****

**Reto 1 UD1 Ejecución de código tras un servidor web**

**Objetivo**: entender la diferencia entre páginas estáticas y páginas generadas dinámicamente. en el funcionamiento de un servidor web.

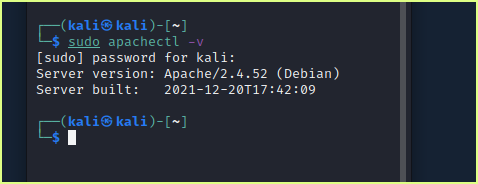
**Introducción**: antes de meternos con PHP vamos a ver la ejecución de código en cualquier lenguaje detrás de en un servidor web con la tecnología CGI, que, aunque no es recomendable en producción nos sirve como prueba de concepto para entender lo que vendrá después.

**Desarrollo**: en grupos de dos o tres alumnos, se trata de empezar a experimentar con un servidor web (Apache) mediante los siguientes pasos:

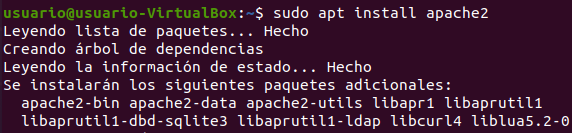
**\*\*\*\* \*\*\*\* \*\*\*\***

*Aprovechando que cada compañero tenemos sistemas operativos diferentes, todo el ejercicio lo hemos hecho tanto en el Sistema Operativo de Kali Linux como en Ubuntu.*

1. **Instalación del servidor web Apache de acuerdo a lo visto en despliegues.**

**S.O. kali Linux:** Verificamos que en el S.O kali linux ya lo tenemos instalado. 

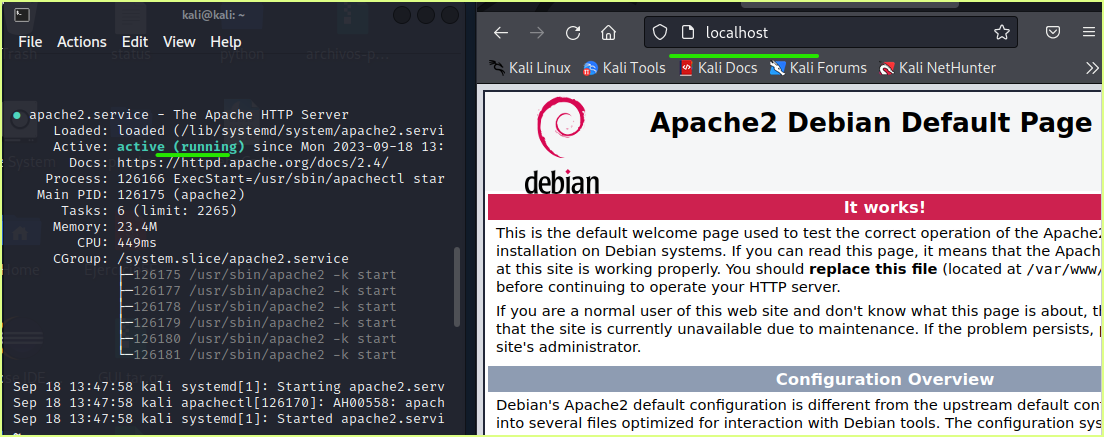
**S.O. Ubuntu 20.04:** Instalamos apache2 desde la terminal.



1. **Comprueba que el servidor funciona, localiza la página web que sirve por defecto y crea carpetas y añade nuevas páginas. Comprueba que funcionan los enlaces (link) relativos entre páginas web obtenidas a través del servidor: en el navegador no abres la página web desde el almacenamiento, si no a través de** [**http://localhost/carpeta/página**](http://localhost/carpeta/p%C3%A1gina) **… (o la IP en vez de localhost, o el nombre ...)**

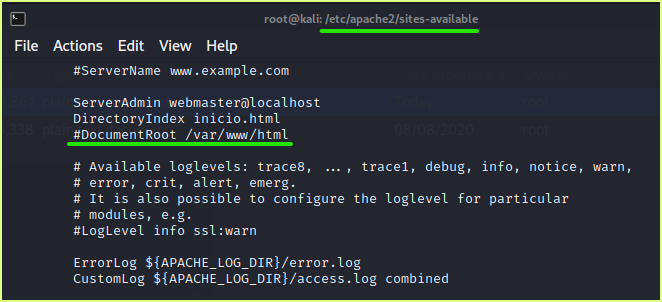
***Kali Linux:***

Comprobamos si apache está activo y probamos en el navegador:

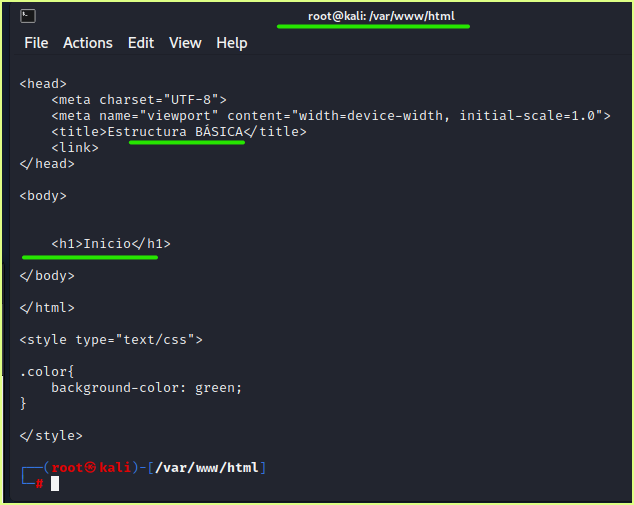


Modificación de la página inicial de apache:

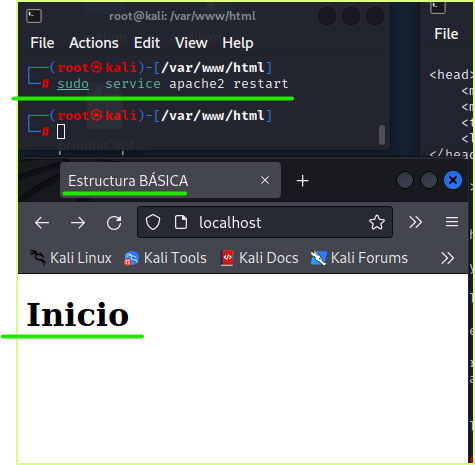
La página inicial de apache se encuentra en la siguiente ruta, en nuestro caso vamos a modificar la página por inicio.html.



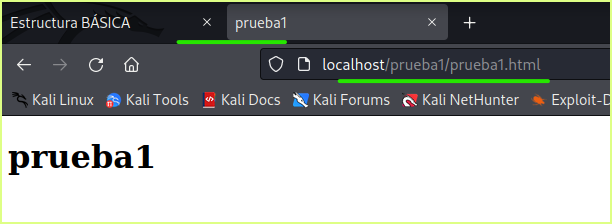
Creación del fichero inicio.html en la ruta /var/www/html, hacemos un html básico para ver los cambios:



Una vez guardado el html, reiniciamos el servidor apache para que tome los cambios y ejecutamos la página inicio.html:

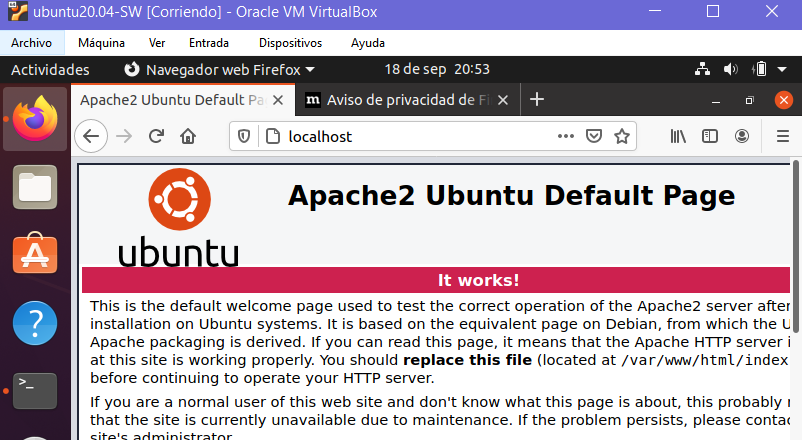


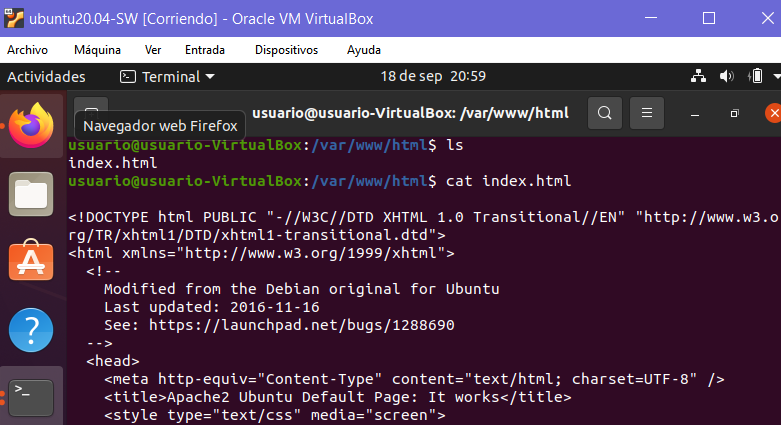
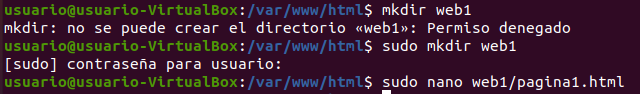
Creación de una nueva página en apache:

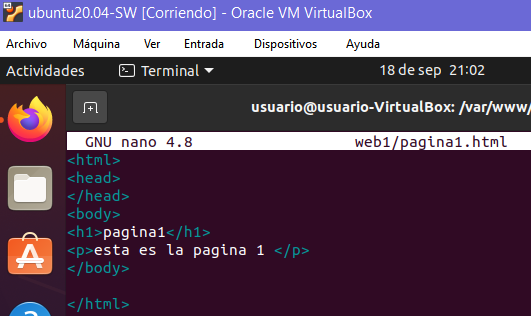


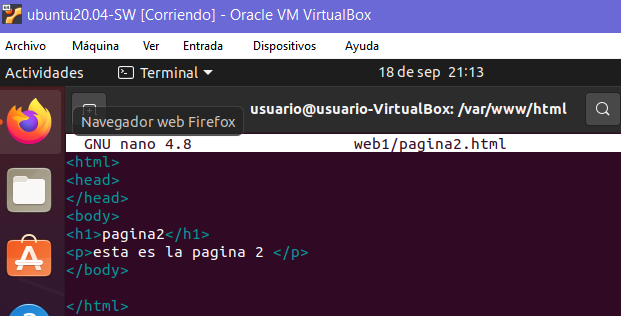
**Sistema operativo Ubuntu 20.04**

* Comprobamos que el servidor funciona.

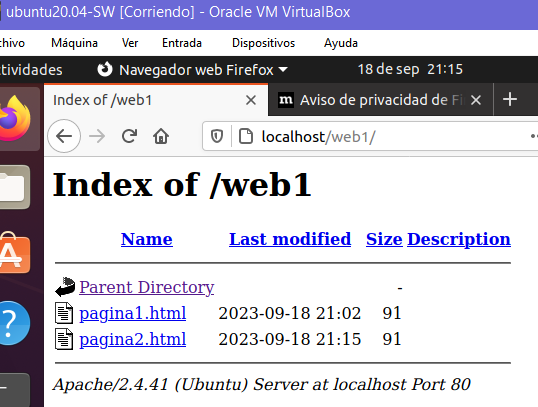


* Localizamos la página por defecto: 
* Creamos carpetas y añadimos páginas nuevas:





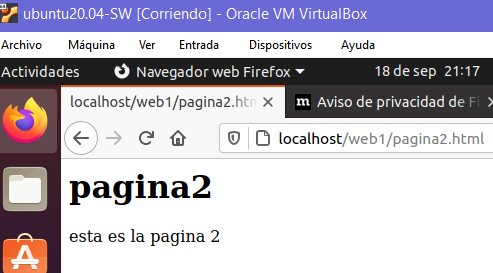
* Comprobamos que funcionan los enlaces relativos entre paginas: Miramos en el navegador el resultado de crear pagina1.html y pagina2.html en web1.



Página1.html: muestra el contenido de pagina1.html



Página2.html: muestra el contenido de pagina2.html

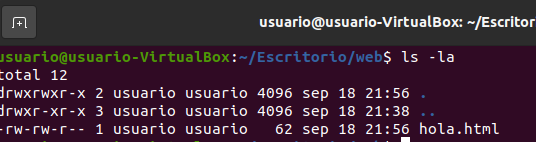


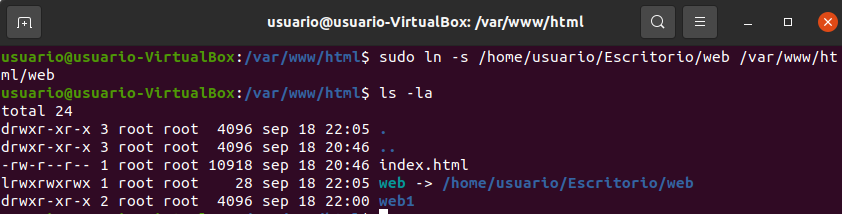
1. **Para evitar tener que editar las páginas como administrador tienes varias opciones:**

Al tener s.o. parecidos Kali Linux y Ubuntu con el uso del servidor apache, no vemos grandes diferencias. Lo cual haremos un mismo resultado para ambos sistemas operativos.

1. **Hacer un acceso directo (symlink) a otra zona donde tengas permisos, por ejemplo, una carpeta en tu home (ej: sudo ln -s /home/alumno/web /var/www/html/web)**

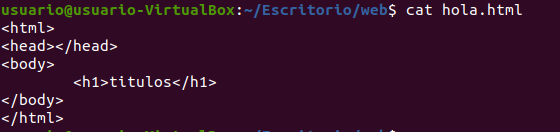
Creamos un enlace simbólico de una ruta donde se tenga permisos de usuario como /home/usuario/Escritorio/web/

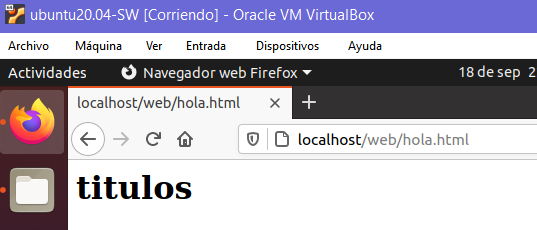




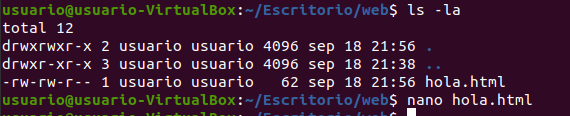
Esto hará que el enlace cree un acceso directo de la ruta origen /home/usuario/Escritorio/web/ a la ruta /var/www/html con el nombre de web.

A partir de aquí, podemos crear un archivo HTML de prueba en la carpeta web del escritorio para comprobar que se ha hecho el enlace simbólico correctamente. Una propiedad del enlace simbólico es que toda modificación hecha en var/www/html/web … se modifica también en la ruta de origen que es /home/usuario/Escritorio/web.





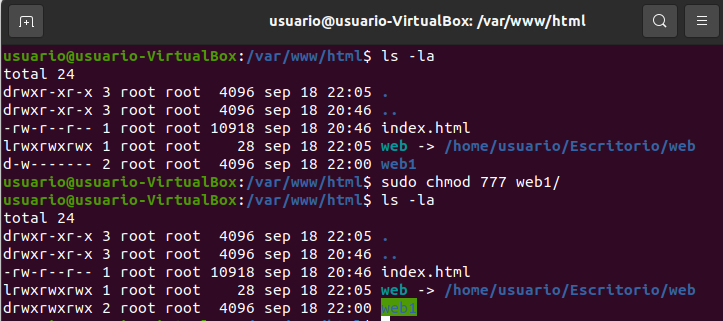
Otra facilidad es también que al tener un enlace a una carpeta con permiso de usuario no se necesita el uso del sudo para crear páginas o carpetas dentro de ella.



1. **Cambiar los permisos de la carpeta que vayas a utilizar en /var/www/html**

Se puede cambiar los permisos a la carpeta web1 para que se evite editar como administrador de la siguiente manera:

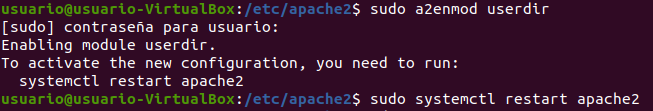
Con sudo chmod 777 web1, este comando dará permisos de lectura, escritura y ejecución al propietario, grupo u otros.



1. **Instalar el módulo de páginas de usuarios (lo veremos en despliegues)**

El módulo userdir.mod de apache2 nos va a permitir que cada usuario tenga un directorio público en su home (public\_html) donde guardará su página personal.

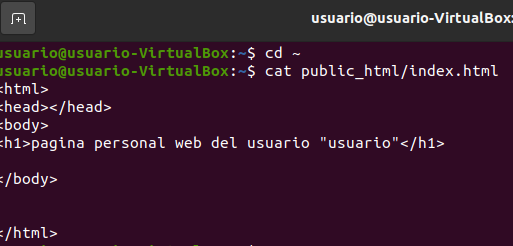
Lo primero que haremos es habilitar el módulo userdir con el comando Sudo a2ensite userdir y luego reiniciamos el servidor apache con systemctl restart apache2:



Una vez hecho esto nos situamos en home/usuario/ y creamos la carpeta public\_html donde creamos la página index.html es decir la página personal del usuario “usuario”.







Comprobamos que funciona en el navegador con la siguiente URL :

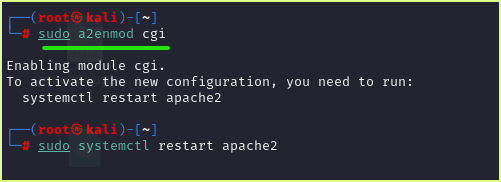
http://localhost/~usuario/index.html



1. **Activa el módulo cgi-bin como hemos visto en despliegues y muestra el resultado de acceder al script (previo permiso de ejecución) adjunto (con una de las tres versiones vale) accediendo al servidor desde un navegador. Accede desde varios navegadores y desde varias máquinas. ¿Cuál es el resultado? ¿Dónde guarda el número de visitas? ¿Qué pasa si quitas al script el echo de una línea en blanco debajo del Content/Type? ¿Y si quitas el Content/Type?**

cgi-bin -> es un programa que se ejecuta en el servidor por petición del navegador de un usuario.

* Activamos el módulo cgi-bin y reiniciamos apache2.

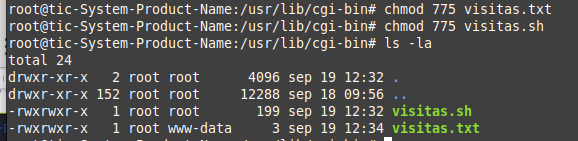


* Una vez activado, creamos la carpeta cgi-bin con permisos y dentro copiamos los scripts adjuntos, se muestra captura del acceso al script:

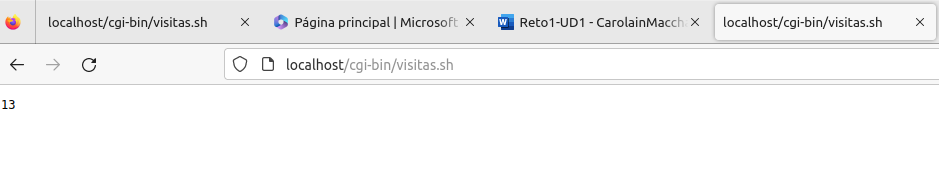


* Damos los permisos necesarios a visitas.sh y visitas.txt 775 y ponemos como propietario del fichero visitas.txt a www-data para que apache pueda tener acceso a él. Con los comandos:

#chmod 775 visitas.txt / #chmod 775 visitas.sh

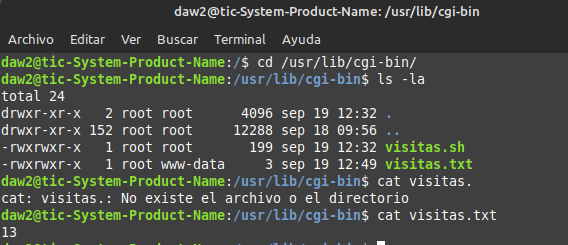


* Accedemos al servidor desde un navegador con la ruta htttp://localhost/cgi-bin/visitas.sh y nos muestra el número 13, que son las visitas que hemos hecho a la web.



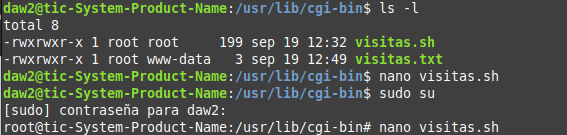
* ¿Dónde guarda el número de vistas?

Lo guarda en visitas.txt como se visualiza en la captura:

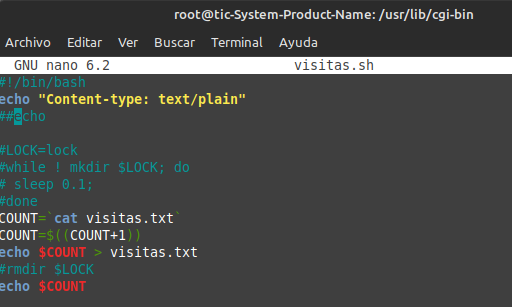


* Quitando al script el echo de una línea en blanco:

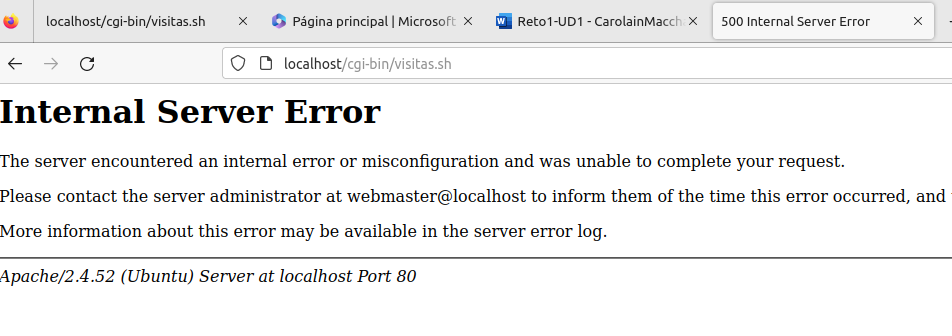
Editamos visitas.sh con el comando nano visitas.sh1.



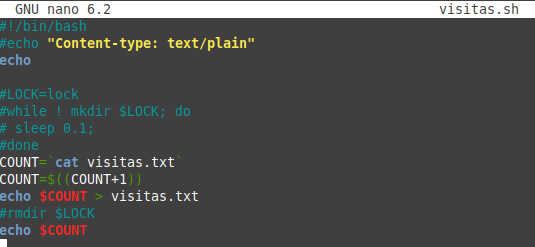
Comentamos el echo de una línea en blanco con #:



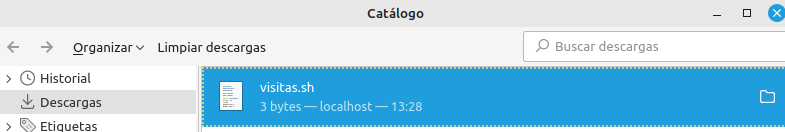
Como resultado nos da el error 500 que quiere decir que el servidor encontró una condición inesperada que le impidió cumplir con la solicitud”. Puede tener diferentes orígenes como que la base de datos se haya corrompido o por un error en el archivo que es el que hicimos de comentar el echo en blanco.

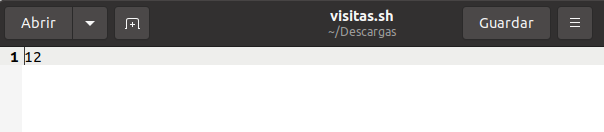


* Quitando el content/type cuando ingresamos al servidor desde un navegador nos aparece una descarga del fichero visitas.sh en vez de mostrarlo dentro de la página web.

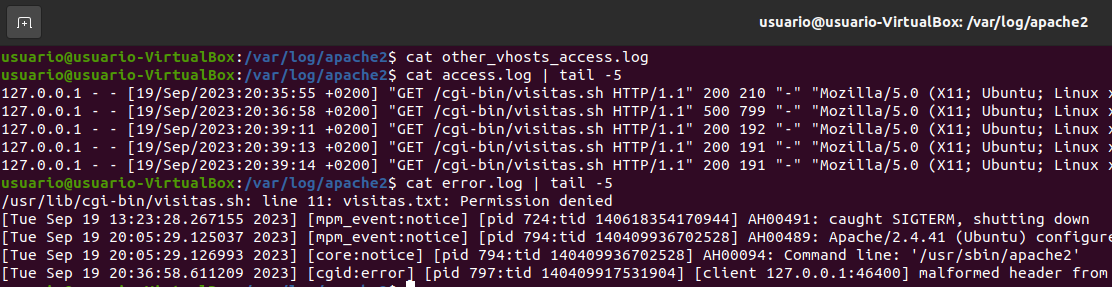


Este archivo mostrará el número de visitas de la web:



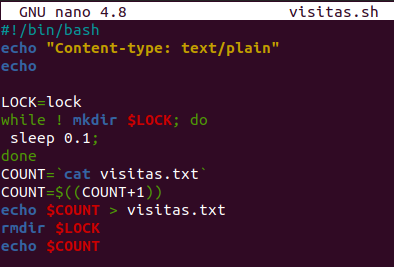


* Otro dato de interés es que en var/log/apache2/ se ve el historial de peticiones:

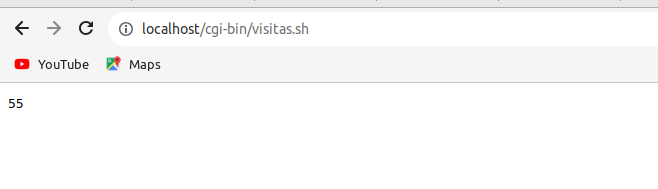


1. **Optativa: peticiones masivas con ab (ApacheBench, comando ab que se instala junto con apache, ej ab -n 1000 -c 10** [**http://localhost/cgi-bin/visitas.sh**](http://localhost/cgi-bin/visitas.sh) **→ realiza mil peticiones de 10 en 10 a la dirección dada) Analiza la diferencia entre las distintas versiones del script.**

**VERSIÓN 1(visitas.sh):**



Actualmente estamos llamando a nuestro localhost y se va actualizando cada vez que llamamos a la url, en la siguiente imagen podemos ver el número.



Lanzamos el comando que

ab -n 1000 -c10 <http://localhost/cgi-bin/visitas.sh>

*ab= > apache benchmarck*

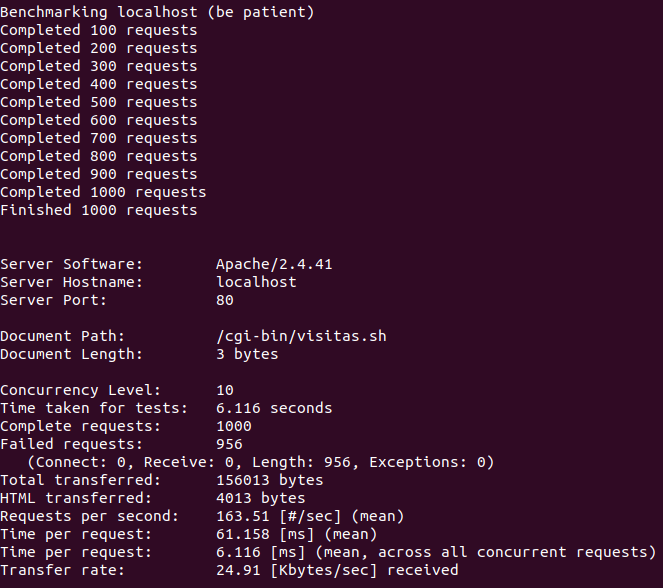
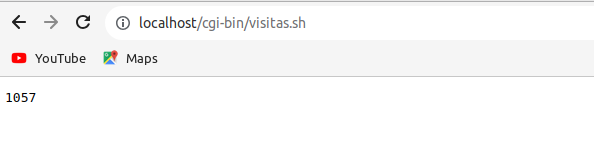
*-n 1000 => es el número de solicitudes que se realizarán al servidor*

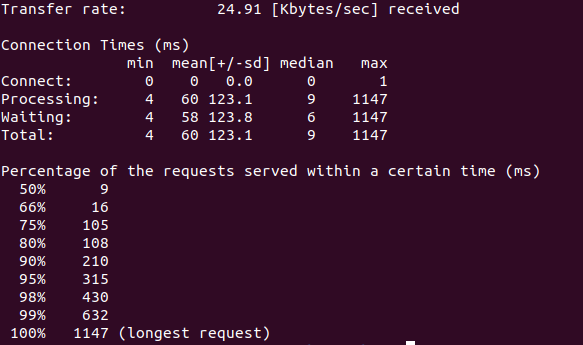
*-c 10 => significa que se realizarán 10 solicitudes simultaneas, simulando 10 usuarios.*

[*http://localhost/cgi-bin/visitas.sh*](http://localhost/cgi-bin/visitas.sh) *=> ruta donde se encuentra el script*

Que llamará a la url 1000 veces y adjuntamos el resultado:

Si actualizamos la url visualizamos el incremento del contador, más de 1000.

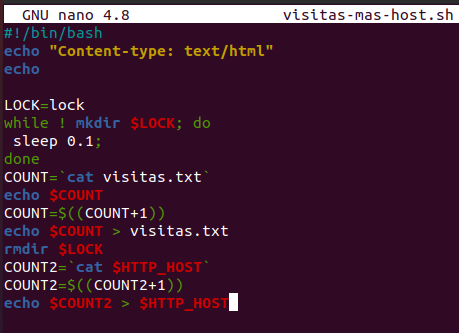


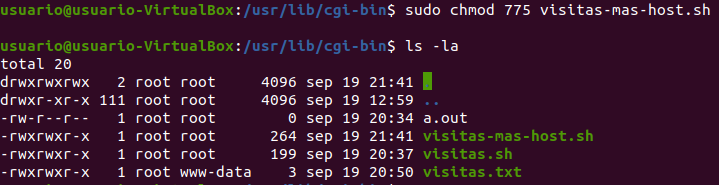


En el gráfico se observa la prueba de rendimientos de apache cuando enviamos peticiones simultaneas a la vez, en este caso, las 1000 peticiones enviadas.

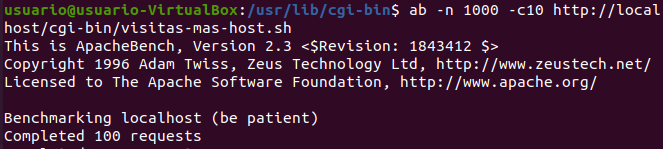
**VERSIÓN 2(visitas-mas-host.sh):**

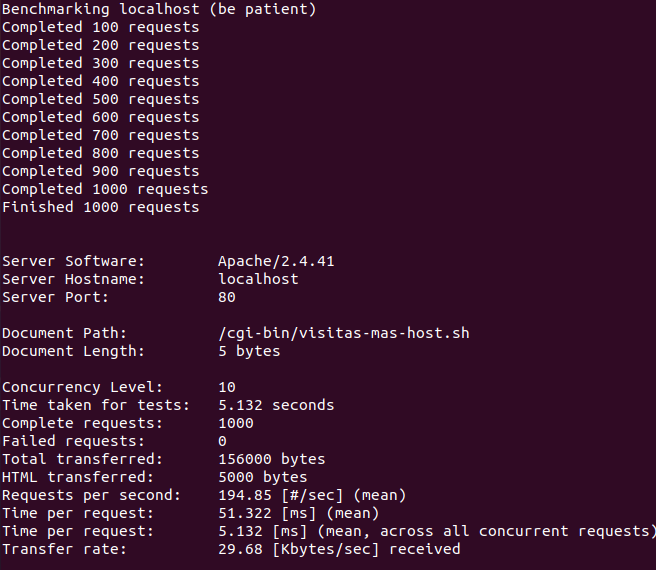
Realizamos la creación del fichero que contendrá al nuevo script con los permisos necesarios.

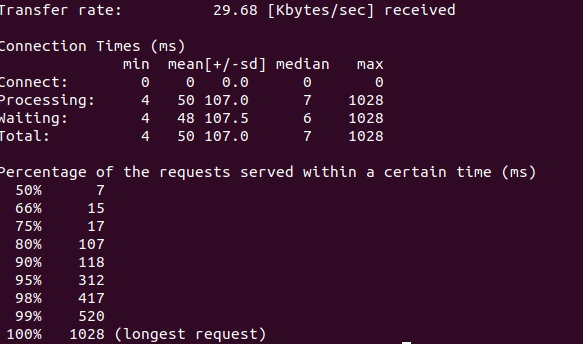




Lanzamos el comando ab -n 1000 -c10 <http://localhost/cgi-bin/visitas-mas-host.sh>:

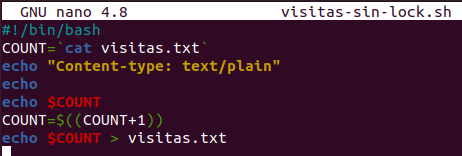


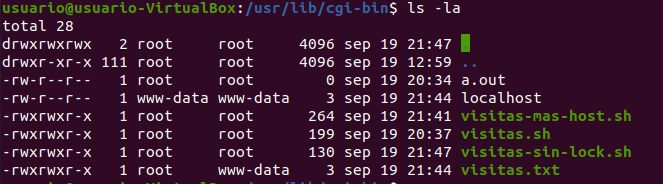




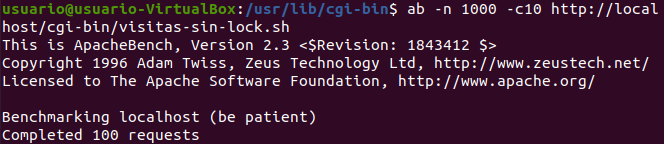
**VERSIÓN 3(visitas-sin-lock):**

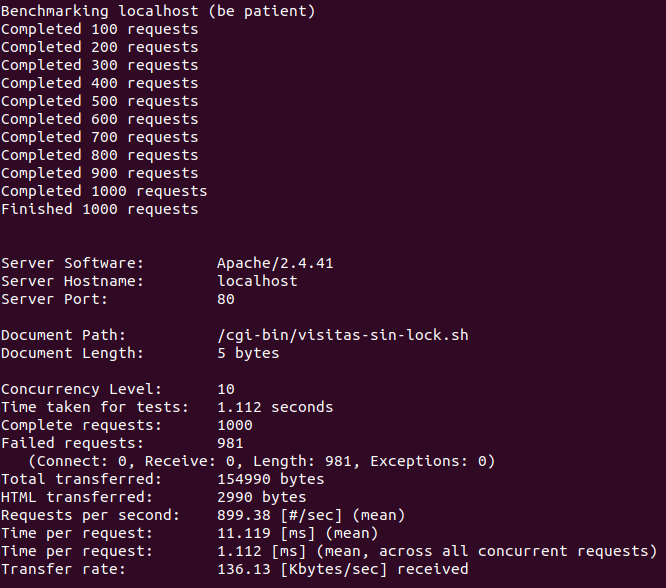
Realizamos la creación del fichero que contendrá al nuevo script con los permisos necesarios.

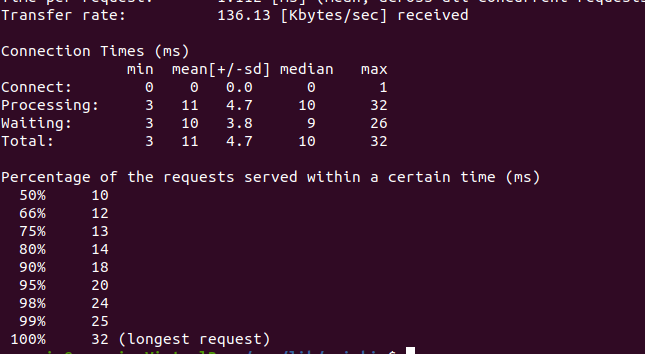




Lanzamos el comando ab -n 1000 -c10 [http://localhost/cgi-bin/visitas-sin-lock.sh](http://localhost/cgi-bin/visitas-mas-host.sh)







|  |  |  |  |
| --- | --- | --- | --- |
|  | **VERSIÓN 1(visitas.sh)** | **VERSIÓN 2(visitas-mas-host.sh)** | **VERSIÓN 3(visitas-sin-lock.sh)** |
| **TTFT** | 6.116 seconds | 5.132 seconds | 1.112 seconds |
| **FR** | 956 | 0 | 981 |
| **RPS(promedio)** | 6.116 | 5.132 | 1.112 |
| **TR** | 24.91 | 29.68 | 136.13 |
| **%(100)** | 1147 seconds | 1028 seconds | 32 seconds |

Haciendo una tabla comparativa sobre las tres versiones encontramos cambios en estos parámetros:

* **Time taken for tests:** El tiempo total que tomó realizar todas las pruebas.
* **Failed requests:** El número de solicitudes que fallaron durante la prueba.
* divide en tres categorías: Connect , Receive y Length (longitud).
* **Requests per second:** La tasa de solicitudes por segundo. solicitudes por segundo en promedio.
* **Time per request (mean, across all concurrent requests):** El tiempo promedio por solicitud, considerando todas las solicitudes simultáneas.
* **Transfer rate:** La velocidad de transferencia de datos.
* **Percentage of the requests served within a certain time (ms):** Muestra el porcentaje de solicitudes que se sirvieron dentro de ciertos tiempos.
* 100%: Indica el tiempo máximo que tomó procesar una solicitud.

- Como conclusión:

El tiempo total de la prueba es mejor la versión 3 que las demás puede ser debido a que el script es más sencillo que los demás y no lleva el LOCK que aumenta el tiempo de ejecución del script.

Las peticiones erróneas en diferencia son mejor la versión 2 que tiene 0 peticiones fallidas. Las demás fallan por el tema de longitud.

La tasa de solicitudes por segundo tarda menos la versión 3 seguramente por ser un código más sencillo y de menor tamaño de bytes, así como tiempo promedio por solicitud.

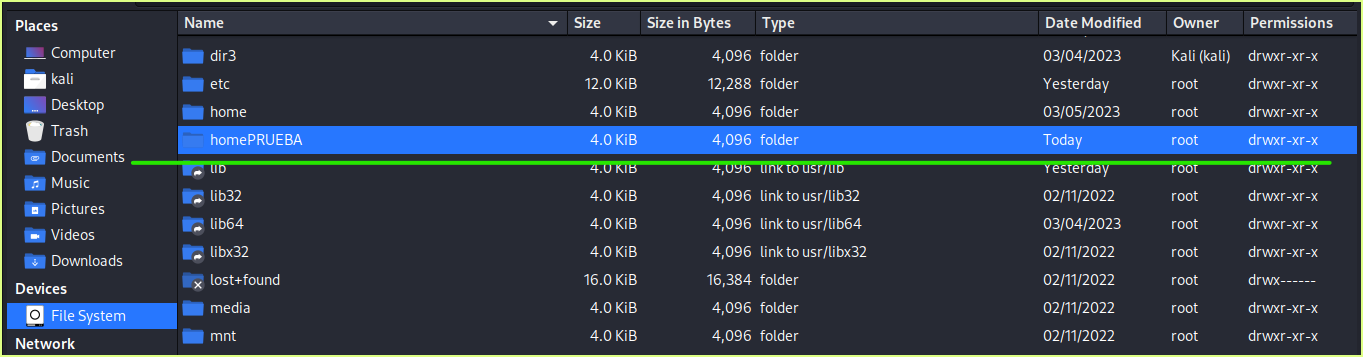
Opinamos que la mejor versión es la que tarda menos en ejecutar las peticiones que es la versión 3(visitas-sin-lock.ssh). además, la velocidad de transferencia es mucho mejor tiene 136 segundos y es mayor a las demás.

No obstante, puede implicar complicaciones ya que no tiene el uso del LOCK que bloquea el acceso a la misma vez de peticiones de varios hosts simultáneamente para evitar bloqueos o fallos en la ejecución. Que cada petición se realice correctamente un tras de otra.

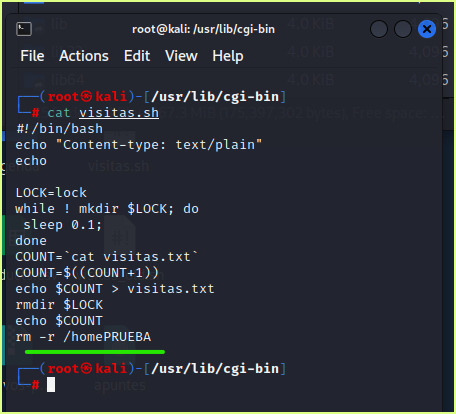
1. **Cambia el script para hacer algo malicioso como borrar el home de un usuario ¿funciona? ¿por qué?**

*No funciona.*

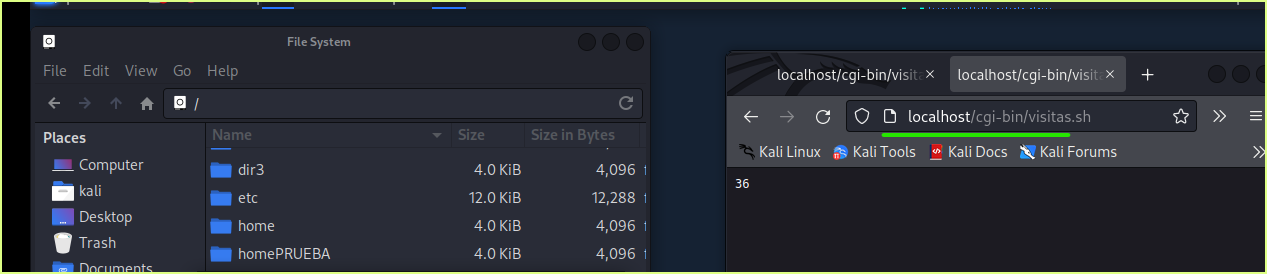
*He creado una carpeta homePRUEBA al mismo nivel que la carpeta home, simulando los mismos permisos que la carpeta home y así no realizar pruebas de test con mi entorno real.*



*Añadimos al script visitas.sh la línea borrar la carpeta /homePRUEBA para que proceda a borrar la carpeta homePRUEBA.*

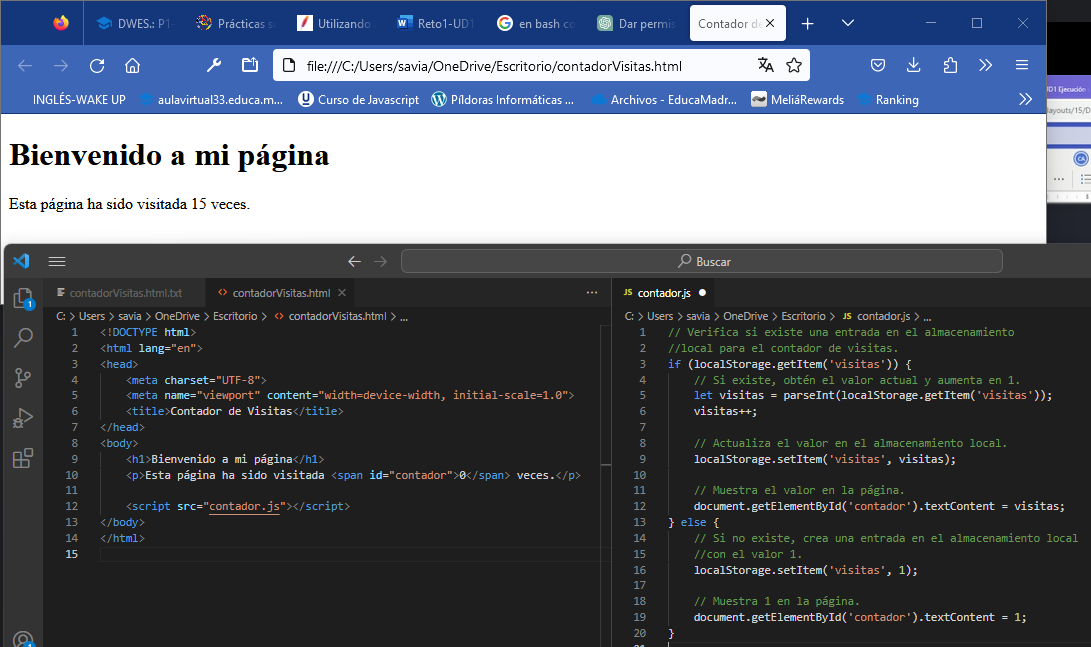


*Actualizamos la página o llamamos a la url y visualizamos que la carpeta homePRUEBA no se ha borrado.*



La razón por la que no funciona es porque homePRUEBA tiene permisos 750, es decir: el usuario de rwx, el grupo de r-x y otros, como www-data, no tienen ningún permiso.

Entonces el usuario/grupo www-data no puede modificar ni acceder a la carpeta, como no es propietario ni está en su mismo grupo, no puede leer, escribir ni ejecutar.

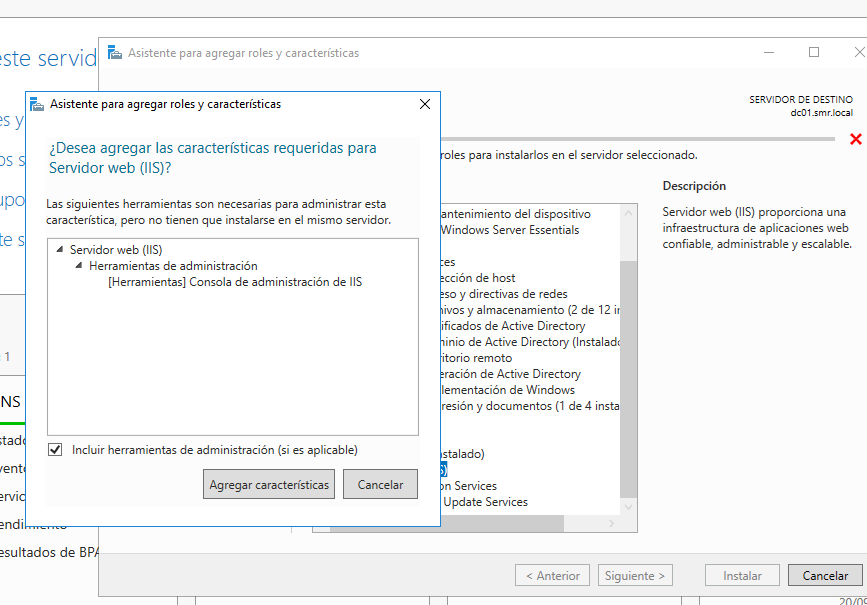
1. **¿Puedes hacer un contador de visitas en JavaScript para Frontend? ¿Funcionaría igual que el anterior en el backend? Justifica la respuesta.**  
     
   Creo archivo HTML y JS (docs adjuntos): 

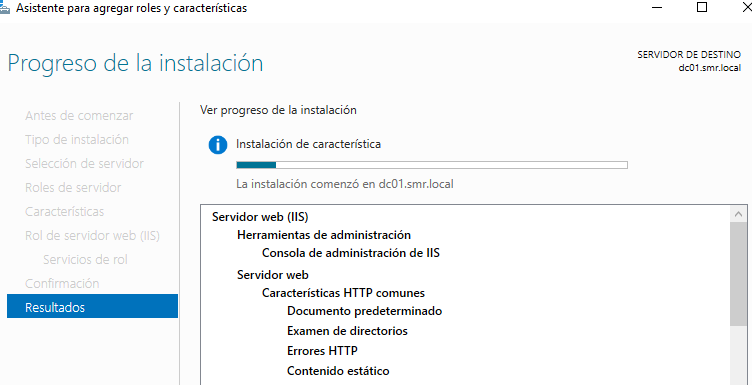
No funcionaría en el lado backend, ya que, este contador de visitas es básico, lo que significa que solo contará las visitas en el dispositivo de cada usuario y no será persistente entre sesiones o dispositivos, es decir, el contador de visitas funciona pero solo a nivel local.

Para un contador de visitas más avanzado y persistente en el lado servidor normalmente se utiliza una base de datos en el lado del servidor. Se necesitaría almacenar y actualiza el contador en una base de datos.

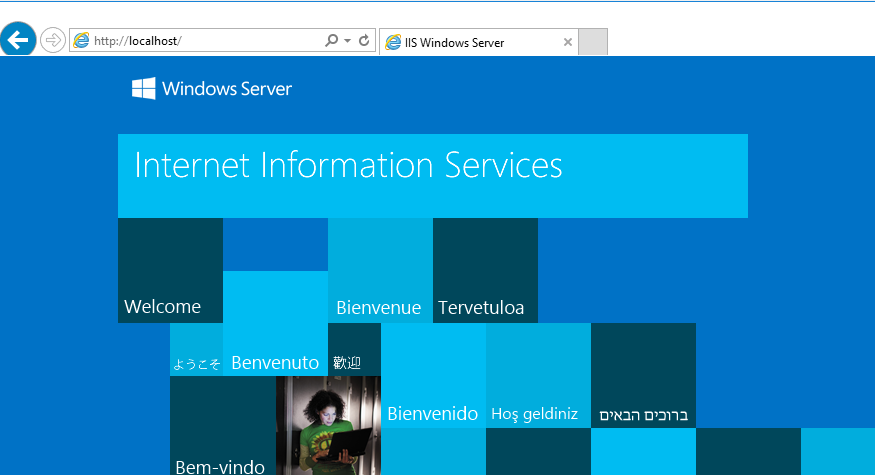
**Adicional: Extensión Instalación de IIS en windows server 2016**

Instalamos el servidor web IIS en window Server 2016

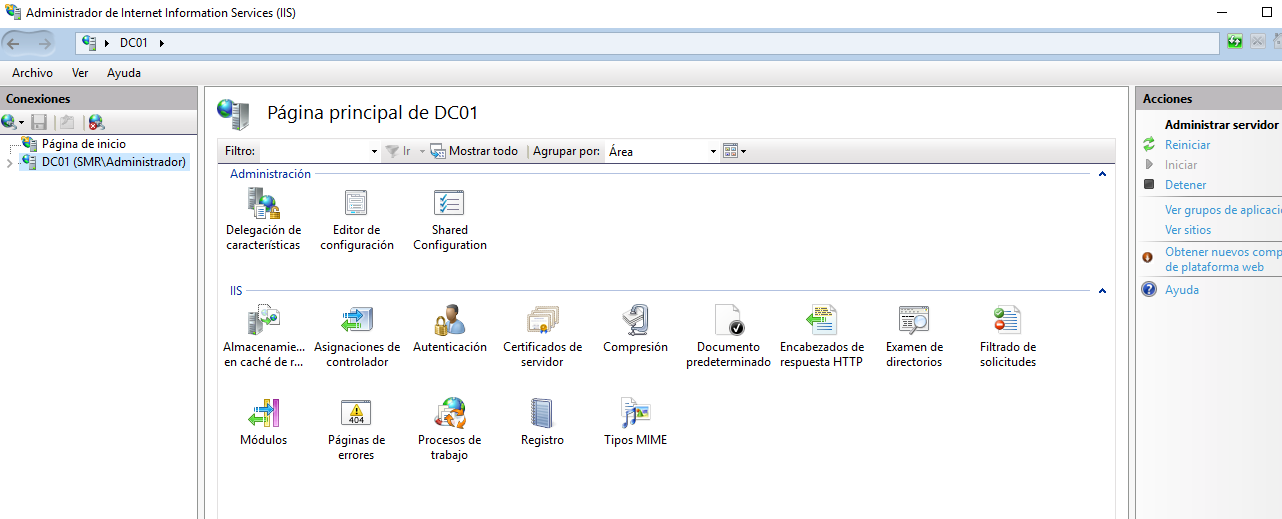




Comprobamos que el servidor web se ha creado correctamente, comprobando en el navegador que nos aparece la página por defecto de IIS.



Este es el administrador de configuración de IIS.



Hemos creado una página nueva en C:\inetpub\wwwroot llamada página nueva y hemos comprobado en el navegador que funciona:

