******UNIVERSIDAD DE EL SALVADOR**

Facultad Multidisciplinaria De Occidente

Departamento de Ingeniería y Arquitectura

Ingeniería en Desarrollo de Software

Asignatura:

**INTRODUCCIÓN AL SOFTWARE LIBRE**

Ciclo IV / Segundo año

Tema:

Proyecto Final

Coordinadora de carrera:

Ing. Ángela de Granillo

Tutor/a GT02:

Ing. Albaluz Escalante

Alumno/a:

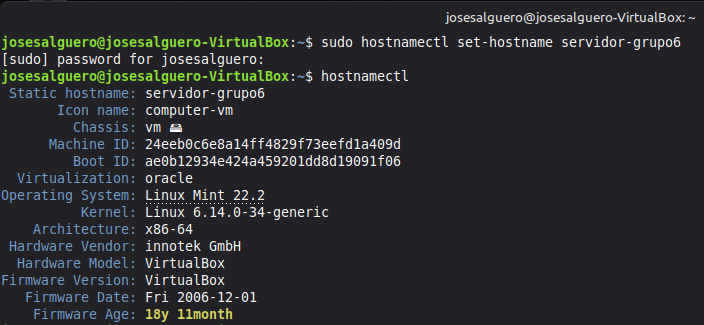
|  |  |
| --- | --- |
| **Nombre** | **Carnet** |
| José Israel Lemus Salguero | LS24009 |
| Andrea Isabel Chávez Mejía | CM24080 |
| José Ricardo González | GG22088 |
| Josseline Stephanie Ramos Ramos |  |

Fecha de entrega:

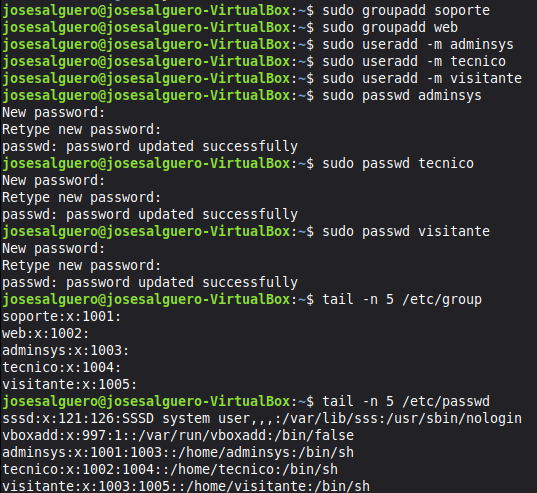
-------

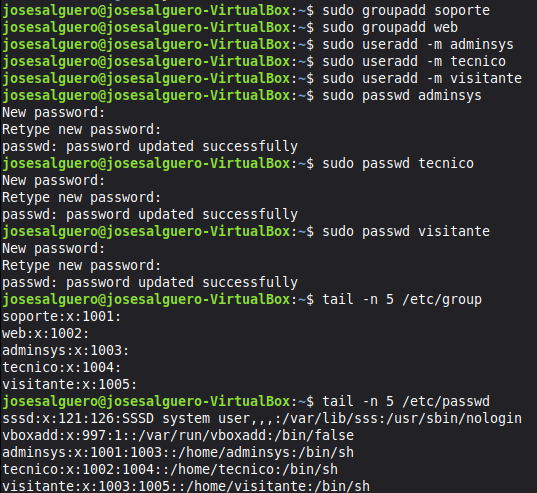
1. **Fase: Preparación del Entorno Servidor ⚙️**
   1. Agregando un nombre al host. En este caso se agregó “servidor-grupo6”

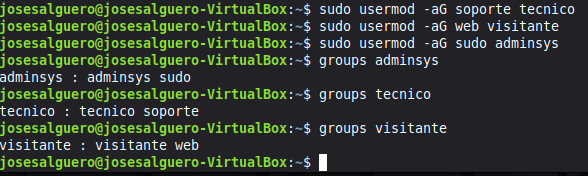
El **Objetivo** de esta fase fue establecer la base del sistema operativo (Linux Mint) con una estructura de usuarios y permisos que simulan un entorno empresarial real, aplicando el **Principio del Mínimo Privilegio (PoLP)**.



* 1. **Creando grupos, usuarios con sus respectivos directorios personales**
* **Usuarios y Grupos:** Creamos usuarios con roles definidos (adminsys, tecnico, visitante) y grupos específicos (sudo, soporte, web). Esto segmenta las responsabilidades, asegurando que el personal tenga acceso limitado a sus áreas de trabajo.







* 1. **Estructura de directorios y permisos**

**Estructura de Directorios:** La carpeta raíz /proyecto/ y sus subdirectorios (web/, datos/) se crearon para organizar el código y los datos del proyecto.

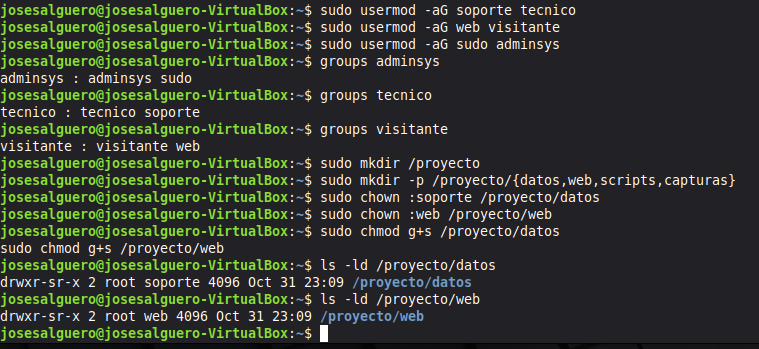
**Permisos de Herencia (sgid):** Aplicamos el *bit* **sgid** (chmod g+s) a los directorios como /proyecto/web/. Esta técnica asegura que **cualquier archivo creado dentro herede**

datos/: para archivos de configuración

web/: para archivos del sitio web

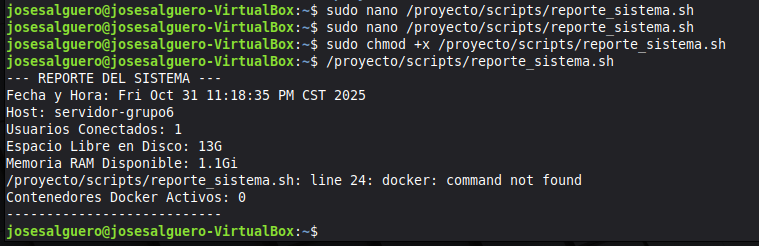
scripts/: para scripts de automatización

capturas/: para evidencias del proyecto



* 1. **Automatización Y Monitoreo**

**automáticamente el grupo propietario** (web), simplificando la gestión de permisos para los usuarios de ese grupo

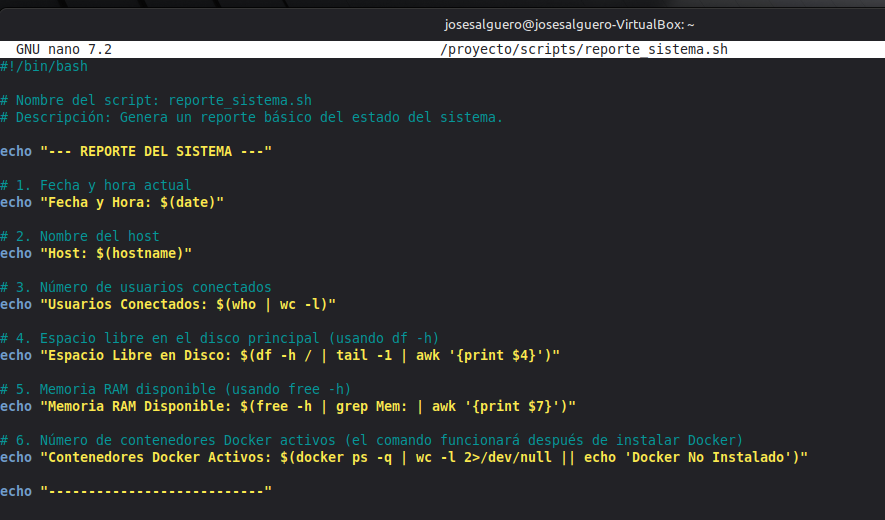


1. **Fase: Automatización y Monitoreo ⏱️**

El **Objetivo** de esta fase fue implementar herramientas nativas de Linux para garantizar la recolección periódica y automática de datos del estado del servidor.

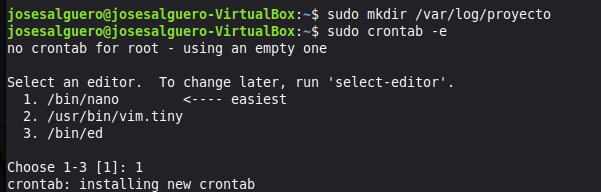
**2.1 Creación de un archivo reporte\_sistema.sh**

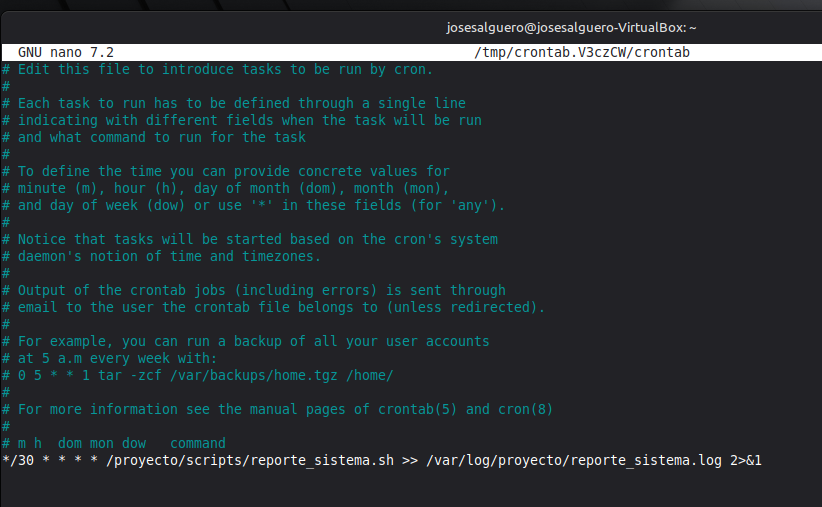
* **Script Shell (.sh):** Creamos el script **reporte\_sistema.sh** para consultar métricas clave del sistema (uso de RAM, disco, estado de contenedores Docker, etc.). Este script es la base para el monitoreo de la salud del servidor.

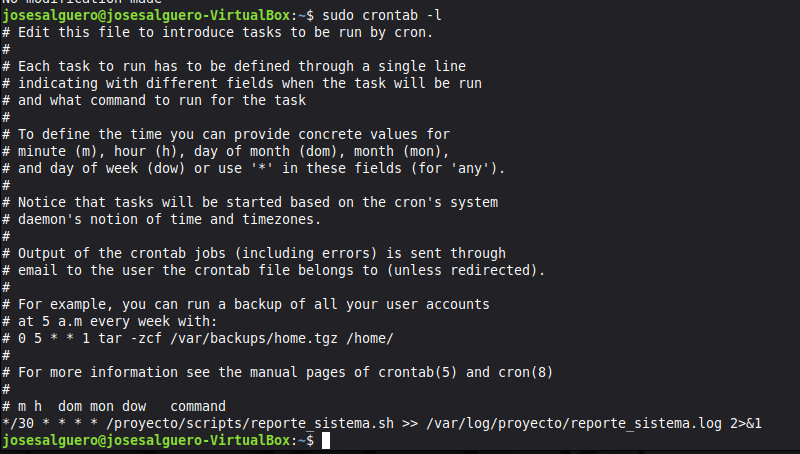


* 1. **Automatización.**

**Cron:** Usamos la utilidad **crontab** del usuario root para programar la ejecución del *script* cada 30 minutos. Esta automatización garantiza que el monitoreo sea **persistente y sin intervención manual**, guardando los resultados en /var/log/proyecto/.

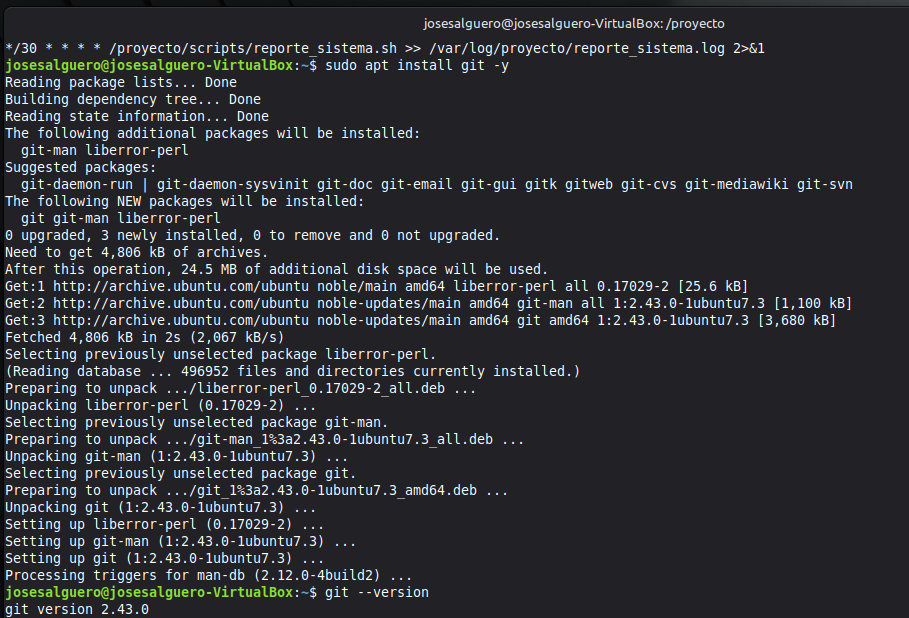




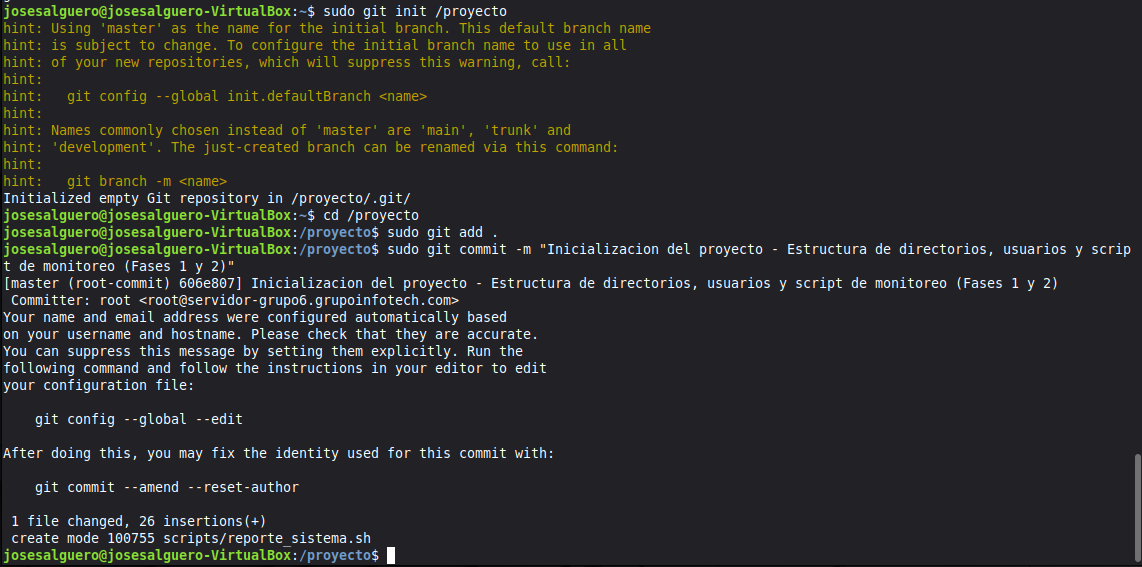


**3. Fase: Control de Versiones (Git y GitHub) 📦**

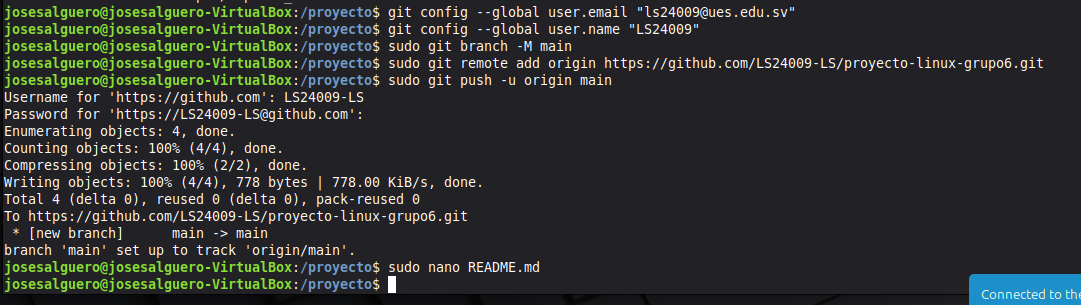
El **Objetivo** fue utilizar Git para la trazabilidad del código y la colaboración estructurada, usando GitHub como repositorio central y respaldo. Instalacion de git.



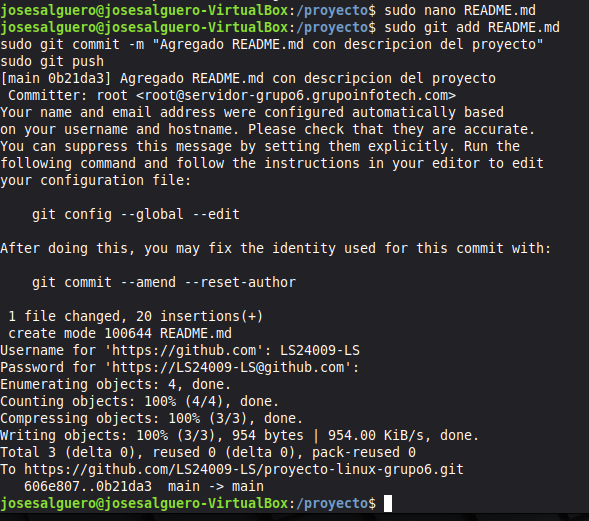
* 1. **Crear repositorio remoto en GitHub.**



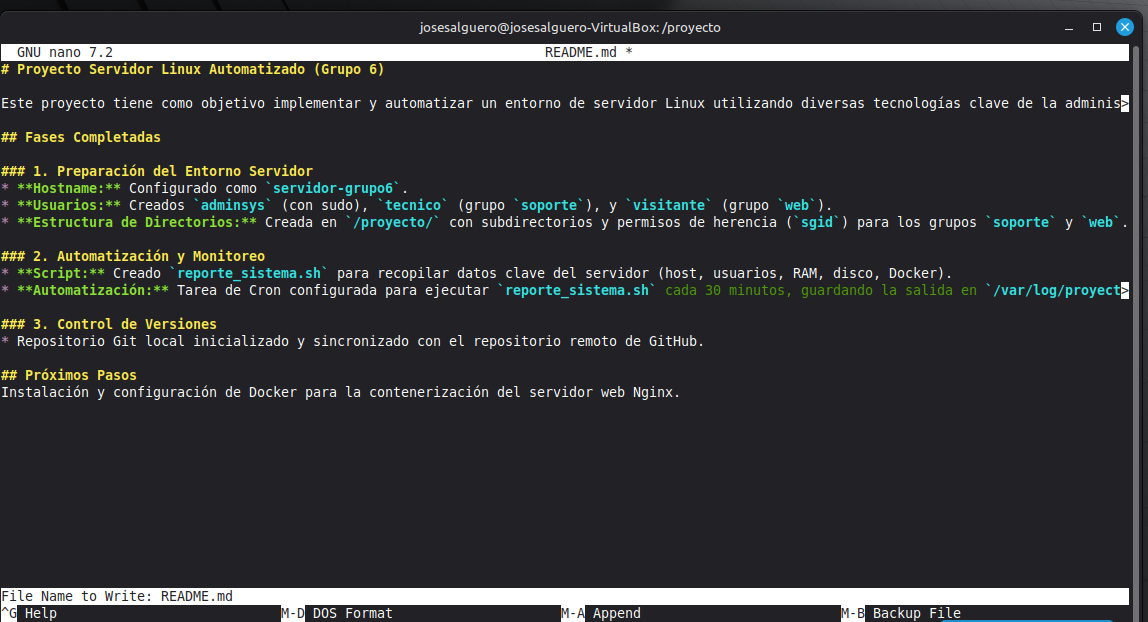
* **Flujo Git:** Se inicializó el repositorio local, se hicieron *commits* con mensajes descriptivos para documentar cada cambio funcional. Se usó un **Token de Acceso Personal (PAT)** en lugar de una contraseña para autenticar las operaciones con GitHub.
  1. Creando repositorio remoto para proyecto.



* 1. **Abriendo el archivo Readme.md**



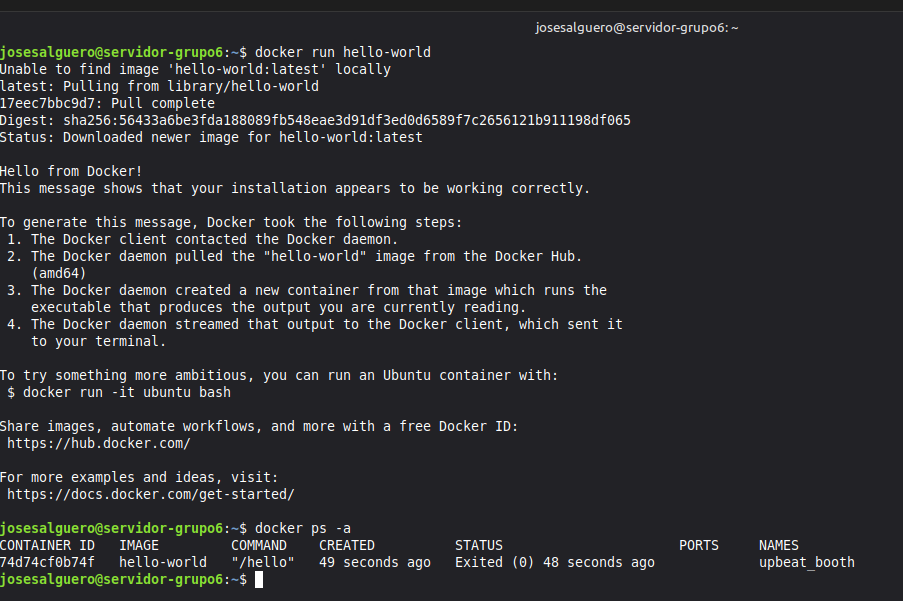
* 1. Editando el archivo Readmi del repositorio



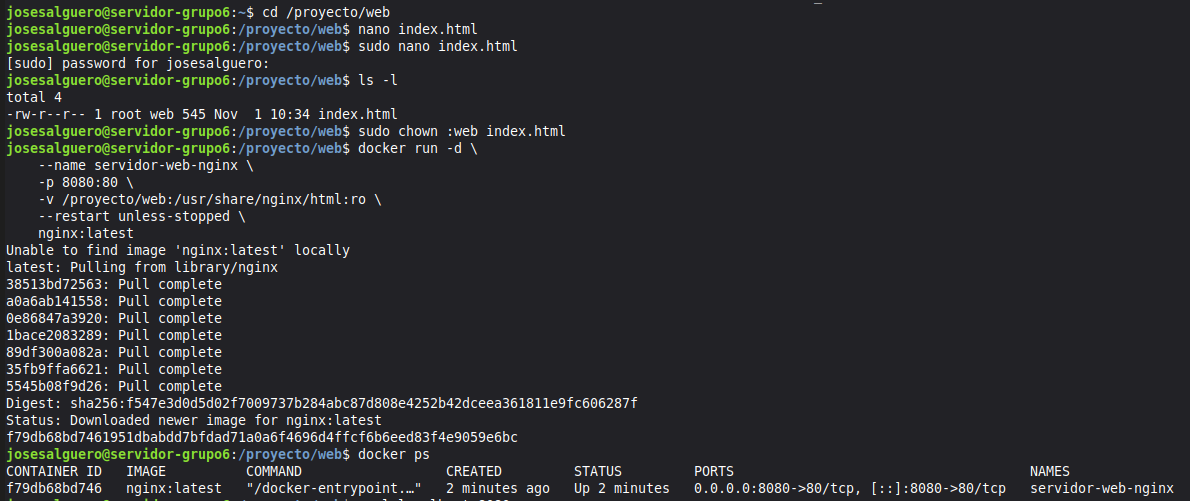
* 1. **Fase: Contenerización (Docker) 🐳**

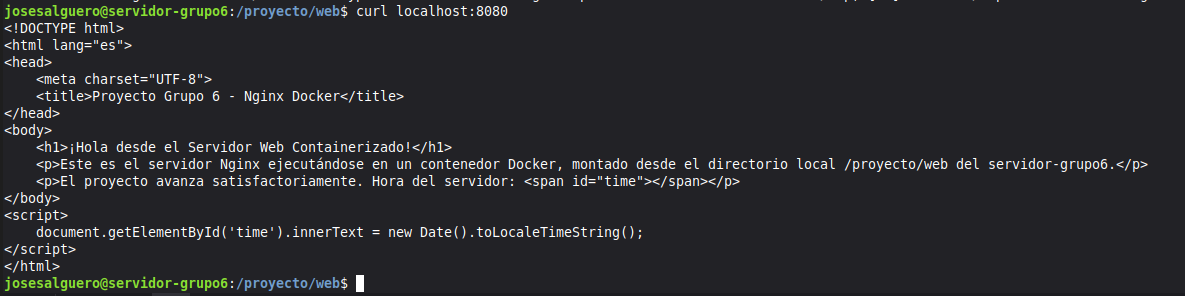
El **Objetivo** fue instalar Docker y configurarlo para un despliegue de aplicaciones aislado, ligero y portable, la base de la infraestructura moderna.

* 1. Instalación y prueba de Docker (**comandos que se usa para probar que Docker está instalado correctamente.**

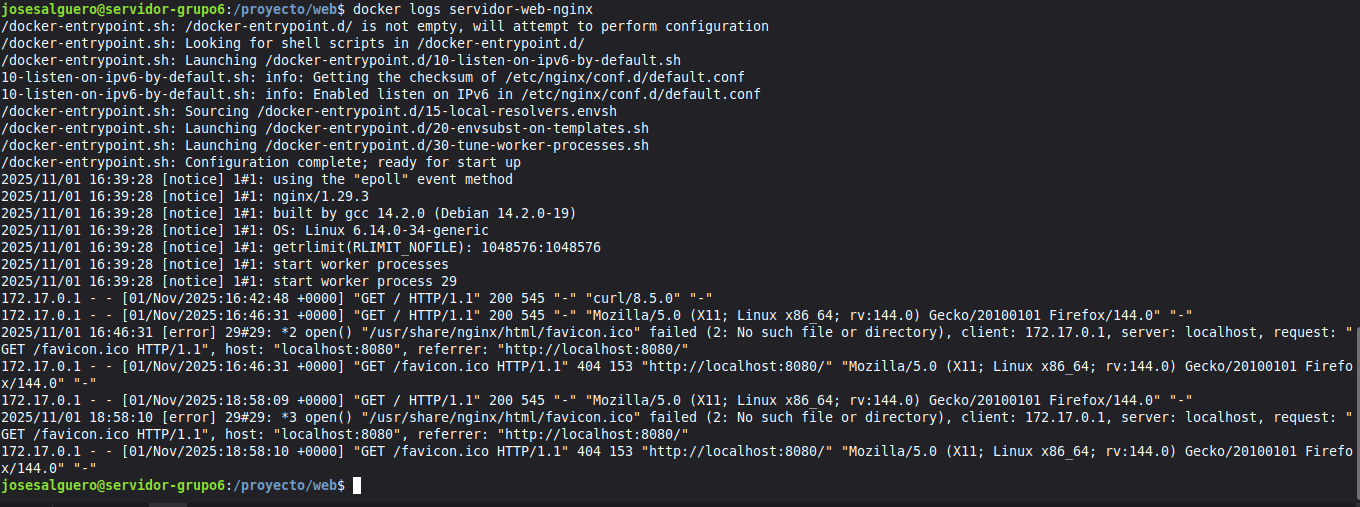


* 1. **Probando que** los contenedores funcionan bien en tu sistema.)





* 1. Ver todo los logs. Comando de Docker que se utiliza para ver los registros (logs) de un contenedor en ejecución.

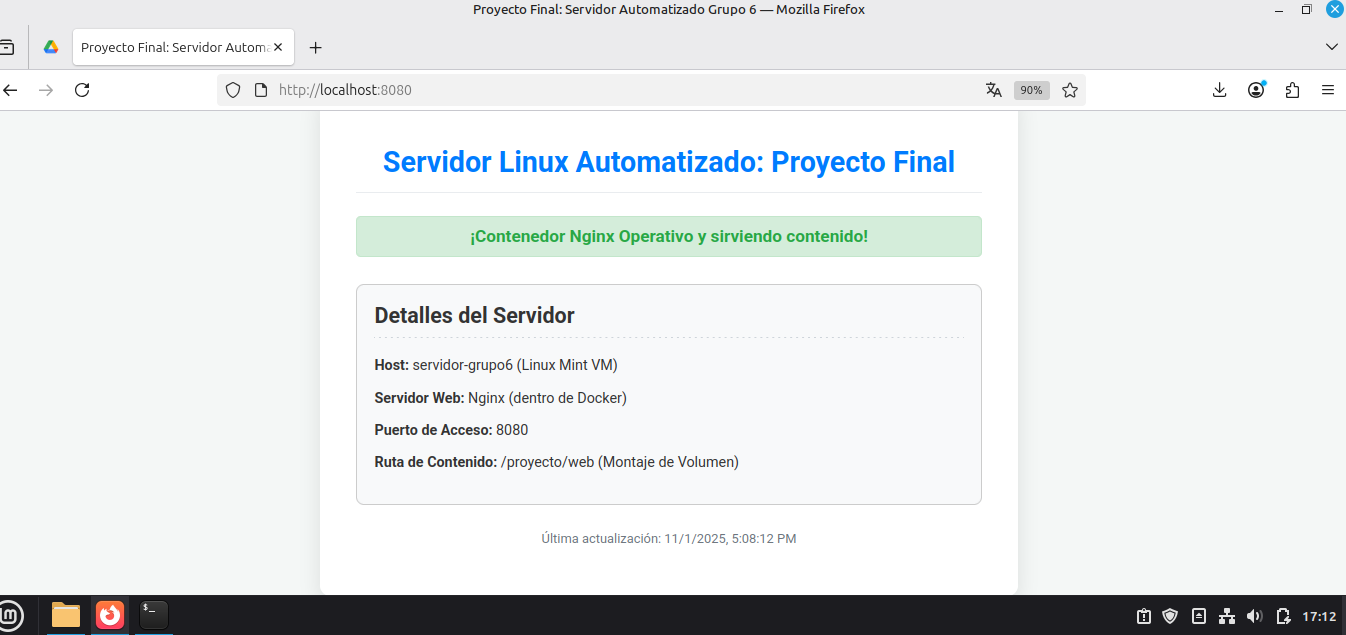


**5. Fase: Servidor Web Containerizado 🌐**

El **Objetivo** fue desplegar el servicio web principal del proyecto de manera eficiente, segura y con alta disponibilidad.

* **Ejecución Persistente:** El contenedor se lanzó con el modo **detached (-d)** y la política **--restart unless-stopped**. Esto garantiza que Nginx se inicie automáticamente cada vez que la VM se encienda, asegurando la **alta disponibilidad** del servicio.

Probando servicios con http://localhost: 8080



* **Montaje de Volumen:**  
  Se utilizó la bandera -v /proyecto/web:/usr/share/nginx/html:ro para montar el directorio local dentro del contenedor.
  + Esta técnica permite separar el código del servidor (por ejemplo, index.html).
  + El modo :ro (solo lectura) añade seguridad, previniendo cambios accidentales o maliciosos desde el contenedor.
* **Verificación Final:**  
  Se comprobó el acceso al servicio a través del puerto 8080 del host (mapeado al puerto 80 del contenedor).  
  La revisión de logs mediante docker logs confirmó la recepción de peticiones con código de estado 200 (OK), demostrando que Nginx está sirviendo el contenido correctamente.