代理端口7897

在 WSL2 的 Ubuntu 环境中配置代理（端口 7897），需要确保与 Windows 主机的代理服务互通。以下是详细步骤：

一、获取 Windows 主机的 IP 地址

WSL2 中需通过 Windows 的 IP 连接代理，先在 Ubuntu 终端执行：

bash

# 查看Windows主机的IP（通常是eth0对应的网关）

cat /etc/resolv.conf | grep nameserver | awk '{print $2}'

输出类似 172.28.192.1，记为 WINDOWS\_IP。

二、临时配置代理（当前终端生效）

在 Ubuntu 终端执行以下命令（替换 WINDOWS\_IP 为实际 IP）：

bash

# 设置HTTP代理192.168.1.104

export http\_proxy="http://192.168.1.104:7897"

export https\_proxy="http://192.168.1.104:7897"

export http\_proxy="http://WINDOWS\_IP:7897"

export https\_proxy="http://WINDOWS\_IP:7897"

# 验证代理是否生效

curl -I https://www.google.com

若返回 HTTP/2 200，说明代理生效。

三、永久配置代理（所有终端生效）

编辑 bash 配置文件：

bash

nano ~/.bashrc

在文件末尾添加（替换 WINDOWS\_IP）：

bash

# 代理配置

export http\_proxy="http://WINDOWS\_IP:7897"

export https\_proxy="http://WINDOWS\_IP:7897"

# 可选：忽略代理的域名（本地服务等）

export no\_proxy="localhost,127.0.0.1,${WINDOWS\_IP}"

保存并生效：

bash

source ~/.bashrc

四、为 pip/conda 配置代理（可选）

1. pip 代理

bash

# 临时使用

pip install <包名> --proxy http://WINDOWS\_IP:7897

# 永久配置

pip config set global.proxy http://WINDOWS\_IP:7897

2. conda 代理

bash

conda config --set proxy\_servers.http http://WINDOWS\_IP:7897

conda config --set proxy\_servers.https http://WINDOWS\_IP:7897

五、注意事项

Windows 代理设置：确保代理软件（如 Clash、V2Ray）已开启 “允许局域网连接”，否则 WSL 无法访问。

IP 变化问题：WSL 重启后 Windows IP 可能变化，需重新执行 cat /etc/resolv.conf 获取新 IP。

测试连通性：若代理失败，先在 Ubuntu 中 ping Windows IP：

bash

ping WINDOWS\_IP

若不通，检查 Windows 防火墙是否阻止 WSL 访问端口 7897。

通过以上配置，Ubuntu 中的命令（如apt、curl、pip）均可通过 Windows 的 7897 端口代理访问网络。

(venv) dministrator@WIN-UB1HRT8QI4S:~$ # 查看Windows主机的IP（通常是eth0对应的网关）

cat /etc/resolv.conf | grep nameserver | awk '{print $2}'

10.255.255.254

(venv) dministrator@WIN-UB1HRT8QI4S:~$

根据你获取的 Windows 主机 IP（10.255.255.254），可以按以下步骤配置代理（端口 7897）：

1. 临时配置代理（当前终端生效）

在虚拟环境中执行：

bash

# 设置HTTP和HTTPS代理（使用获取到的Windows IP）

export http\_proxy="http://10.255.255.254:7897"

export https\_proxy="http://10.255.255.254:7897"

# 测试代理是否生效（访问一个需要代理的网站）

curl -I https://www.github.com

若返回 HTTP/2 200 或类似状态码，说明代理配置成功。

若失败，检查 Windows 端代理软件是否开启 “允许局域网连接”（如 Clash 的 “允许来自局域网的连接” 选项）。

2. 永久配置代理（在虚拟环境中自动生效）

如果需要每次进入该虚拟环境时自动启用代理，可将配置写入虚拟环境的激活脚本：

bash

# 编辑虚拟环境的激活脚本

nano ~/venv/bin/activate

在文件末尾添加：

bash

# 代理配置（Windows IP和端口）

export http\_proxy="http://10.255.255.254:7897"

export https\_proxy="http://10.255.255.254:7897"

export no\_proxy="localhost,127.0.0.1,10.255.255.254" # 本地地址不走代理

保存退出后，重启虚拟环境生效：

bash

deactivate # 退出当前虚拟环境

source ~/venv/bin/activate # 重新激活，代理自动生效

3. 为包管理工具单独配置代理（可选）

如果pip等工具需要单独设置代理（避免依赖系统环境变量）：

bash

# 配置pip代理

pip config set global.proxy http://10.255.255.254:7897

# 验证pip代理

pip install --upgrade pip # 测试是否能通过代理安装包

注意事项：

若 WSL 重启后发现代理失效，可能是 Windows IP 变化导致，需重新执行 cat /etc/resolv.conf | grep nameserver | awk '{print $2}' 获取新 IP 并更新代理配置。

确保 Windows 防火墙未阻止端口 7897 的入站连接（可临时关闭防火墙测试）。

配置完成后，虚拟环境中的网络请求会通过 Windows 的 7897 端口代理进行，方便访问外部资源。

ERROR: Operation cancelled by user

(venv) dministrator@WIN-UB1HRT8QI4S:~$ # 查看Windows主机的IP（通常是eth0对应的网关）

cat /etc/resolv.conf | grep nameserver | awk '{print $2}'

10.255.255.254

(venv) dministrator@WIN-UB1HRT8QI4S:~$ # 设置HTTP和HTTPS代理（使用获取到的Windows IP）

export http\_proxy="http://10.255.255.254:7897"

export https\_proxy="http://10.255.255.254:7897"

# 测试代理是否生效（访问一个需要代理的网站）

curl -I https://www.github.com

curl: (7) Failed to connect to 10.255.255.254 port 7897 after 0 ms: Couldn't connect to server

(venv) dministrator@WIN-UB1HRT8QI4S:~$ curl -I https://www.github.com

curl: (7) Failed to connect to 10.255.255.254 port 7897 after 0 ms: Couldn't connect to server

(venv) dministrator@WIN-UB1HRT8QI4S:~$ curl -I https://www.github.com

curl: (7) Failed to connect to 10.255.255.254 port 7897 after 0 ms: Couldn't connect to server

(venv) dministrator@WIN-UB1HRT8QI4S:~$ pip3 install torch torchvision

WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ProxyError('Cannot connect to proxy.', NewConnectionError('<pip.\_vendor.urllib3.connection.HTTPSConnection object at 0x7b6c22fe3a40>: Failed to establish a new connection: [Errno 111] Connection refused'))': /simple/torch/

WARNING: Retrying (Retry(total=3, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ProxyError('Cannot connect to proxy.', NewConnectionError('<pip.\_vendor.urllib3.connection.HTTPSConnection object at 0x7b6c230aacc0>: Failed to establish a new connection: [Errno 111] Connection refused'))': /simple/torch/

WARNING: Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ProxyError('Cannot connect to proxy.', NewConnectionError('<pip.\_vendor.urllib3.connection.HTTPSConnection object at 0x7b6c22d65760>: Failed to establish a new connection: [Errno 111] Connection refused'))': /simple/torch/

^CERROR: Operation cancelled by user

(venv) dministrator@WIN-UB1HRT8QI4S:~$ curl -I https://www.github.com

curl: (7) Failed to connect to 10.255.255.254 port 7897 after 0 ms: Couldn't connect to server

(venv) dministrator@WIN-UB1HRT8QI4S:~$

从错误信息来看，WSL 的 Ubuntu 环境无法连接到 Windows 主机的 7897 端口代理服务（Connection refused），主要原因可能是代理软件未正确配置或网络连通性问题。以下是逐步排查和解决方法：

1. 检查 Windows 端代理软件配置（关键）

确保代理服务正在运行：在 Windows 中打开你的代理软件（如 Clash、V2Ray、Netch 等），确认其已启动并正常工作（可在 Windows 浏览器中测试能否访问https://www.github.com）。

开启 “允许局域网连接”：这是 WSL 能访问 Windows 代理的核心设置：

Clash：进入 “设置”→“网络”，勾选 “允许局域网连接”。

V2RayN：右键任务栏图标→“设置”→“允许来自局域网的连接”。

其他工具：在代理设置中找到类似 “允许 LAN 访问” 或 “监听所有接口” 的选项并启用。

确认代理端口为 7897：检查代理软件的 “本地端口” 设置，确保是7897（若不是，需将后续命令中的端口改为实际值）。

2. 验证 WSL 与 Windows 的网络连通性

在 Ubuntu 终端执行以下命令，测试能否连接到 Windows 主机的 IP 和代理端口：

bash

# 测试能否ping通Windows主机（IP为10.255.255.254）

ping 10.255.255.254

# 测试代理端口是否开放（7897）

telnet 10.255.255.254 7897

若ping失败：说明 WSL 与 Windows 网络不通，需检查 WSL 网络模式（默认应为 NAT），或重启 WSL（wsl --shutdown后重新启动）。

若telnet失败（提示Connection refused）：确认代理软件已开启、端口正确，且 Windows 防火墙未拦截 7897 端口。

3. 临时关闭 Windows 防火墙测试

防火墙可能阻止 WSL 访问 Windows 的 7897 端口，可临时关闭测试：

打开 Windows “控制面板”→“系统和安全”→“Windows Defender 防火墙”→“关闭防火墙”（临时关闭，测试后建议重新开启）。

关闭后在 Ubuntu 中再次执行curl -I https://www.github.com，若成功则说明防火墙是阻碍因素，需为 7897 端口添加入站规则。

4. 更换 WSL 访问 Windows 的 IP 地址

部分情况下，resolv.conf中的 IP 可能不是 WSL 实际应使用的 Windows IP，可尝试另一种方式获取：

bash

# 在Ubuntu中获取Windows主机的另一个IP（适用于WSL2）

hostname -I | awk '{print $1}' | sed 's/\.[0-9]\*$/.1/'

假设输出为172.28.192.1，则重新配置代理：

bash

export http\_proxy="http://172.28.192.1:7897"

export https\_proxy="http://172.28.192.1:7897"

curl -I https://www.github.com # 再次测试

5. 若所有方法失败，尝试代理软件的 “TUN 模式”（可选）

若代理软件支持 TUN 模式（如 Clash 的 “TUN 旁路”），开启后 WSL 可能无需手动配置代理即可共享 Windows 的网络（依赖软件实现，部分工具支持）。

总结排查流程：

确保 Windows 代理软件正常运行且端口为 7897。

开启 “允许局域网连接” 并关闭防火墙测试。

验证 WSL 能否 ping 通 Windows IP 并连接 7897 端口。

更换 IP 地址重新配置代理。

解决网络连通性后，curl和pip即可通过代理正常访问网络。

# 测试代理端口是否开放（7897）

telnet 10.255.255.254 7897

PING 10.255.255.254 (10.255.255.254) 56(84) bytes of data.

64 bytes from 10.255.255.254: icmp\_seq=1 ttl=64 time=0.118 ms

64 bytes from 10.255.255.254: icmp\_seq=2 ttl=64 time=0.033 ms

64 bytes from 10.255.255.254: icmp\_seq=3 ttl=64 time=0.031 ms

64 bytes from 10.255.255.254: icmp\_seq=4 ttl=64 time=0.032 ms

64 bytes from 10.255.255.254: icmp\_seq=5 ttl=64 time=0.032 ms

64 bytes from 10.255.255.254: icmp\_seq=6 ttl=64 time=0.033 ms

64 bytes from 10.255.255.254: icmp\_seq=7 ttl=64 time=0.034 ms

^C

--- 10.255.255.254 ping statistics ---

7 packets transmitted, 7 received, 0% packet loss, time 6145ms

rtt min/avg/max/mdev = 0.031/0.044/0.118/0.029 ms

Command 'telnet' not found, but can be installed with:

sudo apt install inetutils-telnet # version 2:2.4-3ubuntu1, or

sudo apt install telnet-ssl # version 0.17.41+really0.17-4

(venv) dministrator@WIN-UB1HRT8QI4S:~$ ping 10.255.255.254

PING 10.255.255.254 (10.255.255.254) 56(84) bytes of data.

64 bytes from 10.255.255.254: icmp\_seq=1 ttl=64 time=0.027 ms

64 bytes from 10.255.255.254: icmp\_seq=2 ttl=64 time=0.034 ms

64 bytes from 10.255.255.254: icmp\_seq=3 ttl=64 time=0.033 ms

^C

--- 10.255.255.254 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 2042ms

rtt min/avg/max/mdev = 0.027/0.031/0.034/0.003 ms

(venv) dministrator@WIN-UB1HRT8QI4S:~$ curl -I https://www.github.com

curl: (7) Failed to connect to 10.255.255.254 port 7897 after 0 ms: Couldn't connect to server

(venv) dministrator@WIN-UB1HRT8QI4S:~$

从结果来看，WSL 能正常 ping 通 Windows 主机（10.255.255.254），但无法连接到 7897 端口，核心问题出在代理服务未正确监听或被拦截。以下是针对性解决步骤：

1. 安装 telnet 工具并确认端口是否开放

先安装 telnet 以精确检测 7897 端口状态：

bash

# 安装telnet

sudo apt update && sudo apt install -y inetutils-telnet

# 测试能否连接到Windows的7897端口

telnet 10.255.255.254 7897

若提示 Connection refused：说明 Windows 端 7897 端口未被代理软件占用，或软件未正确启动。

若提示 Connected to 10.255.255.254：端口正常，问题可能在代理软件配置（如仅允许特定 IP 访问）。

2. 确认 Windows 代理软件的监听设置

检查代理软件是否监听 7897 端口：

在 Windows 的 PowerShell 中执行，确认 7897 端口是否被占用：

powershell

# 查看7897端口的占用情况

netstat -ano | findstr :7897

若输出为空：说明代理软件未启动，或未使用 7897 端口（需在软件中重新设置端口为 7897）。

若有输出，记录 PID（最后一列数字），在任务管理器中确认是否为你的代理软件。

确保代理软件监听 “所有接口”：

部分代理软件默认仅监听127.0.0.1（本地回环），导致局域网（WSL）无法访问。需在软件设置中改为监听 0.0.0.0 或 “所有网络接口”。

3. 添加 Windows 防火墙规则放行 7897 端口

即使 ping 通，防火墙仍可能拦截端口连接：

打开 Windows “高级防火墙”→“入站规则”→“新建规则”。

选择 “端口”→“TCP”→“特定本地端口”，输入7897。

选择 “允许连接”→“域、专用、公用”（按需勾选）。

命名规则（如 “Allow WSL Proxy 7897”）并完成。

4. 尝试使用 Windows 的localhost地址（适用于 WSL2）

WSL2 中可通过 host.docker.internal 访问 Windows 的localhost，尝试：

bash

# 使用特殊域名替代IP

export http\_proxy="http://host.docker.internal:7897"

export https\_proxy="http://host.docker.internal:7897"

# 测试

curl -I https://www.github.com

若成功，说明之前的 IP（10.255.255.254）存在访问限制，可将此域名作为代理地址。

5. 临时更换代理端口测试

若 7897 端口始终无法访问，尝试在代理软件中更换为其他端口（如 7890），并在 WSL 中同步更新：

bash

# 假设新端口为7890

export http\_proxy="http://10.255.255.254:7890"

export https\_proxy="http://10.255.255.254:7890"

curl -I https://www.github.com

关键结论：

问题根源是Windows 的 7897 端口未对 WSL 开放，需从三方面确认：

代理软件是否正确监听 7897 端口且允许局域网访问。

防火墙是否放行 7897 端口的入站连接。

代理软件的监听地址是否包含 WSL 所在的网络接口。

完成以上设置后，WSL 应能通过代理正常访问网络。

# 测试能否连接到Windows的7897端口

telnet 10.255.255.254 7897

Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease

Hit:2 http://archive.ubuntu.com/ubuntu noble InRelease

Hit:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease

Hit:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease

Reading package lists... Done

Building dependency tree... Done

Reading state information... Done

All packages are up to date.

Reading package lists... Done

Building dependency tree... Done

Reading state information... Done

The following NEW packages will be installed:

inetutils-telnet

0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.

Need to get 100 kB of archives.

After this operation, 247 kB of additional disk space will be used.

Get:1 http://archive.ubuntu.com/ubuntu noble/main amd64 inetutils-telnet amd64 2:2.5-3ubuntu4 [100 kB]

Fetched 100 kB in 2s (45.5 kB/s)

Selecting previously unselected package inetutils-telnet.

(Reading database ... 40771 files and directories currently installed.)

Preparing to unpack .../inetutils-telnet\_2%3a2.5-3ubuntu4\_amd64.deb ...

Unpacking inetutils-telnet (2:2.5-3ubuntu4) ...

Setting up inetutils-telnet (2:2.5-3ubuntu4) ...

update-alternatives: using /usr/bin/inetutils-telnet to provide /usr/bin/telnet (telnet) in auto mode

Processing triggers for man-db (2.12.0-4build2) ...

Trying 10.255.255.254...

telnet: Unable to connect to remote host: Connection refused

(venv) dministrator@WIN-UB1HRT8QI4S:~$ telnet 10.255.255.254 7897

Trying 10.255.255.254...

telnet: Unable to connect to remote host: Connection refused

(venv) dministrator@WIN-UB1HRT8QI4S:~$

从测试结果来看，telnet仍提示Connection refused，说明 Windows 的 7897 端口确实未对外开放，核心问题出在代理软件的配置上。以下是必须执行的关键步骤：

1. 确认 Windows 代理软件的端口和监听设置

这是解决问题的核心，必须确保代理软件正确占用 7897 端口并允许局域网访问：

步骤 1：检查代理软件是否真的在使用 7897 端口

在 Windows 的 PowerShell 中执行（以管理员身份）：

powershell

# 查看所有端口占用情况，筛选7897

netstat -ano | findstr :7897

若没有任何输出：说明代理软件未启用 7897 端口，需在软件中手动设置端口为 7897 并重启（例如 Clash 的 “本地端口” 设置）。

若有输出：确认对应的进程（PID）是否为你的代理软件（通过任务管理器的 “详细信息” 标签页查看 PID 对应的程序）。

步骤 2：强制代理软件监听所有网络接口

代理软件默认可能只监听127.0.0.1（仅本地可访问），需改为监听0.0.0.0（允许所有网络访问）：

Clash 为例：

打开 Clash→“设置”→“网络”。

找到 “允许局域网连接” 并勾选。

确认 “本地地址” 设置为0.0.0.0（或留空让软件自动处理）。

重启 Clash 使设置生效。

其他软件（如 V2RayN、Netch）：

找到类似 “监听地址” 的设置，改为0.0.0.0，并开启 “允许局域网连接”。

2. 彻底关闭 Windows 防火墙或添加精确规则

防火墙是最常见的拦截原因，按以下步骤操作：

临时关闭防火墙测试（快速验证）

打开 Windows “控制面板”→“系统和安全”→“Windows Defender 防火墙”。

点击左侧 “关闭 Windows Defender 防火墙”，两个网络（专用 / 公用）都选择 “关闭”。

回到 WSL 终端，再次测试：

bash

telnet 10.255.255.254 7897

若此时成功连接（显示Connected），说明防火墙是阻碍因素，需添加规则（见下一步）。

若仍失败，继续排查代理软件设置。

为 7897 端口添加防火墙入站规则（推荐）

打开 “高级防火墙”→“入站规则”→“新建规则”。

选择 “端口”→“TCP”→“特定本地端口”，输入7897。

选择 “允许连接”→勾选 “域”、“专用”、“公用”（根据你的网络环境选择）。

命名为 “Allow Proxy 7897”，完成设置。

重启防火墙后测试连接。

3. 验证代理软件的运行状态

确保代理软件已 “启动” 并处于 “系统代理” 模式（而非仅 “规则模式”）。

在 Windows 浏览器中访问https://www.github.com，确认代理本身能正常工作（排除软件故障）。

若代理软件需要账号登录或订阅，确保已激活且未过期。

4. 最后尝试：更换代理端口

若 7897 端口始终无法开放，可能被系统占用或限制，可更换为其他端口（如 7890）：

在代理软件中修改本地端口为7890并重启。

在 WSL 中重新配置代理：

bash

export http\_proxy="http://10.255.255.254:7890"

export https\_proxy="http://10.255.255.254:7890"

telnet 10.255.255.254 7890 # 测试新端口

总结

问题的本质是Windows 的 7897 端口未被代理软件正确占用并开放给局域网，需按优先级排查：

代理软件是否启用 7897 端口并监听所有接口。

防火墙是否拦截该端口的入站连接。

代理软件本身是否正常运行。

完成上述步骤后，telnet和curl应能成功连接，代理即可生效。

Windows PowerShell

版权所有（C） Microsoft Corporation。保留所有权利。

安装最新的 PowerShell，了解新功能和改进！https://aka.ms/PSWindows

PS C:\Users\Administrator> nvcc --version

nvcc: NVIDIA (R) Cuda compiler driver

Copyright (c) 2005-2025 NVIDIA Corporation

Built on Wed\_Jan\_15\_19:38:46\_Pacific\_Standard\_Time\_2025

Cuda compilation tools, release 12.8, V12.8.61

Build cuda\_12.8.r12.8/compiler.35404655\_0

PS C:\Users\Administrator> # 查看所有端口占用情况，筛选7897

PS C:\Users\Administrator> netstat -ano | findstr :7897

TCP 0.0.0.0:7897 0.0.0.0:0 LISTENING 22420

TCP 127.0.0.1:7897 127.0.0.1:54021 TIME\_WAIT 0

TCP 127.0.0.1:7897 127.0.0.1:54041 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:54063 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:54076 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:55323 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:55335 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:55371 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:55381 TIME\_WAIT 0

TCP 127.0.0.1:7897 127.0.0.1:55383 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:55387 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:55393 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:55398 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:55404 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:55406 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:55420 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:55424 TIME\_WAIT 0

TCP 127.0.0.1:7897 127.0.0.1:55432 TIME\_WAIT 0

TCP 127.0.0.1:7897 127.0.0.1:55534 TIME\_WAIT 0

TCP 127.0.0.1:7897 127.0.0.1:55539 TIME\_WAIT 0

TCP 127.0.0.1:7897 127.0.0.1:55554 TIME\_WAIT 0

TCP 127.0.0.1:7897 127.0.0.1:55586 FIN\_WAIT\_2 22420

TCP 127.0.0.1:7897 127.0.0.1:55588 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:55609 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:55611 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:55615 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:55631 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:55633 ESTABLISHED 22420

TCP 127.0.0.1:7897 127.0.0.1:55639 ESTABLISHED 22420

TCP 127.0.0.1:54005 127.0.0.1:7897 TIME\_WAIT 0

TCP 127.0.0.1:54012 127.0.0.1:7897 TIME\_WAIT 0

TCP 127.0.0.1:54015 127.0.0.1:7897 TIME\_WAIT 0

TCP 127.0.0.1:54018 127.0.0.1:7897 TIME\_WAIT 0

TCP 127.0.0.1:54021 127.0.0.1:7897 TIME\_WAIT 0

TCP 127.0.0.1:54041 127.0.0.1:7897 ESTABLISHED 26908

TCP 127.0.0.1:54063 127.0.0.1:7897 ESTABLISHED 16720

TCP 127.0.0.1:54076 127.0.0.1:7897 ESTABLISHED 16276

TCP 127.0.0.1:55323 127.0.0.1:7897 ESTABLISHED 16720

TCP 127.0.0.1:55335 127.0.0.1:7897 ESTABLISHED 26908

TCP 127.0.0.1:55371 127.0.0.1:7897 ESTABLISHED 16720

TCP 127.0.0.1:55383 127.0.0.1:7897 ESTABLISHED 16720

TCP 127.0.0.1:55387 127.0.0.1:7897 ESTABLISHED 16720

TCP 127.0.0.1:55393 127.0.0.1:7897 ESTABLISHED 16720

TCP 127.0.0.1:55398 127.0.0.1:7897 ESTABLISHED 16720

TCP 127.0.0.1:55404 127.0.0.1:7897 ESTABLISHED 16720

TCP 127.0.0.1:55406 127.0.0.1:7897 ESTABLISHED 16720

TCP 127.0.0.1:55420 127.0.0.1:7897 ESTABLISHED 16720

TCP 127.0.0.1:55586 127.0.0.1:7897 CLOSE\_WAIT 4508

TCP 127.0.0.1:55588 127.0.0.1:7897 ESTABLISHED 4508

TCP 127.0.0.1:55601 127.0.0.1:7897 TIME\_WAIT 0

TCP 127.0.0.1:55609 127.0.0.1:7897 ESTABLISHED 16720

TCP 127.0.0.1:55611 127.0.0.1:7897 ESTABLISHED 16720

TCP 127.0.0.1:55615 127.0.0.1:7897 ESTABLISHED 16720

TCP 127.0.0.1:55629 127.0.0.1:7897 TIME\_WAIT 0

TCP 127.0.0.1:55631 127.0.0.1:7897 ESTABLISHED 16720

TCP 127.0.0.1:55633 127.0.0.1:7897 ESTABLISHED 16720

TCP 127.0.0.1:55639 127.0.0.1:7897 ESTABLISHED 16720

TCP [::]:7897 [::]:0 LISTENING 22420

UDP 0.0.0.0:7897 \*:\* 22420

UDP [::]:7897 \*:\* 22420

PS C:\Users\Administrator>

从 Windows 的端口监听结果来看，代理软件已正确占用 7897 端口（0.0.0.0:7897表示监听所有网络接口），但 WSL 仍无法连接，问题出在WSL 与 Windows 的网络隔离或代理软件的访问控制上。以下是针对性解决方案：

1. 获取 WSL 的子网掩码和网关（关键）

WSL2 使用独立子网，需确认其网络范围是否被代理软件允许：

在 Ubuntu 终端执行：

bash

# 查看WSL的网络信息

ip addr show eth0

输出类似：

plaintext

inet 172.28.231.155/20 brd 172.28.239.255 scope global eth0

记录子网范围（如172.28.224.0/20，即172.28.224.1到172.28.239.255）。

2. 配置代理软件允许 WSL 子网访问

部分代理软件默认仅允许 Windows 本地（127.0.0.1）和特定局域网 IP 访问，需手动添加 WSL 子网：

Clash 为例：

打开 Clash→“设置”→“网络”。

在 “允许局域网连接” 的 “额外允许 IP” 中添加 WSL 子网（如172.28.224.0/20）。

重启 Clash 生效。

其他软件：在 “访问控制” 或 “允许 IP 列表” 中添加 WSL 的子网范围。

3. 通过 Windows 的 “公共网络” 防火墙规则放行

WSL 可能被识别为 “公共网络”，需单独配置规则：

打开 “高级防火墙”→“入站规则”→找到之前创建的 “Allow Proxy 7897” 规则。

右键→“属性”→“高级”→勾选 “公共”（确保规则应用于所有网络类型）。

点击 “范围”→“远程 IP 地址”→“下列 IP 地址”→添加 WSL 子网（如172.28.224.0/20）。

保存后，在 Ubuntu 中测试：

bash

telnet 10.255.255.254 7897

4. 使用 WSL2 的 “主机 IP 映射” 替代原始 IP

WSL2 中可通过/etc/hosts绑定 Windows 主机名，避免 IP 变化问题：

bash

# 编辑hosts文件

sudo nano /etc/hosts

# 添加一行（Windows主机名可在PowerShell中用`hostname`查看）

10.255.255.254 windows-host

保存后用主机名测试代理：

bash

export http\_proxy="http://windows-host:7897"

export https\_proxy="http://windows-host:7897"

curl -I https://www.github.com

5. 终极方案：关闭 WSL2 的网络隔离（临时测试）

若以上步骤仍失败，可尝试将 WSL2 切换为桥接模式（需管理员权限）：

在 Windows PowerShell 中执行：

powershell

# 关闭WSL

wsl --shutdown

# 切换网络模式为桥接（替换"以太网"为你的网卡名称）

wsl --set-version Ubuntu-24.04 2

wsl --networking-mode Ubuntu-24.04 bridged --interface "以太网"

重启 WSL 后，重新获取 IP 并测试代理。

关键结论：

代理软件已正确监听端口，但 WSL 的请求被拦截，核心解决点是：

确保代理软件允许 WSL 所在的子网 IP 访问。

防火墙规则明确包含 WSL 的子网范围。

完成以上配置后，WSL 应能通过10.255.255.254:7897正常使用代理。

Windows PowerShell

版权所有（C） Microsoft Corporation。保留所有权利。

安装最新的 PowerShell，了解新功能和改进！https://aka.ms/PSWindows

PS C:\Users\Administrator> ipconfig

Windows IP 配置

以太网适配器 以太网:

媒体状态 . . . . . . . . . . . . : 媒体已断开连接

连接特定的 DNS 后缀 . . . . . . . : mshome.net

以太网适配器 以太网 6:

连接特定的 DNS 后缀 . . . . . . . :

本地链接 IPv6 地址. . . . . . . . : fe80::4ea8:7c81:66d4:b583%13

IPv4 地址 . . . . . . . . . . . . : 192.168.159.129

子网掩码 . . . . . . . . . . . . : 255.255.255.0

默认网关. . . . . . . . . . . . . : 192.168.159.136

无线局域网适配器 本地连接\* 1:

媒体状态 . . . . . . . . . . . . : 媒体已断开连接

连接特定的 DNS 后缀 . . . . . . . :

无线局域网适配器 本地连接\* 2:

媒体状态 . . . . . . . . . . . . : 媒体已断开连接

连接特定的 DNS 后缀 . . . . . . . :

以太网适配器 VMware Network Adapter VMnet1:

连接特定的 DNS 后缀 . . . . . . . :

本地链接 IPv6 地址. . . . . . . . : fe80::ca56:7618:fc9a:e03a%11

IPv4 地址 . . . . . . . . . . . . : 192.168.61.1

子网掩码 . . . . . . . . . . . . : 255.255.255.0

默认网关. . . . . . . . . . . . . :

以太网适配器 VMware Network Adapter VMnet8:

连接特定的 DNS 后缀 . . . . . . . :

本地链接 IPv6 地址. . . . . . . . : fe80::85a4:51dc:32e1:4f12%24

IPv4 地址 . . . . . . . . . . . . : 192.168.38.1

子网掩码 . . . . . . . . . . . . : 255.255.255.0

默认网关. . . . . . . . . . . . . :

以太网适配器 vEthernet (WSL (Hyper-V firewall)):

连接特定的 DNS 后缀 . . . . . . . :

本地链接 IPv6 地址. . . . . . . . : fe80::49e5:a117:a860:d502%57

IPv4 地址 . . . . . . . . . . . . : 172.22.48.1

子网掩码 . . . . . . . . . . . . : 255.255.240.0

默认网关. . . . . . . . . . . . . :

PS C:\Users\Administrator> wsl --shutdown

PS C:\Users\Administrator>

从你提供的 Windows 网络配置来看，WSL2 对应的虚拟网卡（vEthernet (WSL (Hyper-V firewall))）的 IP 是 172.22.48.1，子网掩码为 255.255.240.0（即子网范围为 172.22.48.1 到 172.22.63.255）。这解释了之前的连接问题：WSL 的 IP 属于该子网，而代理软件可能未允许此范围的访问。

以下是针对性解决步骤：

1. 在 WSL 中确认自身 IP（属于 WSL 子网）

在 Ubuntu 终端执行：

bash

# 查看WSL的IP（应属于172.22.48.0/20子网）

ip addr show eth0 | grep inet

输出类似 inet 172.22.50.123/20 ...，确认其在 172.22.48.1-172.22.63.255 范围内。

2. 修改代理软件配置，允许 WSL 子网访问

关键是让代理软件明确允许 WSL 所在的子网（172.22.48.0/20）：

Clash 为例：

打开 Clash→“设置”→“网络”。

确保 “允许局域网连接” 已勾选。

在 “局域网允许列表” 或 “额外允许 IP” 中添加 172.22.48.0/20（子网范围）。

重启 Clash 使配置生效。

其他代理软件：在 “访问控制” 或 “允许 IP” 设置中添加该子网，确保 WSL 的 IP 被包含。

3. 使用 WSL 虚拟网卡的 IP（172.22.48.1）作为代理地址

WSL 应通过 Windows 的 WSL 虚拟网卡 IP 连接代理，而非之前的 10.255.255.254（可能是其他网络接口的 IP）。在 Ubuntu 终端执行：

bash

# 设置代理地址为WSL虚拟网卡的IP（172.22.48.1）

export http\_proxy="http://172.22.48.1:7897"

export https\_proxy="http://172.22.48.1:7897"

# 测试连接

curl -I https://www.github.com

4. 验证防火墙规则是否包含 WSL 子网

确保 Windows 防火墙允许 WSL 子网访问 7897 端口：

打开 “高级防火墙”→“入站规则”→找到 “Allow Proxy 7897” 规则。

右键→“属性”→“范围”→“远程 IP 地址”→添加 172.22.48.0/20。

保存后再次测试 telnet 172.22.48.1 7897。

5. 永久配置代理（避免 IP 变化问题）

由于 WSL 虚拟网卡 IP（172.22.48.1）通常固定，可将其写入配置文件：

bash

# 编辑bash配置

nano ~/.bashrc

# 添加代理配置（使用WSL虚拟网卡IP）

export http\_proxy="http://172.22.48.1:7897"

export https\_proxy="http://172.22.48.1:7897"

export no\_proxy="localhost,127.0.0.1,172.22.48.1"

# 生效配置

source ~/.bashrc

关键说明：

WSL2 通过虚拟网卡与 Windows 通信，正确的代理地址应为该网卡的 IP（172.22.48.1），而非其他网络接口的 IP。

代理软件必须允许 172.22.48.0/20 子网访问，否则会拦截 WSL 的请求。

若仍失败，可在 Windows 中暂时关闭防火墙，测试是否为防火墙规则问题。

完成以上步骤后，WSL 应能正常通过代理访问网络。

┌──────────────────────────────────────────────────────────────────────┐

│ • MobaXterm Personal Edition v25.2 • │

│ (SSH client, X server and network tools) │

│ │

│ ⮞ Linux distribution: Ubuntu-24.04 │

│ ⮞ Windows drives are mounted into /mnt path (by default) │

│ ⮞ WSL DISPLAY is automatically redirected to Windows desktop │

│ ⮞ WSL filesystem is accessible in the sidebar browser │

│ ⮞ For more info, ctrl+click on help or visit our website. │

└──────────────────────────────────────────────────────────────────────┘

dministrator@WIN-UB1HRT8QI4S:~$ # 设置代理地址为WSL虚拟网卡的IP（172.22.48.1）

export http\_proxy="http://172.22.48.1:7897"

export https\_proxy="http://172.22.48.1:7897"

# 测试连接

curl -I https://www.github.com

HTTP/1.1 200 Connection established

HTTP/2 301

content-length: 0

location: https://github.com/

strict-transport-security: max-age=31536000; includeSubDomains; preload

dministrator@WIN-UB1HRT8QI4S:~$

太好了！从测试结果来看，代理配置已经成功生效（HTTP/2 301 表示通过代理正常访问了 GitHub）。这说明使用 WSL 虚拟网卡的 IP（172.22.48.1）和端口 7897 的配置是正确的。

接下来你可以正常使用代理进行网络操作，例如安装 Python 包：

bash

# 在虚拟环境中安装依赖（以PyTorch为例）

pip3 install torch torchvision

# 更新系统软件包（如需）

sudo apt update && sudo apt upgrade -y

补充说明：

代理持久性：如果希望每次打开终端都自动生效代理，可将配置写入.bashrc（之前提到的方法），避免重复输入。

代理验证：后续若遇到网络问题，可再次用 curl -I https://www.github.com 验证代理是否正常。

软件适配：大部分命令行工具（apt、pip、git等）会自动使用系统环境变量中的代理配置，无需额外设置。

现在你的 WSL 环境已能通过代理正常访问网络，可继续进行后续的开发或部署工作了。