**GUÍA DE INSTALACIÓN, CONFIGURACIÓN Y PRUEBAS — CONTROLADOR SDN FAUCET**

ENTORNO UTILIZADO: Mininet-VM (Ubuntu), Open vSwitch, FAUCET/GAUGE en Docker, consola (PuTTY).

# OBJETIVOS

* Instalar Docker y ejecutar los contenedores FAUCET y GAUGE en la Mininet-VM.
* Crear y gestionar la configuración del controlador FAUCET usando archivos YAML.
* Conectar un switch de Mininet (Open vSwitch) al controlador FAUCET con OpenFlow 1.3.
* Verificar conectividad L2 (h1↔h2) y validarla con pruebas de ping.
* Comprobar programación de flujos y estado del datapath mediante logs y comandos OVS.
* Enviar tráfico de aplicación simple (TCP/UDP) entre hosts con netcat para validar el plano de datos.
* Opcional: demostrar aislamiento por VLAN (dos dominios L2 separados) con h1/h2 en VLAN100 y h3/h4 en VLAN200.
* Dejar un procedimiento reproducible, 100% por consola, para la práctica y exposición.

# FUNDAMENTOS TEÓRICOS

**SDN (Software-Defined Networking):** SDN separa el **plano de control** (donde se decide cómo debe tratarse el tráfico) del **plano de datos** (donde los switches reenvían paquetes). Esta separación permite centralizar la lógica en un **controlador** y aplicar políticas de red como código: versionables, auditables y reproducibles. En este esquema, FAUCET (basado en Ryu) **programa** a los switches mediante OpenFlow, de modo que cambios de topología, VLAN o ACLs se aplican sin tocar físicamente los equipos. El resultado es una red **determinística** y **automatizable**, ideal para laboratorio y producción controlada.

**OpenFlow 1.3:** Es el protocolo estándar que define cómo el controlador instala **flujos (flows)** en tablas del switch usando el modelo **match/action**. En OF1.3 se soportan coincidencias sobre campos L2/L3/L4 (MAC, VLAN ID 802.1Q, EthType, IP src/dst, puertos TCP/UDP, etc.), **prioridades**, **timeouts** (idle/hard), **contadores**, y un **pipeline** de múltiples tablas. Las acciones incluyen **output**, **push/pop VLAN**, **set-field**, **goto\_table** o envío al **CONTROLLER** (packet-in). FAUCET se apoya en estas capacidades para implementar aprendizaje L2, segmentación por VLAN, filtrado con ACLs y, si se habilita, **encaminamiento L3**.

**FAUCET:** Es un **controlador SDN L2/L3** que corre sobre Ryu y usa configuración **declarativa en YAML**. Su diseño enfatiza “infraestructura como código”: describe **qué** política quieres (VLANes, puertos, ACLs, gateways virtuales) y FAUCET la traduce a **flows OpenFlow** de forma consistente. Soporta múltiples **datapaths** (switches), **mirroring**, **ACLs** por puerto/VLAN, **SVI** (interfaces virtuales para L3 mediante faucet\_vips) y telemetría integrada. En contextos educativos y de validación, su sintaxis clara y sus logs facilitan entender cómo una política se convierte en reglas concretas en el switch.

**GAUGE:** Componente de telemetría complementario a FAUCET. Recoge y expone **métricas y eventos** (estadísticas de puertos, cambios de estado, contadores de flujos) útiles para **observabilidad** y **troubleshooting**. En despliegues con Prometheus/Grafana, permite paneles de salud de la red; en laboratorio, sus logs sirven para verificar que el datapath se conectó, que se aplicó la política y que los puertos están UP.

**RYU:** Framework SDN en **Python** que implementa el stack de protocolos (incluido OpenFlow) y provee el **runtime** sobre el cual corre FAUCET. Aporta manejadores de eventos (por ejemplo, packet\_in, cambios de puerto, errores), temporizadores y APIs para construir aplicaciones de control. Aunque el usuario final no programe en Ryu, comprender que FAUCET se apoya en Ryu ayuda a interpretar mensajes y trazas en los logs.

**MININET:** emulador de red que crea hosts, switches Open vSwitch y enlaces en un solo kernel, ideal para pruebas SDN.

**OPEN VSWITCH (OVS):** Switch virtual compatible con **OpenFlow 1.3** que actúa como **plano de datos** en Mininet y en muchos entornos de virtualización. OVS mantiene las **tablas de flujo** que el controlador instala, ejecuta las acciones (output, push/pop VLAN, set-field) y expone herramientas de inspección:

ovs-ofctl show → estado del switch (puertos, **DPID**).

ovs-ofctl dump-flows → reglas activas (tabla, prioridad, match, actions, contadores).  
Este binomio FAUCET+OVS permite ver, de forma transparente, cómo la política declarada en YAML se convierte en comportamiento de reenvío concreto.

# ANTES DE EMPEZAR — REQUISITOS Y PREPARACIÓN

## ¿QUÉ VAMOS A USAR?

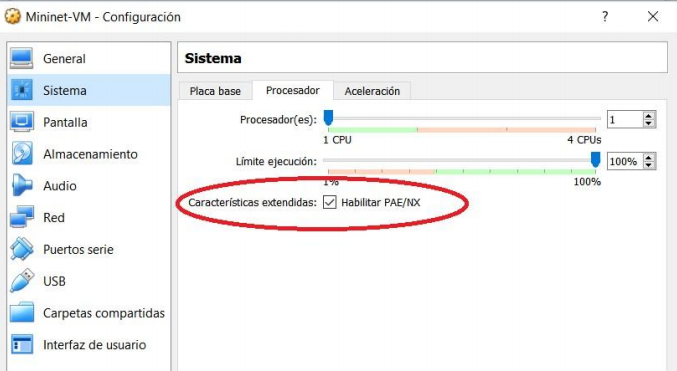
* **-Host**: tu PC (Windows 10/11).
* **-Virtualización**: **VirtualBox** con la **Mininet-VM** (trae Ubuntu + Mininet + OVS).
* **-Acceso remoto**: **PuTTY** (SSH) para abrir consolas a la VM.
* **-Controlador SDN**: **FAUCET** (y **GAUGE**) ejecutados en **Docker** dentro de la VM.
* **-Pruebas**: **Mininet** (topologías), **ping** y **netcat (nc)**.
* -Puertos a usar en la VM: **6653/TCP** (OpenFlow) y **9302/TCP** (estado/métricas de FAUCET).

**PUTTY:** https://putty.org/index.html

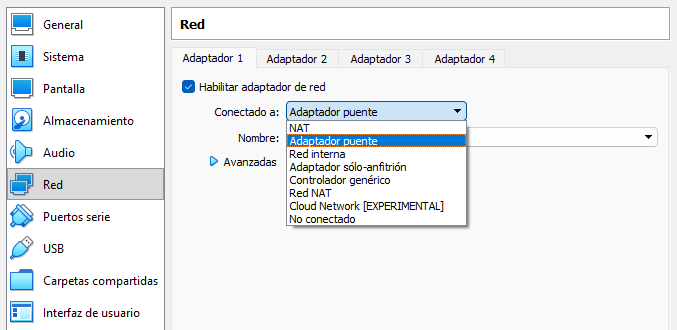
**MININET:** https://mininet-org.translate.goog/download/?\_x\_tr\_sch=http&\_x\_tr\_sl=en&\_x\_tr\_tl=es&\_x\_tr\_hl=es&\_x\_tr\_pto=tc

**VIRTUAL BOX:** https://www.virtualbox.org/wiki/Downloads

# Instalacion de mininet en Virtual Box (Configuracion):

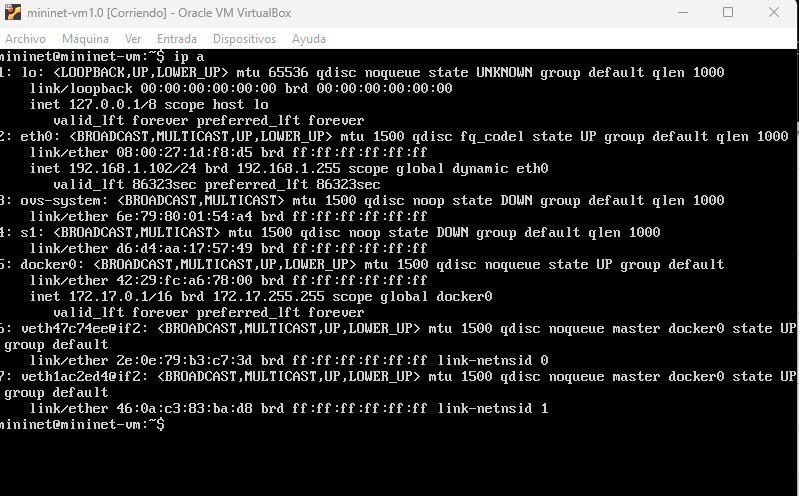


Nos vamos al apartado de sistema en el procesador seleccionamos Habilitar PAE/NX

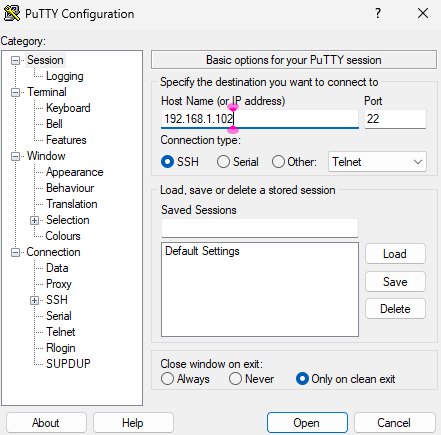


En el apartado de red seleccionamos adaptador puente y guardamos la configuracion.

Iniciamos la maquina virtual, para acceder usamos como Usuario: mininte y como Contraseña: mininet :



Una vez hayamos accedido a la máquina virtual utilizamos el Comando ip a para saber la ip de mininet que nos está brindando la maquina la cual es: 192.168.1.102/24.



La direccion ip que nos dio la máquina virtual la ingresamos en Putty y le damos open.

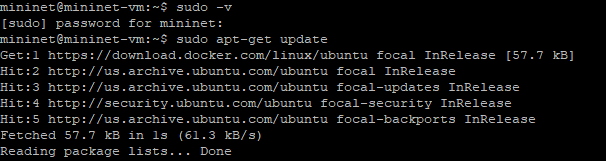
**Desarrollo de la guía paso a paso:**

A continuación se presenta el procedimiento exactamente en el orden en el que fue ejecutado, separando acciones entre Ventana 1 (host) y Ventana 2 (Mininet).

## Ventana 1 — Instalación de Docker

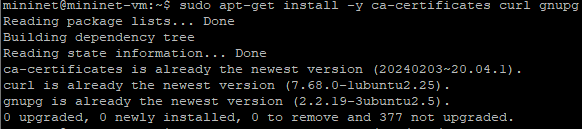
**Probar Docker:**

#sudo -v  
#sudo apt-get update :

  
sudo -v: renueva/valida permisos de superusuario (puede pedir contraseña) para que los siguientes sudo no interrumpan.

sudo apt-get update: actualiza el índice de paquetes de Ubuntu; las líneas Get/Hit y el final Reading package lists... Done indican que se conectó a los repos (incluye Docker) y terminó correctamente.

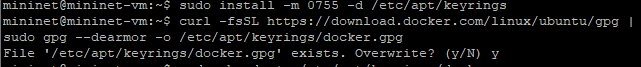
#sudo apt-get install -y ca-certificates curl gnupg :

  
**Comando ejecutado:** sudo apt-get install -y ca-certificates curl gnupg

**Para qué sirve en la guía:** prepara el sistema para añadir el repositorio oficial de Docker de forma segura: ca-certificates valida conexiones HTTPS, curl descarga la clave y el repo, gnupg convierte/gestiona la clave GPG que usará APT.

**Lo que muestra la salida:** esos paquetes **ya estaban en su última versión**, por eso no instaló nada nuevo (0 newly installed).

#sudo install -m 0755 -d /etc/apt/keyrings  
#curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg :

  
**Comandos:**

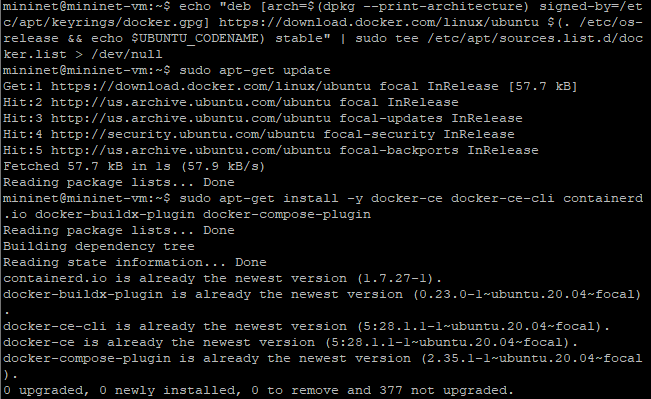
sudo install -m 0755 -d /etc/apt/keyrings → crea (con permisos 0755) la carpeta donde guardaremos claves GPG de repos.

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg → descarga la **clave GPG de Docker** y la convierte a formato .gpg para que APT pueda verificar paquetes del repo oficial.

**Salida relevante:** aparece el mensaje File '/etc/apt/keyrings/docker.gpg' exists. Overwrite? (y/N); respondiste y, es válido (reemplaza la clave si ya existía).

**Estado:** directorio creado y clave GPG guardada.

#sudo chmod a+r /etc/apt/keyrings/docker.gpg  
#echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo $UBUNTU\_CODENAME) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  
#sudo apt-get update  
#sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin :



sudo chmod a+r /etc/apt/keyrings/docker.gpg: da permisos de lectura a la clave GPG para que **APT** pueda usarla.

echo "deb …" | sudo tee /etc/apt/sources.list.d/docker.list: agrega el **repositorio oficial de Docker** (firmado con esa clave) a las fuentes de APT.

sudo apt-get update: recarga el índice de paquetes, ahora **incluyendo** el repo de Docker (se ve download.docker.com … InRelease).

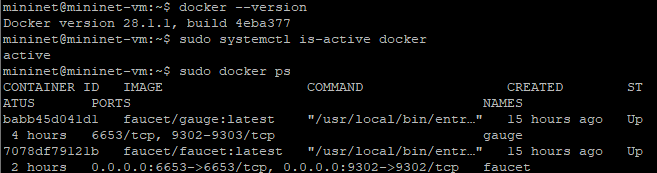
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin: instala el motor de Docker y herramientas.

En tu salida: **“is already the newest version”** → ya estaba instalado y actualizado (0 newly installed).

**Estado:** repo de Docker activo y Docker OK.

**Revisar version y confirmar que este activo el servicio Docker:**

#docker --version  
#sudo systemctl is-active docker  
#sudo docker ps :



docker --version: confirma la versión instalada de Docker (aquí 24.1.1/similar).

sudo systemctl is-active docker: verifica que el servicio Docker esté **activo** (active).

sudo docker ps: lista contenedores en ejecución; se ve faucet y gauge en estado **Up**, con los puertos publicados:

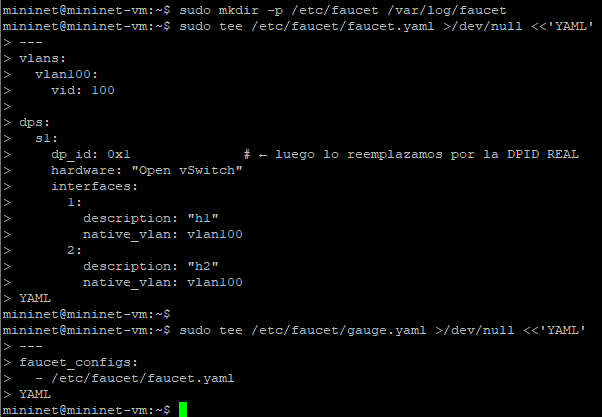
faucet: 6653/tcp (OpenFlow) y 9302/tcp (métricas FAUCET),

gauge: 9302-9303/tcp (según imagen).

**Conclusión:** Docker está operativo y los contenedores **faucet/gauge** están corriendo correctamente.

## Ventana 1 — Crear rutas y configuración inicial de FAUCET

#sudo mkdir -p /etc/faucet /var/log/faucet  
  
#sudo tee /etc/faucet/faucet.yaml >/dev/null <<'YAML'  
---  
vlans:  
 vlan100:  
 vid: 100  
  
dps:  
 s1:  
 dp\_id: 0x1 # ← luego lo reemplazamos por la DPID REAL  
 hardware: "Open vSwitch"  
 interfaces:  
 1:  
 description: "h1"  
 native\_vlan: vlan100  
 2:  
 description: "h2"  
 native\_vlan: vlan100  
YAML  
  
#sudo tee /etc/faucet/gauge.yaml >/dev/null <<'YAML'  
---  
faucet\_configs:  
 - /etc/faucet/faucet.yaml  
YAML :



sudo mkdir -p /etc/faucet /var/log/faucet: crea las carpetas de **configuración** y **logs** que serán montadas dentro de los contenedores.

sudo tee /etc/faucet/faucet.yaml … <<'YAML' … YAML: genera el archivo /etc/faucet/faucet.yaml sin abrir editores.

Define **VLAN 100** y un datapath s1 (hardware “Open vSwitch”).

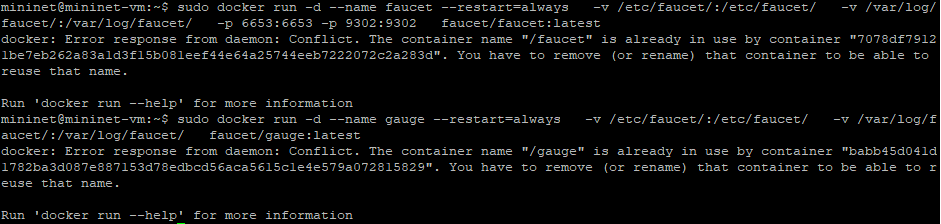
Asigna los **puertos 1 y 2** a vlan100.

El dp\_id: 0x1 es **provisorio**; luego se reemplaza por la **DPID real** del switch s1.

sudo tee /etc/faucet/gauge.yaml …: crea una config mínima para **GAUGE**, apuntando al mismo faucet.yaml.

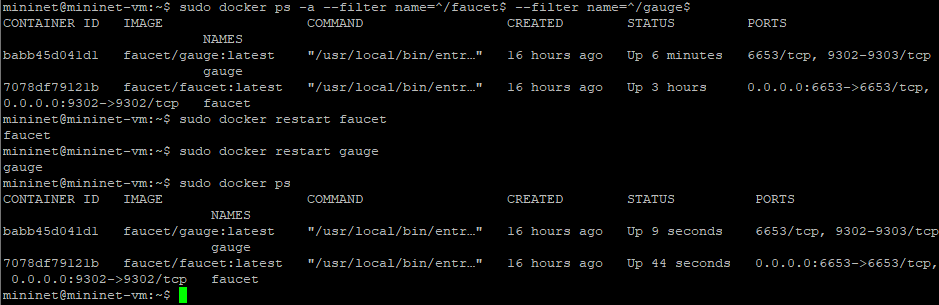
## Ventana 1 — Descargar imágenes y lanzar FAUCET/GAUGE

#sudo docker pull faucet/faucet:latest  
#sudo docker pull faucet/gauge:latest  
  
#sudo docker run -d --name faucet --restart=always -v /etc/faucet/:/etc/faucet/ -v /var/log/faucet/:/var/log/faucet/ -p 6653:6653 -p 9302:9302 faucet/faucet:latest  
  
#sudo docker run -d --name gauge --restart=always -v /etc/faucet/:/etc/faucet/ -v /var/log/faucet/:/var/log/faucet/ faucet/gauge:latest :

  
sudo docker pull faucet/faucet:latest y sudo docker pull faucet/gauge:latest: descargan las imágenes de **FAUCET** y **GAUGE** desde Docker Hub.

sudo docker run -d --name faucet … y sudo docker run -d --name gauge …: crean y arrancan los contenedores, montando /etc/faucet y /var/log/faucet, y publicando puertos (FAUCET: 6653, 9302).

**Solución de los dos errores:**#sudo docker ps -a --filter name=^/faucet$ --filter name=^/gauge$  
#sudo docker restart faucet  
#sudo docker restart gauge  
#sudo docker ps :



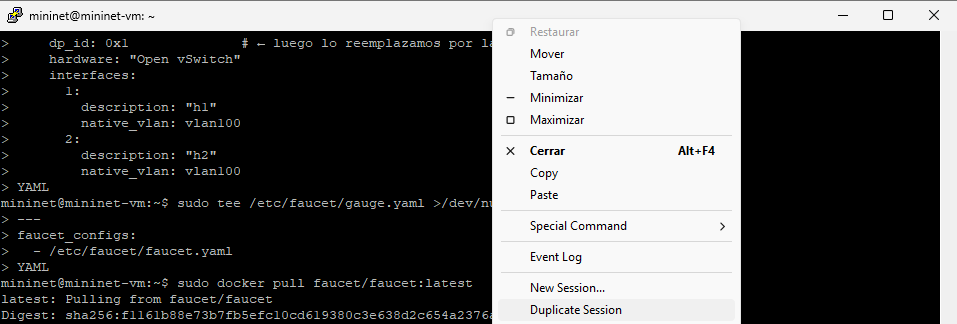
sudo docker ps -a --filter name=^/faucet$ --filter name=^/gauge$: lista **solo** los contenedores llamados exactamente faucet y gauge para comprobar su estado.

sudo docker restart faucet y sudo docker restart gauge: reinician los contenedores existentes (solución al conflicto de nombres).

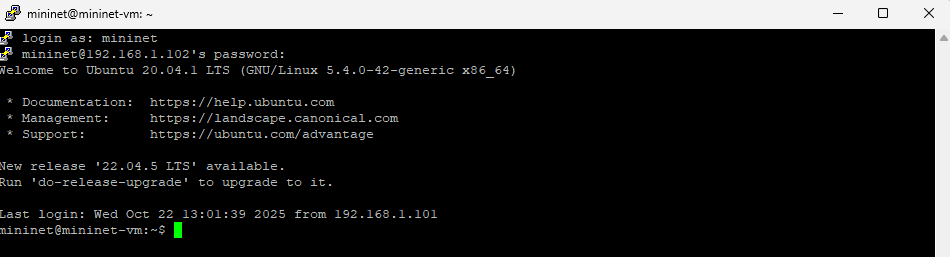
sudo docker ps: confirma que ambos están **Up**; se ven los puertos publicados (faucet: 6653/9302).

**Conclusión:** contenedores **recuperados y activos**; puedes seguir con Mininet, obtener la **DPID** y actualizar faucet.yaml.

## Ventana 2 — Mininet: topología 1 switch/2 hosts y DPID



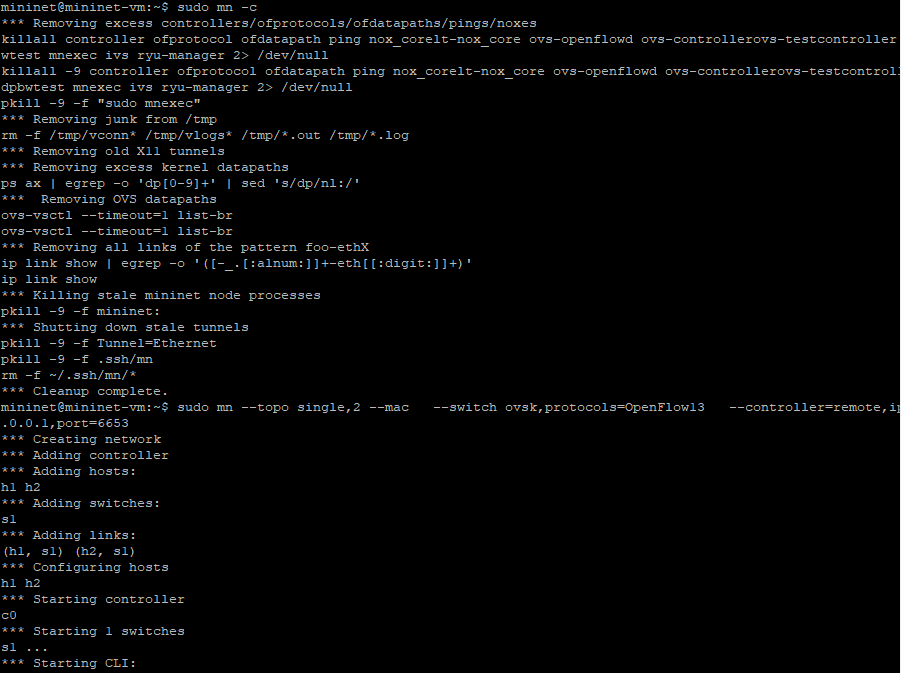
Le damos click derecho a la barra de putty y le damos click a Duplicate session



Ya estando en la Ventana 2 ingresamos el usuario y la contraseña el cual sabemos que es: Usuario: mininet, Contraseña: mininet

**Y seguimos con el proceso:**

#sudo mn -c  
#sudo mn --topo single,2 --mac --switch ovsk,protocols=OpenFlow13 --controller=remote,ip=127.0.0.1,port=6653 :



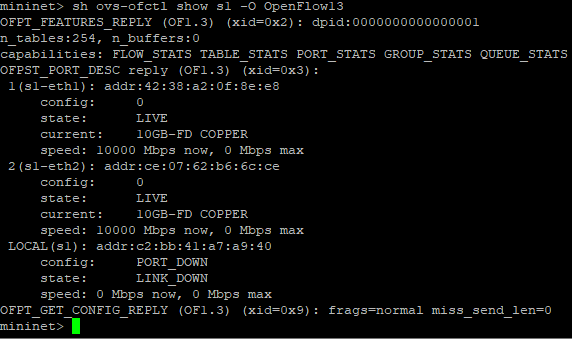
sudo mn -c: limpia restos de sesiones anteriores de Mininet/OVS (interfaces, bridges, túneles).

sudo mn --topo single,2 --mac --switch ovsk,protocols=OpenFlow13 --controller=remote,ip=127.0.0.1,port=6653: levanta una topología con **1 switch**

**(s1)** y **2 hosts (h1, h2)**, fuerza **OpenFlow 1.3** en OVS y conecta el switch al controlador **FAUCET** en 127.0.0.1:6653.

**Salida esperada (lo que ves):**  
“Creating network”, “Adding controller c0”, “Adding hosts h1 h2”, “Adding switches s1”, “Adding links”, “Starting CLI”. Esto confirma que Mininet quedó listo en el prompt mininet>.

**En mininet> (obtén DPID real)**  
#sh ovs-ofctl show s1 -O OpenFlow13 :



**Comando (en** mininet>**):** sh ovs-ofctl show s1 -O OpenFlow13

**Para qué sirve:** obtener la **DPID real** del switch s1 y ver estado/capacidades de puertos bajo **OpenFlow 1.3**.

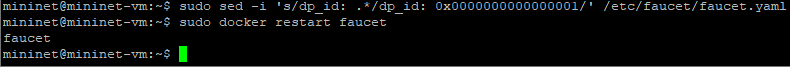
**Cómo leer la salida:**

Línea dpid: 0x0000000000000001 → **copias ese valor**.

Se listan los puertos (1, 2 y LOCAL) con su **state** (LIVE/PORT\_DOWN) y velocidad.

## Ventana 1 — Sustituir DPID real y recargar FAUCET

#sudo sed -i 's/dp\_id: .\*/dp\_id: 0x0000000000000001/' /etc/faucet/faucet.yaml  
#sudo docker restart faucet :



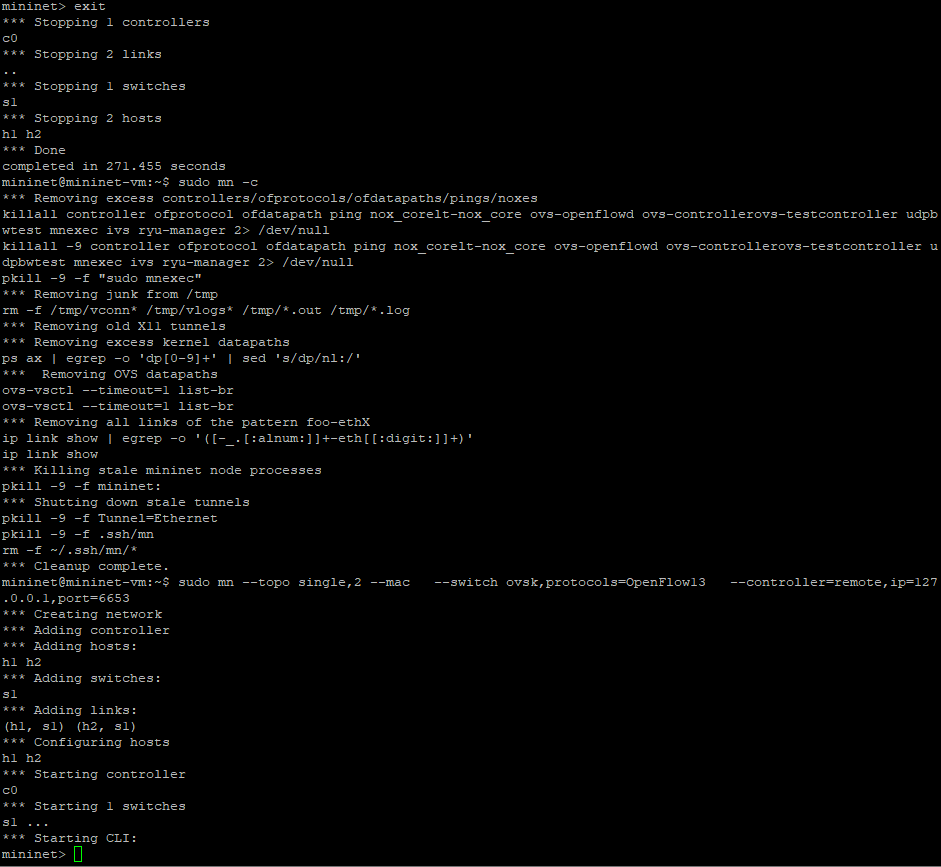
sudo sed -i 's/dp\_id: .\*/dp\_id: 0x0000000000000001/' /etc/faucet/faucet.yaml: reemplaza el dp\_id del YAML por la **DPID real** obtenida de s1.

sudo docker restart faucet: reinicia el contenedor para que FAUCET **recargue** la configuración actualizada.

**Resultado esperado:** ambos comandos retornan al prompt sin error.

## Ventana 2 — Reconectar y pruebas de conectividad

mininet> exit  
#sudo mn -c  
#sudo mn --topo single,2 --mac --switch ovsk,protocols=OpenFlow13 --controller=remote,ip=127.0.0.1,port=6653 :



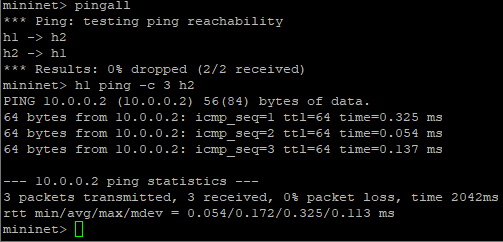
mininet> exit: sales del CLI de Mininet actual para aplicar cambios.

sudo mn -c: limpias completamente recursos de Mininet/OVS (bridges, túneles, procesos).

sudo mn --topo single,2 --mac --switch ovsk,protocols=OpenFlow13 --controller=remote,ip=127.0.0.1,port=6653: vuelves a levantar la topología **1 switch / 2 hosts**, con **OpenFlow 1.3**, apuntando a FAUCET en 127.0.0.1:6653.

**Salida mostrada:** “Creating network… Starting CLI” confirma que el entorno quedó **listo** otra vez y ya puedes ejecutar pingall y h1 ping -c 3 h2.

**En mininet>**#pingall  
#h1 ping -c 3 h2 :



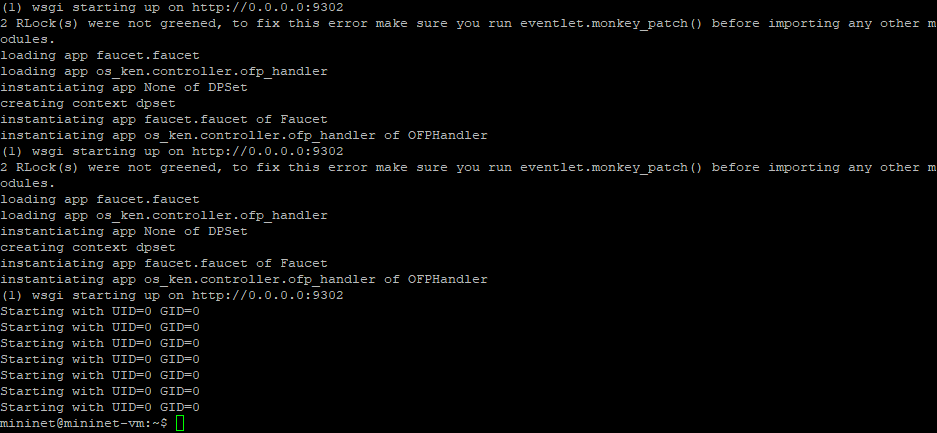
pingall **(en** mininet>**):** prueba conectividad entre **todos los hosts**; aquí muestra 0% dropped (éxito: h1↔h2).

h1 ping -c 3 h2: prueba puntual de ICMP desde h1 hacia h2, enviando **3 paquetes**; la salida confirma **3/3 recibidos, 0% loss** con latencias en ms.

**Conclusión:** la topología está conectada y FAUCET está programando correctamente el switch.

## Ventana 1 — Verificación desde FAUCET (logs y OVS)

#sudo docker logs faucet --since=5m | tail -n +1  
#sudo docker logs faucet | tail -n 100 :



**Comandos:**

sudo docker logs faucet --since=5m | tail -n +1 (o ... | tail -n 100)

**Para qué sirve:** ver los **logs recientes** del contenedor **FAUCET**.

**Lo que muestra la salida:**

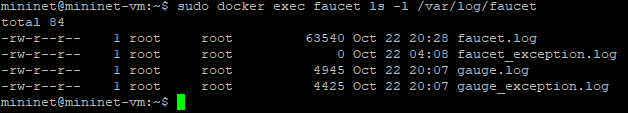
Arranque de **wsgi** en http://0.0.0.0:9302 (endpoint de métricas/estado).

Carga de la app faucet.faucet y handlers OpenFlow (ofp\_handler).

Mensajes “Starting with UID=0 GID=0” indicando que el proceso está activo.

**Conclusión:** FAUCET está **corriendo** y exponiendo su servicio en el puerto **9302**; la configuración se cargó sin errores críticos.

**Archivos de log internos:**  
#sudo docker exec faucet ls -l /var/log/faucet :



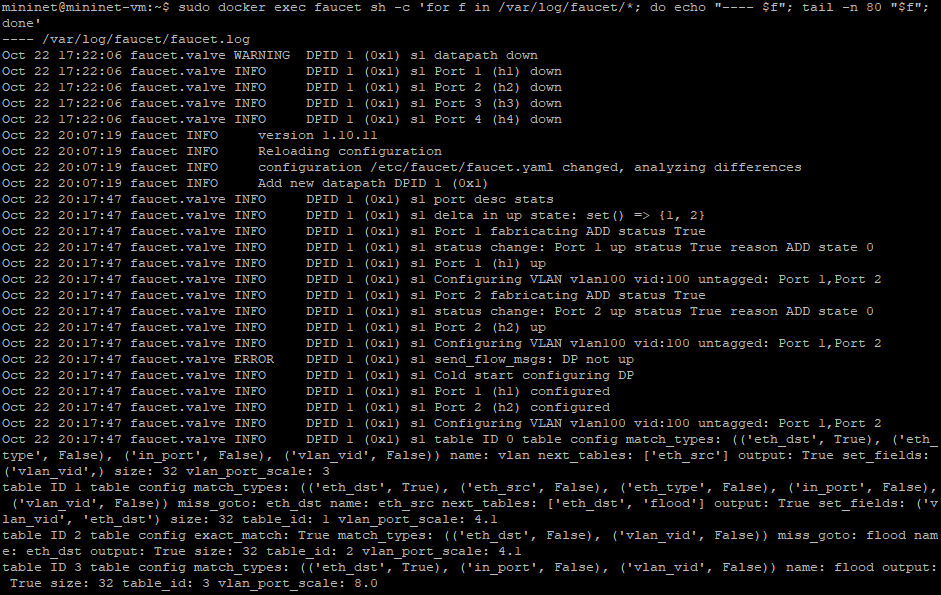
**Comando:** sudo docker exec faucet ls -l /var/log/faucet

**Para qué sirve:** listar los **archivos de log internos** dentro del contenedor faucet.

**Salida:** se ven faucet.log (principal), faucet\_exception.log (excepciones), gauge.log y gauge\_exception.log.

Los tamaños/fechas indican que **se están generando logs** correctamente.

#sudo docker exec faucet sh -c 'for f in /var/log/faucet/\*; do echo "---- $f"; tail -n 80 "$f"; done' :



**Para qué sirve:** mostrar el **final de cada log** dentro del contenedor faucet.

**Lo que se ve en** faucet.log**:**

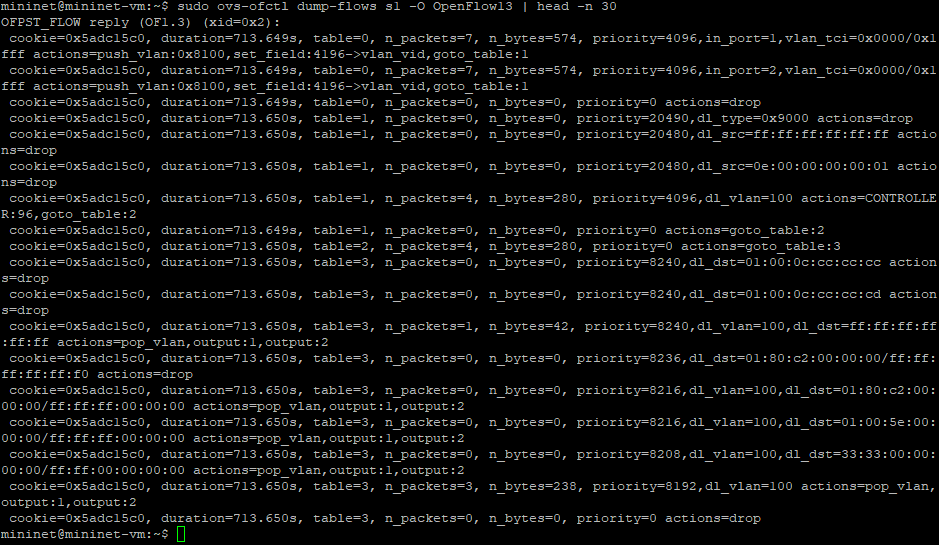
Mensajes INFO/WARNING del módulo faucet.valve.

Eventos de **puertos up/down**, “Reloading configuration”, y líneas donde **aplica la configuración** (Configuring VLAN vlan100, Port 1/2 configured, etc.).

Indica DPID 1 [OK] → el **datapath está conectado** y gestionado por FAUCET.

**Conclusión:** los logs confirman que FAUCET **carga el YAML**, detecta puertos, configura la **VLAN 100** y programa el switch correctamente.

**Confirmación de flujos en OVS:**  
#sudo ovs-ofctl dump-flows s1 -O OpenFlow13 | head -n 30:



**Comando:** sudo ovs-ofctl dump-flows s1 -O OpenFlow13 | head -n 30

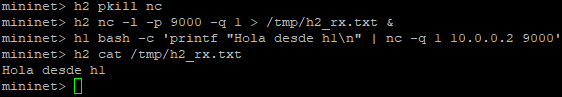
**Para qué sirve:** ver las **reglas (flows)** que FAUCET programó en el switch s1 bajo **OpenFlow 1.3**.

**Lo que confirma la salida:** aparecen entradas con table=…, priority=…, actions=… (por ejemplo dl\_vlan=100, push/pop\_vlan, goto\_table, CONTROLLER, drop). Esto prueba que **FAUCET instaló flujos** coherentes con tu faucet.yaml (VLAN100 y manejo de puertos).

**Conclusión:** el datapath está **configurado y operativo** según la política de FAUCET.

## Ventana 2 — Mensajes entre hosts (TCP/UDP) desde Mininet

**TCP con cierre automático:**#h2 pkill nc  
#h2 nc -l -p 9000 -q 1 > /tmp/h2\_rx.txt &  
#h1 bash -c 'printf "Hola desde h1\n" | nc -q 1 10.0.0.2 9000'  
#h2 cat /tmp/h2\_rx.txt :



**Objetivo:** validar el plano de datos con tráfico de aplicación **TCP** usando netcat y evitar que el proceso quede colgado.

**Comandos (en** mininet>**):**

h2 pkill nc → limpia nc previos.

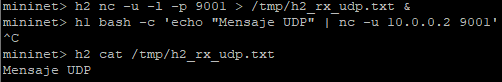
h2 nc -l -p 9000 -q 1 > /tmp/h2\_rx.txt & → servidor TCP en **h2** escuchando en **9000**, cierra 1s después de recibir entrada y guarda en /tmp/h2\_rx.txt.

h1 bash -c 'printf "Hola desde h1\n" | nc -q 1 10.0.0.2 9000' → cliente en **h1** envía el texto y cierra.

h2 cat /tmp/h2\_rx.txt → muestra lo recibido: **“Hola desde h1”**.

**Conclusión:** la comunicación **h1 → h2** funciona correctamente y FAUCET está conmutando tráfico entre hosts.

**UDP (opcional)**  
#h2 nc -u -l -p 9001 > /tmp/h2\_rx\_udp.txt &  
#h1 bash -c 'echo "Mensaje UDP" | nc -u 10.0.0.2 9001'  
#h2 cat /tmp/h2\_rx\_udp.txt :



**Objetivo (opcional):** verificar tráfico **UDP** entre hosts.

**Comandos (en** mininet>**):**

h2 nc -u -l -p 9001 > /tmp/h2\_rx\_udp.txt & → servidor **UDP** en h2 (puerto 9001) guardando lo recibido.

h1 bash -c 'echo "Mensaje UDP" | nc -u 10.0.0.2 9001' → cliente en h1 envía un datagrama UDP a h2.

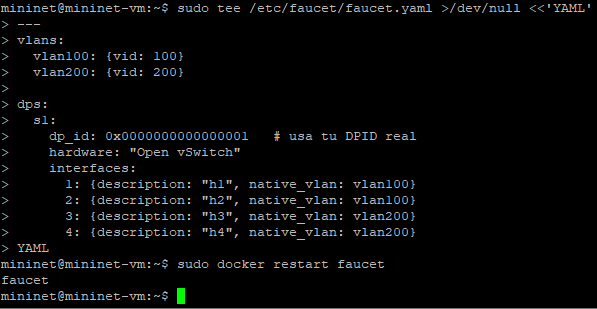
h2 cat /tmp/h2\_rx\_udp.txt → muestra lo recibido: **“Mensaje UDP”**.

**Nota:** aparece ^C porque se corto el comando en el cliente; el datagrama ya se había enviado y se recibió correctamente.

**Conclusión:** conectividad **UDP h1→h2** confirmada; el plano de datos funciona para TCP y UDP.

## PRUEBA EXTRA (OPCIONAL) — AISLAMIENTO L2 CON DOS VLAN

**En Ventana 1: reemplazar YAML y recargar**  
#sudo tee /etc/faucet/faucet.yaml >/dev/null <<'YAML'  
---  
vlans:  
 vlan100: {vid: 100}  
 vlan200: {vid: 200}  
  
dps:  
 s1:  
 dp\_id: 0x0000000000000001 # usa tu DPID real  
 hardware: "Open vSwitch"  
 interfaces:  
 1: {description: "h1", native\_vlan: vlan100}  
 2: {description: "h2", native\_vlan: vlan100}  
 3: {description: "h3", native\_vlan: vlan200}  
 4: {description: "h4", native\_vlan: vlan200}  
YAML  
#sudo docker restart faucet :



**Acción:** sustituyes faucet.yaml para definir **dos VLAN**:

vlan100 (VID 100) para **h1** y **h2**,

vlan200 (VID 200) para **h3** y **h4**,  
con el mismo s1 y tu **DPID real** (0x0000000000000001).

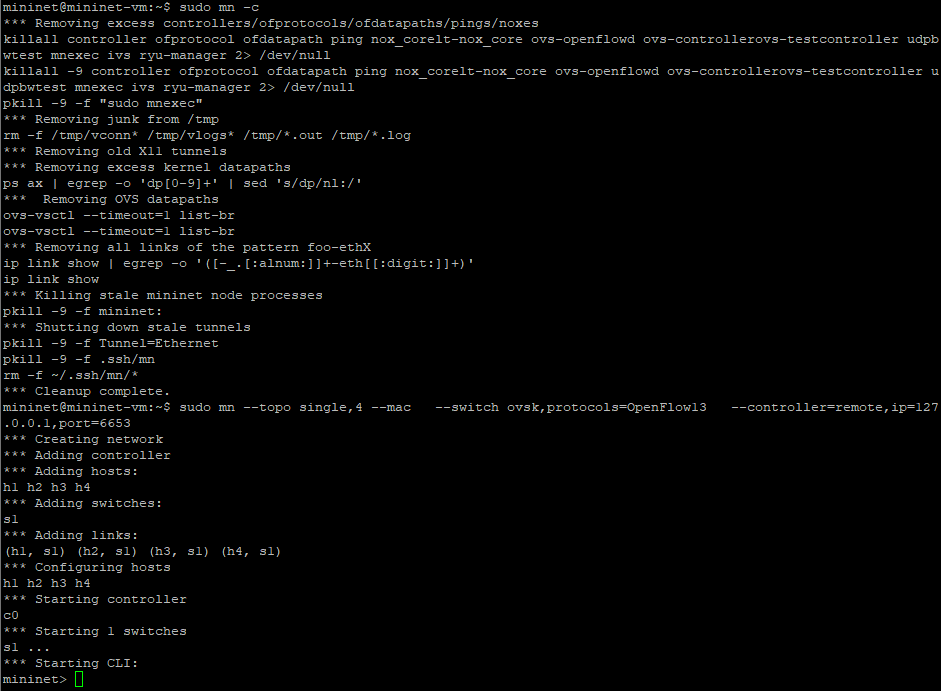
**Comandos:**

sudo tee /etc/faucet/faucet.yaml … <<'YAML' … YAML → reescribe el YAML con la nueva topología de VLAN.

sudo docker restart faucet → recarga FAUCET para aplicar la configuración.

**Objetivo de la prueba:** demostrar **aislamiento L2**: hosts de VLAN100 solo hablan entre sí, y lo mismo para VLAN200.  
  
**En Ventana 2: reiniciar Mininet y probar**

Minenet> exit  
#sudo mn -c  
#sudo mn --topo single,4 --mac --switch ovsk,protocols=OpenFlow13 --controller=remote,ip=127.0.0.1,port=6653 :

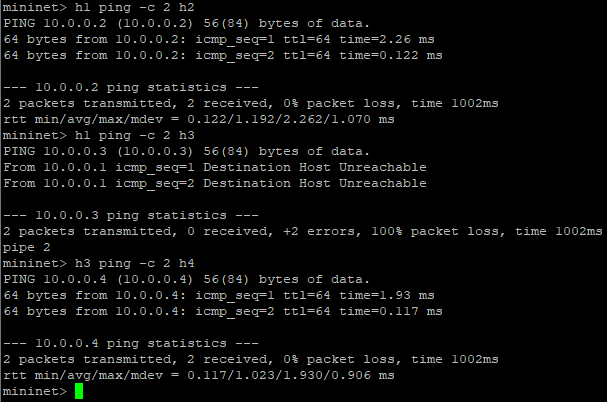


mininet> exit y sudo mn -c: cierras el CLI y limpias recursos previos.

sudo mn --topo single,4 --mac --switch ovsk,protocols=OpenFlow13 --controller=remote,ip=127.0.0.1,port=6653: levantas Mininet con **1 switch s1** y **4 hosts (h1–h4)**, forzando **OpenFlow 1.3** y conectando a FAUCET.

**Salida mostrada:** “Adding hosts h1 h2 h3 h4”, “Adding links”, “Starting CLI” → todo listo para probar VLANs (OK/FAIL/OK).

# En mininet> (orden de pruebas)  
h1 ping -c 2 h2 # OK (VLAN100)  
h1 ping -c 2 h3 # FAIL (VLAN100 -> VLAN200)  
h3 ping -c 2 h4 # OK (VLAN200)



**Comandos (en** mininet> **y en ese orden):**

h1 ping -c 2 h2 → **OK** (ambos en **VLAN100**).

h1 ping -c 2 h3 → **FAIL** (**VLAN100 → VLAN200**, correctamente **aislado**).

h3 ping -c 2 h4 → **OK** (ambos en **VLAN200**).

**CONCLUSIÓN:** queda demostrada la **segmentación L2 por VLAN**: tráfico permitido dentro de la misma VLAN y **bloqueado** entre VLAN100 y VLAN200, tal como define tu faucet.yaml.

## NOTAS Y LIMPIEZA

Si GAUGE arroja error de configuración, puedes ignorarlo (no afecta a la conectividad) o ajustar /etc/faucet/gauge.yaml para exponer métricas por Prometheus.

Al finalizar: salir de Mininet y limpiar:

mininet> exit  
sudo mn -c  
# (opcional) detener gauge si no lo usarás  
sudo docker stop gauge

**REFERENCIAS:**

* FAUCET Project — Documentación oficial: <https://docs.faucet.nz/>
* Ryu SDN Framework — Documentación: <https://ryu.readthedocs.io/>
* Open vSwitch — Documentación: <https://www.openvswitch.org/>
* ONF — OpenFlow Switch Specification 1.3.0: <https://opennetworking.org/>
* Mininet — 'Mininet: An Instant Virtual Network on your Laptop' (SIGCOMM Demo, 2010): <http://mininet.org/>