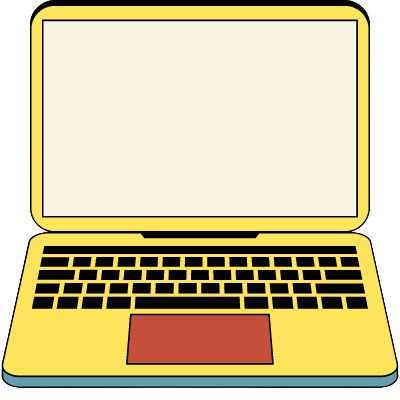
Instalación y configuración básica de un Servidor Web Apache

**Christian Millán Soria**





- Curso: 2023-2024

- Clase: 2º DAW Tarde

- Profesor: David Hormigo Ramírez

- Módulo: Despliegue de Aplicaciones Web

- Fecha: 03/10/2023

**1. Instalación del servidor Apache** 3

**1.1 Conexión** 3

**1.2. Actualización e instalación de Apache** 8

**2. Configuración del servidor web** 10

**2.1. Puerto 8090** 10

**3. Edición de la página web** 12

**4. Archivo default personalizado** 13

**4.1. Configuración del archivo “.conf”** 14

**4.2. Habilitar y deshabilitar sitios** 15

**4.3. Directorio y archivo propios** 15

**5. Postman** 16

**5.1. Método “GET”** 17

**1. Instalación del servidor Apache**

Esta es una guía de instalación, configuración y manejo de un [**Servidor Web Apache**🔗](https://httpd.apache.org/) en nuestra instancia en AWS llamada “WebServer” (pasos para la creación y configuración de la instancia en [**este documento**🔗](https://github.com/cmilsor245/daweb/blob/main/01-servidor_ubuntu_aws/Instalación%20de%20un%20servidor%20Ubuntu%20en%20AWS.pdf)).

**1.1 Conexión**

Cómo vimos en la guía anterior, existen dos formas principales/comunes de realizar una conexión hacia nuestra instancia en AWS:

- Ejecutando el comando correspondiente desde el terminal de comandos de nuestra máquina local.

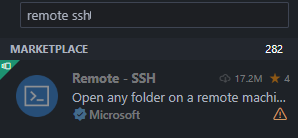
- Mediante la utilización de un entorno gráfico más preparado y personalizable.

En el caso de esta práctica realizaremos la conexión mediante un método adicional: utilizando una extensión de VS Code.

La razón de esto se debe a que necesitaremos editar archivos del propio servidor que vamos a instalar, lo que puede hacerse tedioso o incómodo si se realiza desde el propio terminal del servidor, utilizando editores como “[**nano**](https://en.wikipedia.org/wiki/GNU_nano)”[🔗](https://en.wikipedia.org/wiki/GNU_nano) o “**[vim](https://en.wikipedia.org/wiki/Vim_(text_editor))**”[🔗](https://en.wikipedia.org/wiki/Vim_(text_editor)). Mediante la utilización de VS Code como entorno de trabajo para realizar esta tarea podremos editar dichos archivos de una forma muy visual y cómoda, como si estuviésemos trabajando de normal en el programa.

Vamos a comenzar con la conexión bajo el entorno de Visual Studio Code.

Abrimos el IDE y nos dirigimos a la pestaña de extensiones (**Ctrl + Shift + X**). Una vez aquí ingresamos los términos “**remote ssh**” en el buscador. La extensión que necesitamos instalar es la llamada “[**Remote - SSH**](https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-ssh)”[🔗](https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-ssh).

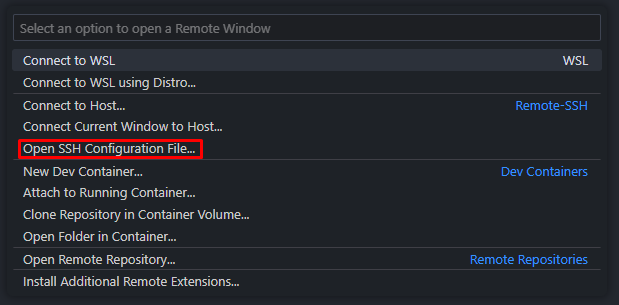


\* Si aparece un icono de advertencia, seguramente se deba a que el IDE ya tiene disponible una opción similar de forma nativa.

Una vez instalada, aparecerá un nuevo icono en la esquina inferior izquierda del IDE (si no se tiene desactivada la barra de estado del mismo).



Hacemos clic en él y se abre un cuadro, donde seleccionaremos la opción “**Open SSH Configuration File…**”.



Esto debería abrir un archivo llamado “**config**”, situado en la ruta “**%UserProfile%\.ssh\config**”, pero en caso de no hacerlo, nos tendríamos que dirigir a la ruta ““**%UserProfile%**” (que hace referencia a la **carpeta personal del usuario de Windows**) y crear el directorio “**.ssh**”, junto con un archivo en su interior llamado “**config**”, sin extensión alguna.

En Linux (al menos, en distros basadas en Ubuntu), la ruta sería “**~/.shh/config**”.

Una vez creado el directorio en la ruta correcta y su archivo, al hacer clic en la opción “Open SSH Configuration File…” se abrirá dicho archivo.

Este fichero será el encargado de establecer los hosts a los que nos queremos conectar y cada una de las dependencias necesarias para ello.

La sintaxis que este archivo requiere es la siguiente:

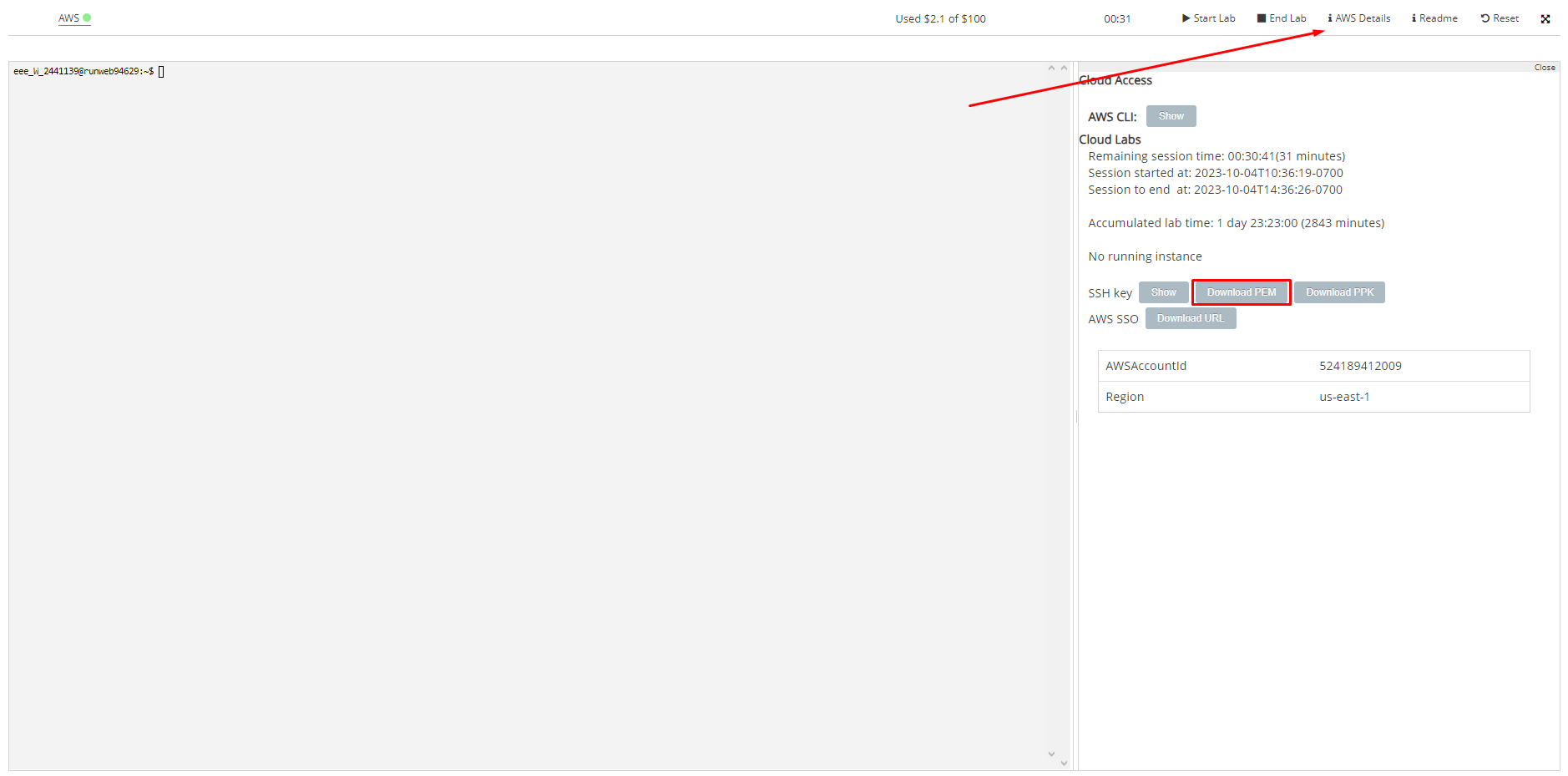
- “**Host**”: se refiere al nombre que tendrá y que el IDE mostrará para la nueva conexión que vamos a crear.

- “**HostName**”: dirección IP o DNS con el que accederemos al servidor en concreto para conectarnos por SSH, como ya hicimos en la primera práctica.

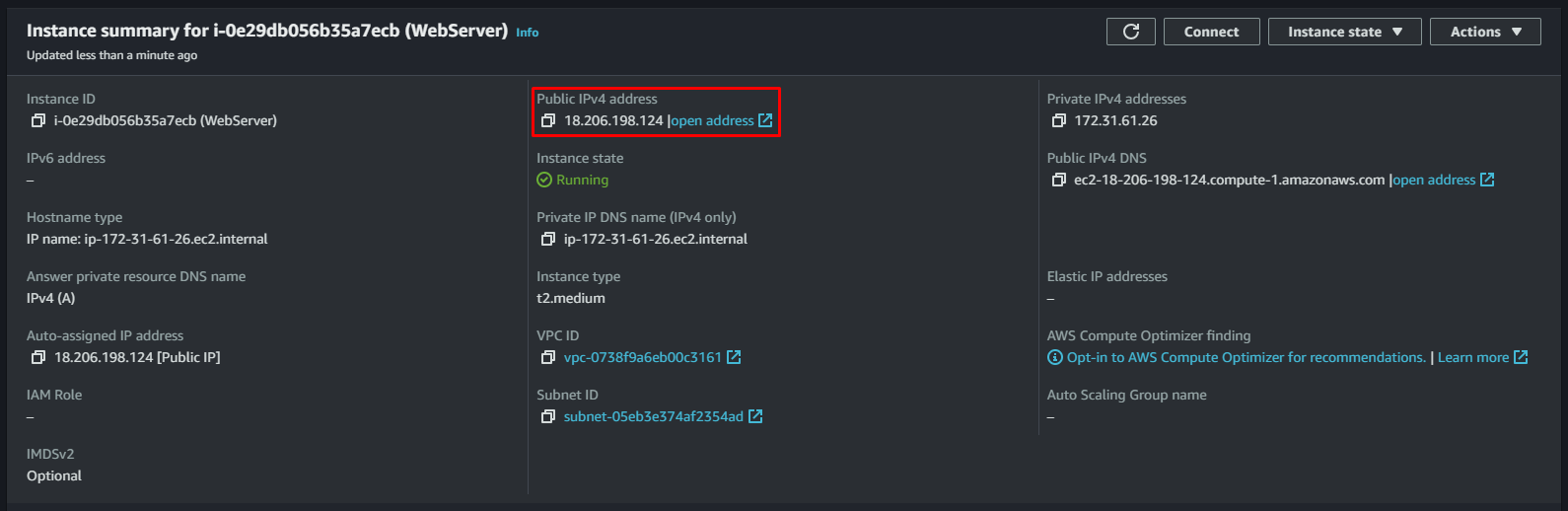
- “**User**”: usuario con el que se realizará la conexión.

- “**IdentityFile**”: ruta hacia el archivo de vockey (par de claves) que autentificará la conexión.

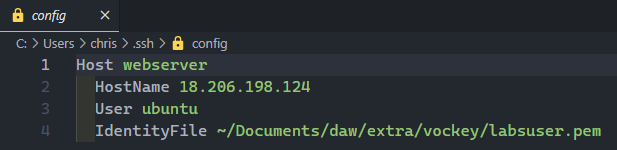
Dado que AWS cambia las credenciales de nuestras instancias y nuestro par de claves cada vez que el laboratorio de AWS se apaga y se vuelve a encender, debemos descargar el vockey de nuevo (esta vez el archivo con extensión “**.pem**”), el cual se encontraba en el workbench de AWS.



La dirección IP también será necesaria, ya que esta ha cambiado desde la última vez. La dirección IP se obtiene desde el menú de instancias de EC2, entrando al resumen de la instancia en concreto.



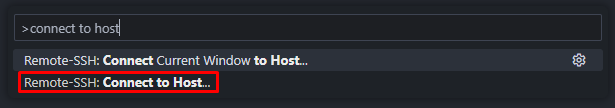
Una vez contemos con todo esto, lo siguiente es rellenar el archivo con las dependencias nombradas anteriormente. Debe quedar con un resultado final similar al siguiente:



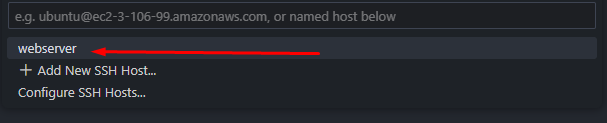
Debido a que la clave vockey lo guardo en un directorio dentro de la carpeta “Documents”, abrevio el acceso al directorio personal de mi usuario en Windows con “~”.

Una vez configurado esto, utilizamos el comando “**Ctrl + Shift + P**” para acceder a la paleta de comandos.

Aquí escribimos “**connect to host**” y elegimos la opción “**Remote-SSH: Connect to Host…**”.

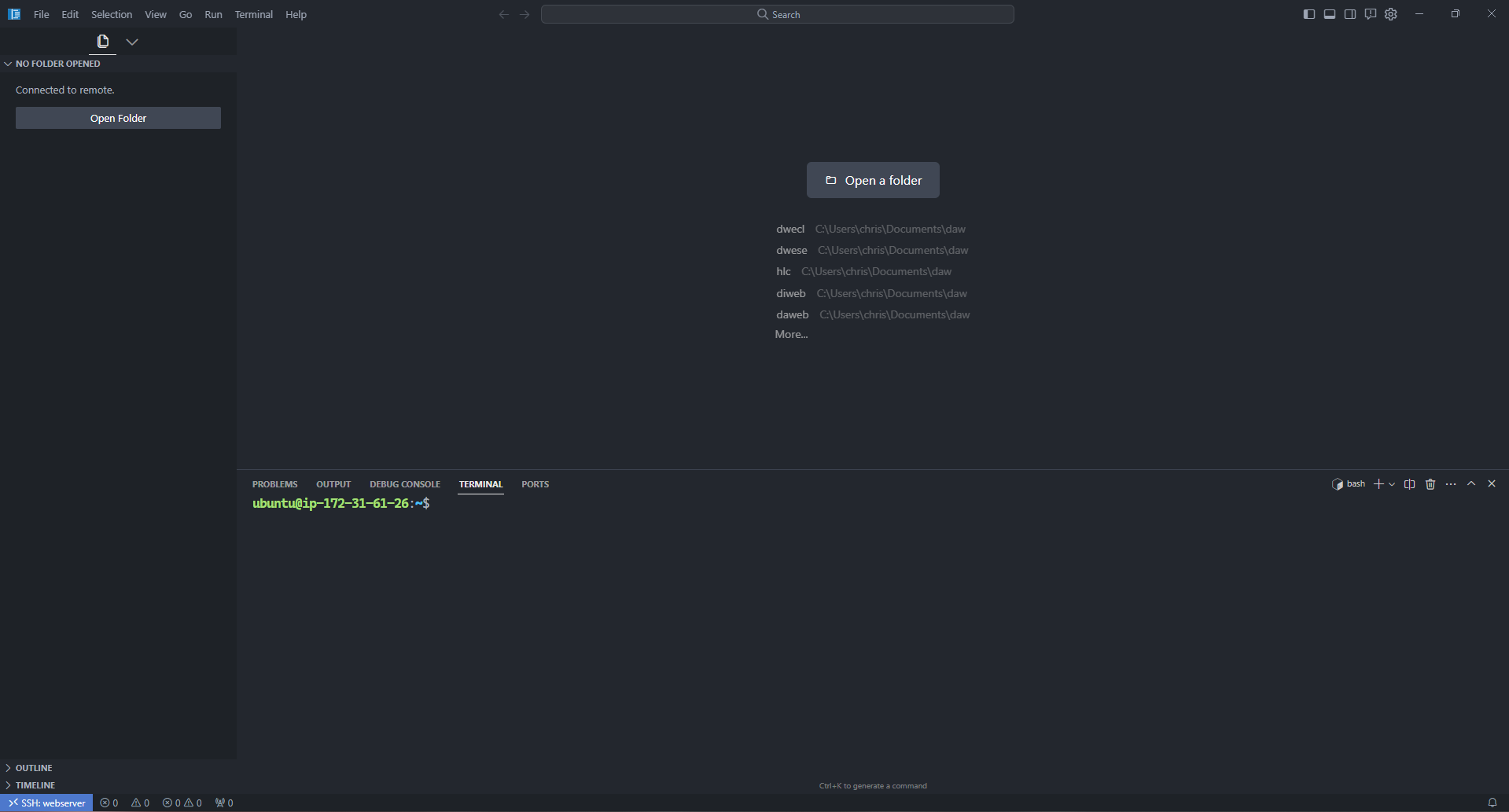


Se nos vuelve a abrir otro cuadro donde seleccionaremos la conexión que hemos creado anteriormente desde el archivo “config”.



Seleccionamos dicha conexión.

Si se han realizado los pasos correctamente, el IDE se preparará automáticamente para conectarse al servidor con las credenciales ofrecidas y nos abrirá el entorno de trabajo en nuestra instancia, como si estuviéramos trabajando en este IDE en nuestra máquina local.

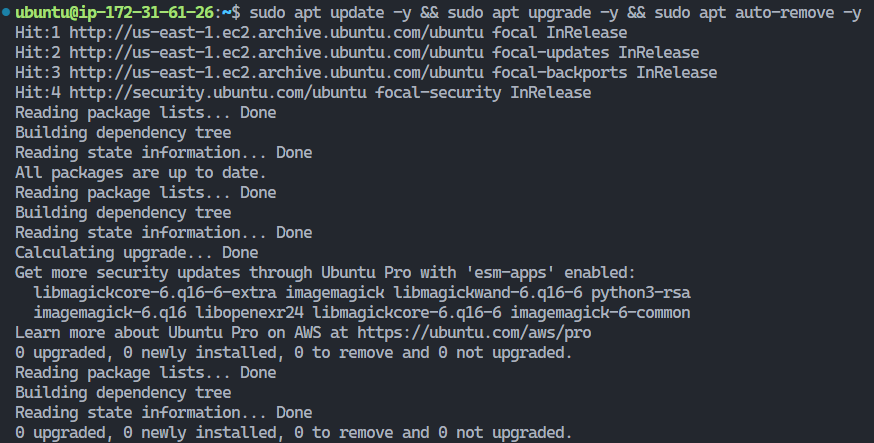


Una vez aquí, cada vez que abramos un terminal (su atajo de teclado es “**Ctrl + Ñ**”) es como si utilizásemos el terminal proporcionado por PuTTy o MobaXterm, con la ventaja de que podemos abrir directorios o archivos directamente desde el IDE para editarlos con el entorno gráfico y profesional que ofrece VS Code.

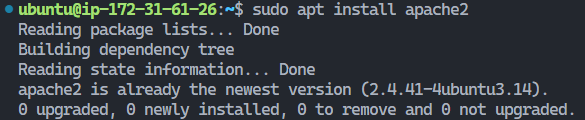
**1.2. Actualización e instalación de Apache**

Lo primero que debemos hacer al acceder de nuevo a la máquina es actualizarla, para que la instalación de Apache no presente problemas debido a algún conflicto con posibles archivos obsoletos del servidor.

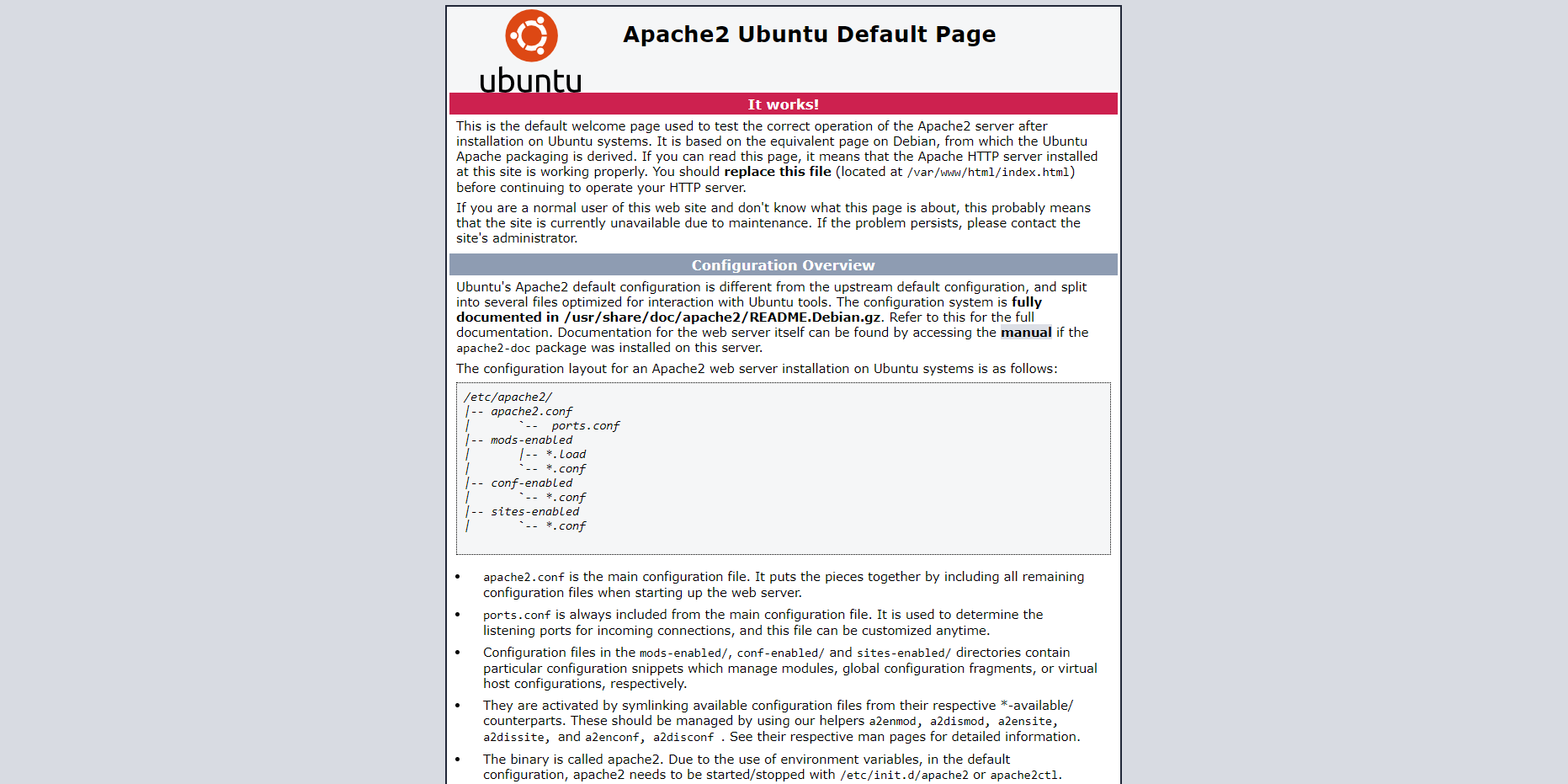
Ejecutamos el comando “**sudo apt update -y && sudo apt upgrade -y && sudo apt auto-remove -y**”.



Una vez actualizado el sistema operativo a su última versión, ejecutamos el comando “**sudo apt install apache2 -y**” para instalar Apache en la máquina.



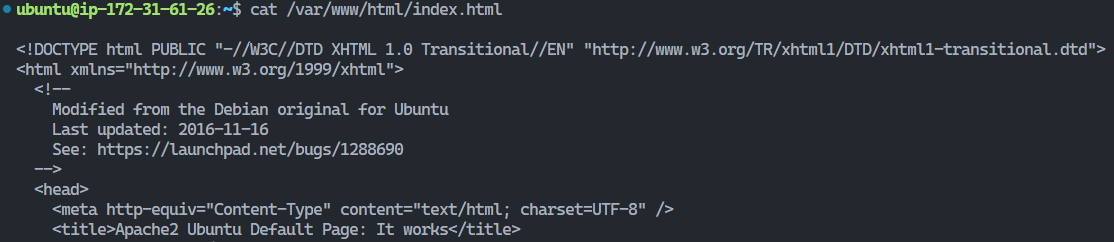
Si nos dirigimos a un navegador web e ingresamos la URL “**{dirección-ip-pública-instancia}**” se nos mostrará una página como la siguiente:



Esto se debe a que se utiliza el puerto 80 de forma predeterminada, aunque no se especifique en la URL ingresada, y este puerto está destinado al protocolo HTTP, que es el que utiliza de base Apache.

La página generada es una página de prueba que Apache se encarga de crear a la hora de instalarse, cuyo único objetivo es demostrar que el servidor está corriendo y en funcionamiento.

Si ejecutamos el comando “**[cat /var/www/html/index.html](https://www.ibm.com/docs/en/aix/7.2?topic=c-cat-command)**”[🔗](https://www.ibm.com/docs/en/aix/7.2?topic=c-cat-command) se nos mostrará el contenido del HTML dedicado a esta página web.



Esto es debido a que la ruta predeterminada para el sitio disponible que Apache establece en su instalación es “**/var/www/html**”, y el archivo que intenta buscar de forma predeterminada debe ser llamado “**index.html**”.

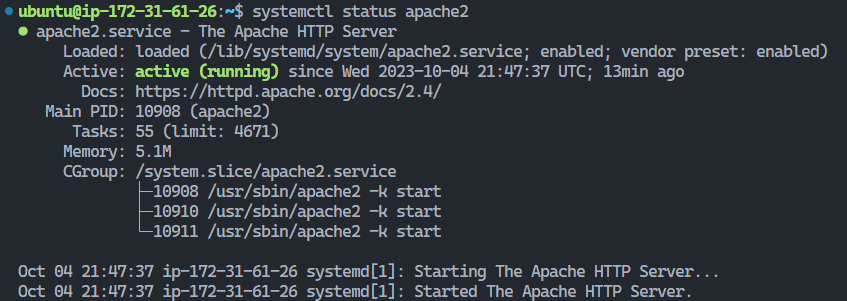
Esto lo cambiaremos más adelante, pero por ahora nos sirve para comprobar que Apache está funcionando.

Existe otra forma generalmente utilizada para la comprobación del estado del servidor, la cual se basa en la ejecución del comando “**[systemctl](https://www.digitalocean.com/community/tutorials/how-to-use-systemctl-to-manage-systemd-services-and-units)**”[🔗](https://www.digitalocean.com/community/tutorials/how-to-use-systemctl-to-manage-systemd-services-and-units).

Con el comando “systemctl”, podemos especificar la acción que queremos realizar y sobre qué servicio o aplicación realizarla.

Esto es muy útil, y nos funcionará sin problemas para comprobar que todos los [**módulos**🔗](https://dev.to/tikam02/apache-modules-basics-and-configurations-5bnh#:~:text=What%20are%20the%20modules) están activos y el servidor no ha encontrado ningún impedimento a la hora de ejecutarse.

Ejecutamos el comando “**systemctl status apache2**” para ver todo esto.



**2. Configuración del servidor web**

El primer paso, según la documentación de esta práctica, es añadir un puerto de escucha para Apache en el 8090, lo cual permitirá que escribamos la URL “**{dirección-ip-pública-instancia}:8090**” y podamos acceder al mismo contenido que desde el puerto 80.

**2.1. Puerto 8090**

Para realizar este primer paso, abriremos un archivo de configuración desde el entorno que nos proporciona nuestro IDE.

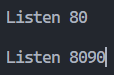
Debido a que los archivos de configuración de Apache necesitan de unos permisos más elevados, no podremos editar estos ficheros desde el entorno gráfico de nuestro IDE, ya que para acceder a ellos necesitamos del uso del comando “sudo”, el cual no podemos utilizar si abrimos un archivo directamente desde el IDE.

Para realizar las configuraciones en estos archivos utilizaremos el terminal de comandos, como de costumbre.

Lo primero que tenemos que ejecutar es el comando “**sudo nano /etc/apache2/ports.conf**”, lo que abrirá el archivo donde se configuran los puertos de escucha del servidor Apache.

Una vez dentro del archivo, nos encontramos con una regla que aparece como “**Listen 80**”. Este es el puerto que Apache utiliza de forma predeterminada para realizar las conexiones y el envío del contenido web.

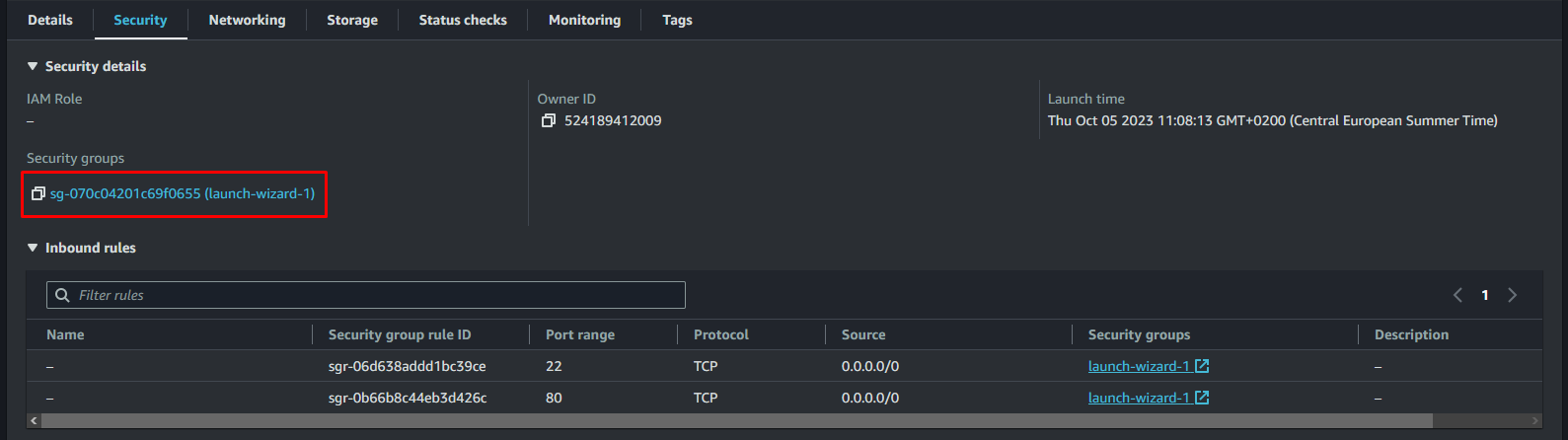
Debajo de esta línea agregamos el mismo contenido que hemos visto, pero cambiando el puerto 80 por el **8090**, que es el que se especifica en la práctica.



“**Ctrl + O -> Enter -> Ctrl + X**” para guardar y salir. Este proceso ha habilitado la escucha para el puerto 8090 de nuestro servidor en la parte de configuración de Apache.

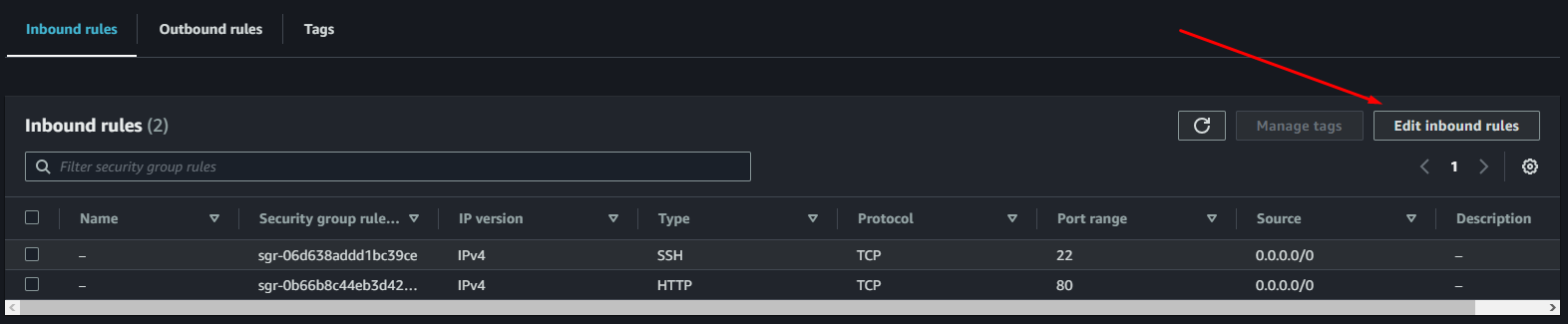
Ahora debemos realizar una configuración en la propia instancia de AWS, ya que necesitamos crear una [**regla de entrada**🔗](https://docs.aws.amazon.com/quicksight/latest/user/vpc-security-groups.html#:~:text=VPC%20User%20Guide.-,Inbound%20rules,-Important) que permita las conexiones mediante la utilización del puerto 8090.

Nos dirigimos a la información de la instancia en AWS y entramos en el apartado “**Security**”. Hacemos clic en el identificador para las reglas de entrada.



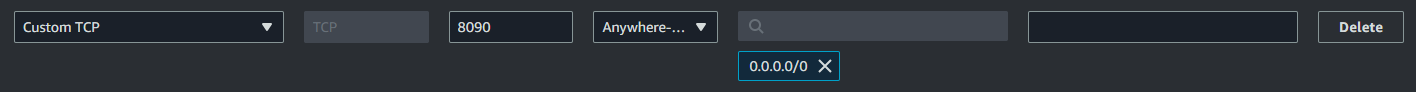
Se nos presenta una lista de las reglas que tenemos activas en el servidor, las cuales configuramos en la práctica anterior en el proceso de creación de la instancia.

Entramos en la edición de reglas de entrada.

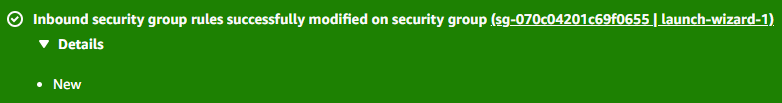


Aquí hacemos clic en el botón “**Add rule**”, situado abajo a la izquierda.

Dejamos el tipo de regla en la opción de serie “**Custom TCP**”, establecemos que el puerto que utilizará será el **8090** y permitimos que el acceso mediante este puerto se pueda realizar desde cualquier dirección IP del mundo, dejando la regla de la derecha en “**0.0.0.0/0**”.



Guardamos los cambios haciendo clic en el botón de la esquina inferior derecha.



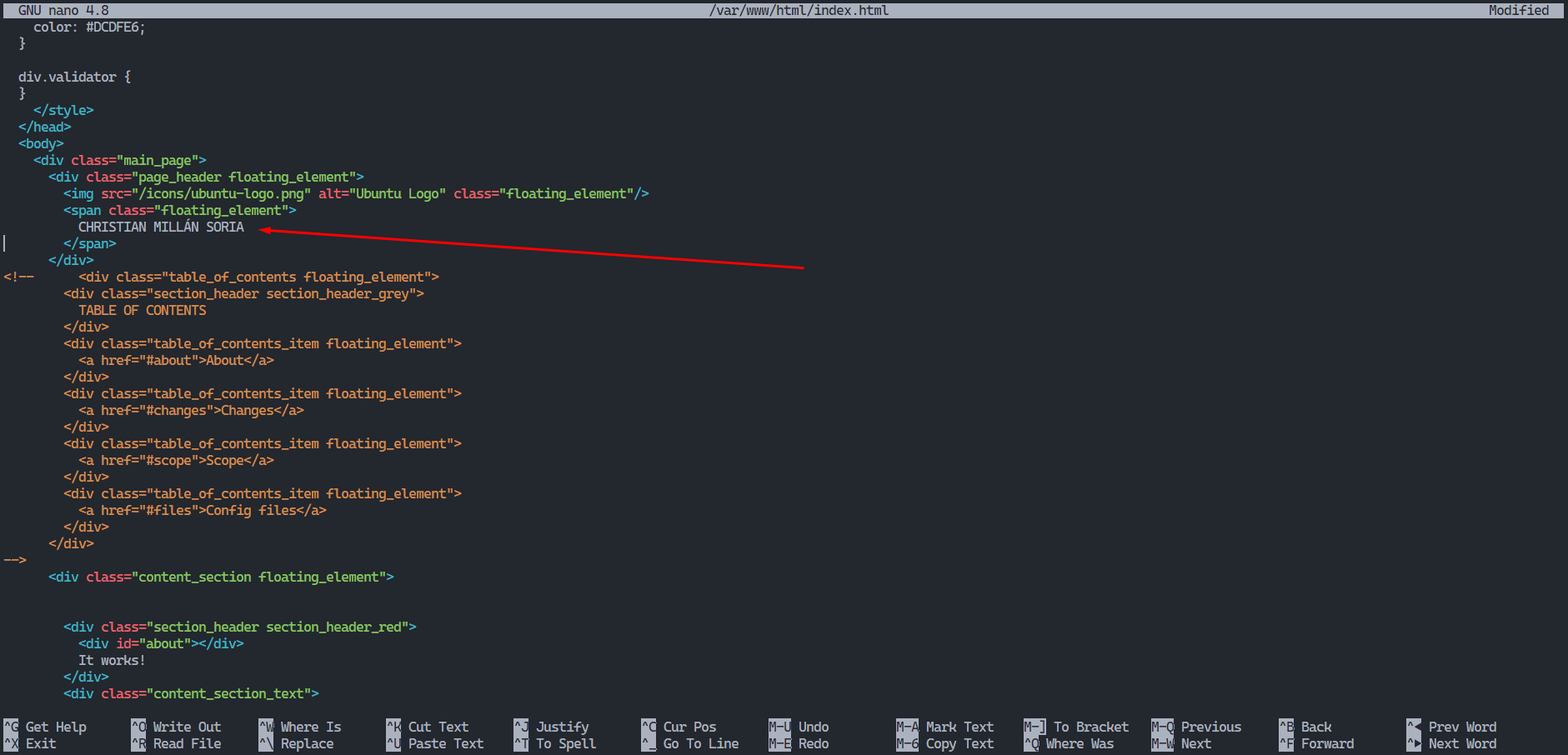
Esto ha hecho que las conexiones entrantes realizadas mediante el uso del puerto 8090 sean posibles, pero todavía no hay nada disponible en nuestro servidor que muestre contenido con este puerto, por lo que aunque ingresemos la URL “{dirección-ip-pública-instancia}:8090” ahora mismo no se nos cargará ninguna página.

**3. Edición de la página web**

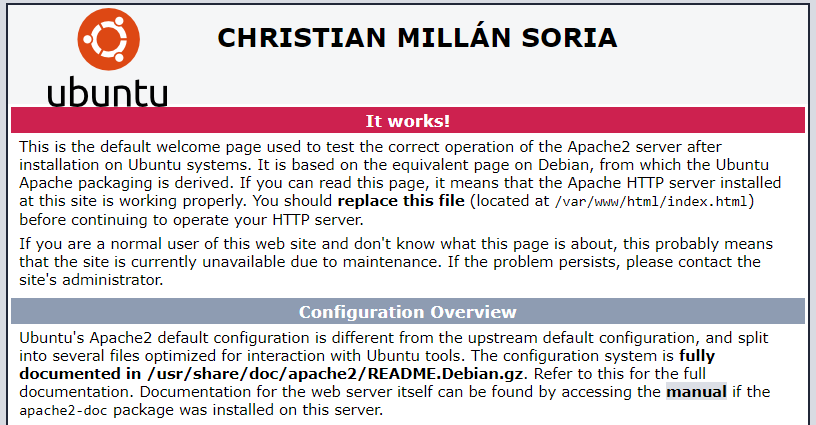
Este paso requiere que editemos el contenido de la página web de prueba para demostrar que podemos realizar cambios y son propios de cada alumno.

Ejecutamos el comando “**sudo nano /var/www/html/index.html**” en el terminal del servidor para poder editar el archivo que Apache está mostrando en la URL de nuestra IP pública.

Nos dirigiremos a la línea de la primera etiqueta “span”, con la clase “**floating\_element**”, y cambiamos el texto “**Apache2 Ubuntu Default Page**” por el texto personalizado “**{NOMBRE ALUMNO}**”.



Si recargamos la página con la URL de la IP pública de nuestra instancia deberíamos ver el cambio en la parte superior de la misma, pero en caso de no ser así, ejecutaremos el comando “[**sudo systemctl reload apache2**](https://www.linode.com/docs/guides/introduction-to-systemctl/#:~:text=can%20issue%20the-,reload,-command%3A)”[🔗](https://www.linode.com/docs/guides/introduction-to-systemctl/#:~:text=can%20issue%20the-,reload,-command%3A), el cual es necesario para que Apache aplique los cambios realizados en el server.



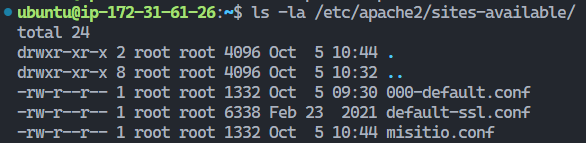
**4. Archivo default personalizado**

El siguiente objetivo es crear un archivo base personalizado, con la configuración pertinente para que, al visitar la URL de la dirección IP pública del servidor se muestre dicha página, y no la “[**000-default**](https://exampleconfig.com/view/apache-ubuntu20-04-etc-apache2-sites-available-000-default-conf)”[🔗](https://exampleconfig.com/view/apache-ubuntu20-04-etc-apache2-sites-available-000-default-conf) que trae por defecto Apache.

**4.1. Configuración del archivo “.conf”**

Volvemos a utilizar el terminal de comandos del servidor, esta vez para replicar el archivo de configuración del sitio “000-default” y tener el nuestro propio.

Utilizaremos el comando “**sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/misitio.conf**” para copiar el archivo base en la misma ruta, pero con un nombre como “**misitio.conf**”, tal y como se especifica en la documentación de la práctica.



Con el comando “**sudo nano /etc/apache2/sites-available/misitio.conf**” entramos al editor de nuestro nuevo archivo.

En la primera línea del archivo se especifica el puerto al que atenderá nuestro “index.html” para que Apache pueda encontrarlo y mostrarlo una vez se utilice el puerto o los puertos establecidos aquí.

La práctica especifica que nuestro archivo base personalizado debe ser accesible desde el puerto 80 y el puerto 8090, por lo que al lado de la especificación del puerto 80 en esta primera línea, añadimos el texto “**\*:8090**” para añadir el nuevo puerto a nuestro archivo.



Por último, editaremos la ruta del directorio raíz donde se alojará nuestro archivo index base.

Bajamos un poco y nos encontramos con una directriz llamada “**DocumentRoot**”, donde cambiaremos la terminación que indica el directorio “**html”** por un nuevo directorio que crearemos llamado “**misitio**”.



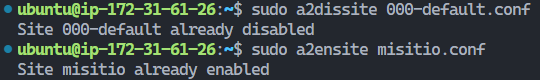
Guardamos y salimos del archivo.

**4.2. Habilitar y deshabilitar sitios**

Una vez configurado el archivo para nuestro nuevo sitio, necesitamos deshabilitar el sitio por defecto de Apache y habilitar el nuestro.

Para este tipo de acciones, Apache ofrece el comando “[**a2**](https://www.2daygeek.com/linux-apache2-commands-cheatsheet-ubuntu-debian/#:~:text=How%20to%20Enable/Disable%20Virtual%20Hosts%20in%20Debian/Ubuntu)”[🔗](https://www.2daygeek.com/linux-apache2-commands-cheatsheet-ubuntu-debian/#:~:text=How%20to%20Enable/Disable%20Virtual%20Hosts%20in%20Debian/Ubuntu), que proporciona una serie de opciones que nos servirán para este proceso.

Para este caso, necesitaremos ejecutar primero el comando “**sudo a2dissite 000-default.conf**”, con el fin de deshabilitar el sitio predeterminado del servidor, y acto seguido el comando “**sudo a2ensite misitio.conf**”, para habilitar el creado por nosotros ahora mismo.



Recargamos el servidor para aplicar los cambios con el comando “**sudo systemctl reload apache2**”.

**4.3. Directorio y archivo propios**

Vamos a crear un directorio (el mismo que hemos indicado en el archivo “misitio.conf”) para alojar nuestro index personalizado.

Para esto, utilizamos el comando “**sudo mkdir /var/www/misitio**”.

Ahora ejecutamos el comando “**sudo cp /var/www/html/index.html /var/www/misitio/index.html**”. Esto copiará el archivo index que Apache trae de forma nativa al nuevo directorio que hemos especificado.

Por último, con el comando “**sudo nano /var/www/misitio/index.html**” entramos a editar dicho archivo y, para poder diferenciarlo del anterior y a la vez del que Apache genera al instalarse, vamos a cambiar el contenido de la etiqueta “**title**”, sustituyendo el texto en su interior por el texto “**Página alojada en /misitio**”.



Además, volveremos a dirigirnos a la etiqueta con la clase “**floating\_element**” (etiqueta “**span**”) y añadiremos la palabra “**V.2**” al final, después de nuestro nombre.



Guardamos y salimos del fichero.

Recargamos el servidor con el comando “sudo systemctl reload apache2”.

Ahora podemos acceder a la página de inicio de Apache desde el puerto 80 y desde el puerto 8090, indistintamente.





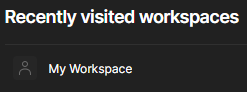
**5. Postman**

El último apartado de esta práctica se basa en mostrar de forma básica cómo podemos realizar peticiones a nuestra página de inicio del servidor Apache utilizando “**[Postman](https://www.postman.com/)**[”](https://www.postman.com/)**[🔗](https://www.postman.com/)**.

“Postman” es una herramienta de desarrollo de API que permite a los desarrolladores crear, probar y documentar API de manera eficiente. Se utiliza principalmente para simplificar el proceso de construcción y prueba de API al proporcionar una interfaz de usuario intuitiva que permite enviar solicitudes HTTP a un servidor y recibir respuestas. “Postman” también ofrece características avanzadas, como la automatización de pruebas, la gestión de entornos y colecciones de solicitudes, y la generación de documentación de API. Es ampliamente utilizado por desarrolladores de software y equipos de desarrollo para mejorar la eficiencia y la calidad en el desarrollo y la integración de API.

Se presupone que ya tenemos una cuenta creada en la página web de”Postman”.

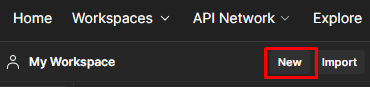
Una vez entramos al panel de control de nuestra cuenta, vemos un elemento llamado “**My Workspace**”. Hacemos clic en él.



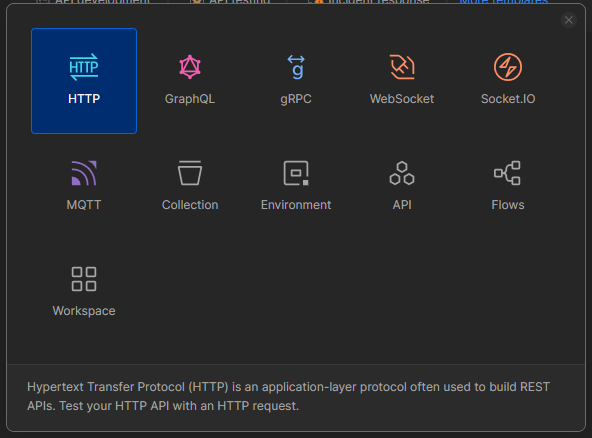
Aquí dentro es donde realizaremos la prueba para esta práctica.

**5.1. Método “GET”**

Una vez dentro del workspace, hacemos clic en el botón “**New**”, arriba a la izquierda.



Elegimos el protocolo “HTTP” en el cuadro de selección que se abre.



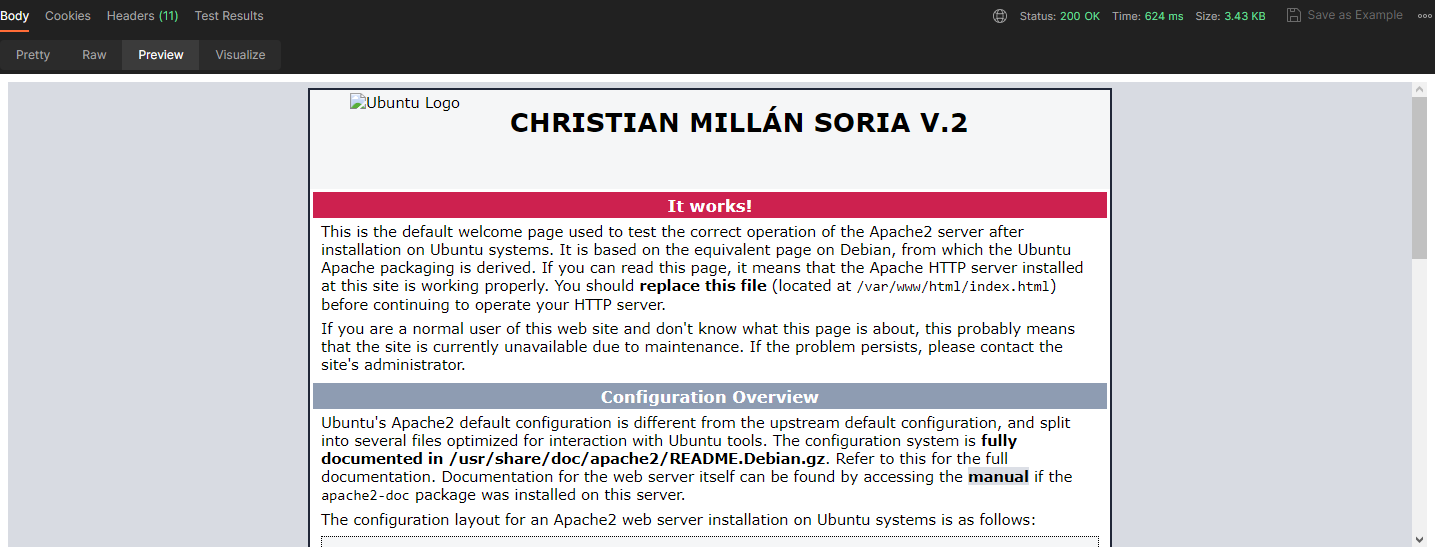
Como podemos ver, se nos presenta una barra de búsqueda donde ingresaremos la URL “**{dirección-ip-pública-instancia}**” o “**{dirección-ip-pública-instancia}:8090**”.

El método de conexión lo dejaremos en “**GET**”.



Al hacer clic en el botón “**Send**”, situado a la derecha, podemos ver como se ejecuta la conexión y la petición mediante el método elegido.

Abajo podemos ver el código de la página que hemos solicitado y en la pestaña “Preview” podemos ver la propia página en sí, como en un navegador web.



Volviendo a la pestaña “**Pretty**”, donde veíamos el código, podemos seleccionar el formato en el que se nos presenta dicho código en un desplegable a la derecha, donde podríamos seleccionar “**JSON**”, por ejemplo, y recibiríamos el mismo contenido en este formato.

