

GUÍA DE RESOLUCIÓN DE LA TAREA 05. Documentación y Control de versiones

DESPLIEGUE DE APLICACIONES WEB

CURSO 2023/2024

Contenido

Ejercicio 1: PHPDocumentor	3
Actividad 1.1 Instala la herramienta phpdocumentor en tu servidor Linux y comenta los aspectos más importantes de tal herramienta, así como las etiquetas principales que se usan.	3
Actividad 1.2 En esta actividad debes crear en tu servidor un script PHP con el nombre practica-XXXXXX.php, donde XXXXXX será tu apellido.	6
Actividad 1.3 Después de revisar la documentación, especialmente el apartado 1.2, genera la documentación del script PHP que hemos creado en la actividad 1.2. A continuación muestra el árbol de carpetas y archivos que genera como salida.	7
Ejercicio 2: Git	9
Actividad 2.1 Elabora un pequeño tutorial donde será necesario instalar git en Windows y configurarlo en tu computadora, donde se detallarán las explicaciones teóricas (¿Qué es git? ¿Para qué sirve? ¿Última versión? etc..)y las capturas de pantallas que sean necesarias.	9
Actividad 2.2 Realiza la siguiente configuración:	11
Actividad 2.3 En este apartado que se escogerá el fichero que hemos realizado en el apartado 1.2 y demostrar cómo funciona git en tal proyecto, por ejemplo, subiendo a git todo el proyecto completo y posteriormente es necesario modificar algo para que se detecten los cambios. Es necesario realizarlo con NetBeans. Hay que darse de alta en la url https://github.com/ y crear un repositorio llamado distanciada2324 para llevar los ficheros del proyecto php. Demostrarlo con pantallas.	12

Este tipo de actividades se realizan en tu servidor Linux ya sea Ubuntu o Debian y Windows, habrá dos tipos de actividades, una relacionada con phpDocumentor y otra con GitHub.

Ejercicio 1: PHPDocumentor

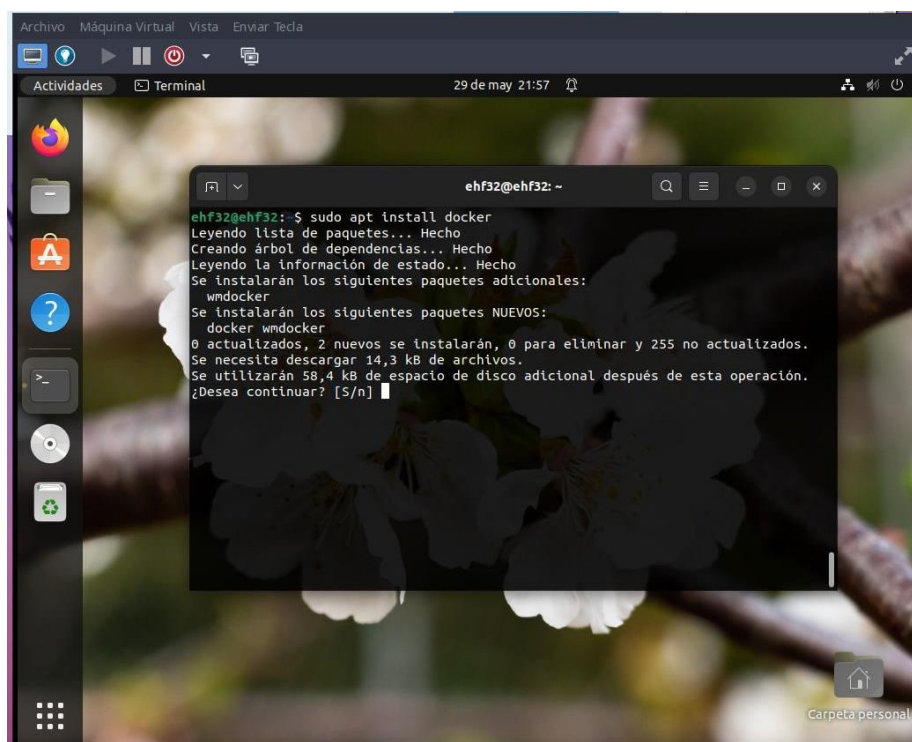
Actividad 1.1 Instala la herramienta phpdocumentor en tu servidor Linux y comenta los aspectos más importantes de tal herramienta, así como las etiquetas principales que se usan.

Hay varias formas de instalar y usar la herramienta según su documentación oficial:

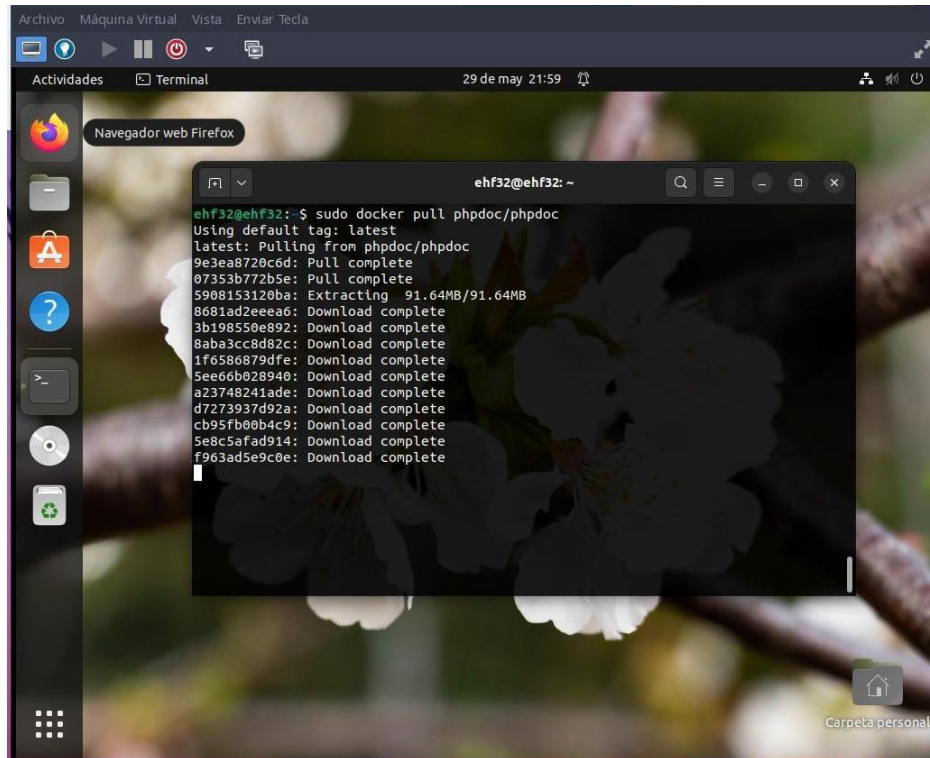
- Utilizando phive
- Utilizando PHAR
- Utilizando Docker
- Utilizando Composer

En este caso vamos a realizar la instalación utilizando docker. Descargamos la última versión de su página de GitHub:

```
1 sudo apt install docker
```



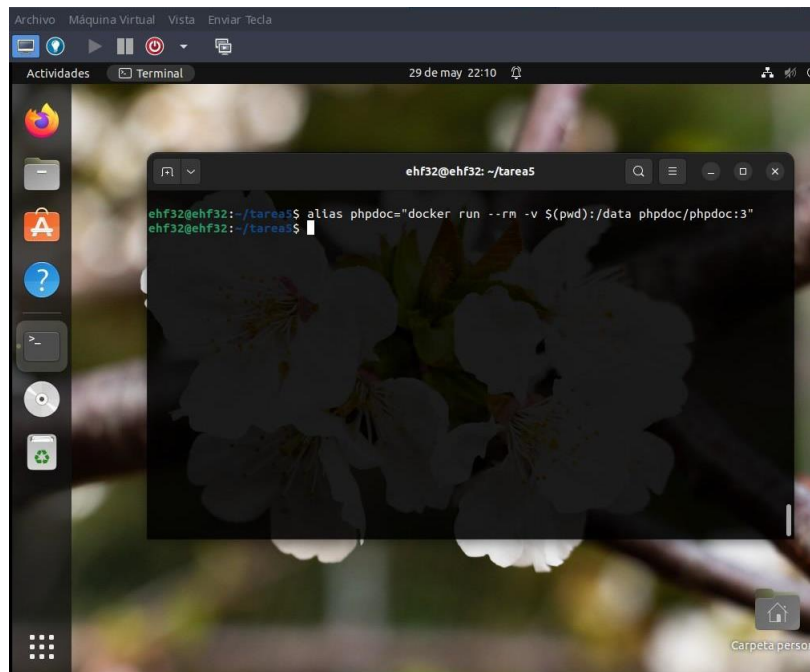
Nos bajamos la imagen oficial de phpdocumentor:



Docker realizará toda la instalación por nosotros en un contenedor virtual, por lo que no es necesario realizar ninguna acción adicional. Las acciones que realiza se pueden consultar en el archivo Dockerfile en el repositorio de GitHub de PHPDocumentor.

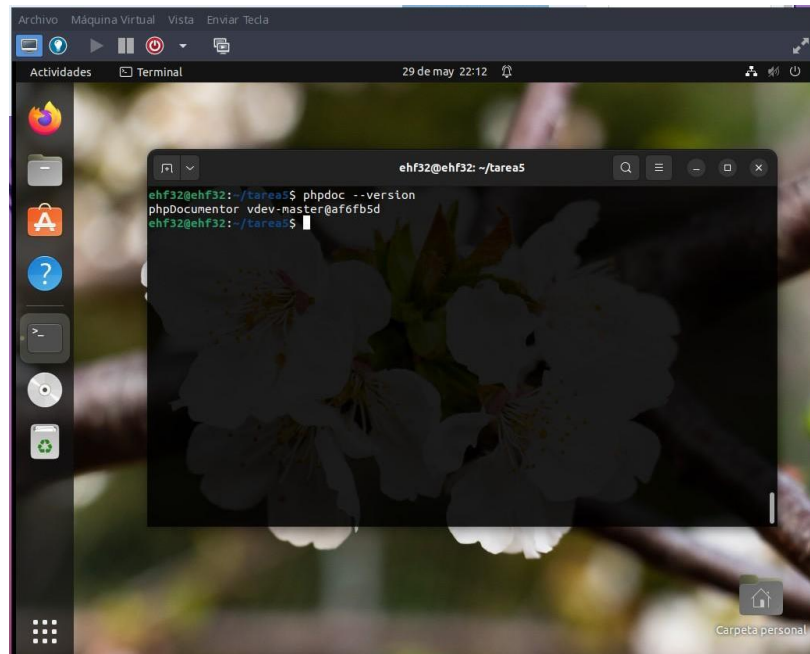
Para poder utilizarlo solo añadimos un alias:

```
1 alias phpdoc="docker run --rm -v $(pwd):/data phpdoc / phpdoc :3 "
```



Y ya comprobamos que lo tenemos instalado:

```
phpdoc --version
```



Aspectos importantes de phpDocumentor:

- phpDocumentor es una herramienta de generación de documentación para código PHP.
- Permite generar documentación en formatos legibles por humanos, como HTML, y también en formatos de intercambio de datos, como XML y JSON.
- phpDocumentor analiza el código fuente PHP y extrae información relevante, como comentarios de documentación y estructura de clases, funciones, métodos, variables, etc.
- La documentación generada por phpDocumentor puede incluir descripciones, ejemplos, listas de parámetros, tipos de retorno, excepciones, entre otros detalles útiles para entender y utilizar el código.
- La herramienta utiliza etiquetas especiales en los comentarios de documentación para anotar información específica, como "@param" para describir los parámetros de una función, "@return" para especificar el tipo de retorno, "@throws" para indicar excepciones lanzadas, etc.

Las etiquetas más comunes son:

- @package: Especifica el nombre o la descripción del paquete al que pertenece el código documentado.
- @subpackage: Permite especificar un subpaquete o una subdivisión adicional dentro del paquete principal.
- @author: Indica el autor o autores del código documentado.
- @version: Muestra la versión del código documentado.

- @param: Describe los parámetros de entrada de una función o método, incluyendo su nombre y una descripción opcional.
- @return: Especifica el tipo de valor de retorno de una función o método.
- @throws: Indica las excepciones que puede lanzar una función o método.
- @var: Describe una variable y su tipo.
- @property: Documenta una propiedad de una clase y su tipo.
- @method: Documenta un método de una clase.
- @internal: Indica que una parte del código solo es visible en la documentación para desarrolladores.
- @deprecated: Marca una función, método, clase o elemento como obsoleto, indicando que no se debe utilizar.
- @inheritdoc: Permite heredar la documentación de una clase o método padre.

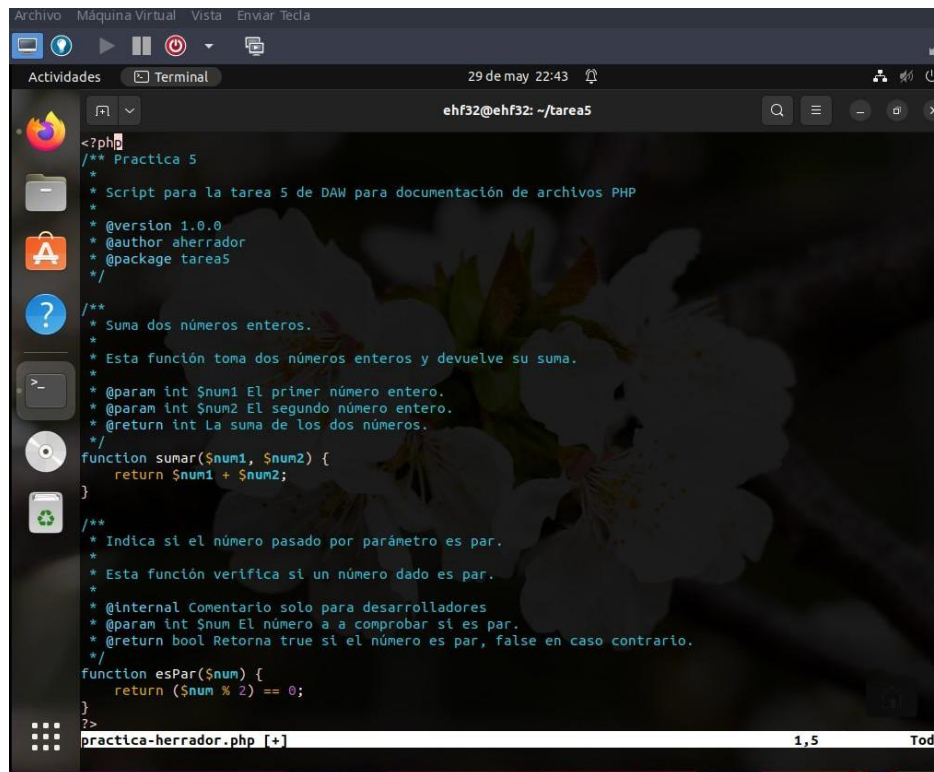
Actividad 1.2 En esta actividad debes crear en tu servidor un script PHP con el nombre practica- XXXXXX.php, donde XXXXXX será tu apellido. A continuación, escribe dentro de este script bloques de código y DocBlocks para que luego se pueda generar la documentación correspondiente. El script debe contener al menos dos funciones documentadas, indicando mediante las etiquetas vistas en la unidad los siguientes elementos:

- Parámetros de entrada de la función.
- Parámetros de devuelve la función.
- Autor y versión del script.
- Una anotación que solo sea visible en la documentación para desarrolladores.

Primero creamos el archivo:

```
1 vim practica - herrador. php
```

Y creamos el archivo:



The image shows a terminal window within a virtual machine environment. The terminal displays PHP code for a practice task. The code includes a docblock for 'Practica 5', a function 'sumar' that adds two integers, and a function 'esPar' that checks if a number is even. The terminal prompt is 'ehf32@ehf32: ~/tarea5'. The background of the terminal window features a faint image of white flowers.

```
<?php
/** Practica 5
 *
 * Script para la tarea 5 de DAW para documentación de archivos PHP
 *
 * @version 1.0.0
 * @author aherrador
 * @package tarea5
 */

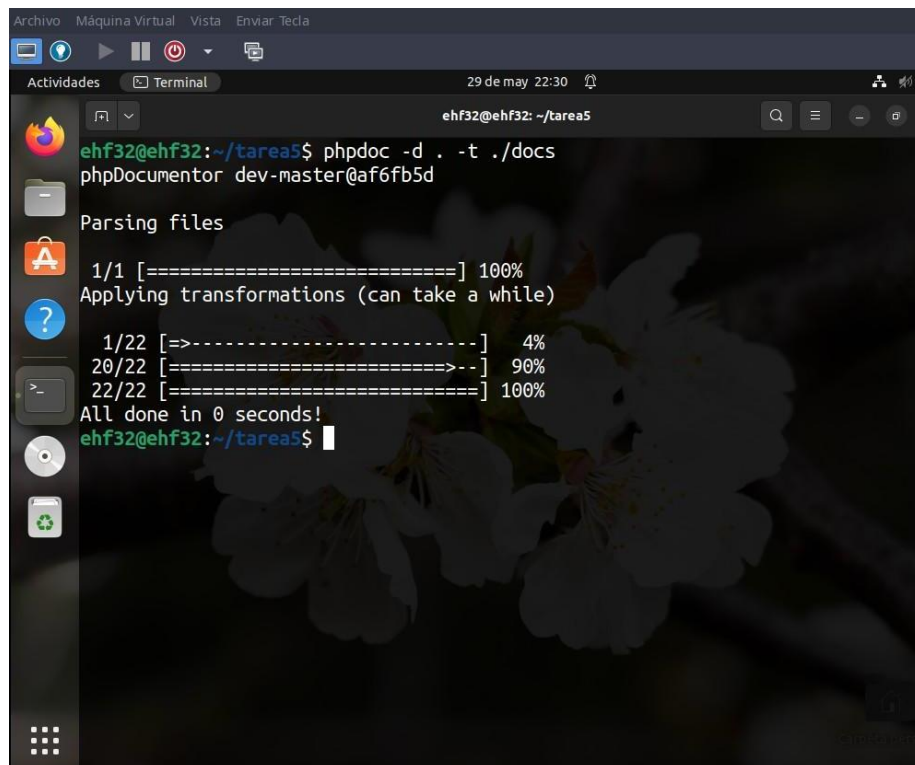
/**
 * Suma dos números enteros.
 *
 * Esta función toma dos números enteros y devuelve su suma.
 *
 * @param int $num1 El primer número entero.
 * @param int $num2 El segundo número entero.
 * @return int La suma de los dos números.
 */
function sumar($num1, $num2) {
    return $num1 + $num2;
}

/**
 * Indica si el número pasado por parámetro es par.
 *
 * Esta función verifica si un número dado es par.
 *
 * @internal Comentario solo para desarrolladores
 * @param int $num El número a a comprobar si es par.
 * @return bool Retorna true si el número es par, false en caso contrario.
 */
function esPar($num) {
    return ($num % 2) == 0;
}
?>
```

Actividad 1.3 Después de revisar la documentación, especialmente el apartado 1.2, genera la documentación del script PHP que hemos creado en la actividad 1.2. A continuación muestra el árbol de carpetas y archivos que genera como salida.

Para generar la documentación:

```
phpdoc -d . -t ./ docs
```

A terminal window titled "ehf32@ehf32: ~/tarea5" showing the execution of the command `phpdoc -d . -t ./docs`. The output indicates that phpDocumentor dev-master@af6fb5d is parsing files and applying transformations. A progress bar shows 1/1 files processed at 100%. A second progress bar shows 1/22 transformations at 4%, 20/22 at 90%, and 22/22 at 100%. The process completes in 0 seconds.

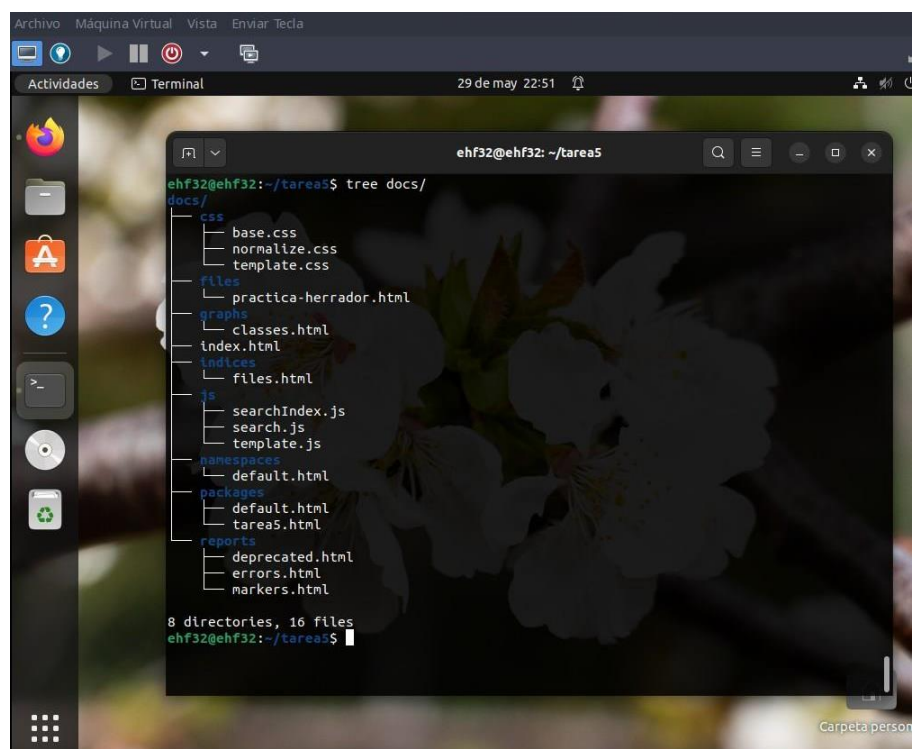
```
ehf32@ehf32: ~/tarea5$ phpdoc -d . -t ./docs
phpDocumentor dev-master@af6fb5d

Parsing files

1/1 [=====] 100%
Applying transformations (can take a while)

1/22 [=>-----] 4%
20/22 [=====] 90%
22/22 [=====] 100%
All done in 0 seconds!
ehf32@ehf32: ~/tarea5$
```

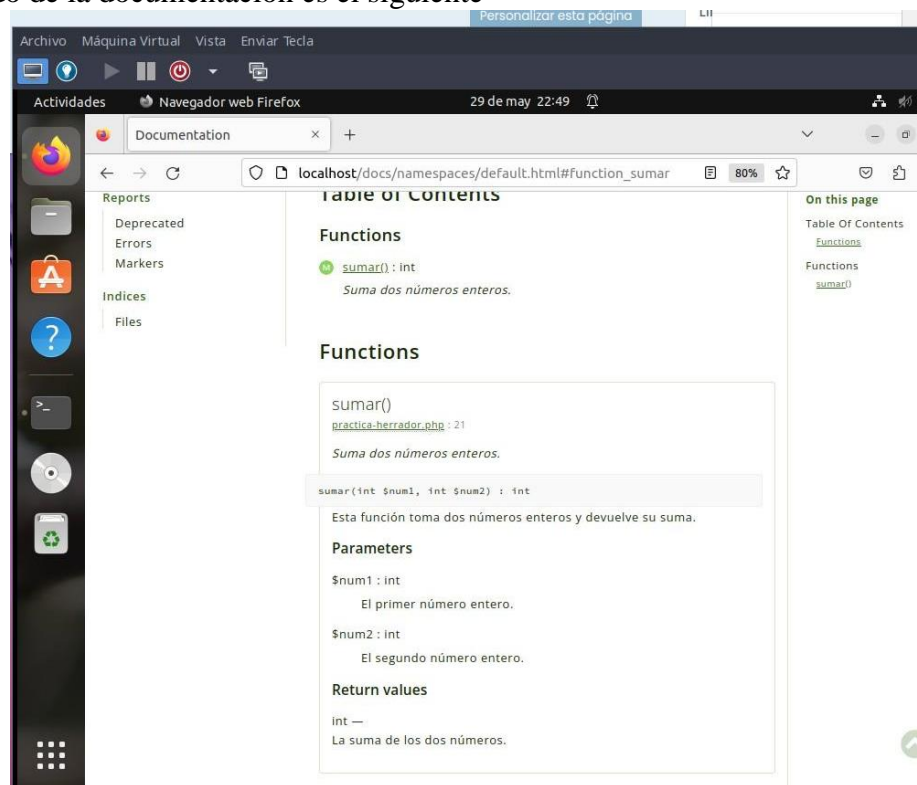
Y nos genera la documentación en la carpeta `./doc`, con el siguiente árbol de carpetas:



A terminal window titled "ehf32@ehf32: ~/tarea5" showing the output of the `tree docs/` command. The output displays a hierarchical tree structure of the generated documentation files and directories.

```
ehf32@ehf32: ~/tarea5$ tree docs/
docs/
├── css
│   ├── base.css
│   ├── normalize.css
│   └── template.css
├── files
│   └── practica-herrador.html
├── graphs
│   ├── classes.html
│   └── index.html
├── indices
│   └── files.html
├── js
│   ├── searchIndex.js
│   ├── search.js
│   └── template.js
├── namespaces
│   └── default.html
├── packages
│   ├── default.html
│   ├── tarea5.html
│   └── reports
│       ├── deprecated.html
│       ├── errors.html
│       └── markers.html
└── 8 directories, 16 files
ehf32@ehf32: ~/tarea5$
```


El resultado de la documentación es el siguiente



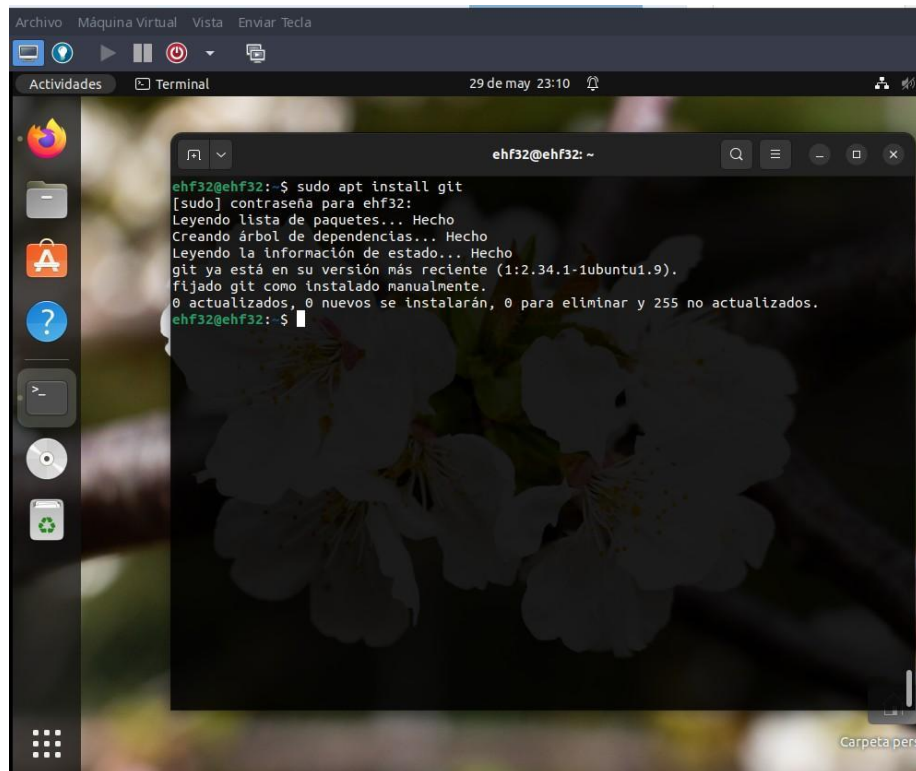
Ejercicio 2: Git

Este ejercicio está relacionado con Github, se puede realizar en windows perfectamente, que consistirá en los siguientes apartados:

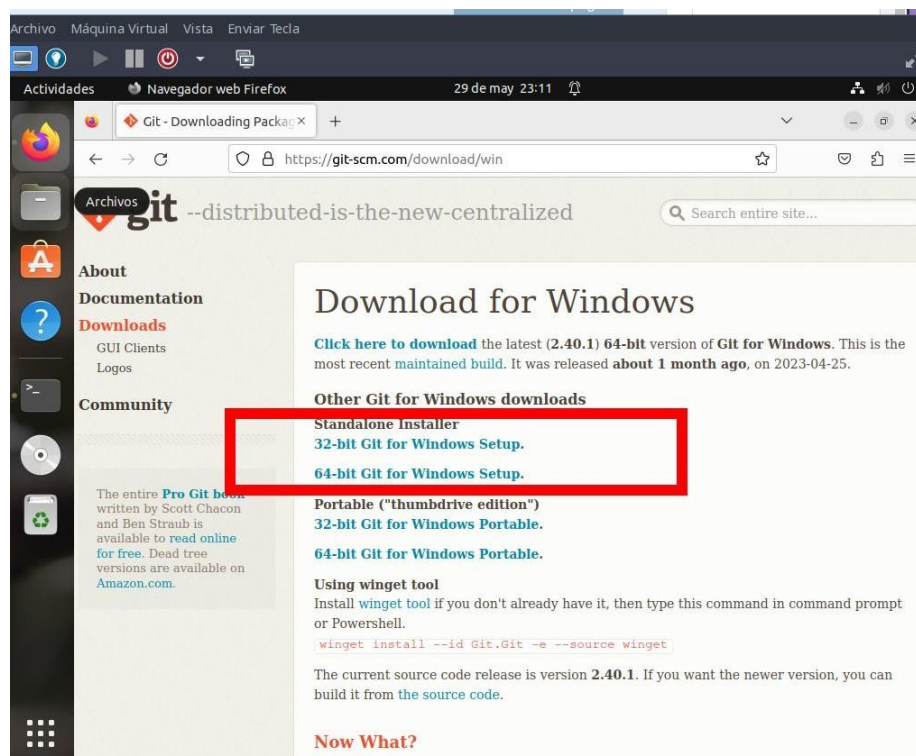
Actividad 2.1 Elabora un pequeño tutorial donde será necesario instalar git en Windows y configurararlo en tu computadora, donde se detallarán las explicaciones teóricas (¿Qué es git? ¿Para qué sirve? ¿Última versión? etc..)y las capturas de pantallas que sean necesarias.

NOTA: Ya que la configuración de Git es idéntica en todos los sistemas operativos, voy a seguir realizándolo en Linux.

Instalamos Git



Para windows descargaríamos uno de los siguientes archivos dependiendo de nuestra máquina:



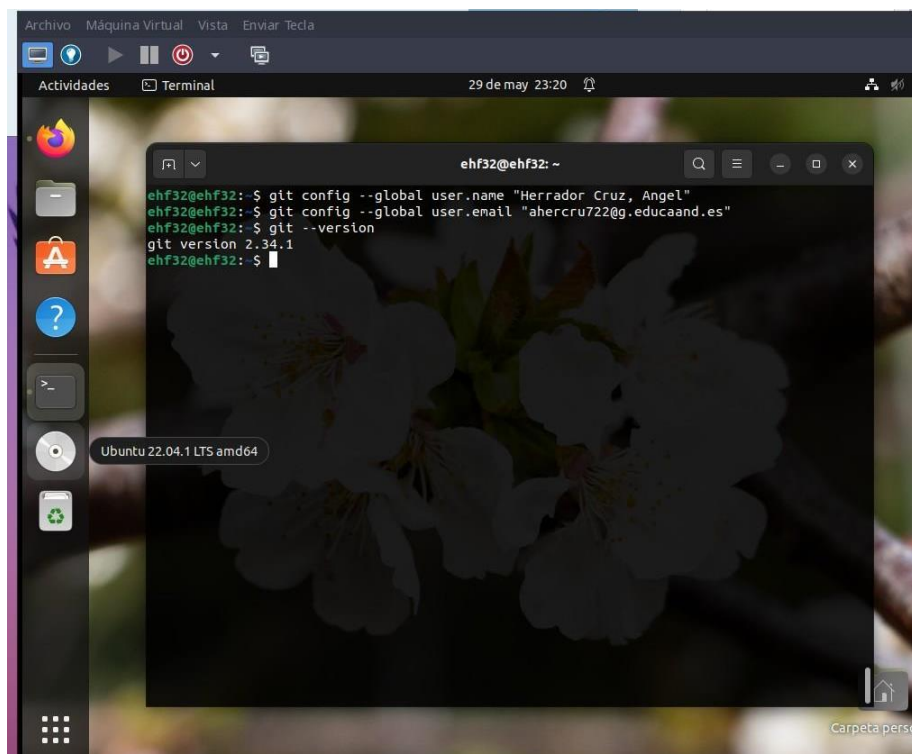
Git es un sistema de control de versiones utilizado en el desarrollo de software. Sirve para rastrear y administrar los cambios en el código fuente a lo largo del tiempo, permitiendo a los desarrolladores colaborar, realizar ramificaciones, fusionar cambios y revertir a versiones anteriores. Además, facilita la sincronización con repositorios remotos para respaldar y compartir el código con otros desarrolladores, es una herramienta fundamental para el control y la gestión de versiones en el desarrollo de software. La última versión de Git es la 2.40.1

Actividad 2.2 Realiza la siguiente configuración:

- Configura tu nombre, apellidos y cuenta de correo
- Muestra la versión instalada.
- Indica cuál es tu directorio de trabajo.

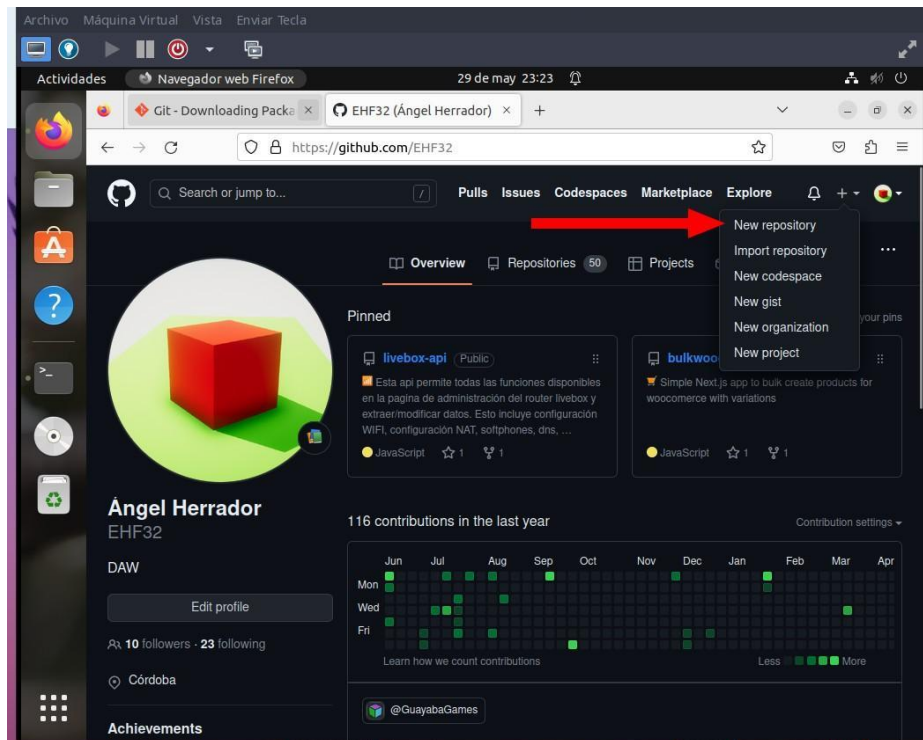
Para configurarlo introducimos los siguientes comandos:

```
git config --global user.name "Herrador Cruz, Angel" #Nombre y apellidos
git config --global user.email "profesor@g.educaand.es" #Correo
git --version #Muestra la version instalada
git init #Inicializa el repositorio , indicando que va a ser nuestro directorio de trabajo para
empezar a realizar el seguimiento de los cambios
```

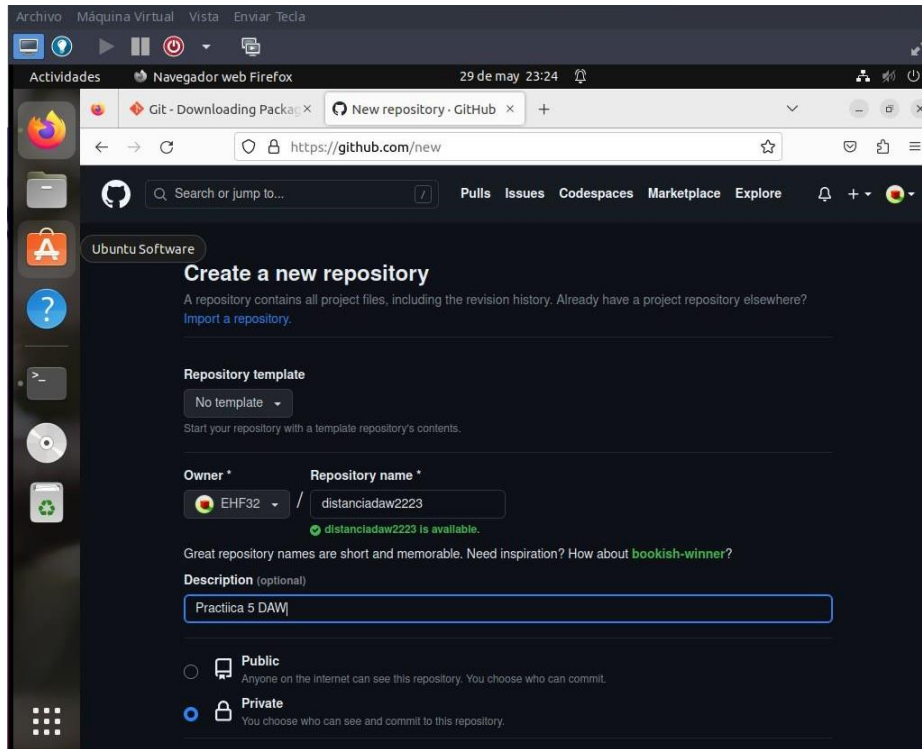


Actividad 2.3 En este apartado que se escogerá el fichero que hemos realizado en el apartado 1.2 y demostrar cómo funciona git en tal proyecto, por ejemplo, subiendo a git todo el proyecto completo y posteriormente es necesario modificar algo para que se detecten los cambios. Es necesario realizarlo con NetBeans. Hay que darse de alta en la url <https://github.com/> y crear un repositorio llamado `distanciadaw2324` para llevar los ficheros del proyecto php. Demostrarlo con pantallas.

Una vez creada la cuenta en github.com, en mi caso ya disponía de una, le damos a crear un repositorio:

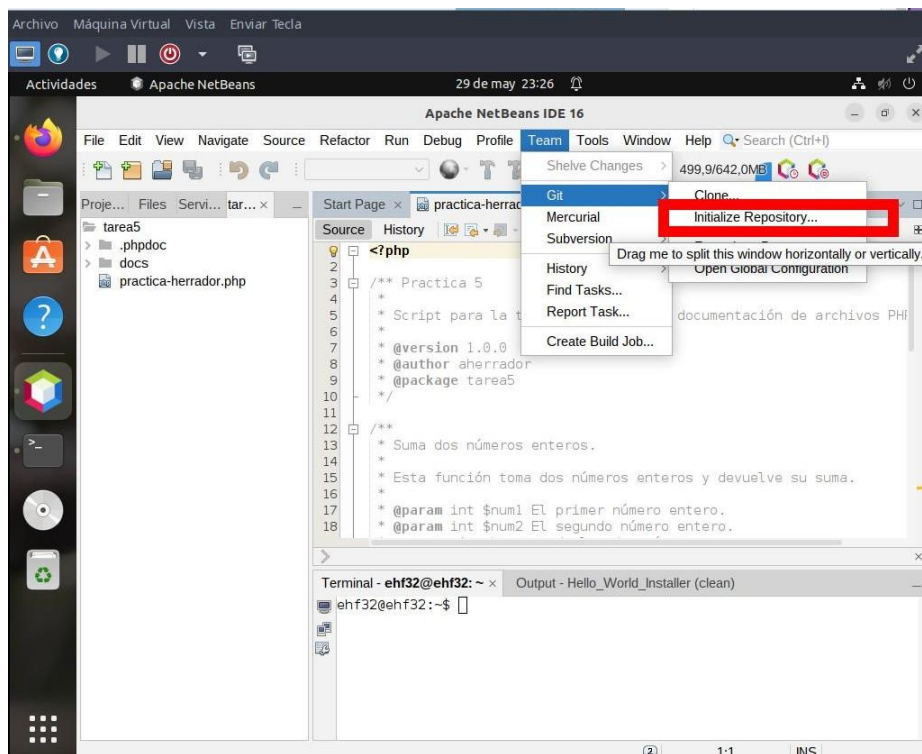


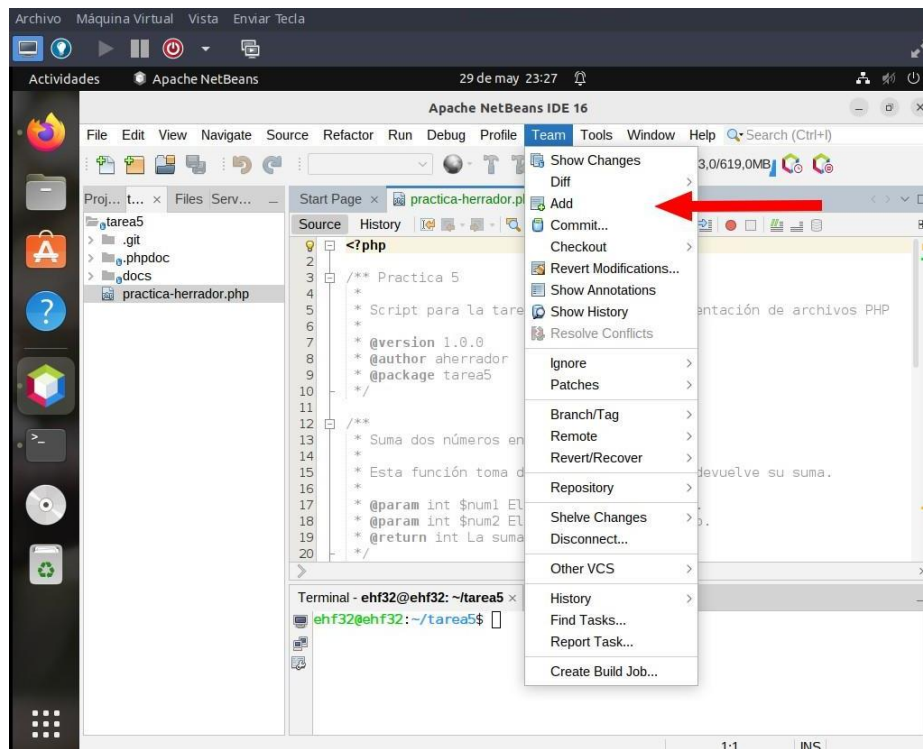
Le ponemos el nombre indicado:



Una vez creado, inicializamos el repositorio después de cargar nuestro proyecto php en netbeans.

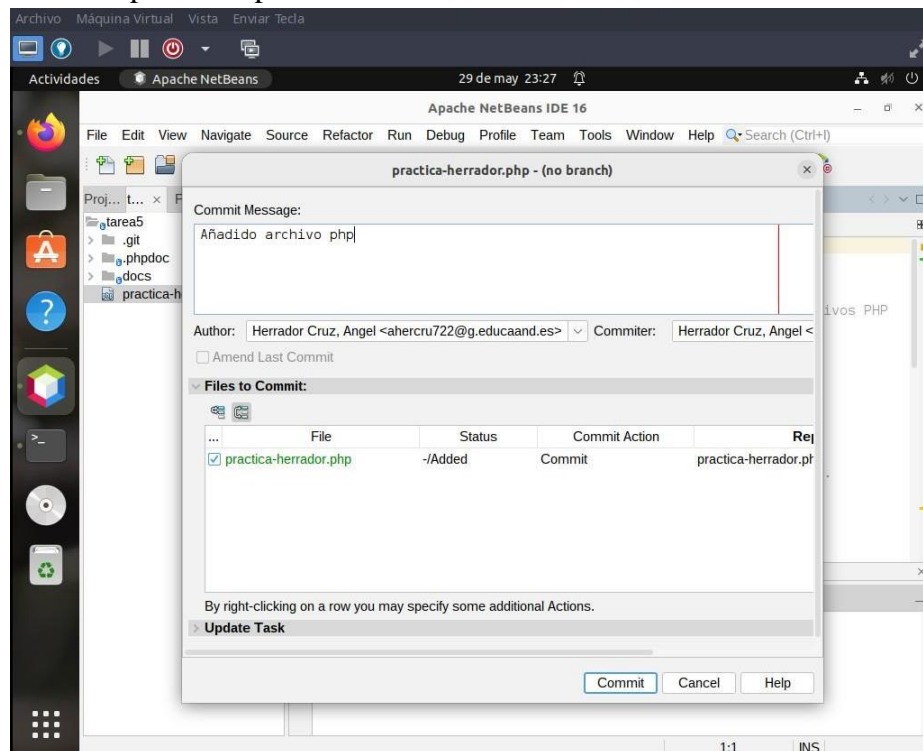
Añadimos el archivo:

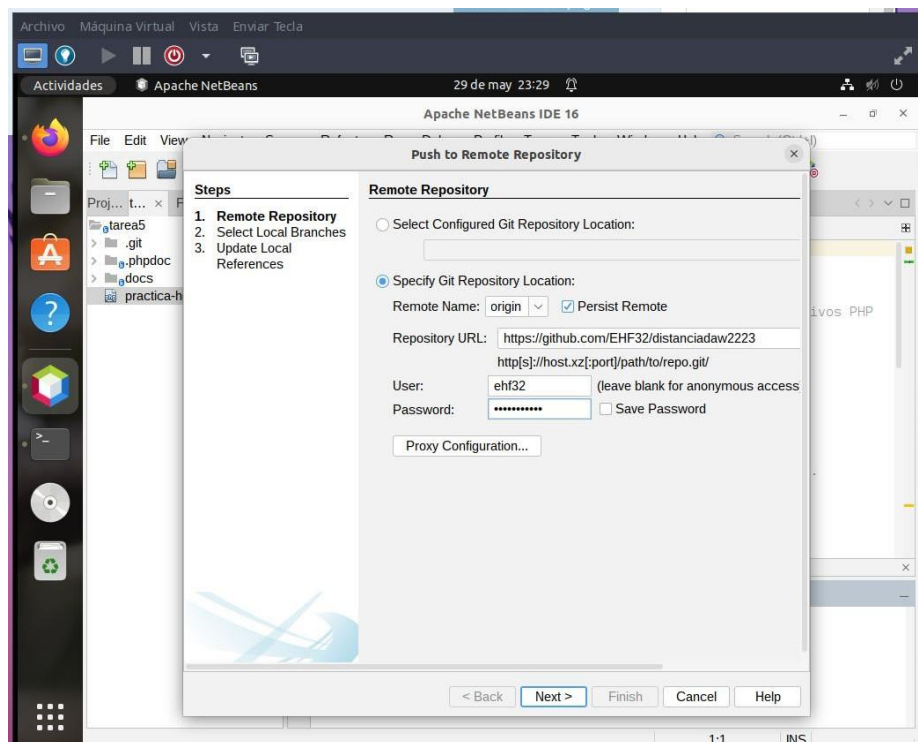
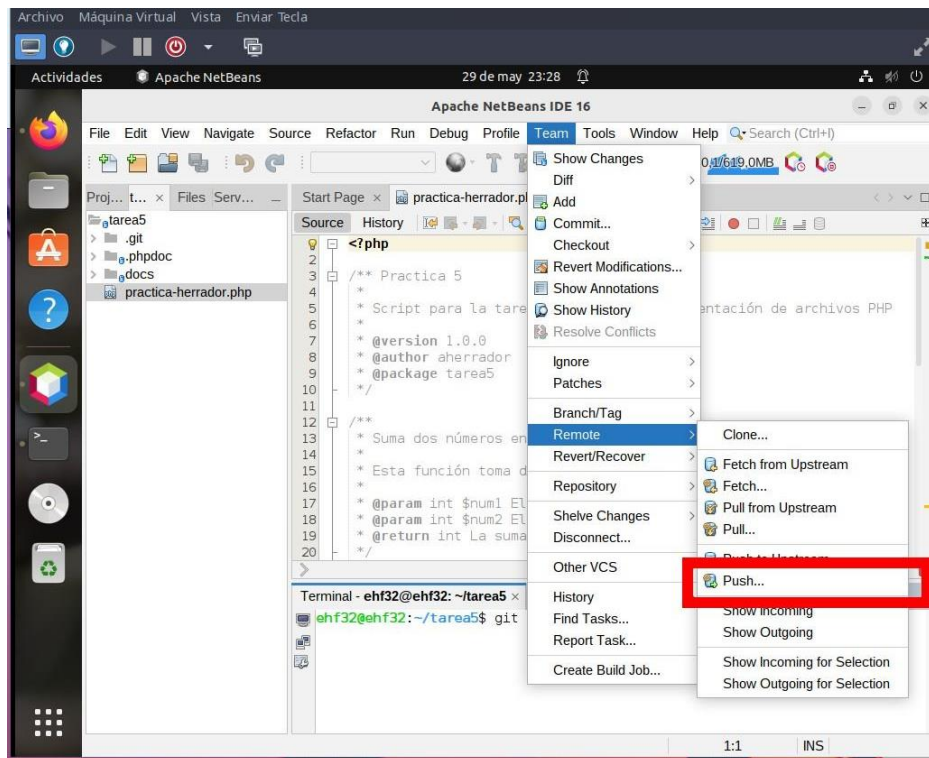




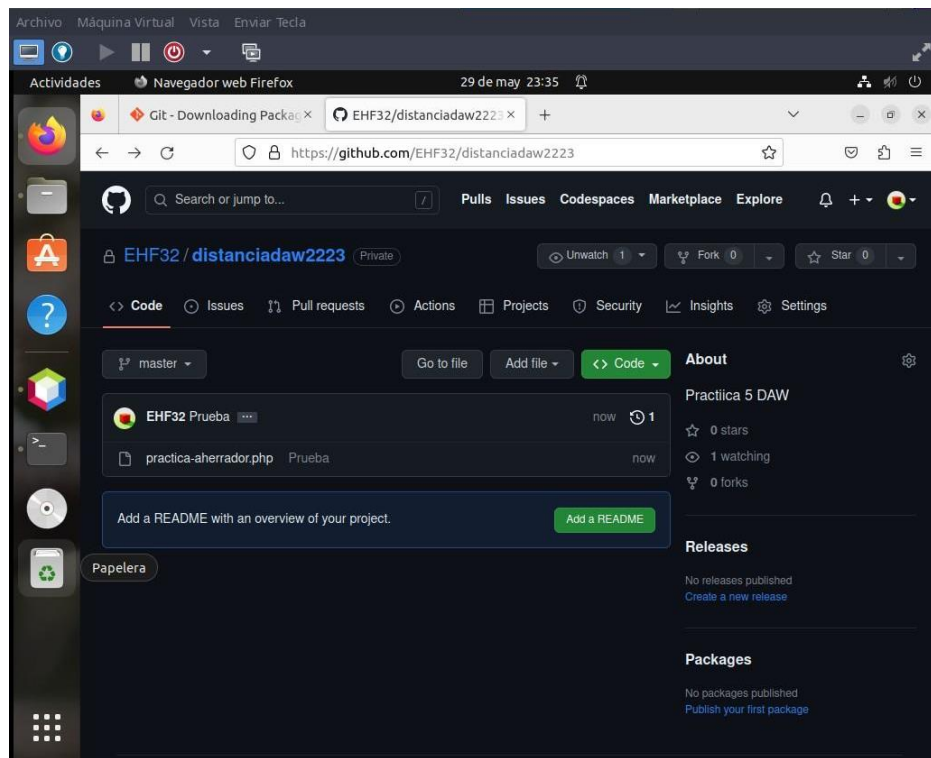
Y realizamos un commit, que registrará los cambios añadidos.

Finalmente hacemos push al repositorio remoto en GitHub:





Y ya podemos ver los cambios en el repositorio en GitHub.



Si realizáramos cambios sobre el archivo, nos lo detectaría:

