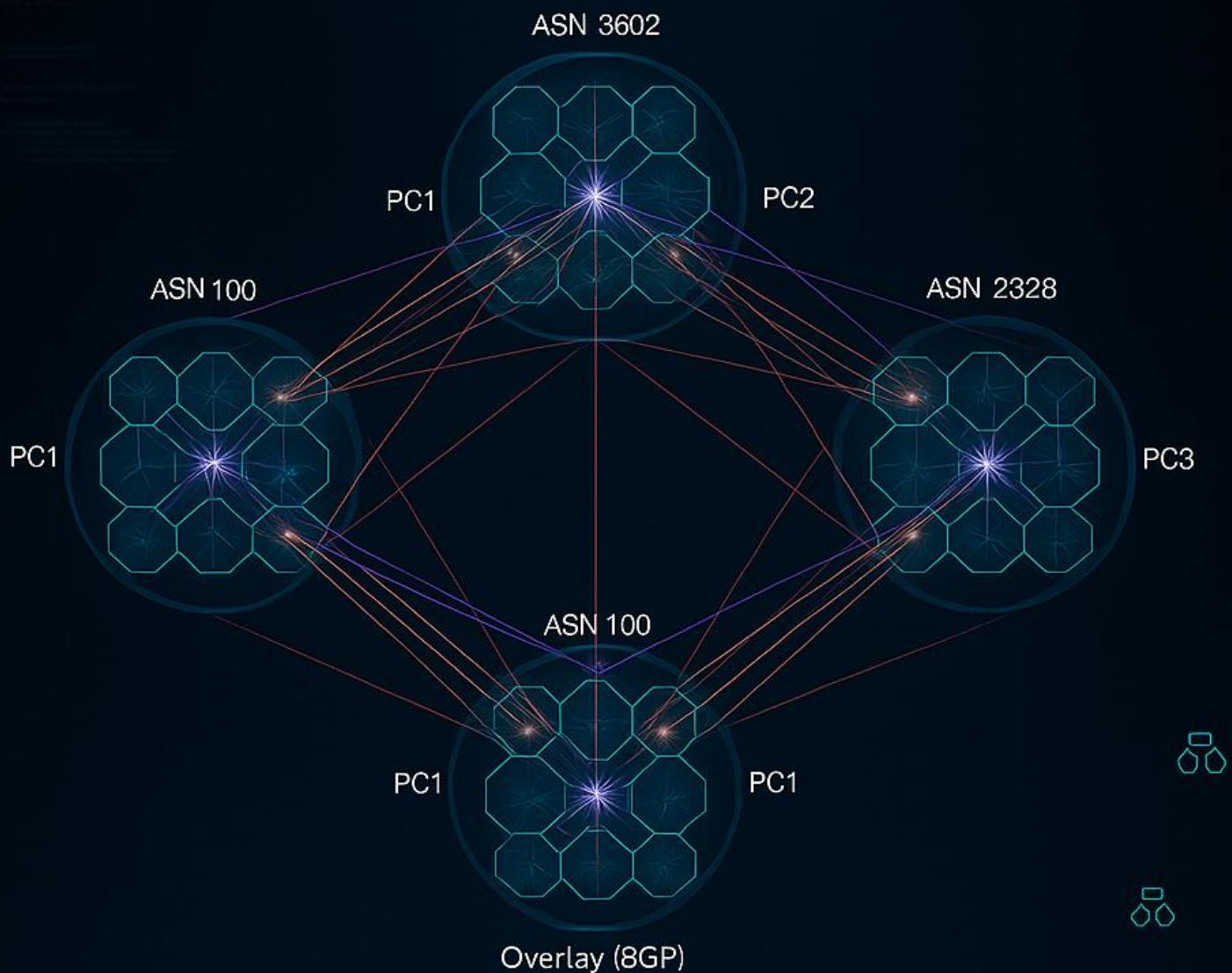


# Red BGP con Containerlab

Autor: Ricardo Sanabria Vega

Asignatura: Arquitectura de redes avanzadas

03-11-2025



**ÍNDICE:**

1. Introducción.....	3
2. Topología y Despliegue.....	3
3. Metodología de Configuración.....	4
4. Ejecución y Puesta en Marcha.....	6
5. Desafíos Técnicos y Soluciones .....	6
6. Rol de la Asistencia por IA .....	7
7. Conclusiones.....	8

## 1. Introducción

El objetivo de esta práctica era recrear fielmente el escenario de red planteado en la “Figura01”. El proceso de aprendizaje ha sido intenso, aportando conocimientos muy útiles sobre la arquitectura de redes desde una perspectiva profundamente técnica.

La ausencia total de una interfaz gráfica (GUI) fue un aspecto definitorio del reto. Esta limitación obligó a realizar todo el proceso de diseño y configuración de forma manual: desde planificar y dibujar la topología completa de 10 routers, 3 PCs y 5 Sistemas Autónomos (ASNs), hasta asignar a mano todo el esquema de direccionamiento IP y las políticas de enrutamiento.

Este enfoque "desde cero" ha sido fundamental para comprender la dependencia mutua de los protocolos (OSPF y BGP) y la importancia crítica de la automatización mediante scripts para gestionar y depurar redes complejas.

Para la realización, se ha requerido la instalación de dos programas principales en un entorno Linux:

- **Docker:** Como motor de contenedores base.
- **Containerlab:** Como herramienta de orquestación para el despliegue de laboratorios de red virtualizados.

## 2. Topología y Despliegue

Para el montaje del escenario se ha utilizado un fichero de topología “*Ejercicio2-2\_3.yml*” que define los 13 nodos (10 routers y 3 PCs) y los 15 enlaces de la red.

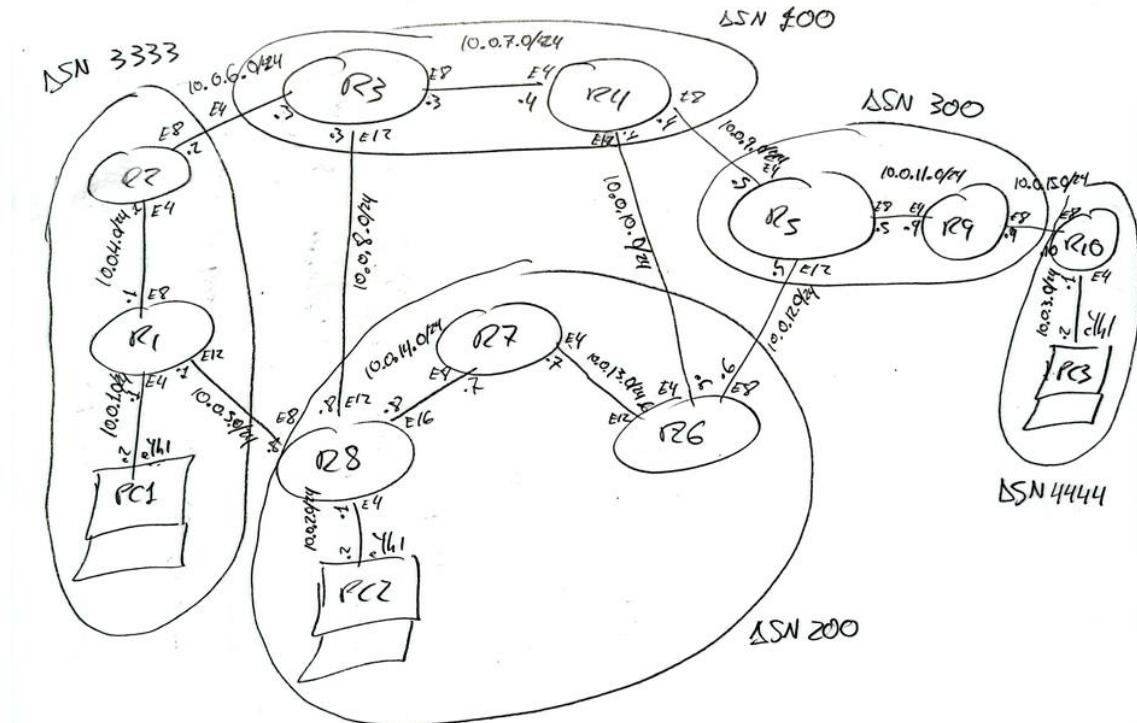


Figura01: Red del escenario a mano

- **Nodos PC:** Se ha empleado la imagen alpine:latest.
- **Nodos Router:** Se ha empleado la imagen jvt911/docker-sonic-lldp:latest.

Un descubrimiento clave durante el despliegue fue una peculiaridad: Las interfaces ethX no son funcionalmente configurables por el software de enrutamiento. Por lo tanto, fue necesario redefinir la topología para usar exclusivamente interfaces EthernetX, donde X es un múltiplo de 4, para asegurar su operatividad.

### 3. Metodología de Configuración

La configuración completa se ha automatizado mediante un script, “*config\_total2.sh*”. Este enfoque es fundamental dada la complejidad y volatilidad del laboratorio. El script se divide en las siguientes fases lógicas:

**Parte 0: Definición de Variables.** Se declaran todas las variables de la red. Esto incluye los nombres de los contenedores, las direcciones IP para nodos e interfaces, las redes que se incluirán en OSPF, los números de ASN para cada router y los Router-IDs (192.168.0.X).

**Parte 1: Bucle de Espera.** Se implementa un bucle de espera que verifica la disponibilidad del servicio vtysh en cada uno de los 10 routers. Esto es crucial, ya que el

script no puede aplicar configuraciones de enrutamiento hasta que los routers estén completamente activos.

**Parte 2: Configuración de PCs.** Se configuran las IPs y las rutas por defecto en los tres PC. Un paso crítico aquí es usar ip route replace default para sobrescribir la ruta de por defecto que Containerlab pone en cada nodo, asegurando que el tráfico de los PCs se dirija a su gateway correcto.

**Parte 3: Configuración de Routers.** En esta fase se establece la conectividad IP fundamental para los 10 routers. A cada interfaz de router (Ethernet4, Ethernet8, etc.) se le asigna su dirección IP y máscara de subred correspondiente, según el plan de direccionamiento definido en las variables.

**Parte 4: Configuración de OSPF.** Se implementa OSPF con el fin de garantizar la conectividad IP fundamental entre todos los routers.

- **Punto Clave:** Para el correcto establecimiento de las sesiones BGP, OSPF debe ejecutarse en todas las subredes. Es fundamental que todos los routers y redes pertenezcan a la misma área (area 0), ya que esto crea una red de conexión unificada. Esta configuración garantiza que todos los routers tengan rutas internas válidas para "encontrar" las IPs de sus vecinos BGP antes de intentar establecer la conexión TCP, evitando así el fallo de la sesión.
- **Automatización:** Se ha definido una función reutilizable, "*config\_ospf*", para centralizar la configuración de OSPF. Esta función es invocada de forma individual por cada uno de los 10 routers, aplicando la configuración necesaria (el router-id y las sentencias network ... area 0) en cada uno.

**Parte 5: Configuración de BGP.** Finalmente, se configura BGP (iBGP y eBGP) "por encima" de OSPF para intercambiar las rutas de los PCs entre los ASNs.

- **Políticas:** Se implementa la lógica de ip prefix-list y route-map (adaptando el script de configuración base proporcionado por el profesor Juan Manuel Vozmediano Torres) para controlar qué rutas se anuncian (out) y cuáles se reciben (in).
- **Comandos Clave:** Para que las sesiones BGP se establecieran:
  - **bgp router-id ...:** Para evitar que el router seleccione automáticamente la IP de gestión (172.21.21.x).
  - **update-source ...:** Para forzar a BGP utilizar la dirección IP de la interfaz de la práctica (ej. 10.0.5.1) como la IP de origen para sus paquetes TCP. Esto soluciona el problema de que el vecino BGP rechace la conexión por recibirla desde la IP de gestión en lugar de la IP esperada en el enlace.
  - **ebgp-multihop 2:** Modifica la regla por defecto de eBGP, que espera que el vecino esté directamente conectado (a 1 salto). Dado que los vecinos en esta red se alcanzan a través de rutas OSPF, este comando permite que la sesión se establezca con un vecino que está a múltiples saltos de distancia.

- o next-hop-self: Se aplica a las sesiones iBGP. Asegura que cuando un router anuncia una ruta aprendida de un vecino externo, se anuncie a sí mismo como el siguiente salto, garantizando que la ruta sea alcanzable dentro de su propio ASN.

## 4. Ejecución y Puesta en Marcha

El despliegue y la configuración del laboratorio se realizan en dos fases principales, seguidas de un período de espera.

**Fase 1: Despliegue de la Topología** Primero, se utiliza Containerlab para desplegar los 13 contenedores (PCs y routers) y crear los enlaces virtuales entre ellos, según lo definido en el fichero .yml.

```
clab deploy -t Ejercicio2-2_3.yml
```

**Fase 2: Ejecución del Script de Configuración** Una vez que todos los contenedores estén activos y en funcionamiento, se ejecuta el script de configuración “*config\_total2.sh*”. Este script realizará todos los pasos de configuración (IPs, OSPF, BGP) en los 13 nodos.

```
./config_total2.sh
```

**Fase 3: Período de Convergencia.** Es fundamental no realizar pruebas de conectividad inmediatamente después de que el script termine. Los protocolos de enrutamiento (OSPF y BGP) no son instantáneos.

Dada la magnitud de la red, se requiere un período de espera de 1 a 2 minutos para permitir que OSPF converja y que todas las sesiones BGP se establezcan e intercambien sus prefijos. Solo después de este período de asentamiento se pueden realizar las pruebas de ping y traceroute para verificar la conectividad global.

## 5. Desafíos Técnicos y Soluciones

El desarrollo de la configuración funcional no fue fácil y requirió la resolución de varios problemas técnicos significativos.

1. **Volatilidad del Laboratorio:** Uno de los principales problemas fue que el entorno de Containerlab es temporal, toda la configuración se pierde en cuanto se destruye el laboratorio con “*clab destroy*”. Esto provocaba que tener que configurar todo a mano repetidamente fuera muy ineficiente. Por esta razón, se hizo indispensable automatizar todo el proceso de configuración en un único

script “config\_total2.sh”, permitiendo así desplegar y probar la red de forma rápida y fiable.

2. **Peculiaridad de la Imagen de Sonic:** La imagen de Docker seleccionada (jvt911/docker-sonic-lldp:latest) presentó un problema. Se detectó que las interfaces ethX estándar no eran operativas. Esto obligó a rediseñar el fichero de topología .yml y a renombrar todas las interfaces de los routers a EthernetX (utilizando múltiplos de 4)
3. **Fallo de Establecimiento de BGP (State/PfxRcd 0):** Este fue el problema más complejo. Aunque la configuración de BGP era sintácticamente correcta, las sesiones no se establecían. La depuración reveló múltiples causas:
  - o **Fallo de Conectividad:** BGP no podía encontrar a sus vecinos. La solución fue activar OSPF en todas las subredes, haciendo que todos los routers pertenecieran a la misma área.
  - o **Conflicto de IP de Origen:** BGP intentaba usar la IP de gestión de Containerlab (172.21.21.x) como IP de origen. Esto se solucionó borrando la ruta por defecto (ip route del default) y forzando la IP de origen correcta con neighbor ... update-source ....
  - o **Comprobación de eBGP:** Al usar OSPF, los vecinos eBGP ya no estaban directamente conectados, lo que requirió el comando neighbor ... ebgp-multihop 2.

## 6. Rol de la Asistencia por IA

La Inteligencia Artificial ha sido una herramienta de soporte clave en este proyecto, facilitando varias fases del desarrollo.

- **Automatización.** Tras definir manualmente la lógica de configuración (qué comandos ejecutar y en qué orden), se solicitó a la IA que ayudara a refactorizar estos comandos en un script (.sh). Esto incluyó la creación de plantillas de código repetitivo (como los bloques docker exec para cada router) y la implementación de funciones (config\_ospf) para hacer el código más modular y reutilizable.
- **Soporte en la Depuración y Análisis de Logs.** La IA actuó como una herramienta de "segundo par de ojos" durante la depuración. Al proporcionarle las salidas de error ( Create the peer-group..., Unknown command...) o los resultados de show ip bgp summary (como State/PfxRcd 0 o (Policy)), la IA pudo ayudar a interpretar estos logs. Sugirió posibles que luego se verificaron y probaron manualmente.

- **Herramienta de Consulta Rápida** Finalmente, se empleó como una herramienta de consulta para clarificar conceptos técnicos puntuales que surgieron durante el desarrollo

## 7. Conclusiones

Este proyecto ha representado un desafío técnico de considerable complejidad. Ha exigido trabajar con un conjunto de tecnologías no utilizadas previamente, como la virtualización de redes a gran escala con Containerlab

El proceso de aprendizaje ha sido intenso, aportando conocimientos muy útiles sobre la arquitectura de redes. La ausencia total de una interfaz gráfica (GUI) fue un aspecto definitorio del reto. Esta limitación obligó a realizar todo el proceso de diseño y configuración de forma manual, un proceso que resultó especialmente laborioso, ya que requirió duplicar el esfuerzo: primero, se debió planificar y dibujar la topología completa a mano y, posteriormente, trasladar todo ese esquema de direccionamiento IP y políticas de enrutamiento al código del script de automatización.

Trabajar "desde cero" fue clave para entender dos puntos: la importancia de la automatización para gestionar redes complejas y la dependencia mutua de los protocolos. Sin embargo, la lección más valiosa fue el proceso de depuración sistemática por capas. Se comprobó que los problemas debían resolverse en orden (primero la conectividad IP, luego OSPF y finalmente BGP), demostrando que la red de conexión básica debe estar plenamente funcional para que el enrutamiento BGP pueda operar.

El proyecto se ha completado con éxito, logrando la conectividad total entre los tres PCs, cada uno ubicado en un Sistema Autónomo distinto. Esta práctica ha sido una excelente oportunidad para conectar los conceptos teóricos de la asignatura con un escenario real.

Ver en acción la configuración de BGP y OSPF, y especialmente depurar sus interacciones, ha demostrado ser un ejercicio fundamental para consolidar y asentar los conocimientos teóricos.