

ACTIVIDAD 12: Servidor Video Streaming – Ubuntu Server



KEVIN NICOLÁS SIERRA GONZÁLEZ 20182020151

LUIS MIGUEL POLO 20182020158

YEISON ALEXANDER FARFAN PERALTA 20201020138

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

INGENIERÍA DE SISTEMAS

TELEINFORMATICA I

ANDRES ALEXANDER RODRIGUEZ FONSECA

2024-III

Objetivos

- Implementar servicios sobre Ubuntu Server
- Implementar un servidor de Streaming

Materiales

- Computador personal con acceso a Internet
- Máquina virtual con Ubuntu Server 24.04 y servicio SSH instalado

Procedimiento

Se crea una instancia con el nombre de “Streaming” y se selecciona la distribución de Ubuntu, el grupo de seguridad creado para la actividad anterior y la misma clave generada de extensión .ppk para usar con PuTTY y se lanza la instancia.

Nombre

Streaming

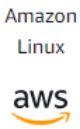

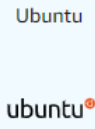




[Agregar etiquetas adicionales](#)

▼ **Imágenes de aplicaciones y sistemas operativos (Imagen de máquina de Amazon)**

[Información](#)

Una AMI es una plantilla que contiene la configuración de software (sistema operativo, servidor de aplicaciones y aplicaciones) necesaria para lanzar la instancia. Busque o examine las AMI si no ve lo que busca a continuación.

[Recientes](#) | [Inicio rápido](#)

 Amazon Linux	 macOS	 Ubuntu	 Windows	 Red Hat	 SUSE	 Buscar más AMI Inclusión de AMI de AWS, Marketplace y la comunidad
-----------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

- 1) Comenzamos utilizando la instancia que nos ofrece AWS para montar la maquina y servidor de Ubuntu.
 - a. Elegimos el mismo par de claves ssh de la práctica pasada, al igual que el mismo grupo de seguridad pasado.
- 2) Una vez entrada en ejecución la instancia de Ubuntu, copiamos la dirección IPv4 pública.

<input checked="" type="checkbox"/>	Streaming	i-00ceb5560de805f92	✓ En ejecución	t2.micro	Inicializando	Ver alarmas +	us-e
<input type="checkbox"/>	ServerVoIP	i-09806a9055aa874e1	⏸ Detenida	t2.small	-	Ver alarmas +	us-e

i-00ceb5560de805f92 (Streaming)

ID de la instancia
i-00ceb5560de805f92

Dirección IPv6
-

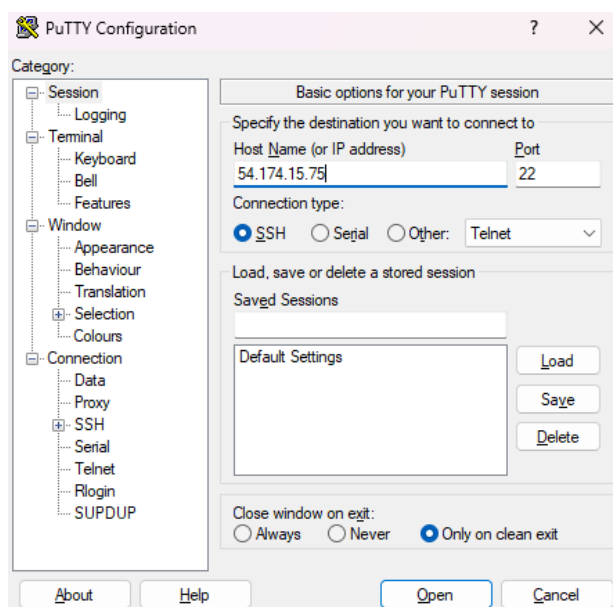
Dirección IPv4 pública
54.174.15.75 | [dirección abierta](#)

Estado de la instancia
✓ En ejecución

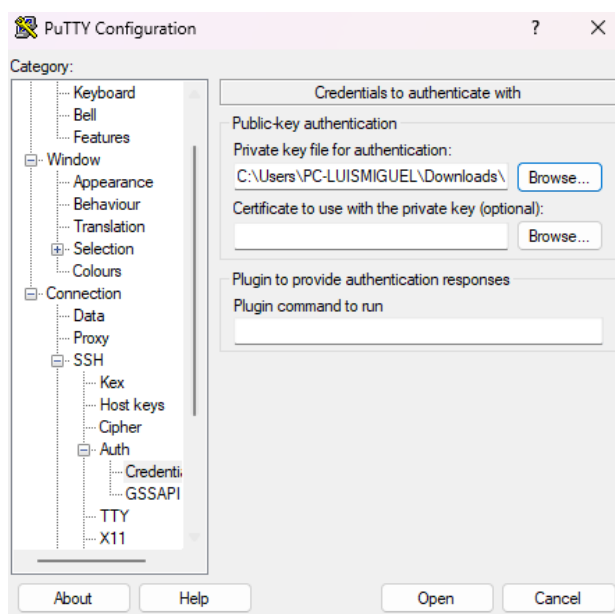
Direcciones IPv4 privadas
172.31.32.246

DNS de IPv4 pública
ec2-54-174-15-75.compute-1.amazonaws.com | [dirección abierta](#)

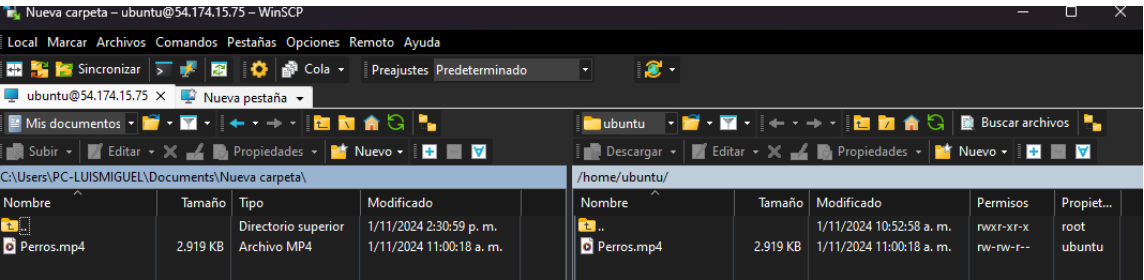
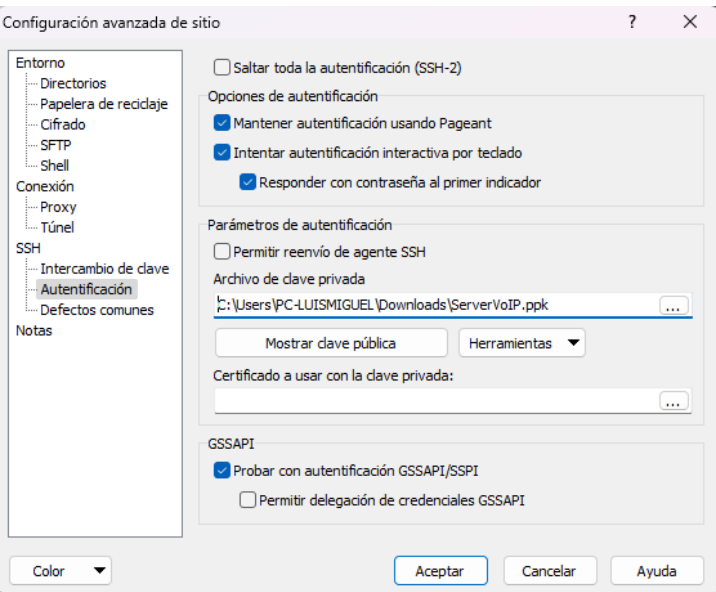
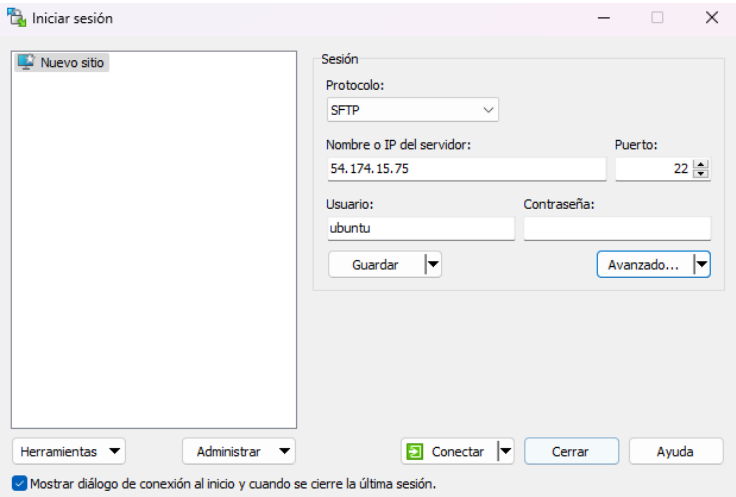
3) Abrimos el PUTTY para realizar la conexión y copiamos la misma dirección pública del servidor.



4) Mantenemos las mismas par de claves de ssh.



Utilizando WinSCP realizamos la transferencia del video descargado desde nuestra computadora local a ubuntu a través del protocolo SFTP copiando la misma dirección publica anterior y con la misma clave ssh.



Inicializamos Ubuntu en PUTTY como máquina virtual.

```
login as: ubuntu
Authenticating with public key "ServerVoIP"
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Fri Nov  1 19:34:13 UTC 2024

System load:  0.0               Processes:    114
Usage of /:   34.6% of 6.71GB   Users logged in:  1
Memory usage: 25%              IPv4 address for enX0: 172.31.32.246
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

38 updates can be applied immediately.
19 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

10 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm
```

Cargamos el video previamente montado al servidor de Ubuntu

```
root@ip-172-31-32-246:/home/ubuntu# ls
Perros.mp4
root@ip-172-31-32-246:/home/ubuntu#
```

Instalamos el servidor NGINX - RTMP

```
root@ip-172-31-32-246:/home/ubuntu# apt install libnginx-mod-rtmp
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

Configuración de fichero nginx.conf agregando las siguientes líneas al final del archivo:

```
GNU nano 7.2 /etc/nginx/nginx.conf
#       server {
#           listen        localhost:143;
#           protocol      imap;
#           proxy          on;
#       }
#}

rtmp {
    server {
        listen 1935;
        chunk_size 4096;
        allow publish 127.0.0.1;
        deny publish all;
        application live {
            live on;
            record off;
        }
    }
}

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute  ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify  ^_ Go To Line
```

Instalamos FFMPEG para transmitir audio y video en el servidor.

```
root@ip-172-31-32-246:/home/ubuntu# apt install ffmpeg
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

Reiniciamos el servicio de nginx.

```
root@ip-172-31-32-246:/home/ubuntu# systemctl restart nginx
root@ip-172-31-32-246:/home/ubuntu# systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Fri 2024-11-01 17:15:25 UTC; 8s ago
     Docs: man:nginx(8)
   Process: 4830 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited)
   Process: 4833 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited)
  Main PID: 4834 (nginx)
    Tasks: 2 (limit: 1130)
   Memory: 1.8M (peak: 2.0M)
      CPU: 11ms
   CGroup: /system.slice/nginx.service
           └─4834 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─4835 "nginx: worker process"

Nov 01 17:15:25 Ubserver systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server:
Nov 01 17:15:25 Ubserver systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server:
lines 1-16/16 (END) ...skipping...
```

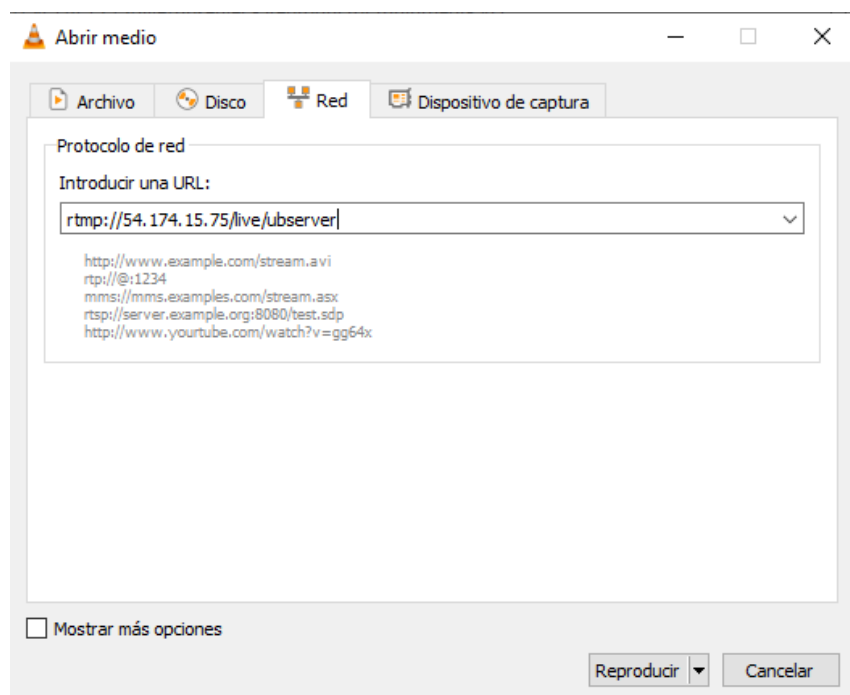
Se debe realizar el envío de video descargado por RTMP.

```
root@ip-172-31-32-246:/home/ubuntu# ffmpeg -re -i "Perros.mp4" -c copy -f flv rtmp://localhost/live/ubserver
```

Comienza a correr el servicio de streaming en el servidor de Ubuntu.

```
root@ip-172-31-32-246: /home/ubuntu
handler_name : ISO Media file produced by Google Inc. Created on: 08/15/2023.
vendor_id    : [0][0][0][0]
Output #0, flv, to 'rtmp://localhost/live/ubserver':
Metadata:
  major_brand      : mp42
  minor_version    : 0
  compatible_brands: isommp42
  encoder         : Lavf60.16.100
Stream #0:0(und): Video: h264 (Main) ([7][0][0][0] / 0x0007), yuv420p(tv, bt709, progres
bn (default)
Metadata:
  creation_time : 2023-08-15T18:20:08.000000Z
  handler_name  : ISO Media file produced by Google Inc. Created on: 08/15/2023.
  vendor_id    : [0][0][0][0]
Stream #0:1(eng): Audio: aac (LC) ([10][0][0][0] / 0x000A), 44100 Hz, stereo, fltp, 127
Metadata:
  creation_time : 2023-08-15T18:20:08.000000Z
  handler_name  : ISO Media file produced by Google Inc. Created on: 08/15/2023.
  vendor_id    : [0][0][0][0]
Stream mapping:
  Stream #0:0 -> #0:0 (copy)
  Stream #0:1 -> #0:1 (copy)
Press [q] to stop, [?] for help
size= 1559kB time=00:00:33.16 bitrate= 385.0kbits/s speed=1.02x
```

En esta parte un integrante el grupo descargo e hizo la configuración correspondiente para la conexión con el servidor desde VLC.



Correcta visualización con éxito del video y correcto funcionamiento del servicio de streaming.



Análisis y Discusión

1. Explique el funcionamiento del protocolo RTMP (Real-Time Messaging Protocol) y cómo se utiliza en la transmisión de video en tiempo real. ¿Cuáles son las ventajas y desventajas de utilizar RTMP en comparación con otros protocolos de transmisión de video en tiempo real como HLS o DASH?

Rta: El protocolo RTMP (Real-Time Messaging Protocol) fue desarrollado por Adobe para la transmisión de video, audio y datos en tiempo real. Funciona mediante una conexión TCP que permite un flujo constante de datos, asegurando una baja latencia, por lo que es ideal para transmisiones en vivo. RTMP divide la transmisión en pequeñas partes (chunks) y las envía a través de una conexión persistente entre el servidor y el cliente, permitiendo que el usuario reciba el contenido de forma continua sin interrupciones. Este protocolo fue diseñado inicialmente para trabajar con Adobe Flash Player, aunque también se utiliza en aplicaciones de transmisión en vivo hacia plataformas como YouTube y Facebook Live.

En comparación con otros protocolos como HLS (HTTP Live Streaming) o DASH (Dynamic Adaptive Streaming over HTTP), RTMP tiene la ventaja de ofrecer una latencia baja y una transmisión en tiempo real más fluida debido a su transmisión directa y continua. Sin embargo, presenta algunas desventajas: requiere una conexión constante entre el servidor y el cliente, lo cual puede ser demandante para la red, y no está optimizado para funcionar en redes móviles o conexiones con limitaciones de

ancho de banda. Además, HLS y DASH son más compatibles con dispositivos modernos y navegadores web, ya que funcionan sobre HTTP y permiten una adaptación dinámica de la calidad del video según la velocidad de la red, lo cual mejora la experiencia del usuario en condiciones de red variables.

2. Discuta el uso del protocolo HTTP en la configuración de un servidor Nginx para transmitir video en tiempo real. ¿Cómo se configura Nginx para manejar solicitudes HTTP y RTMP simultáneamente, y cuáles son los desafíos de seguridad asociados con esta configuración?

Rta: El protocolo HTTP se utiliza ampliamente en la configuración de servidores Nginx para la transmisión de video en tiempo real, especialmente mediante el uso de protocolos de streaming adaptativo como HLS (HTTP Live Streaming). Nginx actúa como un servidor de medios que almacena y distribuye segmentos de video mediante HTTP, permitiendo a los clientes acceder a la transmisión en directo con baja latencia y una calidad adaptable. Para esta configuración, Nginx necesita el módulo `nginx-rtmp-module`, que permite agregar funcionalidad RTMP y así manejar transmisiones en vivo, lo cual es útil para soportar una mayor cantidad de formatos de streaming en paralelo.

Configurar Nginx para manejar solicitudes HTTP y RTMP simultáneamente implica añadir configuraciones específicas en el archivo de configuración de Nginx para definir diferentes bloques de servidores, uno para cada protocolo. Se define un servidor HTTP para las solicitudes HLS y otro para las solicitudes RTMP, permitiendo así que el servidor administre ambas conexiones y entregue contenidos de video en tiempo real a los usuarios en distintos formatos según su dispositivo o necesidad.

Entre los desafíos de seguridad asociados, uno de los principales es el riesgo de ataques DDoS, que pueden sobrecargar el servidor al enviar múltiples solicitudes HTTP o RTMP simultáneamente. Además, tanto HTTP como RTMP pueden ser susceptibles a intentos de acceso no autorizado, por lo que se recomienda implementar autenticación, cifrado SSL/TLS para HTTP, y técnicas de filtrado de IP y restricciones de acceso para RTMP. También es importante monitorizar el servidor y utilizar firewalls para proteger el ancho de banda y mantener el rendimiento y la estabilidad del servicio.

3. Explique cómo el protocolo TCP asegura la transmisión confiable de datos en la transmisión de video en tiempo real con RTMP. ¿Cuáles son las limitaciones de

usar TCP para transmisión en tiempo real, y cómo se compara con el uso de UDP en este contexto?

Rta: El protocolo TCP asegura una transmisión confiable de datos en la transmisión de video en tiempo real con RTMP mediante un proceso de control de flujo y corrección de errores. TCP fragmenta los datos en paquetes, envía cada uno, y garantiza su entrega mediante un sistema de confirmación (acknowledgment): cada vez que el receptor recibe un paquete, envía una confirmación al emisor. Si algún paquete se pierde o llega corrupto, TCP lo reenvía, asegurando que el contenido llegue completo y en el orden correcto. Este enfoque es adecuado para RTMP, ya que ayuda a mantener la integridad del video y evita que se generen saltos o interrupciones en la transmisión.

Sin embargo, el uso de TCP en transmisiones en tiempo real tiene limitaciones. La retransmisión de paquetes perdidos genera latencia, lo que puede afectar la sincronización y fluidez en transmisiones en vivo, ya que el protocolo prioriza la precisión sobre la velocidad. En este sentido, UDP es una alternativa comúnmente usada para transmisión en tiempo real, ya que no garantiza la entrega de todos los paquetes y, en lugar de retransmitir, permite que la transmisión continúe sin interrupciones. Esto reduce la latencia y mejora la experiencia de visualización en tiempo real, aunque con el riesgo de perder algunos datos en la calidad de la transmisión.

4. Analice la importancia de los codecs de video y audio en la transmisión en tiempo real utilizando FFmpeg y RTMP. ¿Cómo afectan los codecs utilizados la calidad del video, la latencia y el ancho de banda requerido? Proporcione ejemplos de codecs comúnmente utilizados y sus características.

Rta: Los codecs de video y audio son esenciales en la transmisión en tiempo real con FFmpeg y RTMP, ya que comprimen y descomprimen los datos para optimizar su transmisión sin sacrificar la calidad. FFmpeg utiliza estos codecs para codificar y empaquetar el contenido de forma eficiente, permitiendo reducir el tamaño de los archivos y adaptarse al ancho de banda disponible. La elección del codec influye directamente en la calidad del video y audio, la latencia, y la cantidad de ancho de banda necesario: un codec bien optimizado permite una transmisión fluida y de buena calidad, incluso en redes limitadas.

Entre los codecs más utilizados para video en transmisiones en tiempo real están H.264 y H.265 (HEVC). H.264 es popular por su balance entre calidad y ancho de banda, siendo ampliamente compatible con diversos dispositivos y plataformas. H.265, aunque requiere más potencia de procesamiento, ofrece una mayor compresión con la misma calidad, reduciendo el ancho de banda necesario. En cuanto al audio, el codec AAC es común, ya que proporciona una buena calidad de sonido a tasas de bits bajas, lo cual es útil para minimizar el ancho de banda sin comprometer la claridad. Estos codecs ayudan a reducir la latencia al adaptar la calidad de la transmisión en función de las condiciones de red, lo que es crucial para lograr una experiencia de transmisión en tiempo real confiable y eficiente.

5. Discuta las consideraciones de seguridad y métodos de autenticación utilizados en la transmisión de video en tiempo real con RTMP. ¿Cuáles son las mejores prácticas para proteger una transmisión RTMP de accesos no autorizados y ataques? Proporcione ejemplos de configuraciones de Nginx para implementar estas prácticas.

Rta: La seguridad en la transmisión de video en tiempo real con RTMP es fundamental para evitar accesos no autorizados, robo de contenido y ataques que puedan afectar la estabilidad del servidor. Entre las principales consideraciones de seguridad está la implementación de métodos de autenticación para controlar el acceso a la transmisión, como la autenticación básica o tokens de autenticación, que permiten validar que solo usuarios autorizados puedan conectarse. Estos métodos ayudan a prevenir el acceso a contenido privado o sensible, asegurando que solo los clientes con permisos adecuados accedan a la transmisión.

Para proteger una transmisión RTMP, se recomienda también restringir el acceso por IP, limitar el ancho de banda y el número de conexiones por cliente, y utilizar cifrado TLS para las conexiones HTTP asociadas a las solicitudes de autenticación. Además, el uso de un firewall puede prevenir ataques DDoS, que intentan sobrecargar el servidor mediante múltiples solicitudes simultáneas. En cuanto a la configuración en Nginx, una práctica común es utilizar el módulo `nginx-rtmp-module` junto con autenticación tokenizada. Un ejemplo de configuración sería establecer una clave secreta para generar tokens de autenticación y usar variables de IP en el archivo de configuración de Nginx.

Bibliografía

https://en.wikipedia.org/wiki/Real-Time_Messaging_Protocol

https://en.wikipedia.org/wiki/Transmission_Control_Protocol

Crea tu servidor RTMP para streaming con nginx

<https://adictosaltrabajo.com/2017/06/21/crea-tu-servidor-rtmp-para-streaming-con-nginx/>

Alojamiento de servidores RTMP: Qué es y cómo acceder a él.

<https://www.dacast.com/es/blog-es/rtmp-servidor-hosting/>

NGINX: ¿qué es y en qué se distingue de Apache?

<https://rockcontent.com/es/blog/nginx/>