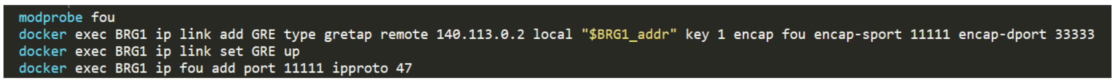
1. Show the configuration commands you made on each node to provide Internet connectivity for hosts and briefly explain the purpose of the commands

a) BRG1

- GRE over UDP (statically)

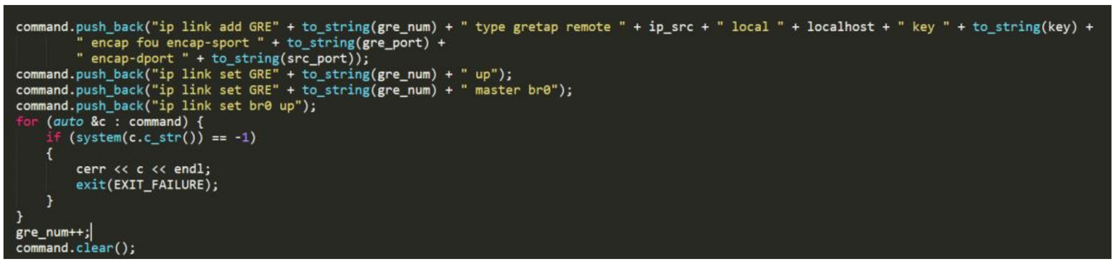


```
modprobe fou
docker exec BRG1 ip link add GRE type gretap remote 140.113.0.2 local "$BRG1_addr" key 1 encap fou encap-sport 11111 encap-dport 33333
docker exec BRG1 ip link set GRE up
docker exec BRG1 ip fou add port 11111 ipproto 47
```

載入 fou module，才可以讓 ip link 設定 UDP 相關網路通道

b) BRGr

- GRE over UDP (dynamically)

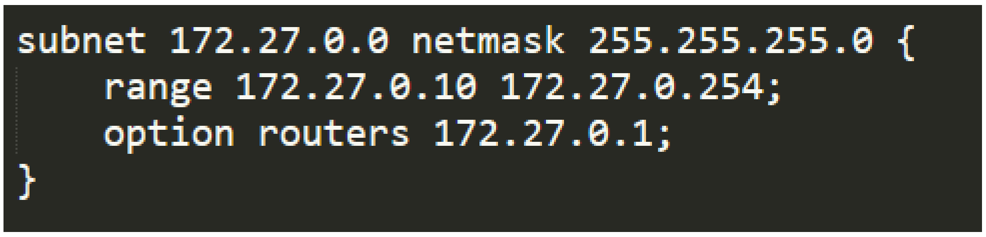


```
command.push_back("ip link add GRE" + to_string(gre_num) + " type gretap remote " + ip_src + " local " + localhost + " key " + to_string(key) +
        " encap fou encap-sport " + to_string(gre_port) +
        " encap-dport " + to_string(src_port));
command.push_back("ip link set GRE" + to_string(gre_num) + " up");
command.push_back("ip link set GRE" + to_string(gre_num) + " master br0");
command.push_back("ip link set br0 up");
for (auto &c : command) {
    if (system(c.c_str()) == -1)
    {
        cerr << c << endl;
        exit(EXIT_FAILURE);
    }
}
gre_num++;
command.clear();
```

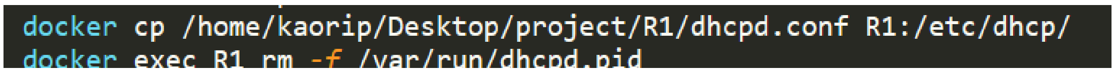
基本上使用的指令同 statically，不同的是 dinamically 的部分需要等待封包解析擷取 IP 的部分，再做 tunnel 建立。

c) Edge Router

- DHCP for BRG1, BRG2



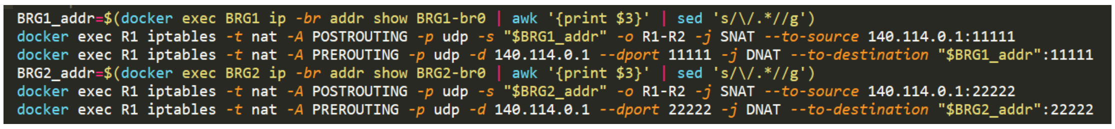
```
subnet 172.27.0.0 netmask 255.255.255.0 {
    range 172.27.0.10 172.27.0.254;
    option routers 172.27.0.1;
}
```



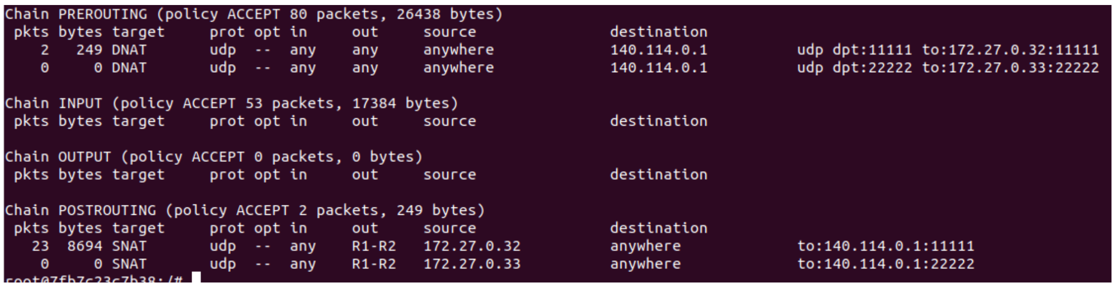
```
docker cp /home/kaorip/Desktop/project/R1/dhcpd.conf R1:/etc/dhcp/
docker exec R1 rm -f /var/run/dhcpd.pid
```

設定好 IP 位置後就能將 dhcpd.conf 複製到 R1 的 dhcp 資料夾底下運行。另外避免 IP 範圍過大出錯，所以 range 設定為 10~254…。

- NAT rules for BRG1 (show NAT tables to justify your answer)



```
BRG1_addr=$(docker exec BRG1 ip -br addr show BRG1-br0 | awk '{print $3}' | sed 's/\/.*//g')
docker exec R1 iptables -t nat -A POSTROUTING -p udp -s "$BRG1_addr" -o R1-R2 -j SNAT --to-source 140.114.0.1:11111
docker exec R1 iptables -t nat -A PREROUTING -p udp -d 140.114.0.1 --dport 11111 -j DNAT --to-destination "$BRG1_addr":11111
BRG2_addr=$(docker exec BRG2 ip -br addr show BRG2-br0 | awk '{print $3}' | sed 's/\/.*//g')
docker exec R1 iptables -t nat -A POSTROUTING -p udp -s "$BRG2_addr" -o R1-R2 -j SNAT --to-source 140.114.0.1:22222
docker exec R1 iptables -t nat -A PREROUTING -p udp -d 140.114.0.1 --dport 22222 -j DNAT --to-destination "$BRG2_addr":22222
```

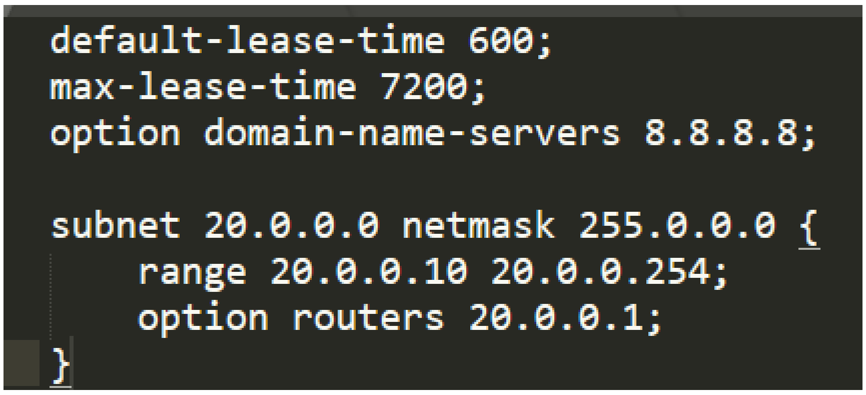


```
Chain PREROUTING (policy ACCEPT 80 packets, 26438 bytes)
 pkts bytes target     prot opt in     out     source               destination
    2   249 DNAT       udp  --  any    any     anywhere             140.114.0.1          udp dpt:11111 to:172.27.0.32:11111
    0     0 DNAT       udp  --  any    any     anywhere             140.114.0.1          udp dpt:22222 to:172.27.0.33:22222

Chain INPUT (policy ACCEPT 53 packets, 17384 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 2 packets, 249 bytes)
 pkts bytes target     prot opt in     out     source               destination
   23  8694 SNAT       udp  --  any    R1-R2   172.27.0.32          anywhere             to:140.114.0.1:11111
    0     0 SNAT       udp  --  any    R1-R2   172.27.0.33          anywhere             to:140.114.0.1:22222
```

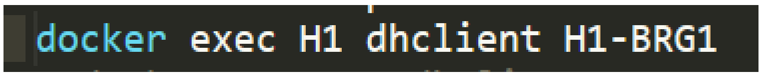
參照 lab3 的設定將 port 的部分從 tcp 改成 udp，且 source ip 與 destination ip 的 port 也須分別修改為 11111 與 33333

d) GWr

- DHCP for hosts



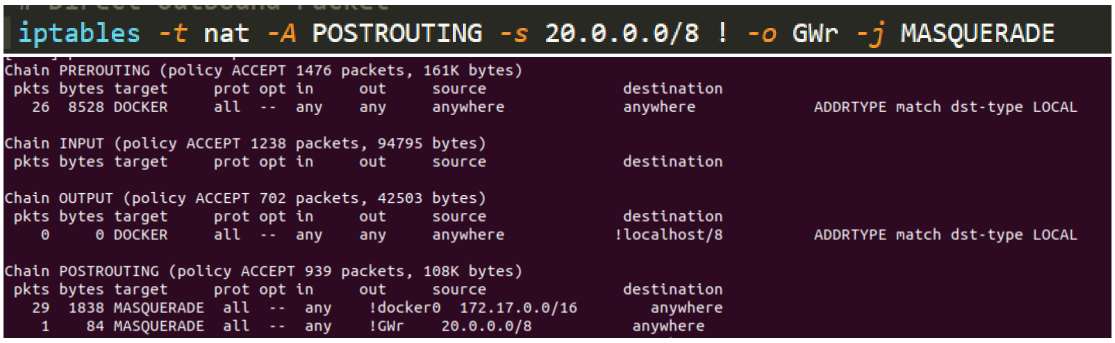
```
default-lease-time 600;
max-lease-time 7200;
option domain-name-servers 8.8.8.8;

subnet 20.0.0.0 netmask 255.0.0.0 {
    range 20.0.0.10 20.0.0.254;
    option routers 20.0.0.1;
}
```



```
docker exec H1 dhclient H1-BRG1
```

設定好 VM 內部的 DHCP server 因為題目要求是 20.0.0.0/8，因此 netmask 是 255.0.0.0

- NAT rules for hosts (show NAT tables to justify your answer)



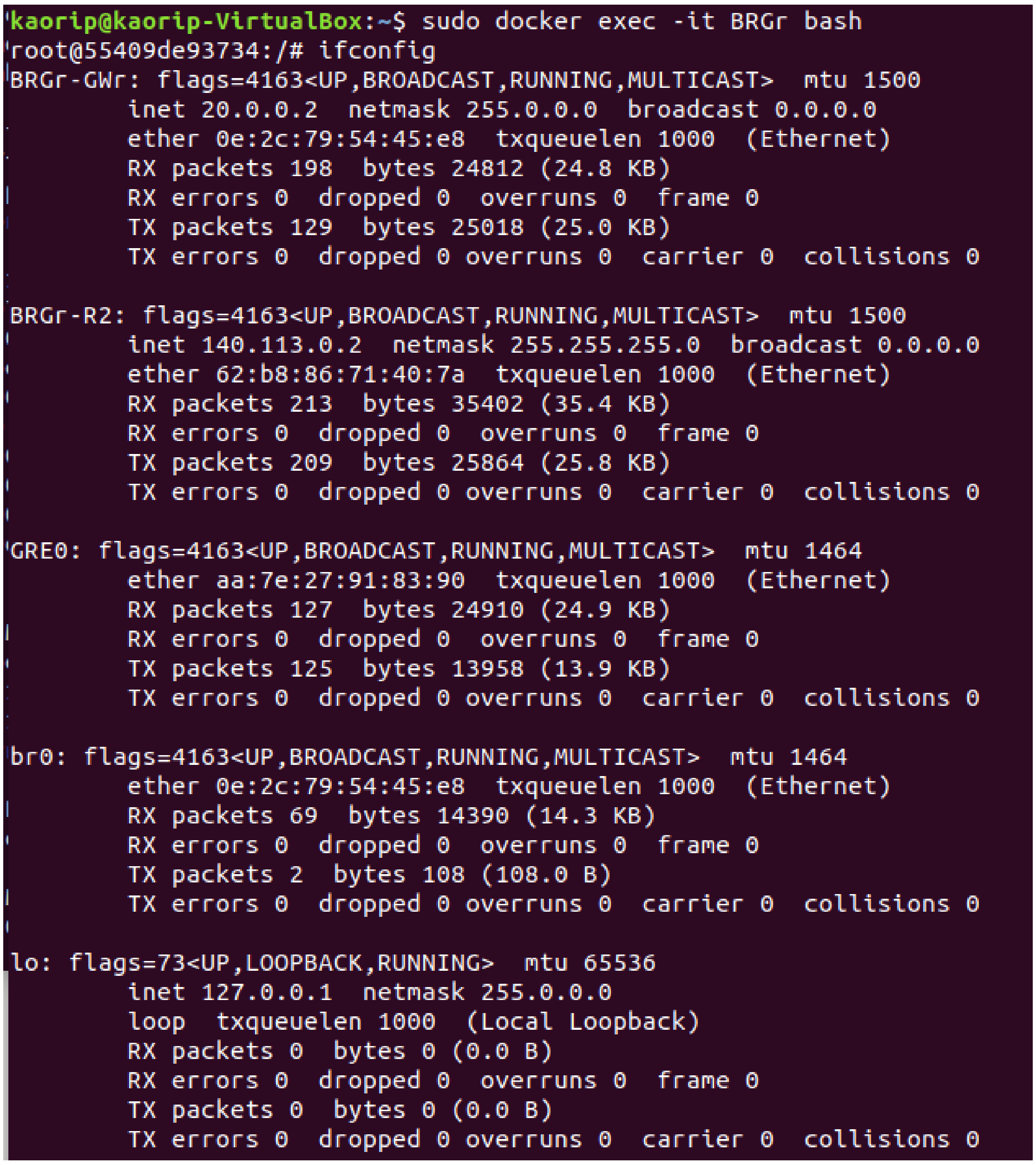
```
iptables -t nat -A POSTROUTING -s 20.0.0.0/8 ! -o GWr -j MASQUERADE
```

```
Chain PREROUTING (policy ACCEPT 1476 packets, 161K bytes)
 pkts bytes target     prot opt in     out     source               destination
   26  8528 DOCKER     all  --  any    any     anywhere             anywhere             ADDRTYPE match dst-type LOCAL

Chain INPUT (policy ACCEPT 1238 packets, 94795 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 702 packets, 42503 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DOCKER     all  --  any    any     anywhere            !localhost/8          ADDRTYPE match dst-type LOCAL

Chain POSTROUTING (policy ACCEPT 939 packets, 108K bytes)
 pkts bytes target     prot opt in     out     source               destination
   29  1838 MASQUERADE  all  --  any    !docker0  172.17.0.0/16        anywhere
    1    84 MASQUERADE  all  --  any    !GWr    20.0.0.0/8           anywhere
```

只要 source 是 20.0.0.0/8 這個網域且經過 GWr 的封包都需經過 NAT 轉換

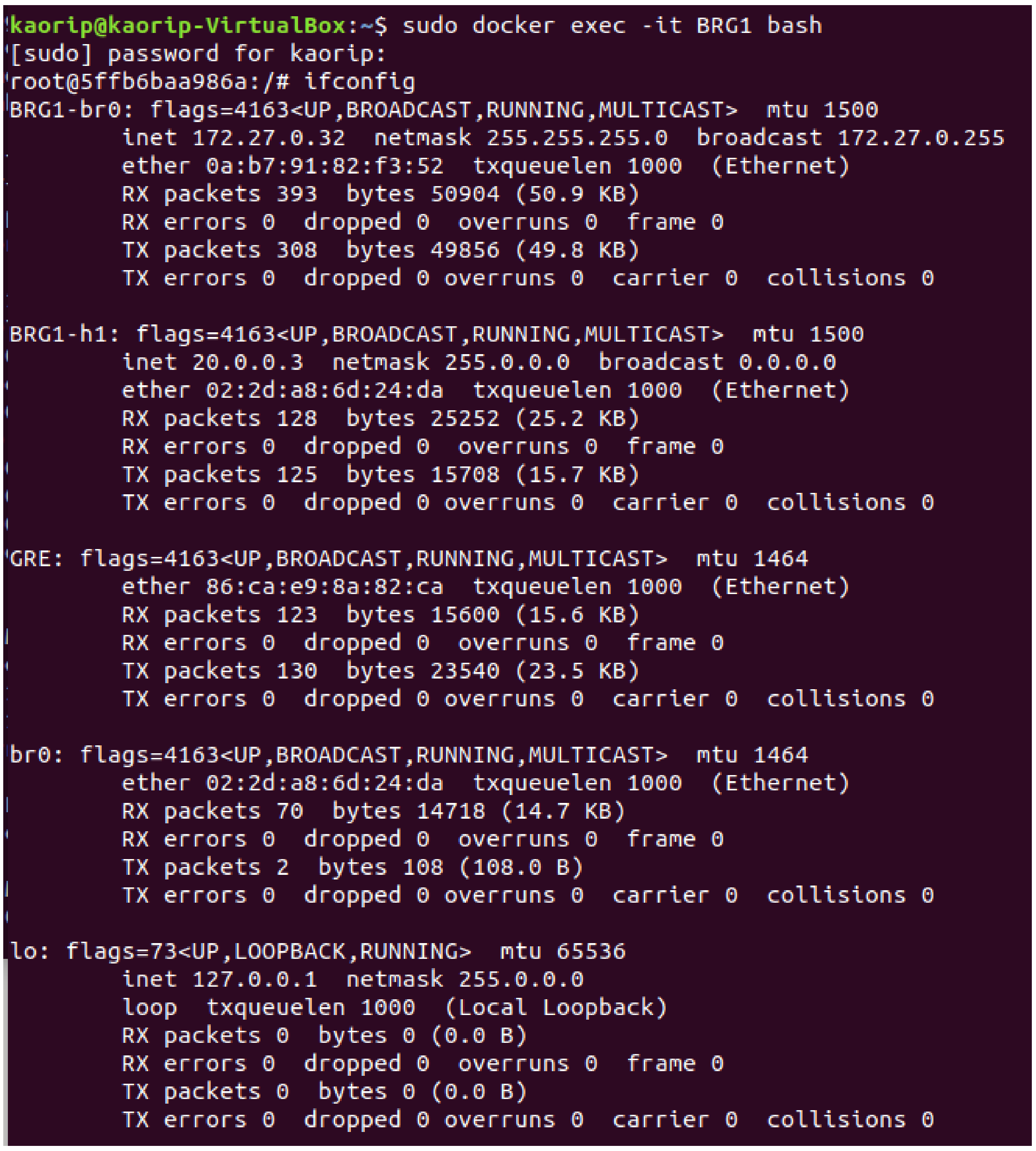
2. Show interfaces list on node BRGr and BRG1, 2

BRGr:



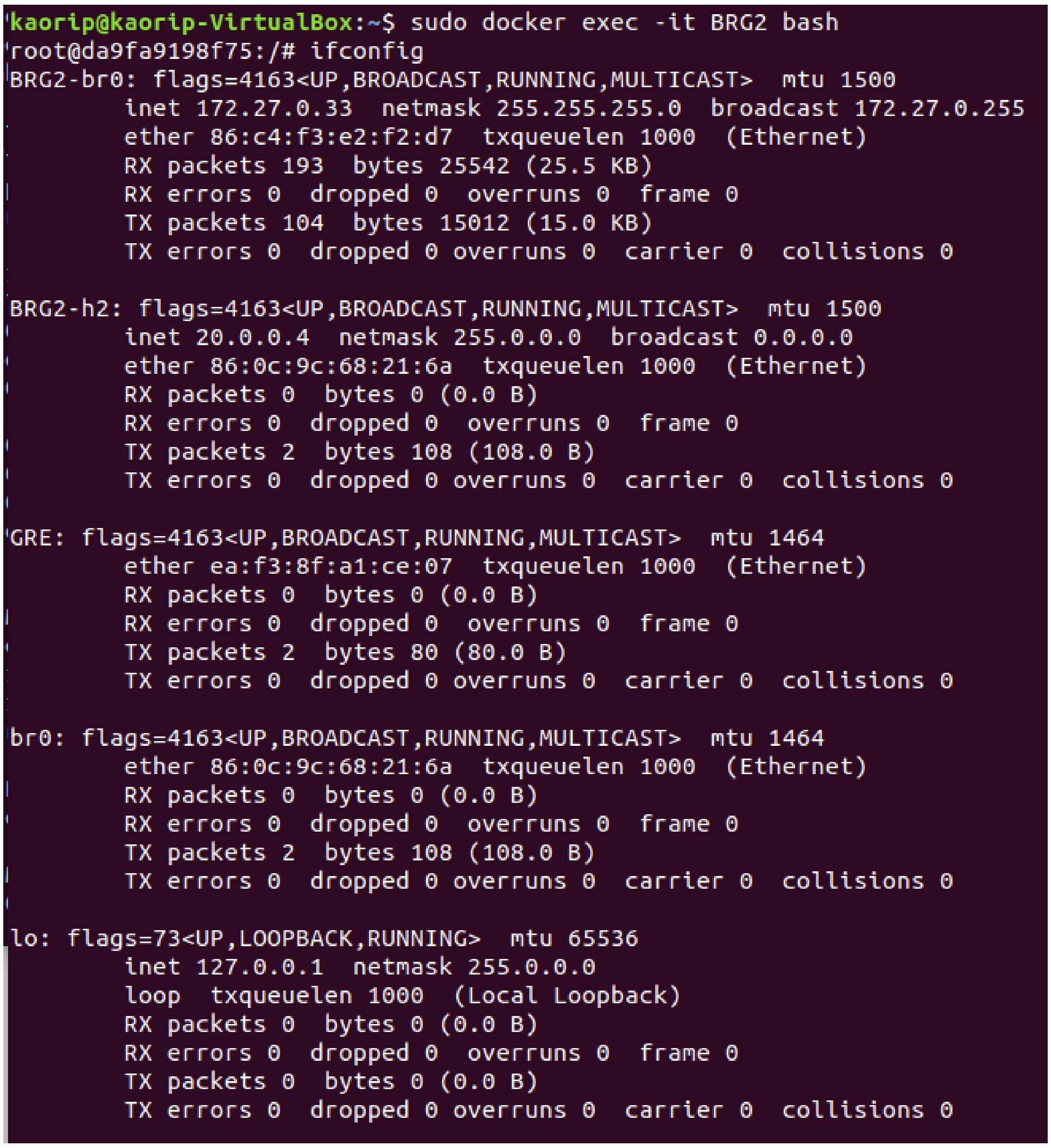
```
kaorip@kaorip-VirtualBox:~$ sudo docker exec -it BRGr bash
root@55409de93734:/# ifconfig
BRGr-GWr: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 20.0.0.2  netmask 255.0.0.0  broadcast 0.0.0.0
        ether 0e:2c:79:54:45:e8  txqueuelen 1000  (Ethernet)
        RX packets 198  bytes 24812 (24.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 129  bytes 25018 (25.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

BRGr-R2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 140.113.0.2  netmask 255.255.255.0  broadcast 0.0.0.0
        ether 62:b8:86:71:40:7a  txqueuelen 1000  (Ethernet)
        RX packets 213  bytes 35402 (35.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 209  bytes 25864 (25.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

GRE0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1464
        ether aa:7e:27:91:83:90  txqueuelen 1000  (Ethernet)
        RX packets 127  bytes 24910 (24.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 125  bytes 13958 (13.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1464
        ether 0e:2c:79:54:45:e8  txqueuelen 1000  (Ethernet)
        RX packets 69  bytes 14390 (14.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2  bytes 108 (108.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

BRG1:



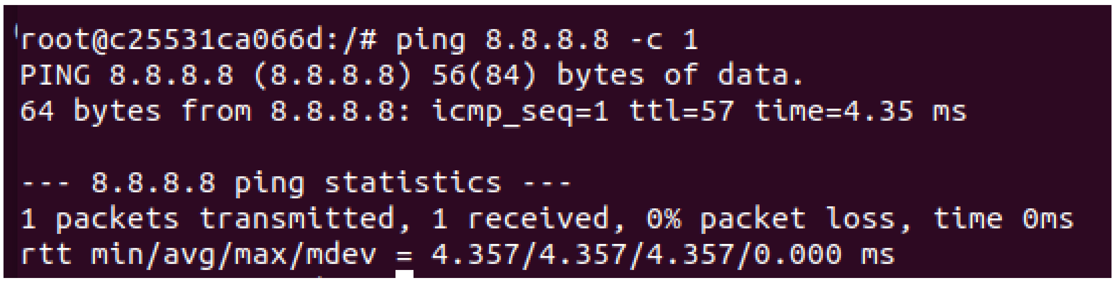
```
kaorip@kaorip-VirtualBox:~$ sudo docker exec -it BRG1 bash
[sudo] password for kaorip:
root@5ffb6baa986a:/# ifconfig
BRG1-br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.27.0.32  netmask 255.255.255.0  broadcast 172.27.0.255
        ether 0a:b7:91:82:f3:52  txqueuelen 1000  (Ethernet)
        RX packets 393  bytes 50904 (50.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 308  bytes 49856 (49.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

BRG1-h1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 20.0.0.3  netmask 255.0.0.0  broadcast 0.0.0.0
        ether 02:2d:a8:6d:24:da  txqueuelen 1000  (Ethernet)
        RX packets 128  bytes 25252 (25.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 125  bytes 15708 (15.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

GRE: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1464
        ether 86:ca:e9:8a:82:ca  txqueuelen 1000  (Ethernet)
        RX packets 123  bytes 15600 (15.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 130  bytes 23540 (23.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1464
        ether 02:2d:a8:6d:24:da  txqueuelen 1000  (Ethernet)
        RX packets 70  bytes 14718 (14.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2  bytes 108 (108.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

BRG2:



```
kaorip@kaorip-VirtualBox:~$ sudo docker exec -it BRG2 bash
root@da9fa9198f75:/# ifconfig
BRG2-br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.27.0.33  netmask 255.255.255.0  broadcast 172.27.0.255
        ether 86:c4:f3:e2:f2:d7  txqueuelen 1000  (Ethernet)
        RX packets 193  bytes 25542 (25.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 104  bytes 15012 (15.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

BRG2-h2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 20.0.0.4  netmask 255.0.0.0  broadcast 0.0.0.0
        ether 86:0c:9c:68:21:6a  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2  bytes 108 (108.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

GRE: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1464
        ether ea:f3:8f:a1:ce:07  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2  bytes 80 (80.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1464
        ether 86:0c:9c:68:21:6a  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2  bytes 108 (108.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

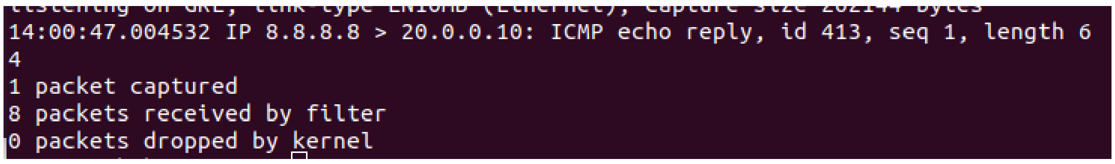
Let h1 ping google DNS server 8.8.8.8.



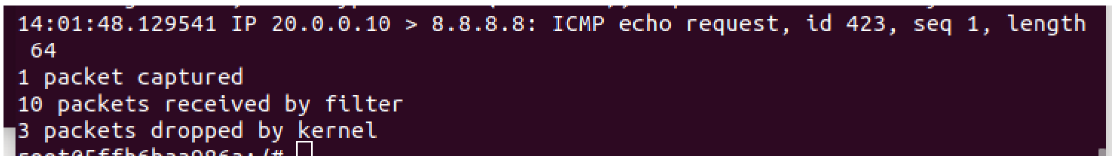
```
root@c25531ca066d:/# ping 8.8.8.8 -c 1
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=57 time=4.35 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 4.357/4.357/4.357/0.000 ms
```

3. Capture packets and take screenshots on node

- BRG1 input/output

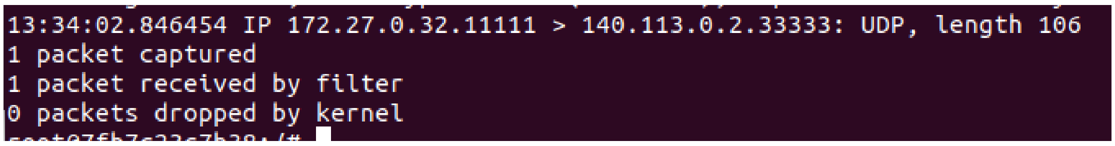


```
14:00:47.004532 IP 8.8.8.8 > 20.0.0.10: ICMP echo reply, id 413, seq 1, length 6
4
1 packet captured
8 packets received by filter
0 packets dropped by kernel
```

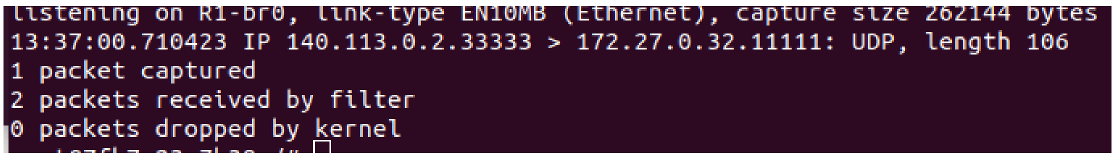


```
14:01:48.129541 IP 20.0.0.10 > 8.8.8.8: ICMP echo request, id 423, seq 1, length
 64
1 packet captured
10 packets received by filter
3 packets dropped by kernel
```

▪ Access Router input/output

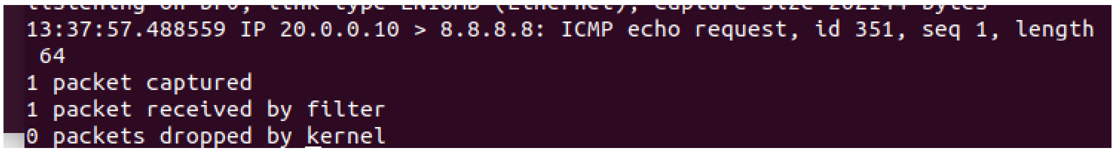


```
13:34:02.846454 IP 172.27.0.32.11111 > 140.113.0.2.33333: UDP, length 106
1 packet captured
1 packet received by filter
0 packets dropped by kernel
```



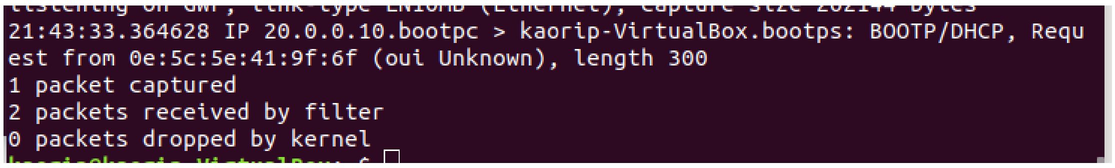
```
listening on R1-br0, link-type EN10MB (Ethernet), capture size 262144 bytes
13:37:00.710423 IP 140.113.0.2.33333 > 172.27.0.32.11111: UDP, length 106
1 packet captured
2 packets received by filter
0 packets dropped by kernel
```

▪ BRGr input/output

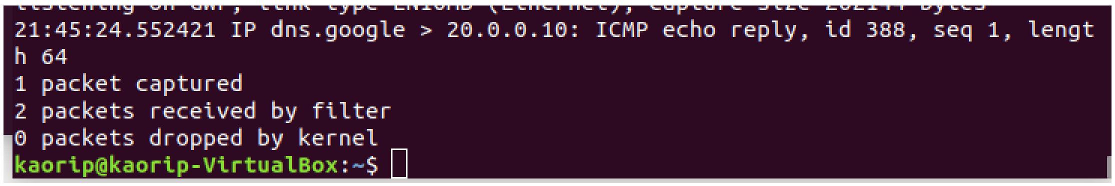


```
13:37:57.488559 IP 20.0.0.10 > 8.8.8.8: ICMP echo request, id 351, seq 1, length
 64
1 packet captured
1 packet received by filter
0 packets dropped by kernel
```

▪ GWr input/output



```
21:43:33.364628 IP 20.0.0.10.bootpc > kaorip-VirtualBox.bootps: BOOTP/DHCP, Requ
est from 0e:5c:5e:41:9f:6f (oui Unknown), length 300
1 packet captured
2 packets received by filter
0 packets dropped by kernel
```



```
21:45:24.552421 IP dns.google > 20.0.0.10: ICMP echo reply, id 388, seq 1, lengt
h 64
1 packet captured
2 packets received by filter
0 packets dropped by kernel
kaorip@kaorip-VirtualBox:~$
```

Briefly describe the header changes made by each node, by using the screenshots, to deliver packets from h1 to 8.8.8.8, and from 8.8.8.8 to h1, and briefly explain why such changes are required.

由上圖可見，由 h1 送出的封包到 R1 會改變 IP header 到 BRGr 改變一次後傳送到外部網路，用以增加內部網路的安全性。

4. BRGr will receive ping responses from Google DNS. Briefly describe how BRGr determines the GRE interface to tunnel the response packets back to BRG1.

確認到傳出對象為已建立好 GRE tunnel 的網域，BRGr 會再封裝封包加上

GRE header，BRG1 便可以辨識此封包的傳輸來源與目的。