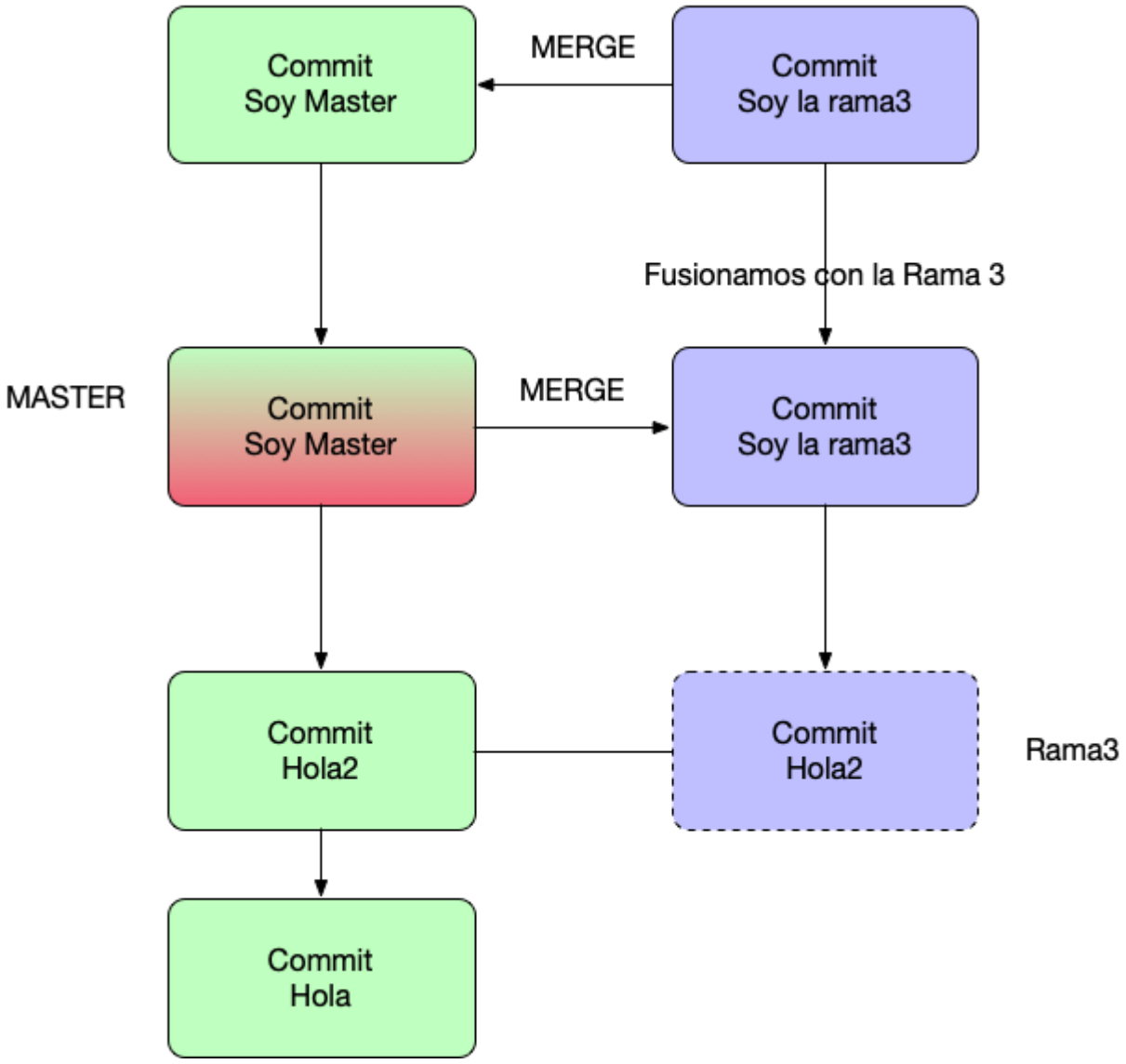
	public static void main(String[] args) {
		System.out.println("hola");
		System.out.println("hola2");
		System.out.println("soy el master");
		System.out.println("soy la rama2");
		System.out.println("hola3 rama1");
		System.out.println("soy la rama3");
```

```
                System.out.println("soy el master");
                System.out.println("hola4 rama1");

        }

}
```

Una vez hecha esta operación podemos solicitar sin problemas un Merge con la Rama Master y los cambios se incorporarán sin problemas.



## Otros artículos relacionados

- Eclipse Git , Repositorios locales y remotos
- Eclipse GIT commit y como crear un repositorio
- Visual Studio Code Java y Extensiones
- Eclipse GIT

[/ihc-hide-content]