Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Redes (IC 7602)

Proyecto II

Valor: 30%

Segundo Semestre 2022

Integrantes de grupo:

- Aaron Vargas Valerin
- Ingrid Fernández Arce
- Daniel Barrantes Esquivel
- Adriana López Calderón

Instrucciones de cómo ejecutar el proyecto en linux(Ubuntu)

Pre requisitos:

Descargar o clonar los archivos del proyecto que se encuentra en el repositorio de Github

<Aaron70/dns-interceptor (github.com)>

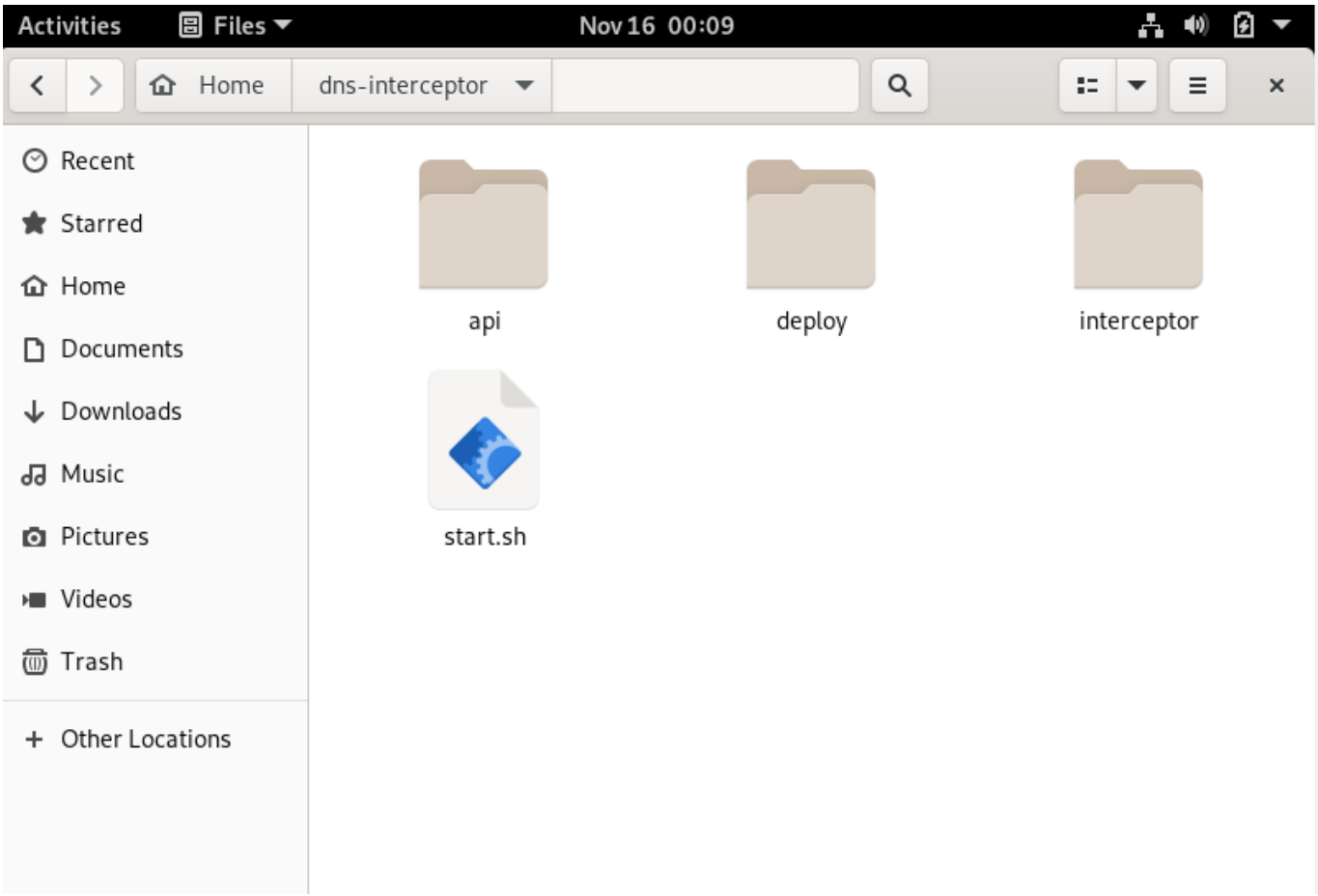
> Para más información sobre cómo clonar un repositorio de GitHub, ingresar al siguiente link:

https://docs.github.com/es/repositories/creating-and-managing-repositories/cloning-a-repository?platform=linux

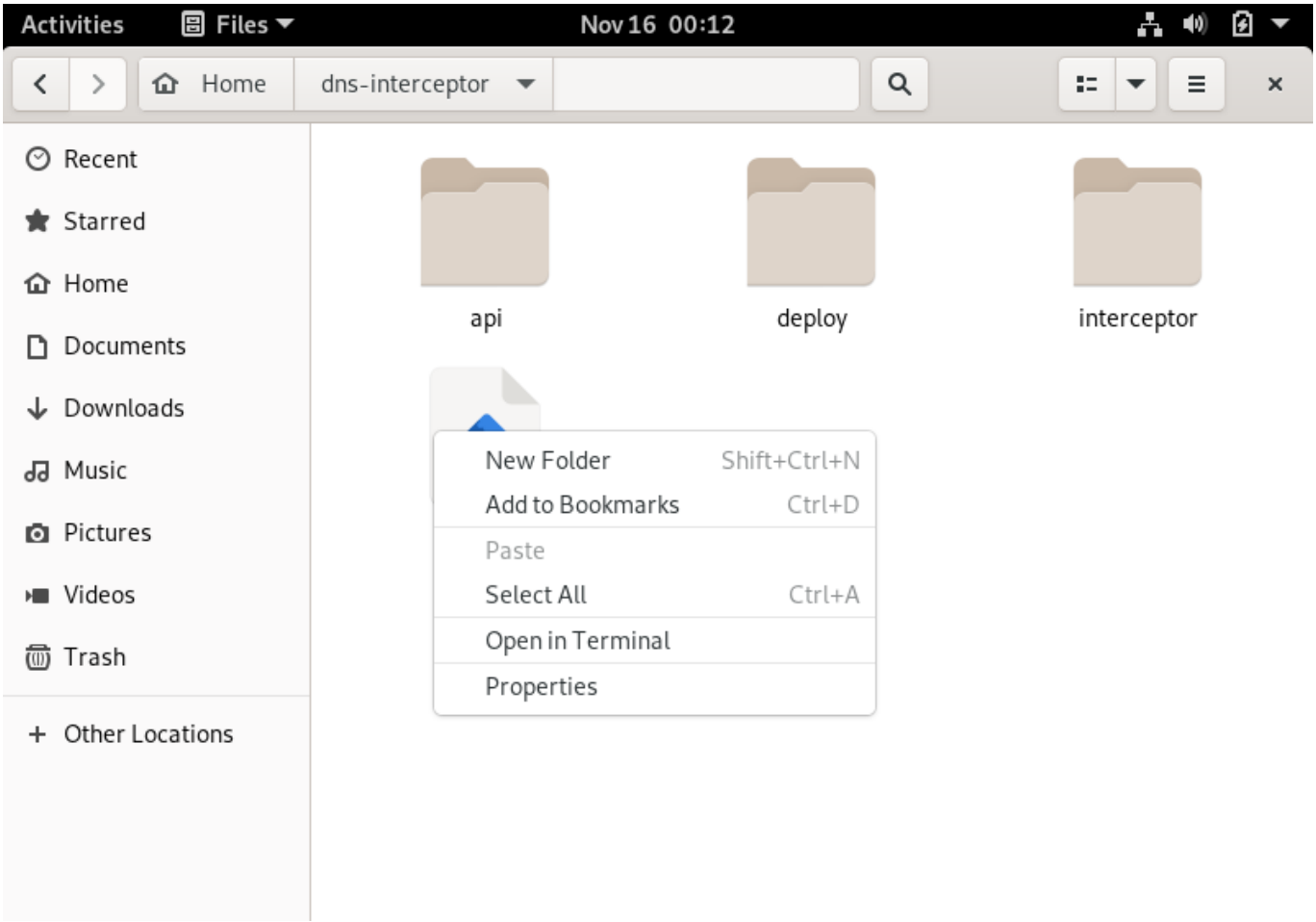
Instrucciones de ejecución:

Ir a la carpeta donde se guardó o clonó los documentos del Proyecto

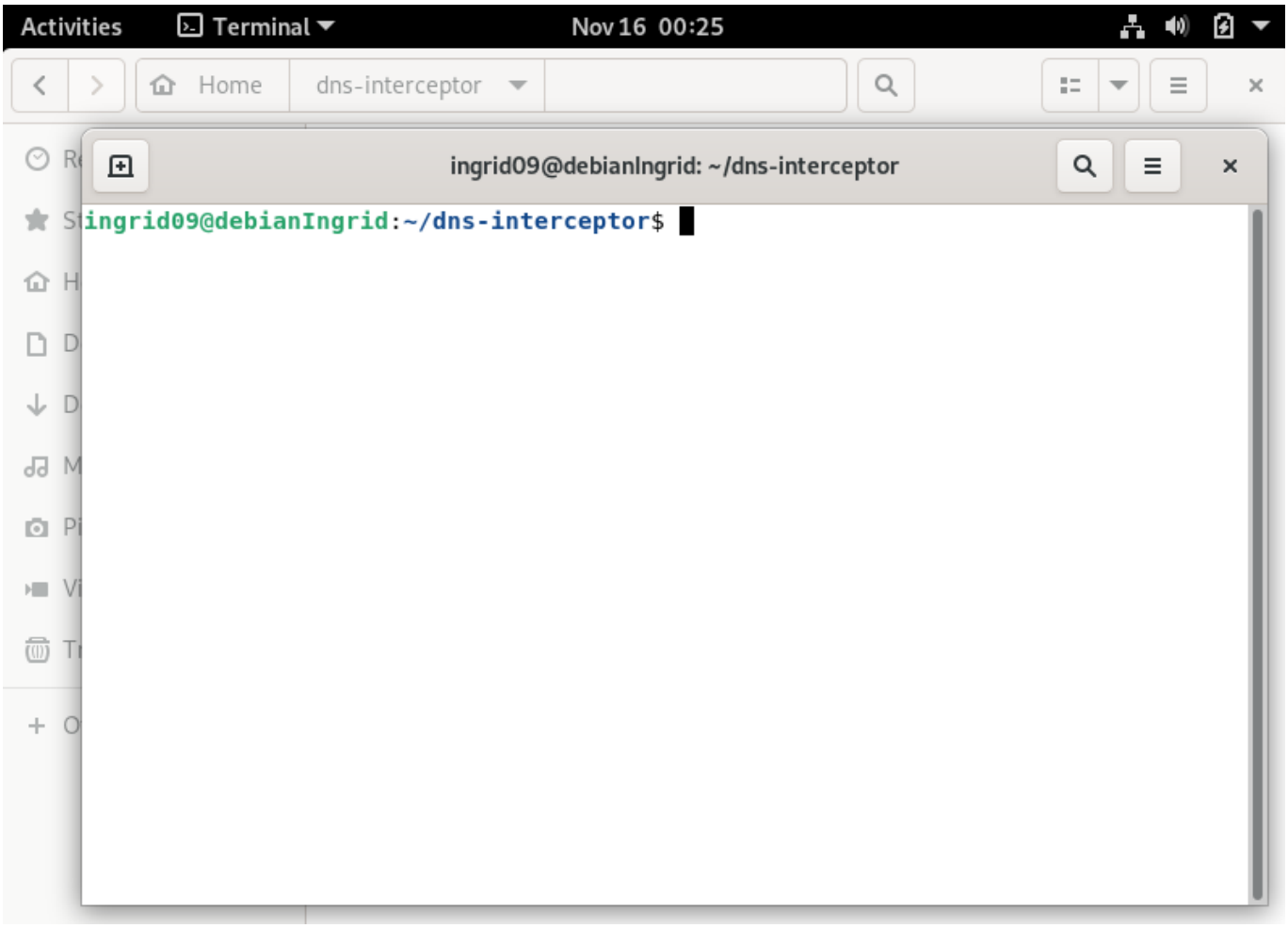
Abrir la carpeta llamada “dns-interceptor”. En ella se encontrará los siguientes archivos



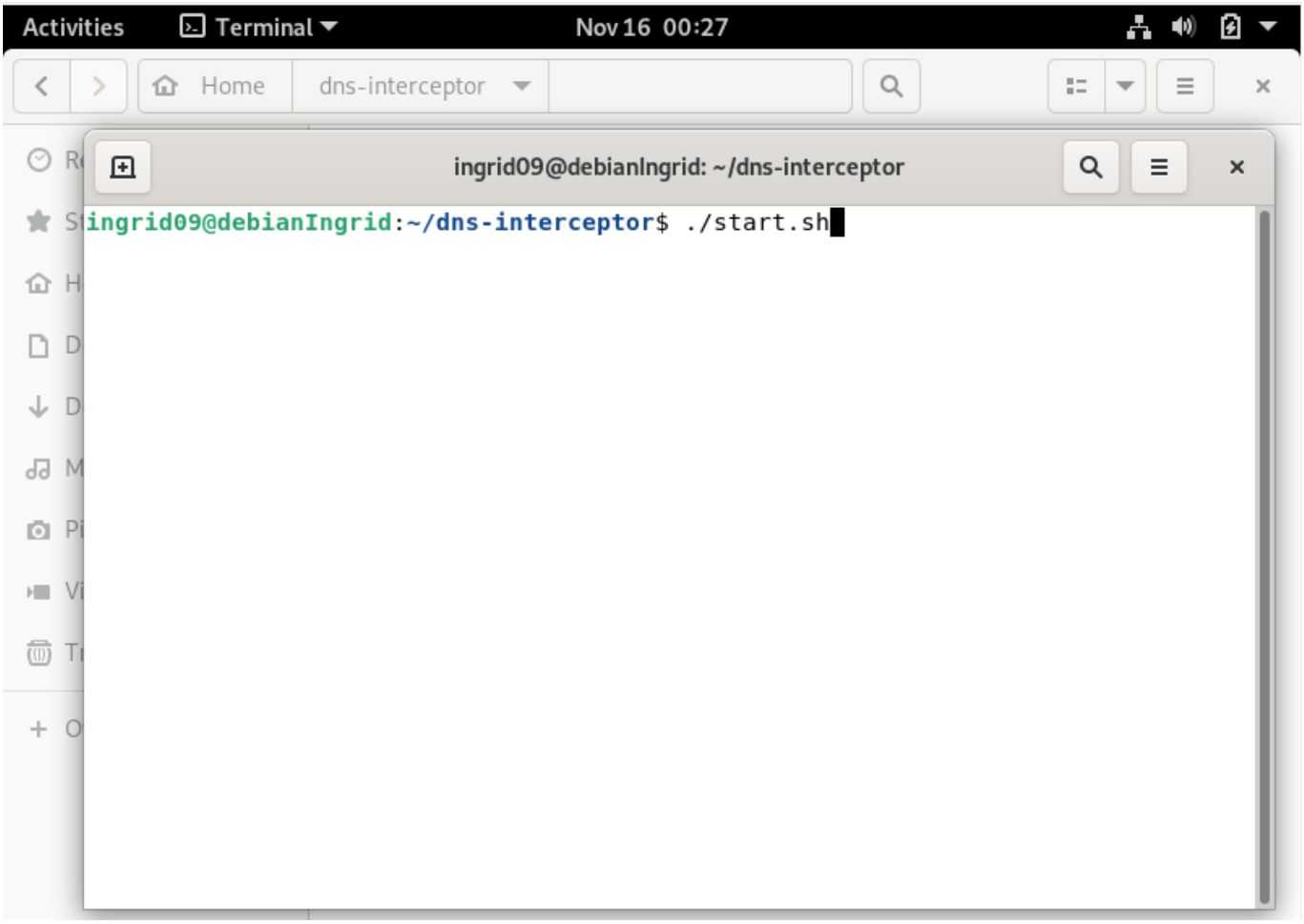
> Dar click derecho y seleccionar la opción “Open in terminal”



Deberá aparecer una ventana similar a la siguiente



Ejecutar el archivo "./start.sh"



Luego se empezará a construir los componentes necesarios

Instalaciones necesarias

Kubectl

Para saber cómo instalarlo, ingresar al siguiente link: Install and Set Up kubectl on Linux | Kubernetes

Minikube

Para saber cómo instalarlo, ingresar al siguiente link: minikube start | minikube (k8s.io)

Helmchart

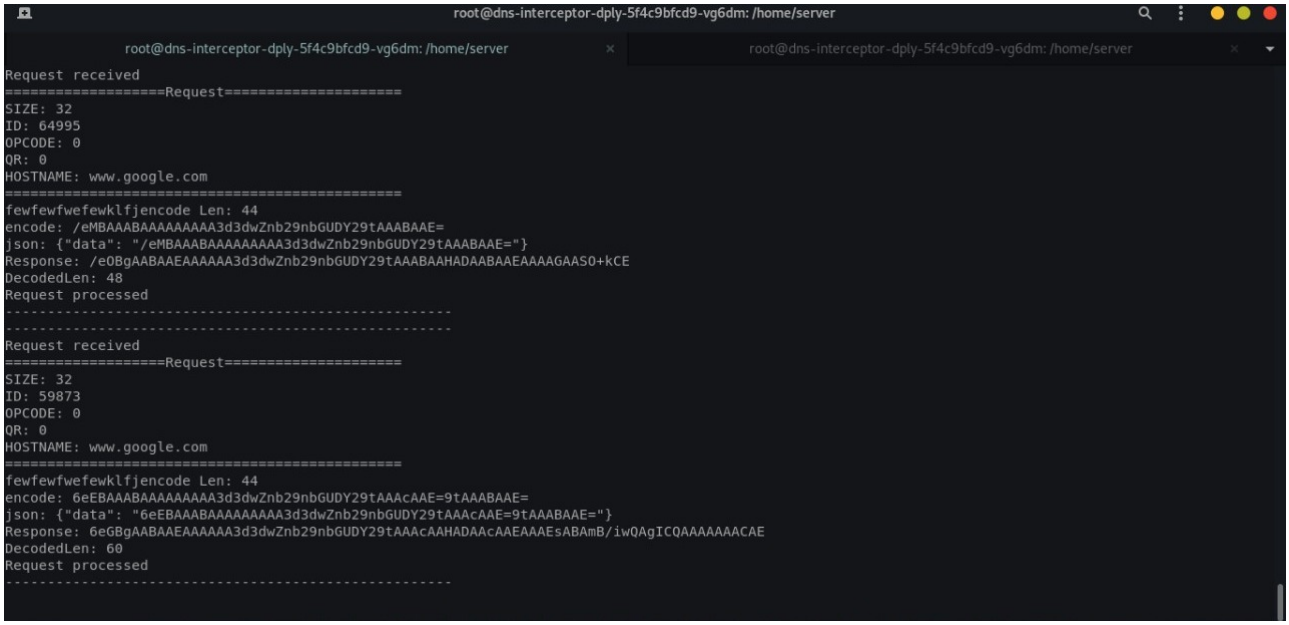
Para saber cómo instalarlo, ingresar al siguiente link: Helm | Installing Helm

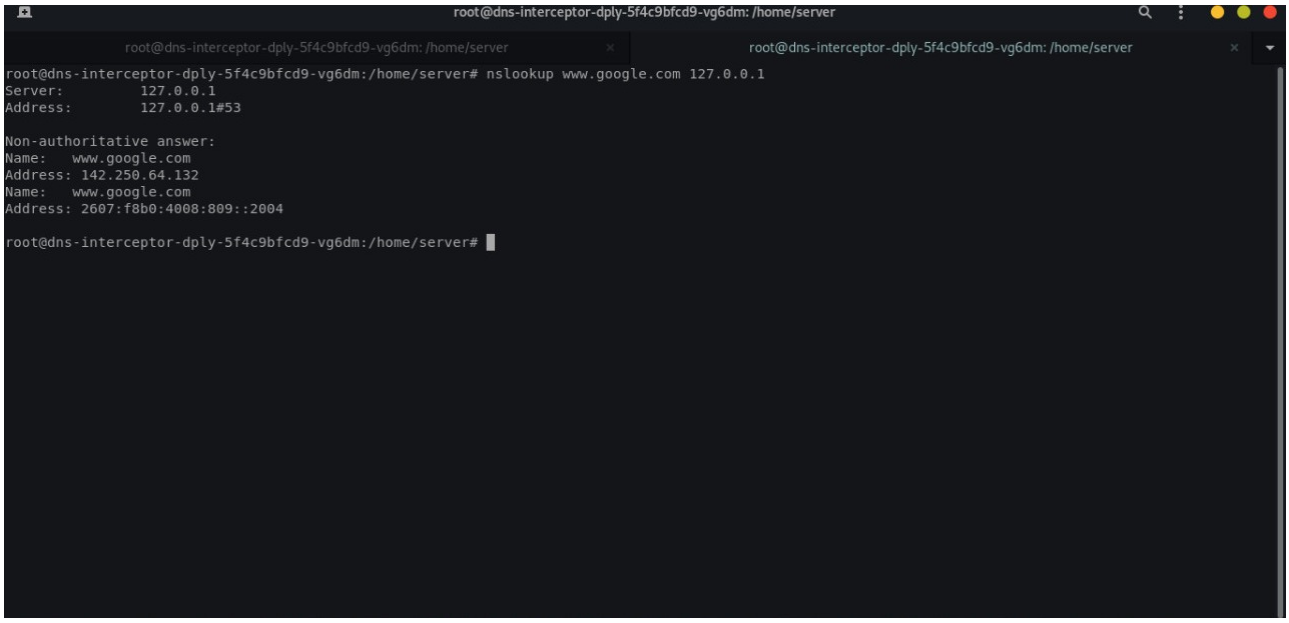
Docker

Para saber cómo instalarlo, ingresar al siguiente link: Install on Ubuntu | Docker Documentation

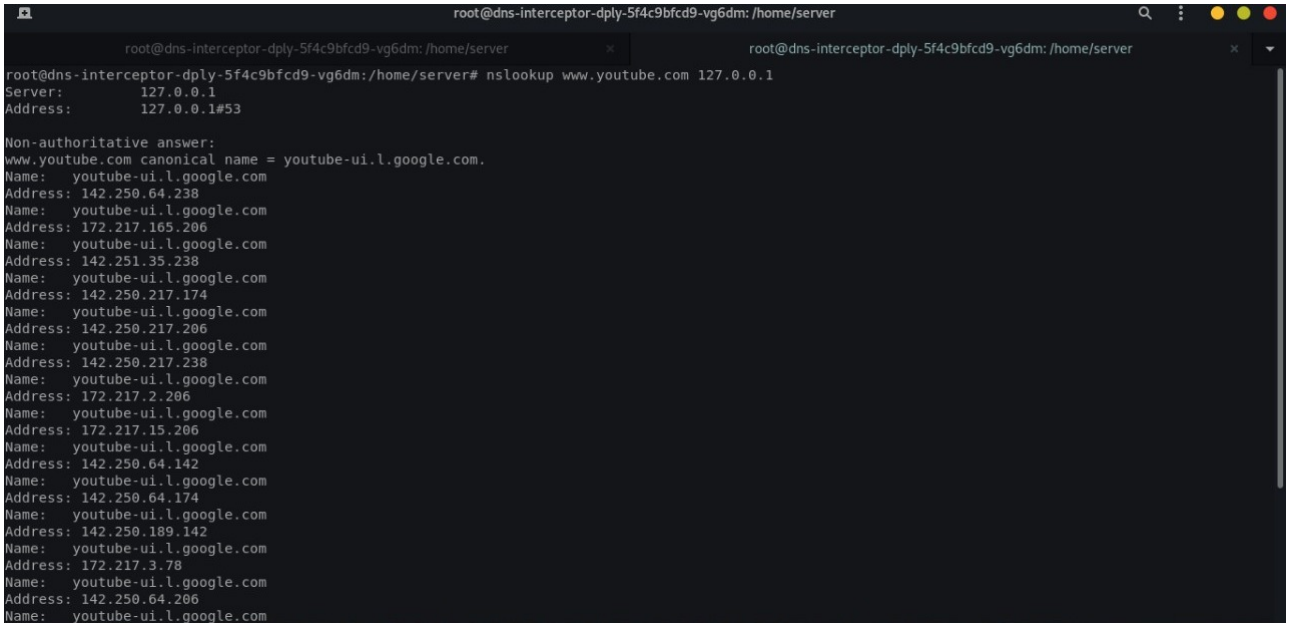
Pruebas de funcionalidad

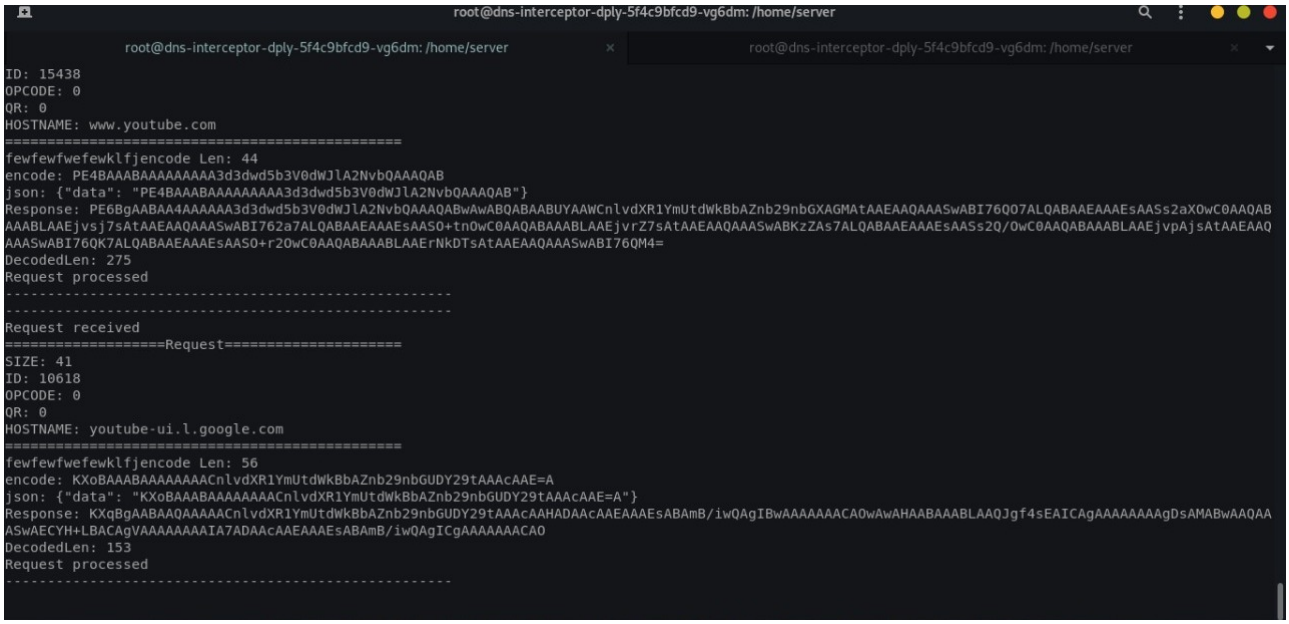
1. Prueba del DNS al resolver www.google.com



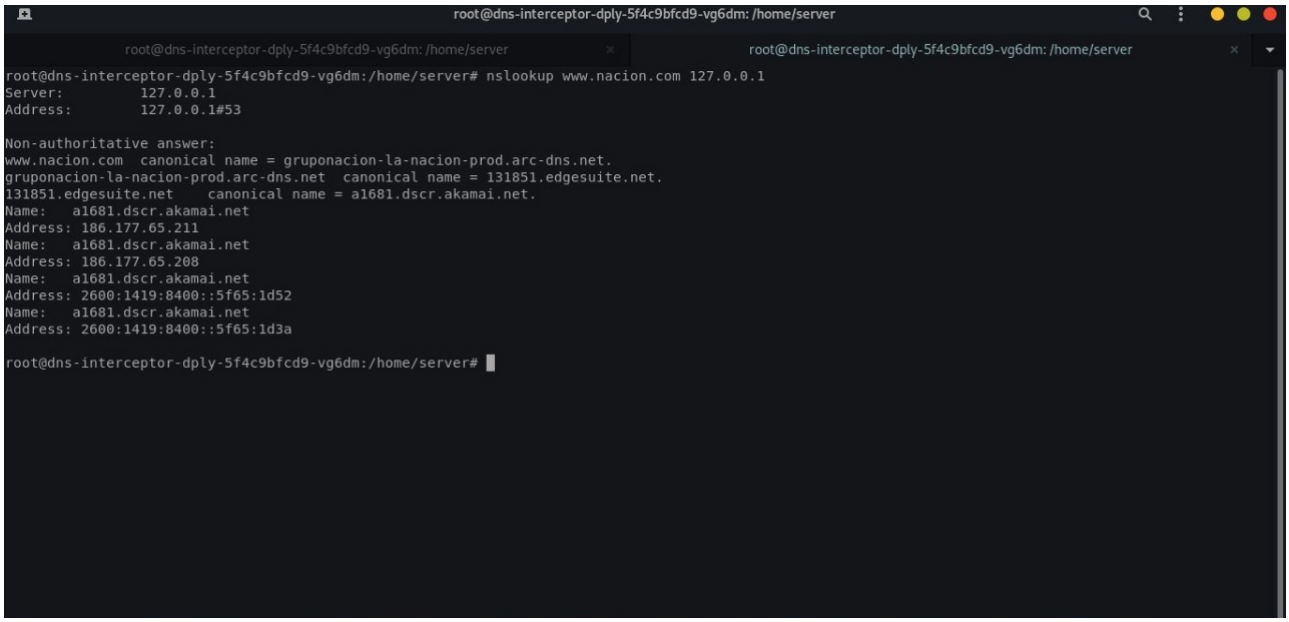


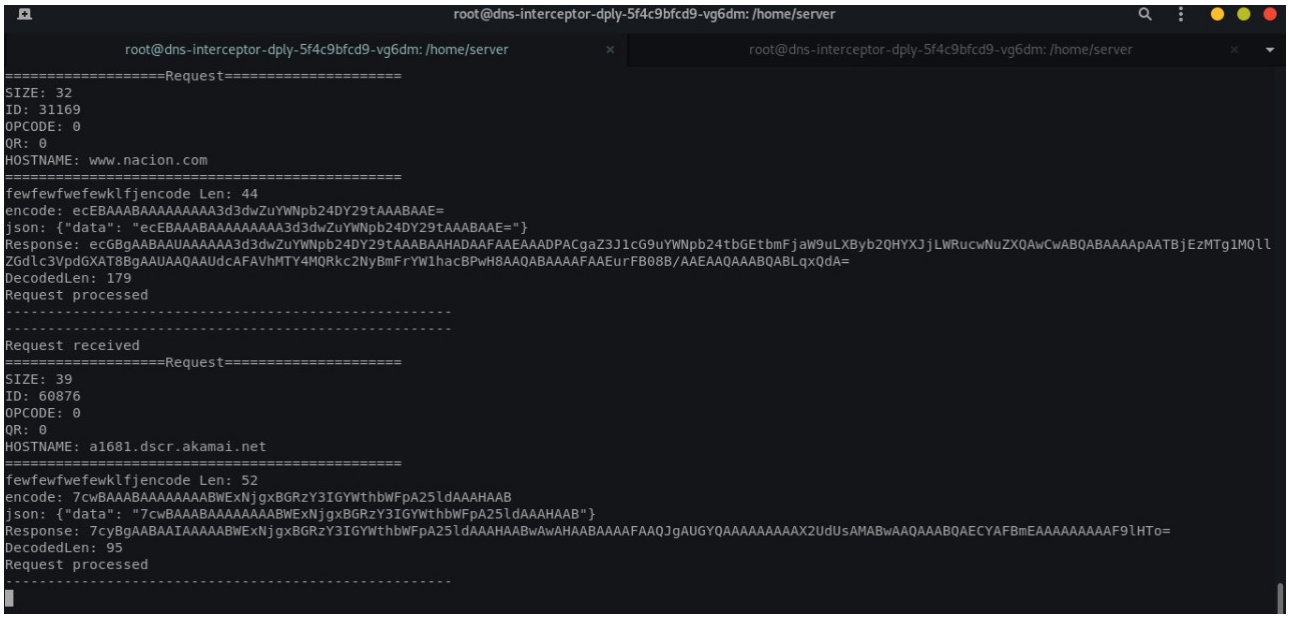
2. Prueba del DNS al resolver www.youtube.com





3. Prueba del DNS al resolver www.nacion.com





Recomendaciones

- Al iniciar se debe asegurar tener todos los paquetes y dependencias necesarias instaladas correctamente
- Instalar aquellas dependencias faltantes
- La mejor manera de crear archivos JSON o la estructura de uno en C es con la librería llamada Jansson.
- Si se usa la librería Jansson en C no hay que olvidar usar la bandera -ljansson a la compilar el archivo con gcc.
- Para trabajar con curl en C y hacer peticiones HTTPS, la manera más sencilla es usar la librería Libcurl.
- Si se usa la librería Libcurl en C no hay que olvidar usar la bandera -lcurl a la hora de compilar.
- Al usar la Libcurl y crear la función no hay que olvidar configurarlo para que pueda trabajar con JSON y configurar la función añadiendo al header "Accept: application/json" y "Content-Type: application/json".
- Si se instala elascticsearch en una pc sin usar contenedor no hay que olvidar que por defecto usa un 50 % de RAM.
- Antes de instalar elascticsearch no hay que olvidar instalar JAVA.
- Si se está trabajando con una estructura de NSLOOKUP y se tiene un array de bytes, para poder interpretar esos bytes y verlos de una manera legible se podría usar DNSRECORD en python de la librería DNSLIB.
- Para hacer codificación a base 64 en python la manera más fácil es por medio de la librería (base64).
- Si se reserva memoria MALLOC no hay que olvidar incluir el carácter '\0' dado que si se calcula mal el tamaño daría un error en la reserva del bloque de memoria.

Conclusiones

- La librería Jansson realmente resulta muy útil para trabajar JSON en C, realmente pensábamos que iba ser más difícil dado el tipo de lenguaje de programación sin embargo con esta herramienta resultó más fácil.
- La librería Libcurl al usarla también resultó de mucha utilidad, las peticiones se recibieron y se enviaron de una manera muy sencilla gracias a esta herramienta.
- Poder recibir el paquete de la manera correcta del NSLOOKUP resultó todo un reto dado que tuvimos que hacer muchas pruebas de lo recibido, y así poder armar la estructura correspondiente de un paquete dado el RFC2929.
- El usar elascticsearch nos resultó muy interesante. Es una herramienta que no conocíamos por lo que al trabajar con él nos pareció una herramienta muy potente y muy útil, sin embargo el consumo de la memoria RAM al inicio nos pareció algo sorprendente, dado que no creíamos que consumiera tanto.
- Todos los miembros del equipo trabajamos bajo el entorno Linux (Debian y Ubuntu) por lo que muchas cosas se nos facilitó dado que teníamos algo de experiencia con la terminal de este sistema operativo, sin embargo de igual manera muchas veces aparecieron retos que tuvimos que ir

solucionando por lo que nos llevó a una curva de aprendizaje en Linux, fortaleciendo nuestros conocimientos en el mismo.

- Las documentaciones de muchas herramientas eran muy útiles, por lo que nos brindó un mejor conocimiento de algunas usadas, de igual manera muchos foros de internet nos aportaban muchas ideas o nos brindaban mucha ayuda en distintas dificultades que se nos presentaron en el momento.
- Docker es una herramienta muy utilizada en el mercado, trabajar con él nos ayudó a fortalecer nuestros conocimientos y tener un mejor conocimiento de las herramientas utilizadas en el ámbito laboral.
- El poder levantar una imagen de docker con Ubuntu por ejemplo, nos ayudó a hacer muchas pruebas a la hora de trabajar, ayudándonos a no hacerlas en la máquina física, por lo que si aparecía algún error no habría ningún problema, por lo que fue una herramienta de gran utilidad para este proyecto, y estamos seguros que para próximos también nos ayudará.
- Para crear el API con el método POST en python se usó FLASK, resultó ser muy sencillo una vez leyendo la documentación e información general en distintos sitios de internet, por lo que resultó ser de gran utilidad aprender cómo usar flask con python.
- Helm Charts y Kubernetes son herramientas que nos parecieron muy útiles, ningún miembro del equipo había trabajado nunca con ninguno de los 2, por lo que tuvo una curva de aprendizaje en cada uno de nosotros, realmente nos pareció bastante fascinante como con estas herramientas se pueden automatizar muchas haciendo distintas configuraciones con cada uno.