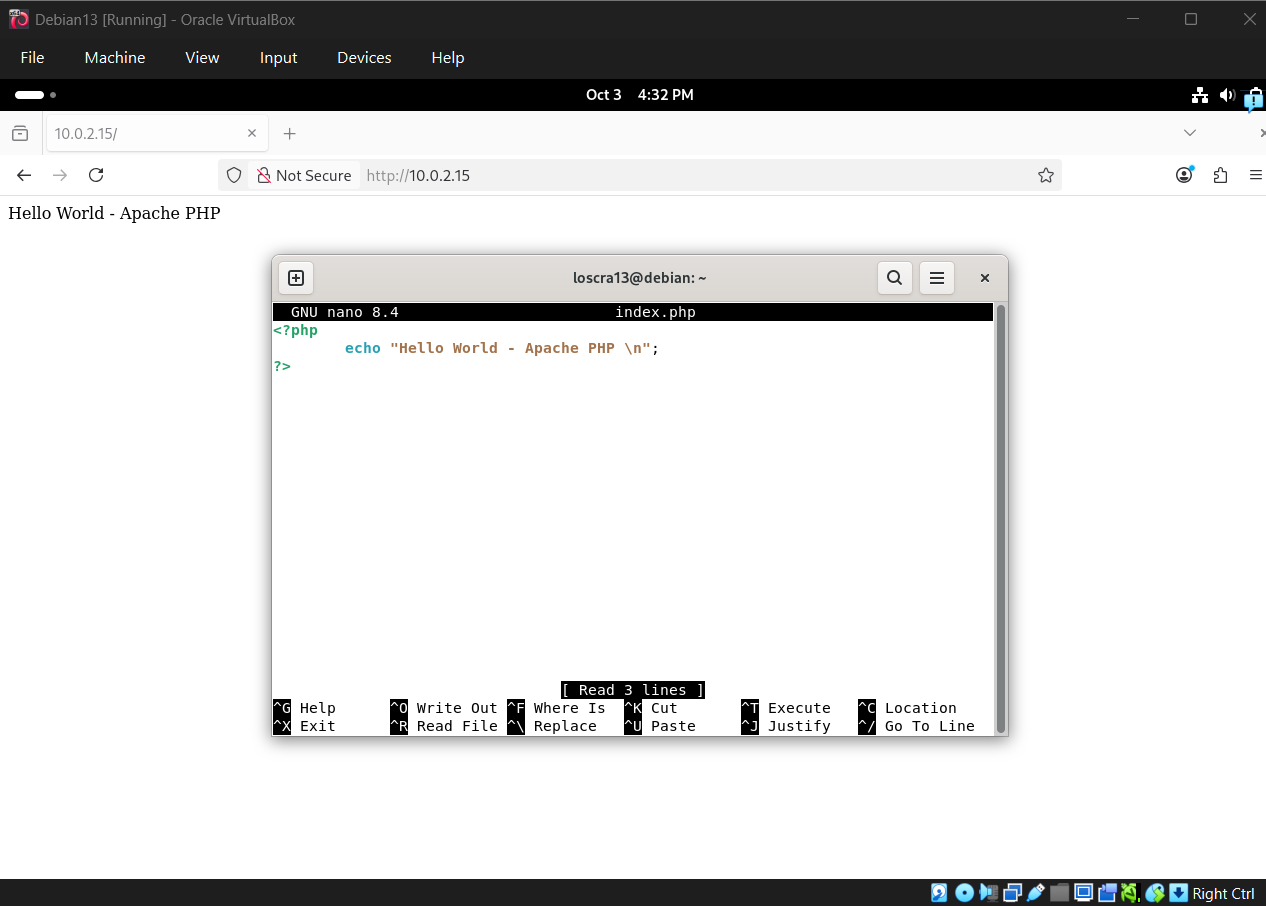
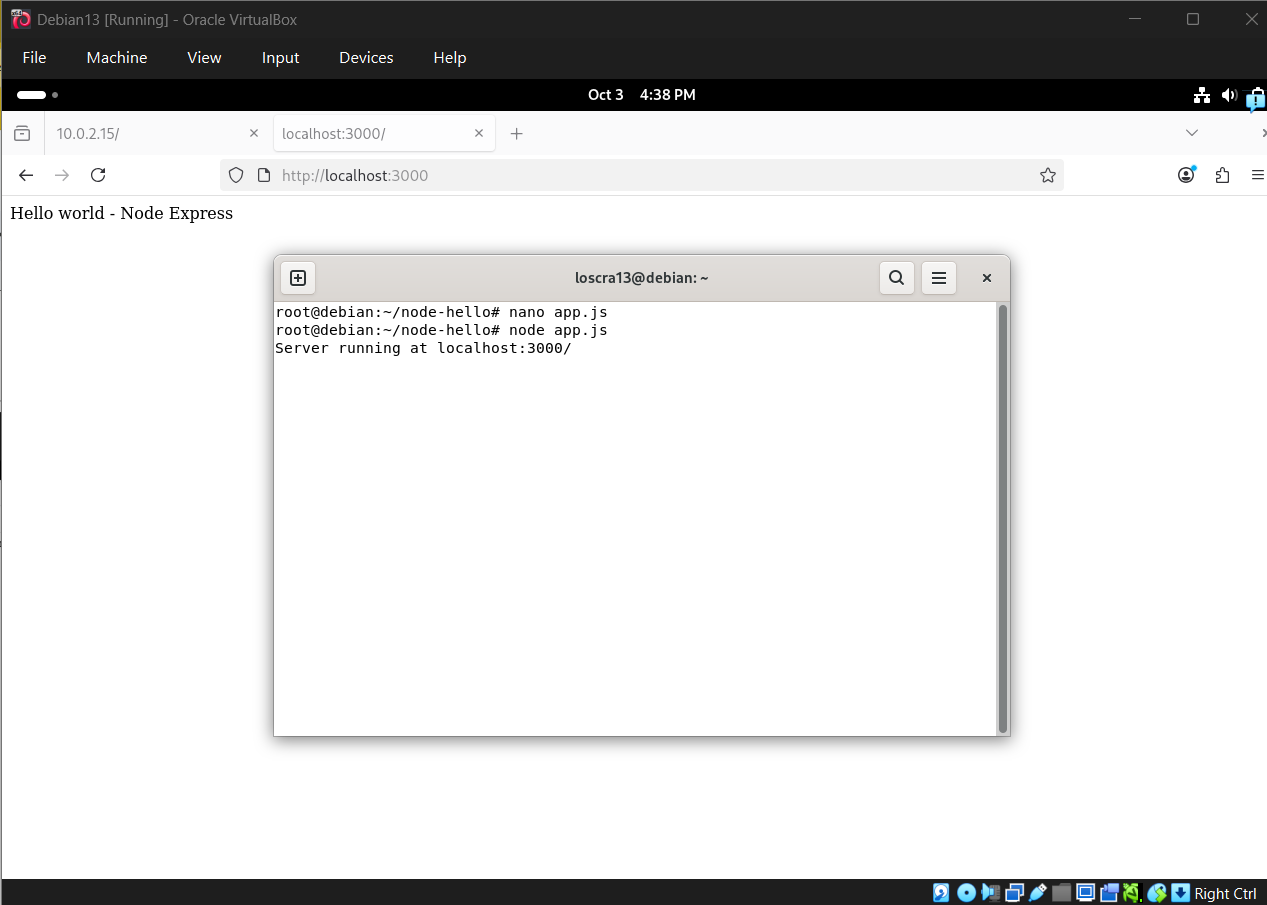
# Punto 1. Instalación del Software

Se instala Apache y PHP, creando un código simple de “Hello World” y desplegando en pantalla. Tras reiniciar la máquina virtual, el servicio seguía ejecutándose tras acceder al localhost.



Posteriormente, se instala Node y Express para desarrollar una lógica similar, pero en este caso el servidor no estaba encendido automáticamente tras el reinicio; tuvo que iniciarse manualmente para poder desplegar el “Hello world”.

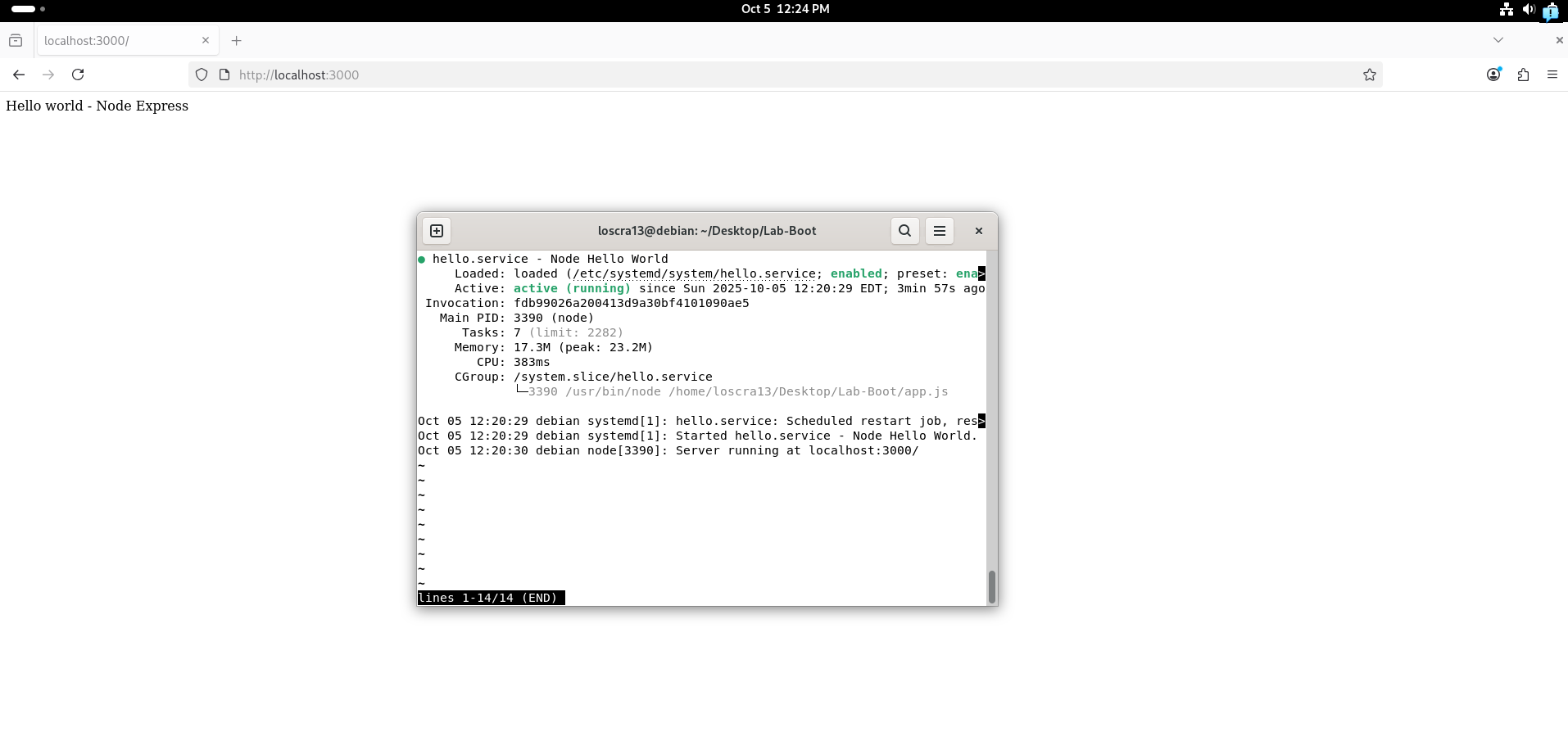
# Punto 2. Comparación del Software

En primer lugar, Apache2 es un popular servicio web que la distribución utilizada (Debian 13), administra por medio de “systemd” utilizando una Unit File (archivo de texto plano con la configuración de un servicio) ubicada en la ruta “/lib/systemd/system/apache2.service”. De acuerdo con la información presentada por Ibeh (2024) y Wisniowski (s.f.), Apache funge como un servidor web desde la máquina instalada, acomodándose en el Linux Network Stack y utilizando HTTP para establecer una conexión con el Internet. Esto provoca que esté intrínsecamente enlazado con el Sistema Operativo, constantemente ejecutándose en segundo plano y que recibe el tratamiento de “sudo systemctl enable apache2”, siendo esto lo que permite que el servicio se mantenga activo tras reiniciar la máquina.

En segundo lugar, Node JS, de acuerdo con Wisniowski (s.f.), utiliza un modelo centrado en un “event loop”, el cual consiste en un ciclo que constantemente notifica al Kernel cuando una tarea se ha completado. De esta forma, cada instancia de Node JS se maneja en un mismo hilo de forma concurrente y asincrónica con otros, por lo que per sé no es un servicio, sino un proceso más que el sistema operativo administra como cualquier otro. Por su parte, según lo recuperado de Geek for Geeks (2025), Express JS consiste en un Framework que se acomoda encima del servidor web the Node JS para simplificar su API y proveer nuevas funcionalidades para manejar los objetos HTTP.

De esta forma, la principal distinción entre ambos software es que el primero es un servicio, mientras que el otro es un proceso. Asimismo, Apache es administrado por Linux systemd, por lo que tiene soporte para activarse automáticamente tras el boot de la computadora, mientras que Express y Node no tienen esta funcionalidad por defecto.

Con esto, se escribe un Unit File llamado “hello.service” para que Node JS pueda inicializarse tras reiniciar la máquina virtual.



# Punto 3. Continuous Deployment

Se creó un nuevo archivo temporizador “deploy.timer” que se ejecuta en el fondo con un lapso de 5 minutos y toda su responsabilidad es llamar a un servicio "deploy.service", el cual actualiza un repositorio en Github al verificar si la branch Deploy tiene cambios o no por medio de la ejecución de un archivo de Shell tras registrar una llave SSH en la cuenta de GitHub:

Esto es una forma de Continuous Deployment, definido por Susnjara y Smalley (s.f.) como una estrategia en el desarrollo de software en la cual el código se actualiza automáticamente a la última versión que exista en el entorno de producción. Con esta implementación, no es necesario esperar ni manualmente programar una actualización, sino que el control de versiones toma esta responsabilidad y de forma transparente para el usuario en el caso de este laboratorio.

# Punto 4. Conclusiones

Tras haber realizado el laboratorio y verificado lo encontrado por The Hosting (2023), se puede afirmar que los servicios de Linux desempeñan un rol fundamental para garantizar el funcionamiento adecuado del sistema operativo. Desde proporcionar servicios web para cambiar información hasta procesar y gestionar datos como se vio en el caso del Continuous Deployment, se puede ver cómo los servicios permiten automatizar una serie de tareas que de otra forma serían repetitivas y tediosas.

Asimismo, el proceso de crear un servicio se puede afirmar que es bastante sencillo de completar exitosamente: basta con crear un archivo “.service” o “.timer” según la lógica que se desee implementar y activarlos en unos pocos comandos (haciendo uso de *systemctl*) desde la terminal. Son especialmente útiles al saber que estos Unit Files son compatibles con programas Shell que permiten expandir la libertad en el manejo que se puede hacer del dispositivo por medio de esta técnica.

Para terminar, se recalca la importancia y versatilidad de systemd al momento de automatizar procesos por medio de servicios, donde verdaderamente aportan bastante al proceso de desarrollo. Ya sea que se ejecute código al momento de arranque o durante una sesión normal de uso de la computadora, los servicios ahorran tiempo que se puede aprovechar en otras actividades. Un programador que trabaja en Linux tiene plena capacidad de que las tareas más comunes de su cotidianidad laboral se hagan correctamente y en el momento que se desee al aprovechar todo el poder que los daemons ofrecen, sin tener que estar al pendiente de lo que ocurre en el fondo al ejecutarse de forma plenamente transparente para el usuario.

# Referencias

* GeeksforGeeks. (23 de julio, 2025). *Node vs Express*. <https://www.geeksforgeeks.org/node-js/node-js-vs-express-js/>
* Ibeh, J. (31 de octubre, 2024). *Getting Started with systemd: Managing Services and Configuring Apache on Linux*. DEV Community. <https://dev.to/sir-j/understanding-systemd-the-modern-service-manager-for-linux-13i0>
* Susnjara, S &, Smalley, I. (s.f.). *What is continuous deployment?* IBM. <https://www.ibm.com/think/topics/continuous-deployment>
* The Hosting. (15 de septiembre, 2023). *¿Qué son los servicios en Linux?*. <https://the.hosting/es/help/que-son-los-servicios-en-linux>
* Wisniowski, K. (s.f.). *Apache vs Node.js - What’s the Difference? (Pros and Cons)*. Cloud Infrastructure Services. <https://cloudinfrastructureservices.co.uk/apache-vs-node-js-whats-the-difference/>