ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

UNIVERSIDAD

Computing Networks

Laboratory No. 7.2

Basic Infrastructure and Network Layer

Students:

Andrea Camila Torres González

Jorge Andrés Gamboa Sierra

Presented to:

Fabian Eduardo Sierra Sánchez

Semester 2024-2

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

UNIVERSIDAD

Content

Objective

Continue learning about the operation of operating systems and network services.

Install network management tools.

Configure routers and static routing.

Abstract

In this lab, we will deepen our understanding of operating systems and network services. We will install network management tools, configure routers, and set up static and dynamic routes. We will create shell scripts for network diagnostics and configure monitoring servers on different operating systems. Using Azure, we will deploy and monitor web applications. Practical activities include tracing routes with ICMP, configuring routers, and setting up serial interconnections, gaining skills in network management and configuration. Finally, we will demonstrate our configurations to the professor to ensure everything is functioning correctly.
Key words: monitoring, configuration, networking, router, static, dynamic

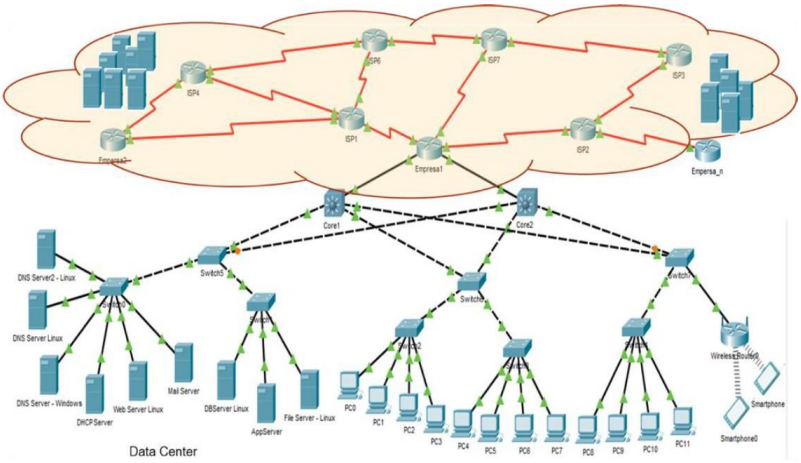
Tools to be Used

Computers
Internet access
Virtualization software
Packet Tracer and Wireshark
Routers and switches

Introduction

A company typically has several IT infrastructure services. It includes wired and wireless user stations and servers (both physical and virtualized), all connected through switches (layer 2 and 3), wireless devices, and routers that connect it to the Internet. It is also common to have cloud infrastructures where resources are provisioned according to the organization's needs. Among the servers, there can be web services, DNS, email, database, storage, applications, and others.

The following is a possible configuration:



Theoretical framework

IP Protocol (Internet Protocol) and IP Addressing

The IP Protocol is the backbone of the network layer in the OSI model, responsible for addressing and routing data packets from their origin to their final destination. There are two predominant versions: IPv4, which uses 32-bit addresses, and IPv6, which uses 128-bit addresses to allow for a greater number of unique addresses. IP addressing assigns unique addresses to each device on a network, enabling efficient segmentation and management of the network. Proper use of subnets and subnet masks is essential for organizing and optimizing network communication.

ICMP (Internet Control Message Protocol), Traceroute, and Online Routing Tools

The ICMP Protocol is used to send error messages and operational information about an IP network. It is crucial for network diagnostic tools like ping and traceroute. Traceroute tracks the path that data packets take from the source to the destination, providing information about each intermediate hop, useful for identifying network problems and verifying routing functionality. Online routing tools allow these diagnostics to be performed via web interfaces, facilitating analysis from any device with Internet access. These tools are valuable for monitoring and troubleshooting connectivity and performance issues in both local and remote networks.

Routing and Static Routes

Routing is the process of selecting paths for data to travel from one device to another across a network. **Routers** use **routing tables** to determine the best path for each data packet, based on factors like distance, cost, and network type. **Static routes** are manually configured in a router and do not change unless modified by the network administrator. They are useful in smaller or more controlled networks where routes do not change frequently, and there is no need for dynamic updates.

Static routes are simple and efficient for small networks but have limitations in terms of flexibility and scalability. In larger or dynamic networks, if a link fails or the network topology changes, the administrator must manually update static routes on each router, which can be prone to errors and lead to downtime.

Dynamic Routing

Is a more flexible and automatic approach where routers exchange routing information to adapt to network changes without manual intervention by the administrator. Through dynamic routing protocols, routers learn about network topology and automatically adjust their routing tables based on changes, such as link failures or the addition of new routers. This type of routing is essential for large networks or those that experience frequent structural changes, as it minimizes human error and reduces downtime.

Dynamic Routing Protocols:

- **RIP (Routing Information Protocol)**:
  RIP is one of the oldest dynamic routing protocols. It uses **hop count** as its metric to

determine the best route to a destination. The maximum allowed hop count in RIP is 15, meaning any route requiring more than 15 hops is considered unreachable. While it is an easy protocol to configure, RIP has significant disadvantages:

- **Limited scalability**: due to its 15-hop limit, it is not suitable for large or complex networks.
- **Slow convergence**: in case of network changes (like a link failure), RIP can take a considerable amount of time to react and update routes.
- **Lacks advanced features**: such as load balancing and more sophisticated security measures, making it less efficient for larger networks or enterprise environments.

Although RIP is no longer widely used in large networks due to these limitations, it remains useful for smaller networks and educational scenarios.

- **OSPF (Open Shortest Path First)**:
  OSPF is a link-state routing protocol, meaning each router maintains a complete representation of the network topology. Instead of relying on hop count, OSPF uses **cost** (usually based on bandwidth) as its metric to determine the best route. Some of the features that make OSPF more advanced and suitable for large networks are:
  - **Fast convergence**: OSPF updates its routing tables quickly when a change occurs in the network, minimizing downtime.
  - **Scalability**: OSPF can handle large networks by dividing the network into **areas**. Each area has an independent topology, reducing the amount of information each router needs to process.
  - **Redundancy and load balancing**: OSPF supports **multiple paths** to a destination, allowing for load balancing across multiple links.

NAT (Network Address Translation) and VLANs (Virtual Local Area Networks)

NAT is a technique that maps private IP addresses to public IP addresses, allowing multiple devices on a local network to share a single public IP address. This conserves IP addresses and improves security by hiding the internal network structure from external devices. VLANs allow a physical network to be segmented into multiple logical networks, improving traffic management and security. Each VLAN behaves as an independent network, even though devices share the same physical hardware. This facilitates the management of large networks and improves performance by limiting the amount of traffic in each segment.

SNMP (Simple Network Management Protocol)

SNMP is a protocol used for monitoring and managing devices on a network. Network administrators use SNMP to collect information about the performance, availability, and status of network devices such as routers, switches, and servers. SNMP allows communication between devices and a management station, facilitating the automation of network administration tasks. It uses a hierarchical structure of objects called MIB (Management Information Base) to organize the data that can be queried and controlled.

Nagios: Network Monitoring Tool

Nagios is a network and system monitoring tool that enables IT administrators to identify and resolve infrastructure issues before they affect critical business processes. Nagios monitors the activity of servers, switches, applications, and network services, alerting administrators to any problems detected. Nagios' ability to integrate with other systems and monitoring tools makes it a versatile and powerful solution for maintaining network stability and performance.

CDN Networks (Content Delivery Network)

CDNs are distributed networks of servers that work together to deliver Internet content quickly and efficiently to users. CDNs cache copies of content at various geographically dispersed points, reducing latency and improving access speed. They are essential for enhancing user experience in web applications and streaming services by reducing load times and increasing reliability. CDNs also help mitigate DDoS (Denial of Service) attacks by distributing traffic evenly and ensuring greater content availability.

Basic information commands on the network

Linux

1. **ifconfig**: Displays and configures network interfaces.

Parameters:

- **-a**: Displays all interfaces, including inactive ones.
- **<interface>**: Shows configuration of a specific interface (e.g., eth0 or lo).
- **up**: Activates the network interface.
- **down**: Deactivates the network interface.
- **netmask <mask>**: Sets the subnet mask for the interface.
- **broadcast <address>**: Sets the broadcast address for the interface.
- **inet <IP>**: Assigns an IP address to the interface.
- **mtu <size>**: Sets the maximum transmission unit (MTU) size.

2. **Netstat:** Displays network statistics and connections.

Parameters:

- **-i**: Displays information about network interfaces.
- **-r**: Displays the routing table.
- **-a**: Displays all active connections and listening ports.
- **-n**: Displays IP addresses and ports in numeric format (without resolving names).
- **-t**: Displays only TCP connections.
- **-u**: Displays only UDP connections.
- **-p**: Displays the name of the process using the connection.

3. **route**: Displays and manipulates the routing table.

Parameters:

- **add**: Adds a new route.
- **delete**: Deletes an existing route.
- **default**: Sets a default route.
- **-n**: Displays the routing table in numeric format (without resolving names).
- **-v**: Displays the routing table with more detail (verbose).

4. **ping**: Verifies network connectivity using ICMP packets.

Parameters:

- **-c <n>**: Specifies the number of ICMP packets to send.
- **-s <size>**: Sets the size of the ICMP packets.
- **-i <interval>**: Sets the interval between ICMP packets.
- **-t <ttl>**: Sets the TTL (Time to Live) value for the packets.
- **-w <timeout>**: Sets a timeout in seconds to wait for a response.

5. **traceroute**: Displays the route that packets take to reach a destination.

Parameters:

- **-n**: Displays the hop addresses in numeric format (without resolving names).
- **-m <hops>**: Limits the number of hops.
- **-q <queries>**: Sets the number of queries per hop.

6. **nslookup:** Performs DNS queries.

Parameters:

- **<domain>**: Queries a specific domain.
- **server <dns-server>**: Specifies a DNS server to query.
- **set type=<type>**: Sets the query type (e.g., A, MX, NS).

7. **ethtool**: Displays or configures parameters for Ethernet network interfaces.

Parameters:

- **-i <interface>**: Displays information about the network interface, such as the driver.
- **-s <interface>**: Configures settings for the interface (e.g., speed and duplex).
- **-a <interface>**: Displays the auto-negotiation status of the interface.
- **-p <interface>**: Blinks the network interface LED for identification.

8. **hostname**: Displays or sets the system's hostname.

Parameters:

- **-f**: Displays the fully qualified domain name (FQDN).
- **<hostname>**: Sets a new hostname.

PowerShell

1. **ipconfig**: Displays the current network configuration of all network interfaces.

Parameters:

- **/all**: Displays full information for all interfaces (including DNS and MAC addresses).
- **/release**: Releases the current DHCP lease.
- **/renew**: Renews the DHCP lease.
- **/flushdns**: Clears the DNS resolver cache.
- **/displaydns**: Displays the contents of the DNS resolver cache.
- **/registerdns**: Refreshes all DHCP leases and re-registers DNS names.

2. **netstat**: Displays network statistics and current active connections.

Parameters:

- **-a**: Displays all active connections and listening ports.
- **-n**: Displays the addresses and port numbers in numeric format (without resolving names).
- **-o**: Displays the owning process ID (PID) associated with each connection.
- **-p**: Displays the protocol used for each connection.
- **-r**: Displays the routing table.
- **-s**: Displays network statistics by protocol.

3. **tracert**: Traces the route packets take to a destination.

Parameters:

- **-h <max_hops>**: Limits the number of hops to the destination.
- **-d**: Prevents DNS resolution, shows IP addresses only.
- **-w <timeout>**: Sets the timeout in milliseconds for each reply.

4. **Ping:** Sends ICMP echo requests to verify network connectivity.

Parameters:

- **-t**: Pings the target indefinitely until stopped (Ctrl + C).
- **-n <count>**: Specifies the number of echo requests to send.
- **-l <size>**: Specifies the size of the ping packet.
- **-4**: Forces the use of IPv4.
- **-6**: Forces the use of IPv6.

5. **Route:** Displays and modifies the network routing table.

Parameters:

- **print**: Displays the routing table.

- **add**: Adds a new route.
- **delete**: Deletes a route.
- **change**: Modifies an existing route.
- **-f**: Clears the routing table and deletes all routes.

6. **nslookup**: Queries the DNS to resolve hostnames to IP addresses.

Parameters:

- **<hostname>**: Queries a specific domain name.
- **server <dns_server>**: Specifies a DNS server to use for the query.
- **set type=<record_type>**: Specifies the record type (e.g., A, MX, NS).
- **-type=<type>**: Specifies the record type (e.g., A, MX, PTR).

7. **getmac**: Displays the MAC (Media Access Control) address of the network interfaces.

Parameters:

- **/v**: Displays additional information (e.g., interface description).
- **/fo <format>**: Specifies the output format (e.g., table, csv).
- **/s <computer>**: Specifies a remote computer to query.

Base Software Installation

1. Other Useful Commands

a. Network Information Shell

Familiarize yourself with commands such as netstat, vnstat, route, and ethtool.
Create a Shell program that includes a menu to execute these commands, providing an easy-to-understand interface.

First, we created our shell script and designed a menu with the options shown in *Figure 1.* We used **netstat** to display network information such as ports, services, statistics, and UDP and TCP connections. We also used **route** to view the routing table and **ethtool** to check network interface information

```
 GNU nano 6.0                                  netstat.sh
#!/bin/bash
while true; do
        echo "Network information menu:"
        echo "1) Show network connections and listening ports"
        echo "2) Show network usage"
        echo "3) Show routing table"
        echo "4) Show network interface details"
        echo "5) Show connections with an especific protocol"
        echo "6) Exit"
        read option
        case $option in
                1)
                        echo "Showing listening ports and services:"
                        netstat -tuln
                ;;
                2)
                        echo "Network stadistics: "
                        netstat -i
                ;;
                3)
                        echo "Routing table:"
                        route -n
                ;;
                4)
                        echo "Network interface details"
                        ethtool eth1
                ;;
                5)
                        echo "Select protocol:"
                        echo "1. UDP"
                        echo "2. TCP"
                        read protocol
                        case $protocol in

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

Figure 1. First part of the shell script for network information

```
 GNU nano 6.0                                  netstat.sh
                ;;
                4)
                        echo "Network interface details"
                        ethtool eth1
                ;;
                5)
                        echo "Select protocol:"
                        echo "1. UDP"
                        echo "2. TCP"
                        read protocol
                        case $protocol in
                                1)
                                        echo "TCP Connections: "
                                        netstat -tan
                                ;;
                                2)
                                        echo "UDP Connections: "
                                        netstat -uan
                                ;;
                                *)
                                        echo "Invalid Protocol Option"
                                ;;
                        esac
                ;;
                6)
                        echo "Killing Program..."
                        break
                ;;
                *)
                        echo "Invalid Option"
                ;;
        esac
done
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line M-E Redo
```

Figure 2. Second part of the shell script for network information

We tested option 1 on the menu (Network connections and listening ports). As we can see, the transport layer protocols are displayed with the ports they are listening on, along with the allowed IP addresses

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:5432            0.0.0.0:*               LISTEN
tcp        0      0 192.168.20.100:53       0.0.0.0:*               LISTEN
tcp        0      0 192.168.20.100:53       0.0.0.0:*               LISTEN
tcp        0      0 192.168.20.100:53       0.0.0.0:*               LISTEN
tcp        0      0 192.168.20.100:53       0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 ::1:53                  :::*                    LISTEN
tcp6       0      0 ::1:53                  :::*                    LISTEN
tcp6       0      0 ::1:53                  :::*                    LISTEN
tcp6       0      0 ::1:53                  :::*                    LISTEN
tcp6       0      0 :::5432                 :::*                    LISTEN
tcp6       0      0 fe80::a00:27ff:fe98::53 :::*                    LISTEN
tcp6       0      0 fe80::a00:27ff:fe98::53 :::*                    LISTEN
tcp6       0      0 fe80::a00:27ff:fe98::53 :::*                    LISTEN
tcp6       0      0 fe80::a00:27ff:fe98::53 :::*                    LISTEN
udp        0      0 192.168.20.100:53       0.0.0.0:*
udp        0      0 192.168.20.100:53       0.0.0.0:*
udp        0      0 192.168.20.100:53       0.0.0.0:*
udp        0      0 192.168.20.100:53       0.0.0.0:*
udp        0      0 192.168.20.100:123      0.0.0.0:*
udp        0      0 127.0.0.1:123           0.0.0.0:*
udp        0      0 0.0.0.0:123             0.0.0.0:*
udp        0      0 0.0.0.0:42114           0.0.0.0:*
udp        0      0 0.0.0.0:161             0.0.0.0:*
udp        0      0 0.0.0.0:55768           0.0.0.0:*
udp6       0      0 ::1:53                  :::*
udp6       0      0 ::1:53                  :::*
udp6       0      0 ::1:53                  :::*
udp6       0      0 ::1:53                  :::*
udp6       0      0 fe80::a00:27ff:fe98::53 :::*
udp6       0      0 fe80::a00:27ff:fe98::53 :::*
udp6       0      0 fe80::a00:27ff:fe98::53 :::*
:_
```

Figure 3. Testing the first option of the script

We tested option 2 on the menu (Network usage). We can see information about eth1 (Network interface) and lo (Loopback)

```
Network information menu:
1) Show network connections and listening ports
2) Show network usage
3) Show routing table
4) Show network interface details
5) Show connections with an especific protocol
6) Exit
2
Network stadistics:
Kernel Interface table
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth1      1500    43797      0      0 0          4088      0      0      0 BMRU
lo       65536       12      0      0 0            12      0      0      0 LRU
(END)_
```

Figure 4. Testing second option of the script

We tested option 3 on the menu (Routing table). We can see the routes at the routing table

```
Network information menu:
1) Show network connections and listening ports
2) Show network usage
3) Show routing table
4) Show network interface details
5) Show connections with an especific protocol
6) Exit
3
Routing table:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.20.1    0.0.0.0         UG    0      0        0 eth1
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
192.168.20.0    0.0.0.0         255.255.255.0   U     0      0        0 eth1
(END)
```

Figure 5. Testing third option of the script

We tested option 4 on the menu (Network interface details). We can see the details of eth1

```
Network information menu:
1) Show network connections and listening ports
2) Show network usage
3) Show routing table
4) Show network interface details
5) Show connections with an especific protocol
6) Exit
4
Network interface details
Settings for eth1:
        Supported ports: [ TP ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Supported pause frame use: No
        Supports auto-negotiation: Yes
        Supported FEC modes: Not reported
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Advertised pause frame use: No
        Advertised auto-negotiation: Yes
        Advertised FEC modes: Not reported
        Speed: 1000Mb/s
        Duplex: Full
        Auto-negotiation: on
        Port: Twisted Pair
        PHYAD: 0
        Transceiver: internal
        MDI-X: off (auto)
        Supports Wake-on: umbg
        Wake-on: d
        Current message level: 0x00000007 (7)
                               drv probe link
        Link detected: yes
(END)
```

Figure 6. Testing fourth option of the script

We tested option 5 on the menu (UDP connections and TCP connections)
First, we select the TCP option. We can see the ports that TCP is listening to

```
Network information menu:
1) Show network connections and listening ports
2) Show network usage
3) Show routing table
4) Show network interface details
5) Show connections with an especific protocol
6) Exit
5
Select protocol:
1. UDP
2. TCP
1
TCP Connections:
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:5432            0.0.0.0:*               LISTEN
tcp        0      0 192.168.20.100:53       0.0.0.0:*               LISTEN
tcp        0      0 192.168.20.100:53       0.0.0.0:*               LISTEN
tcp        0      0 192.168.20.100:53       0.0.0.0:*               LISTEN
tcp        0      0 192.168.20.100:53       0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 ::1:53                  :::*                    LISTEN
tcp6       0      0 ::1:53                  :::*                    LISTEN
tcp6       0      0 ::1:53                  :::*                    LISTEN
tcp6       0      0 ::1:53                  :::*                    LISTEN
tcp6       0      0 :::5432                 :::*                    LISTEN
tcp6       0      0 fe80::a00:27ff:fe98::53 :::*                    LISTEN
tcp6       0      0 fe80::a00:27ff:fe98::53 :::*                    LISTEN
tcp6       0      0 fe80::a00:27ff:fe98::53 :::*                    LISTEN
tcp6       0      0 fe80::a00:27ff:fe98::53 :::*                    LISTEN
(END)
```

Figure 7. Testing fifth option with TCP

Then select UDP option. We can see the ports that UDP is listening to

```
UDP Connections:
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 192.168.20.100:53       0.0.0.0:*
udp        0      0 192.168.20.100:53       0.0.0.0:*
udp        0      0 192.168.20.100:53       0.0.0.0:*
udp        0      0 192.168.20.100:53       0.0.0.0:*
udp        0      0 192.168.20.100:123      0.0.0.0:*
udp        0      0 127.0.0.1:123           0.0.0.0:*
udp        0      0 0.0.0.0:123             0.0.0.0:*
udp        0      0 0.0.0.0:42114           0.0.0.0:*
udp        0      0 0.0.0.0:161             0.0.0.0:*
udp        0      0 0.0.0.0:55768           0.0.0.0:*
udp6       0      0 ::1:53                  :::*
udp6       0      0 ::1:53                  :::*
udp6       0      0 ::1:53                  :::*
udp6       0      0 ::1:53                  :::*
udp6       0      0 fe80::a00:27ff:fe98::53 :::*
udp6       0      0 fe80::a00:27ff:fe98::53 :::*
udp6       0      0 fe80::a00:27ff:fe98::53 :::*
udp6       0      0 fe80::a00:27ff:fe98::53 :::*
udp6       0      0 fe80::a00:27ff:fe98:123 :::*
udp6       0      0 ::1:123                 :::*
udp6       0      0 :::123                  :::*
(END)
```

Figure 8. Testing fifth option with UDP

2. Network Management

On the other hand, part of an organization's technological platform is the monitoring server. Through it, administrators can check the status of the network equipment they manage. These platform management tools enable remote monitoring of disk space, CPU usage, network performance, memory usage, installed software, among other aspects of the network devices.

2.1. Installing Nagios on Solaris

To monitor our virtual machines, we installed Nagios, a real-time monitoring software. First, we need to install some libraries:

- ✓ gd
- ✓ gcc-53

```
root@solaris:~# pkg install gd
                      Paquetes que instalar: 15
                      Servicios que cambiar:  2
                    Crear entorno de inicio: No
Crear copia de seguridad de entorno de inicio: No

DOWNLOAD                                  PKGS         FILES    XFER (MB)   SPEED
Completado                              15/15       434/434    73.2/73.2  3.0M/s

PHASE                                          ITEMS
Instalando acciones nuevas                   995/995
Actualizando base de datos de estado de paquete      Listo
Actualizando cachÃ© de paquete                  0/0
Actualizando estado de imagen                Listo
Creando base de datos de bÃºsqueda rÃ¡pida  en proceso -Loading smf(7) service d
Creando base de datos de bÃºsqueda rÃ¡pida  en proceso -
Creando base de datos de bÃºsqueda rÃ¡pida      Listo
Actualizando cachÃ© de paquete                  1/1
root@solaris:~#
```

Figure 9. Installing gd on Solaris

```
root@solaris:~# pkg install gcc-53
                      Paquetes que instalar: 11
                    Mediadores que cambiar:  1
                      Servicios que cambiar:  1
                    Crear entorno de inicio: No
Crear copia de seguridad de entorno de inicio: No

DOWNLOAD                                  PKGS         FILES    XFER (MB)   SPEED
Completado                              11/11     1679/1679  263.4/263.4  1.5M/s

PHASE                                          ITEMS
Instalando acciones nuevas                 2028/2028
Actualizando base de datos de estado de paquete      Listo
Actualizando cachÃ© de paquete                  0/0
Actualizando estado de imagen                Listo
Creando base de datos de bÃºsqueda rÃ¡pida      Listo
Actualizando cachÃ© de paquete                  1/1
root@solaris:~#
```

Figure 10. Installing gcc on Solaris

Then, we go to the official Nagios website, download the **.tar.gz** file and transfer it to the Solaris machine using Samba

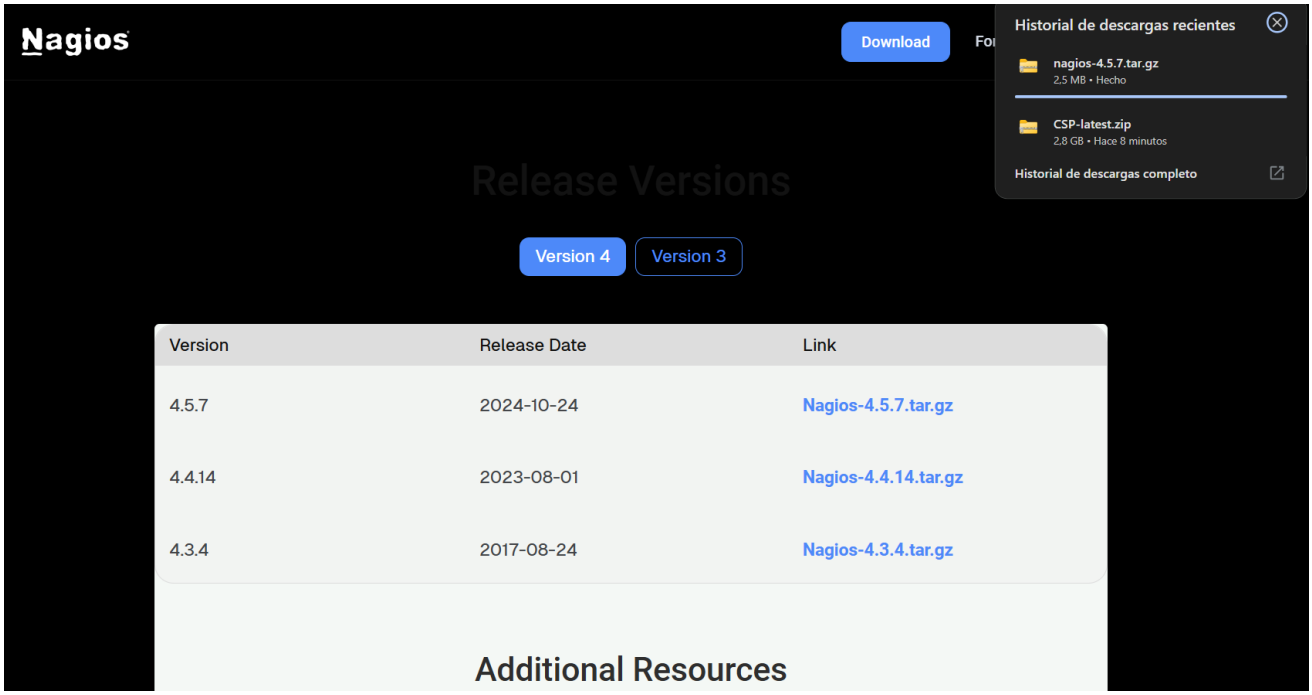


Figure 11. Downloading nagios.tar.gz file

We decompress the file using **tar -xzf nagios-4.5.7.tar.gz**

```
root@solaris:~# ls
CSP-latest.zip        nagios-4.5.7.tar.gz
root@solaris:~# tar -xzf nagios-4.5.7.tar.gz
```

Figure 12. Decompressing .tar.gz file

We move the generated file (nagios-4.5.7) to the **/nagios** directory and navigate to it

```
root@solaris:~# mkdir /nagios
root@solaris:~# mv nagios-4.5.7 nagios
root@solaris:~# cd nagios
root@solaris:~/nagios# ls
aclocal.m4        debian            LICENSE           t
autoconf-macros   docs              make-tarball      t-tap
base              doxy.conf         Makefile.in       tap
cgi               functions         mkpackage         test
Changelog         html              module            THANKS
common            include           nagios.spec       update-version
config.guess      indent-all.sh     nagios.sysconfig  UPGRADING
config.sub        indent.sh         pkginfo.in        worker
configure         install-sh        README.md         xdata
configure.ac      INSTALLING        sample-config
contrib           LEGAL             startup
CONTRIBUTING.md   lib               subst.in
root@solaris:~/nagios#
```

Figure 13. Moving nagios-4.5.7 into Nagios directory

We configure Nagios package using **./configure**

```
config.status: creating lib/iobroker.h

Creating sample config files in sample-config/ ...


*** Configuration summary for nagios 4.5.7 2024-10-24 ***:

 General Options:
 -------------------------
        Nagios executable:  nagios
        Nagios user/group:  nagios,nagios
       Command user/group:  nagios,nagios
             Event Broker:  yes
        Install ${prefix}:  /usr/local/nagios
    Install ${includedir}:  /usr/local/nagios/include/nagios
                Lock file:  /var/run/nagios.lock
   Check result directory:  /usr/local/nagios/var/spool/checkresults
           Init directory:  /etc/init.d
  Apache conf.d directory:  /etc/httpd/conf.d
             Mail program:  /usr/bin/mail
                  Host OS:  solaris2.11
          IOBroker Method:  poll

 Web Interface Options:
 ------------------------
                HTML URL:  http://localhost/nagios/
                 CGI URL:  http://localhost/nagios/cgi-bin/
 Traceroute (used by WAP):  /usr/sbin/traceroute


Review the options above for accuracy.  If they look okay,
type 'make all' to compile the main program and CGIs.

root@solaris:~/nagios#
```

Figure 14. Configuring Nagios

Before we can start compiling Nagios, there are a couple of source file changes that need to be made, since this version of Nagios defines a structure (**struct comment**) that conflicts with a system structure of the same name in **/usr/include/sys/pwd.h.**

For that reason, we add the following line as line 28 of the **./worker/ping/worker-ping.c** file:
#include <pwd.h>

```
root@solaris:~/nagios# nano ./worker/ping/worker-ping.c
```

Figure 15. Opening worker-ping.c file

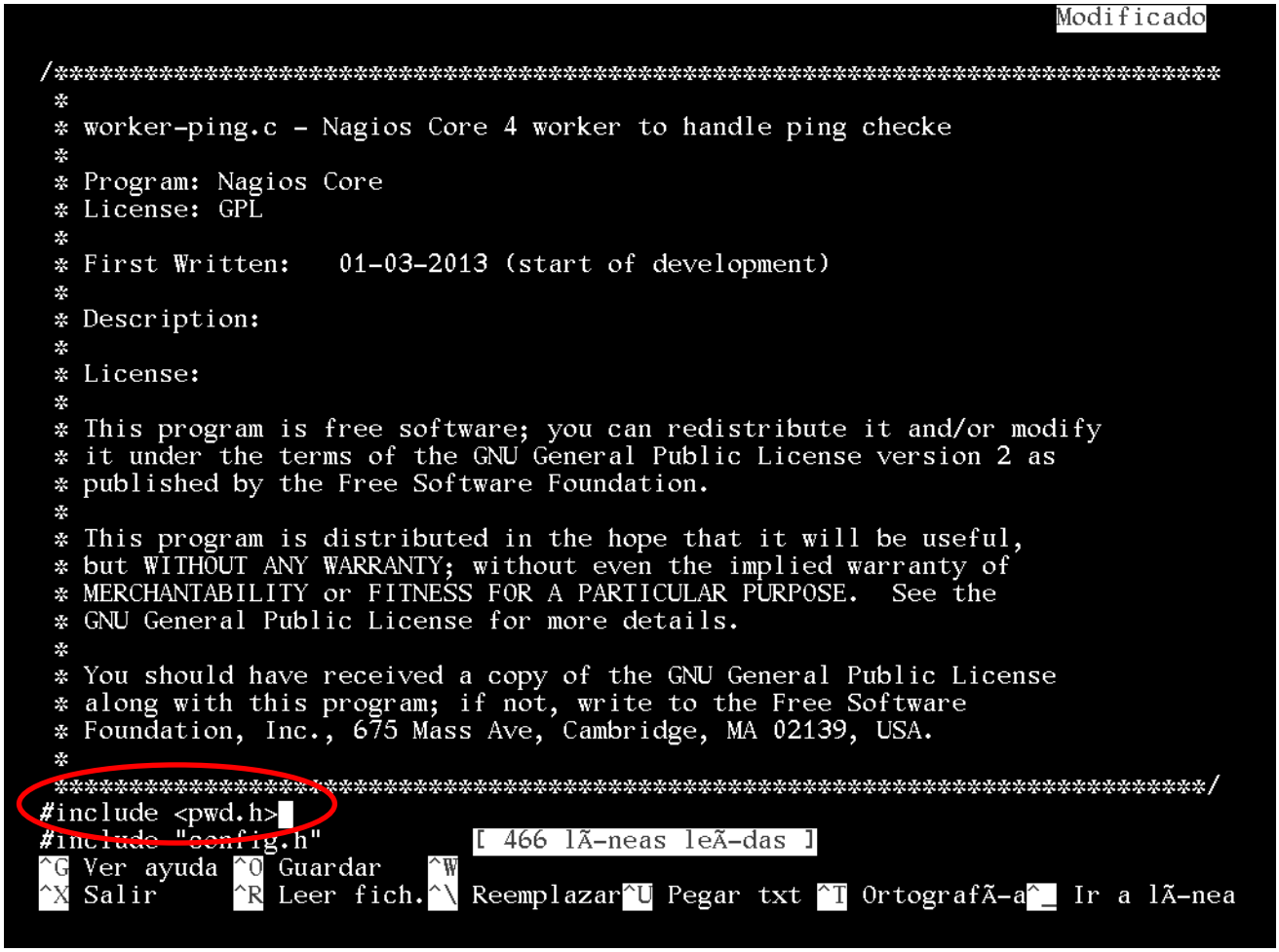


Figure 16. Changing worker-ping.c file

We change the following line (line 27) in the **./base/utils.c** file from this:

#include "../include/comments.h" to this

“#include <pwd.h>”

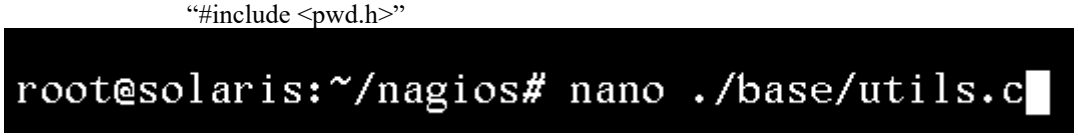


Figure 17. Opening utils.c file

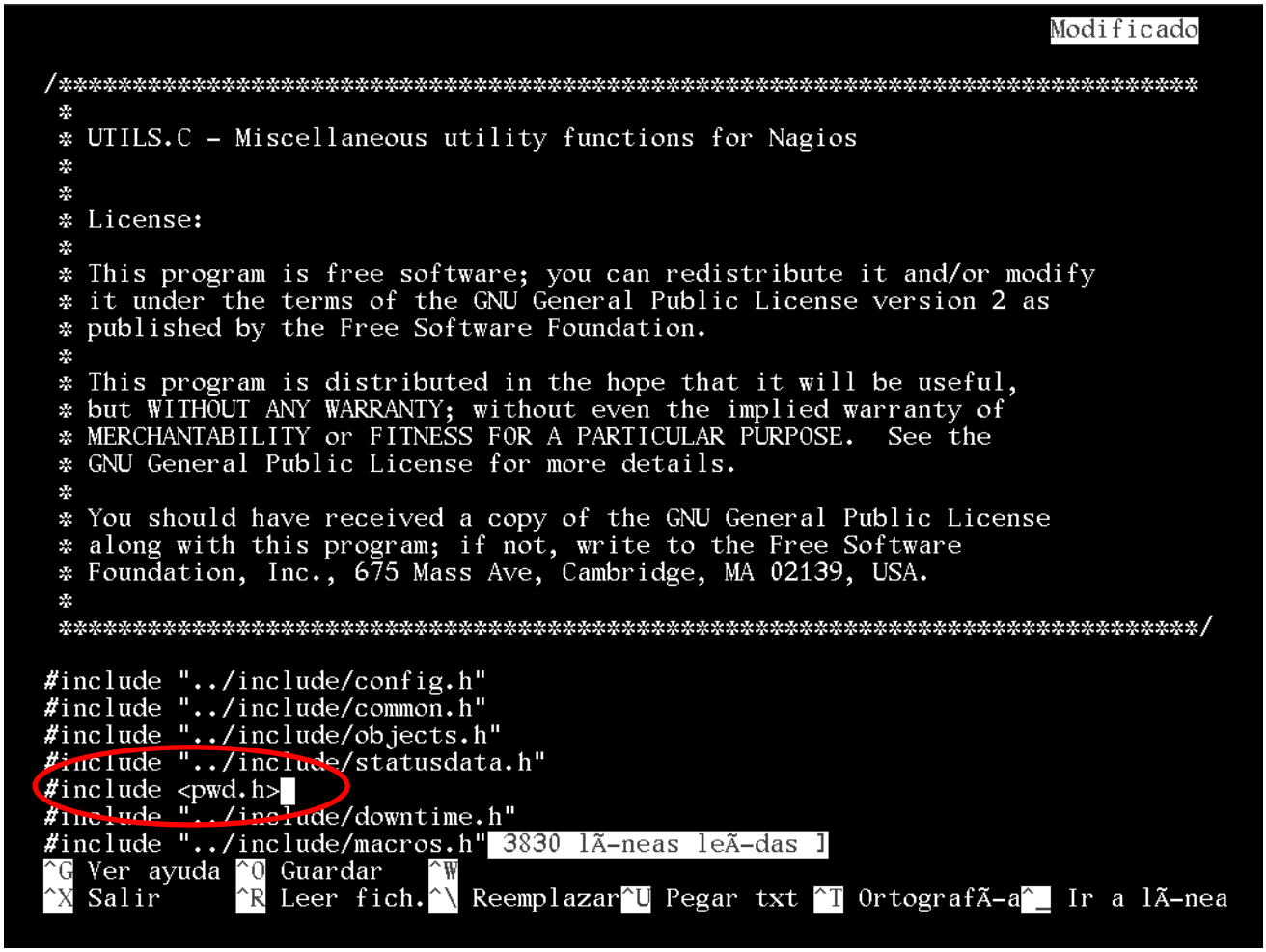


Figure 18. Changing utils.c file

We change the following line (line 29) in the **./cgi/Makefile** file from this:
CFLAGS=-Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCGI
To this:
CFLAGS=-Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCGI -I/usr/include/gd2

Figure 19. Opening /cgi/Makefile file

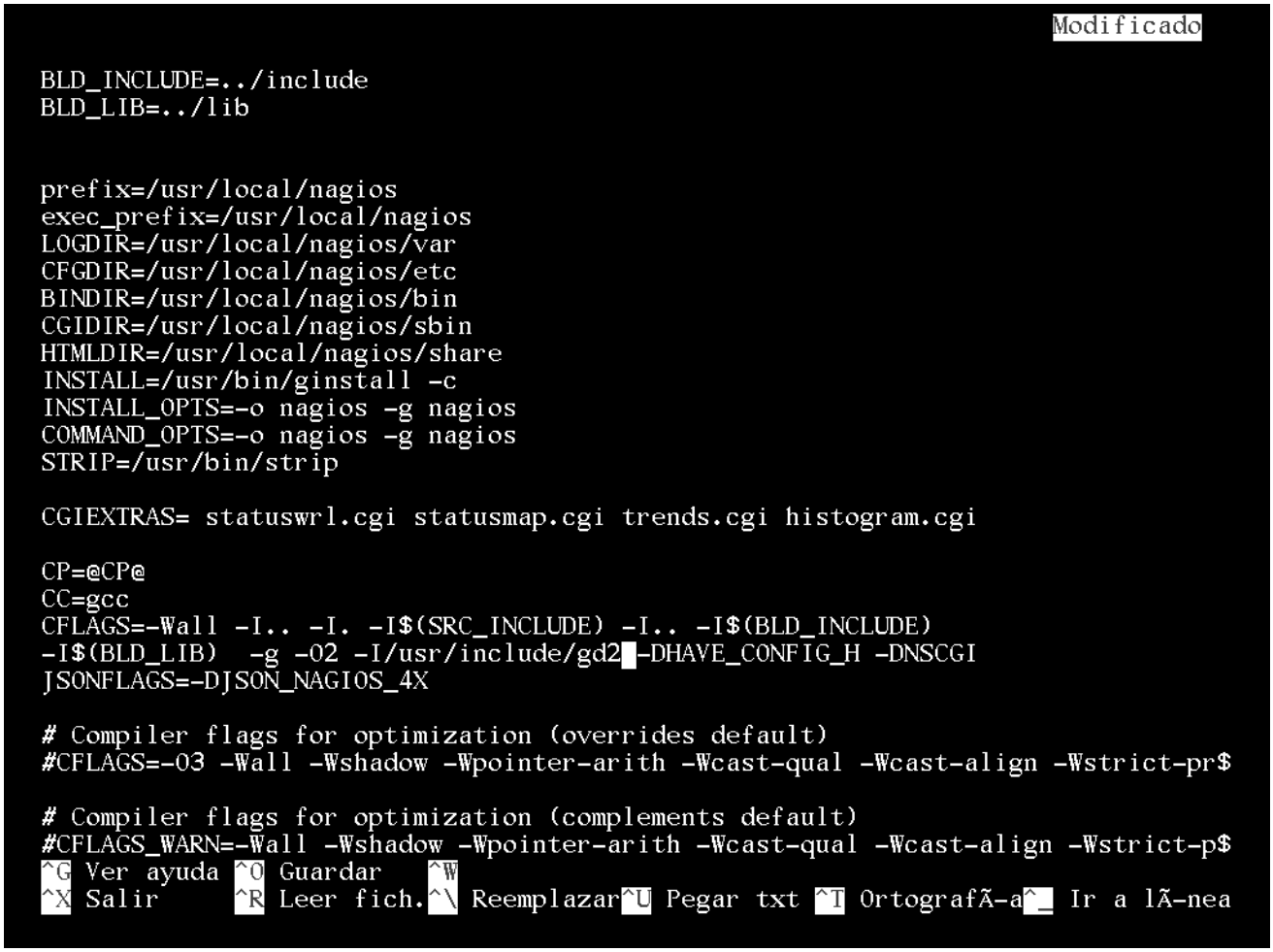


Figure 20. Changing /cgi/Makefile file

Then we run the following command to compile Nagios

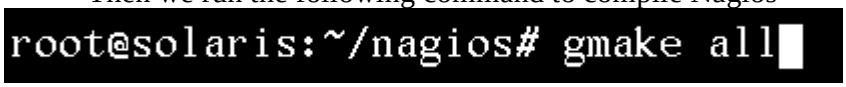


Figure 21. Compile Nagios command

```
        web interface

   make install-classicui
      - This installs the classic theme for the Nagios
        web interface


*** Support Notes *******************************************

If you have questions about configuring or running Nagios,
please make sure that you:

     - Look at the sample config files
     - Read the documentation on the Nagios Library at:
           https://library.nagios.com

before you post a question to one of the mailing lists.
Also make sure to include pertinent information that could
help others help you.  This might include:

     - What version of Nagios you are using
     - What version of the plugins you are using
     - Relevant snippets from your config files
     - Relevant error messages from the Nagios log file

For more information on obtaining support for Nagios, visit:

       https://support.nagios.com

************************************************************

Enjoy.

root@solaris:~/nagios-4.5.7#
```

Figure 22. Compiling Nagios

Once everything has been completed, we use the **gmake install, gmake install-commandmode, and gmake install-config** commands. But first, we need to create the nagios user and nagios group using **groupadd** and **useradd** commands

```
root@solaris:~/nagios# groupadd nagios
root@solaris:~/nagios# useradd -g nagios nagios
root@solaris:~/nagios# gmake install DESTDIR=/root/PROTO
```

Figure 23. Creating Nagios user

```
*** Exfoliation theme installed ***
NOTE: Use 'make install-classicui' to revert to classic Nagios theme

gmake[1]: se sale del directorio '/root/nagios'
gmake install-basic
gmake[1]: se entra en el directorio '/root/nagios'
/usr/bin/ginstall -c -m 775 -o nagios -g nagios -d /root/PROTO/usr/local/nagios/
libexec
/usr/bin/ginstall -c -m 775 -o nagios -g nagios -d /root/PROTO/usr/local/nagios/
var
/usr/bin/ginstall -c -m 775 -o nagios -g nagios -d /root/PROTO/usr/local/nagios/
var/archives
/usr/bin/ginstall -c -m 775 -o nagios -g nagios -d /root/PROTO/usr/local/nagios/
var/spool/checkresults
chmod g+s /root/PROTO/usr/local/nagios/var/spool/checkresults

*** Main program, CGIs and HTML files installed ***

You can continue with installing Nagios as follows (type 'make'
without any arguments for a list of all possible options):

  make install-init
     - This installs the init script in /root/PROTO/etc/init.d

  make install-commandmode
     - This installs and configures permissions on the
       directory for holding the external command file

  make install-config
     - This installs sample config files in /root/PROTO/usr/local/nagios/etc

gmake[1]: se sale del directorio '/root/nagios'
root@solaris:~/nagios#
```

Figure 24. Installing Nagios using 'gmake install' command

```
root@solaris:~/nagios# gmake install-commandmode
/usr/bin/ginstall -c -m 775 -o nagios -g nagios -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***

root@solaris:~/nagios#
```

Figure 25. Finishing the Nagios configuration with the command 'gmake install-commandmode'

```
root@solaris:~/nagios# gmake install-config
/usr/bin/ginstall -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/ginstall -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/ginstall -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr
/local/nagios/etc/nagios.cfg
/usr/bin/ginstall -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/lo
cal/nagios/etc/cgi.cfg
/usr/bin/ginstall -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /u
sr/local/nagios/etc/resource.cfg
/usr/bin/ginstall -c -b -m 664 -o nagios -g nagios sample-config/template-object
/templates.cfg /usr/local/nagios/etc/objects/templates.cfg
/usr/bin/ginstall -c -b -m 664 -o nagios -g nagios sample-config/template-object
/commands.cfg /usr/local/nagios/etc/objects/commands.cfg
/usr/bin/ginstall -c -b -m 664 -o nagios -g nagios sample-config/template-object
/contacts.cfg /usr/local/nagios/etc/objects/contacts.cfg
/usr/bin/ginstall -c -b -m 664 -o nagios -g nagios sample-config/template-object
/timeperiods.cfg /usr/local/nagios/etc/objects/timeperiods.cfg
/usr/bin/ginstall -c -b -m 664 -o nagios -g nagios sample-config/template-object
/localhost.cfg /usr/local/nagios/etc/objects/localhost.cfg
/usr/bin/ginstall -c -b -m 664 -o nagios -g nagios sample-config/template-object
/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/ginstall -c -b -m 664 -o nagios -g nagios sample-config/template-object
/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/ginstall -c -b -m 664 -o nagios -g nagios sample-config/template-object
/switch.cfg /usr/local/nagios/etc/objects/switch.cfg

*** Config files installed ***

Remember, these are *SAMPLE* config files.  You'll need to read
the documentation for more information on how to actually define
services, hosts, etc. to fit your particular needs.

root@solaris:~/nagios#
```

Figure 26. Finishing the Nagios configuration with the command 'gmake install-config'

We go to the created directory and open the **cgi.cfg** file, and in the "**default_user_name**" variable we set it **nagiosadmin** to get the correct authorization for Nagios

```
root@solaris:~/nagios# cd /root/PROTO/usr/local/nagios/etc/
root@solaris:~/PROTO/usr/local/nagios/etc# ls
cgi.cfg       nagios.cfg    objects       resource.cfg
root@solaris:~/PROTO/usr/local/nagios/etc# nano cgi.cfg
```

Figure 27. Opening cgi.cfg file

Figure 28. Configuring cgi.cfg file

Now, we download the Nagios plugins from the official website

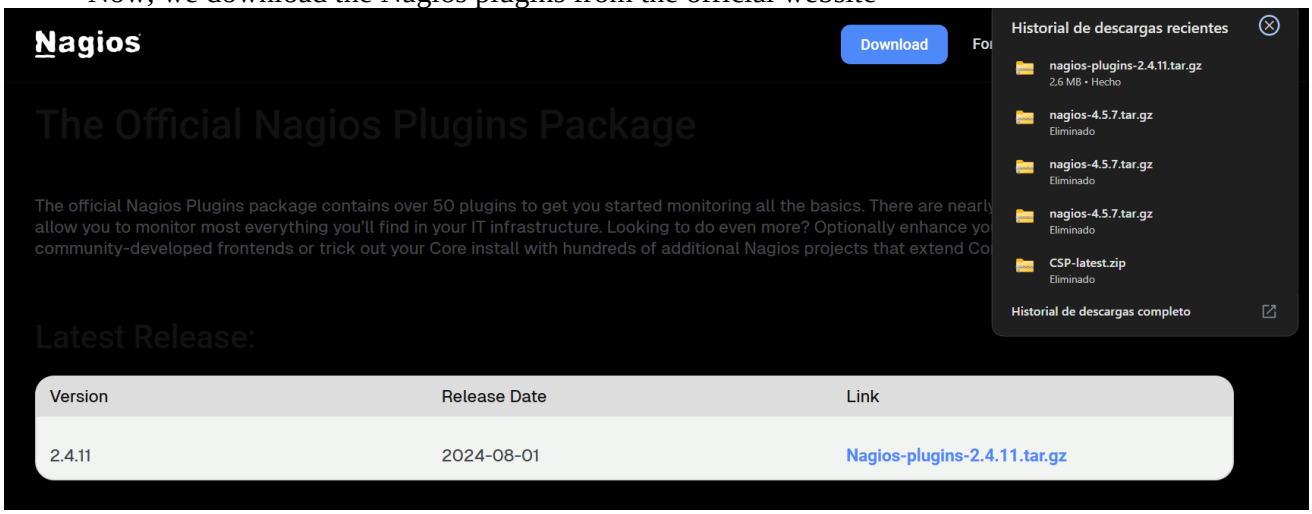


Figure 29. Downloading nagios plugins

We decompress the .tar.gz file, then we use ./configure and make install

```
root@solaris:~# mv /compartir/nagios-plugins-2.4.11.tar.gz /root/
root@solaris:~# ls
nagios                          pax_global_header
nagios-4.5.7.tar.gz             PROTO
nagios-plugins-2.4.11.tar.gz
root@solaris:~# tar -zxf nagios-plugins-2.4.11-tar.gz
```

Figure 30. Decompressing nagios-plugins

```
                  --with-gnutls: no
              --enable-extra-opts: yes
                     --with-perl: /usr/bin/perl
             --enable-perl-modules: no
                  --with-cgiurl: /nagios/cgi-bin
              --with-trusted-path: /usr/local/sbin:/usr/local/bin:/sbin:/bin:/u
sr/sbin:/usr/bin
                 --enable-libtap: no
checking that generated files are newer than configure... done
configure: creating ./config.status
config.status: creating gl/Makefile
config.status: creating nagios-plugins.spec
config.status: creating tools/build_perl_modules
config.status: creating Makefile
config.status: creating tap/Makefile
config.status: creating lib/Makefile
config.status: creating plugins/Makefile
config.status: creating lib/tests/Makefile
config.status: creating plugins-root/Makefile
config.status: creating plugins-scripts/Makefile
config.status: creating plugins-scripts/utils.pm
config.status: creating plugins-scripts/utils.sh
config.status: creating perlmods/Makefile
config.status: creating test.pl
config.status: creating pkg/solaris/pkginfo
config.status: creating po/Makefile.in
config.status: creating config.h
config.status: config.h is unchanged
config.status: executing depfiles commands
config.status: executing libtool commands
config.status: executing po-directories commands
config.status: creating po/POTFILES
config.status: creating po/Makefile
root@solaris:~/nagios-plugins-2.4.11#
```

Figure 31. Configuring nagios plugins

```
root@solaris:~/nagios-plugins-2.4.11# gmake install DESTDIR=/root/PROTO
```

Figure 32. Using 'gmake install' to install nagios plugins

```
 chmod ug=rx,u+s /root/PROTO/usr/local/nagios/libexec/check_icmp
 /usr/bin/ginstall -c  pst3 /root/PROTO/usr/local/nagios/libexec/pst3
 chown root /root/PROTO/usr/local/nagios/libexec/pst3
 chmod ug=rx,u+s /root/PROTO/usr/local/nagios/libexec/pst3
gmake[2]: No se hace nada para 'install-data-am'.
gmake[2]: se sale del directorio '/root/nagios-plugins-2.4.11/plugins-root'
gmake[1]: se sale del directorio '/root/nagios-plugins-2.4.11/plugins-root'
Making install in po
gmake[1]: se entra en el directorio '/root/nagios-plugins-2.4.11/po'
/usr/bin/gmkdir -p /root/PROTO/usr/local/nagios/share
installing fr.gmo as /root/PROTO/usr/local/nagios/share/locale/fr/LC_MESSAGES/na
gios-plugins.mo
installing de.gmo as /root/PROTO/usr/local/nagios/share/locale/de/LC_MESSAGES/na
gios-plugins.mo
if test "nagios-plugins" = "gettext-tools"; then \
  /usr/bin/gmkdir -p /root/PROTO/usr/local/nagios/share/gettext/po; \
  for file in Makefile.in.in remove-potcdate.sin    Makevars.template; do \
    /usr/bin/ginstall -c  -m 644 ./$file \
                  /root/PROTO/usr/local/nagios/share/gettext/po/$file; \
  done; \
  for file in Makevars; do \
    rm -f /root/PROTO/usr/local/nagios/share/gettext/po/$file; \
  done; \
else \
  : ; \
fi
gmake[1]: se sale del directorio '/root/nagios-plugins-2.4.11/po'
gmake[1]: se entra en el directorio '/root/nagios-plugins-2.4.11'
gmake[2]: se entra en el directorio '/root/nagios-plugins-2.4.11'
gmake[2]: No se hace nada para 'install-exec-am'.
gmake[2]: No se hace nada para 'install-data-am'.
gmake[2]: se sale del directorio '/root/nagios-plugins-2.4.11'
gmake[1]: se sale del directorio '/root/nagios-plugins-2.4.11'
root@solaris:~/nagios-plugins-2.4.11#
```

Figure 33. Installing nagios plugins

2.2. Configuring Nagios and SNMP on Solaris

We open the file **/usr/local/nagios/etc/objects/hosts.cfg** to configure the hosts (machines to be monitored) and add the following configuration (we must specify the IP address of the host):

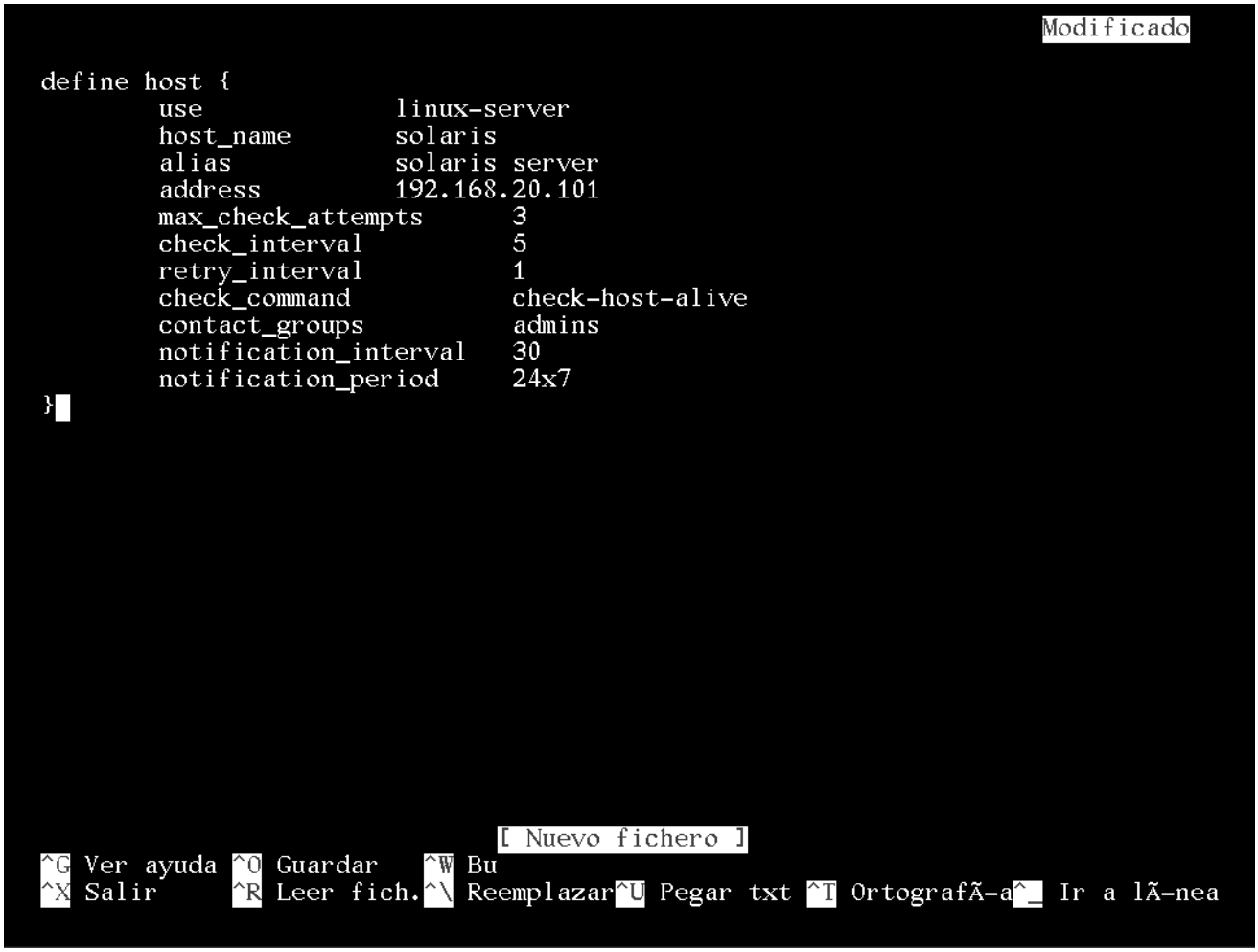


Figure 34. Solaris host monitoring configuration

We open the file **/usr/local/nagios/etc/objects/commands.cfg** to add a command that allows us to retrieve CPU information:

Define command {
Command_name check_cpu_load
Command_line /usr/local/nagios/libexec/check_snmp -H $HOSTADDRESS$ -C public -o 1.3.6.1.4.1.2021.10.1.3.1 -w 1.0 -c 2.0
}

Where .**1.3.6.1.4.1.2021.10.1.3.1** is the OID identifier that allows me to know the CPU status of the host using SNMP

```
define command {

    command_name    process-host-perfdata
    command_line    /usr/bin/printf "%b" "$LASTHOSTCHECK$\t$HOSTNAME$\t$HOSTSTA$
}



define command {

    command_name    process-service-perfdata
    command_line    /usr/bin/printf "%b" "$LASTSERVICECHECK$\t$HOSTNAME$\t$SERV$
}
define command {
    command_name        check_cpu_load
    command_line        /usr/local/nagios/libexec/check_snmp -H $HOSTADDRESS$ -$
}
```

^G Ver ayuda ^O Guardar ^W
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar txt ^T OrtografÃ-a ^_ Ir a lÃ-nea

Figure 35. CPU monitoring command

We open the file **/usr/local/nagios/etc/objects/services.cfg** to configure the CPU service and add the following configuration (we must specify the command and the host that we created before):

```
define service {
        use                     generic-service
        host_name               solaris-server
        service_description     CPU Load
        check_command           check_cpu_load
        max_check_attempts      3
        check_interval          5
        retry_interval          1
}

                                [ 9 lÃ-neas leÃ-das ]
^G Ver ayuda ^O Guardar   ^W Buscar     ^K Cortar txt^J Justificar^C PosiciÃ³n
^X Salir     ^R Leer fich.^\ Reemplazar^U Pegar txt ^T OrtografÃ-a^_ Ir a lÃ-nea
```

Figure 36. Configuration of the CPU service for Solaris

We verify that the configuration has no syntax errors with the following command

```
root@solaris:~# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cf
g
```

Figure 37. Command to verify the Nagios configuration

We can see that there's no errors

```
License: GPL

Website: https://www.nagios.org
Reading configuration data...
   Read main config file okay...
   Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
        Checked 9 services.
        Checked 2 hosts.
        Checked 1 host groups.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
        Checked 24 commands.
        Checked 5 time periods.
        Checked 0 host escalations.
        Checked 0 service escalations.
Checking for circular paths...
        Checked 2 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
root@solaris:~#
```

Figure 38. Verifying Nagios configuration

We set a password for the user nagiosadmin using the command “htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin”

```
root@solaris:~# htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
root@solaris:~#
```

Figure 39. Creating a password for the user nagiosadmin

Now, we need to configure SNMP, we open **/etc/net-snmp/snmp/snmpd.conf**

```
root@solaris:~# nano /etc/net-snmp/snmp/snmpd.conf
```

Figure 40. Opening snmpd.conf

We add the following configuration to the file:

Rocommunity public
Syslocation “Solaris Machine”
Syscontact “[email to contact]”

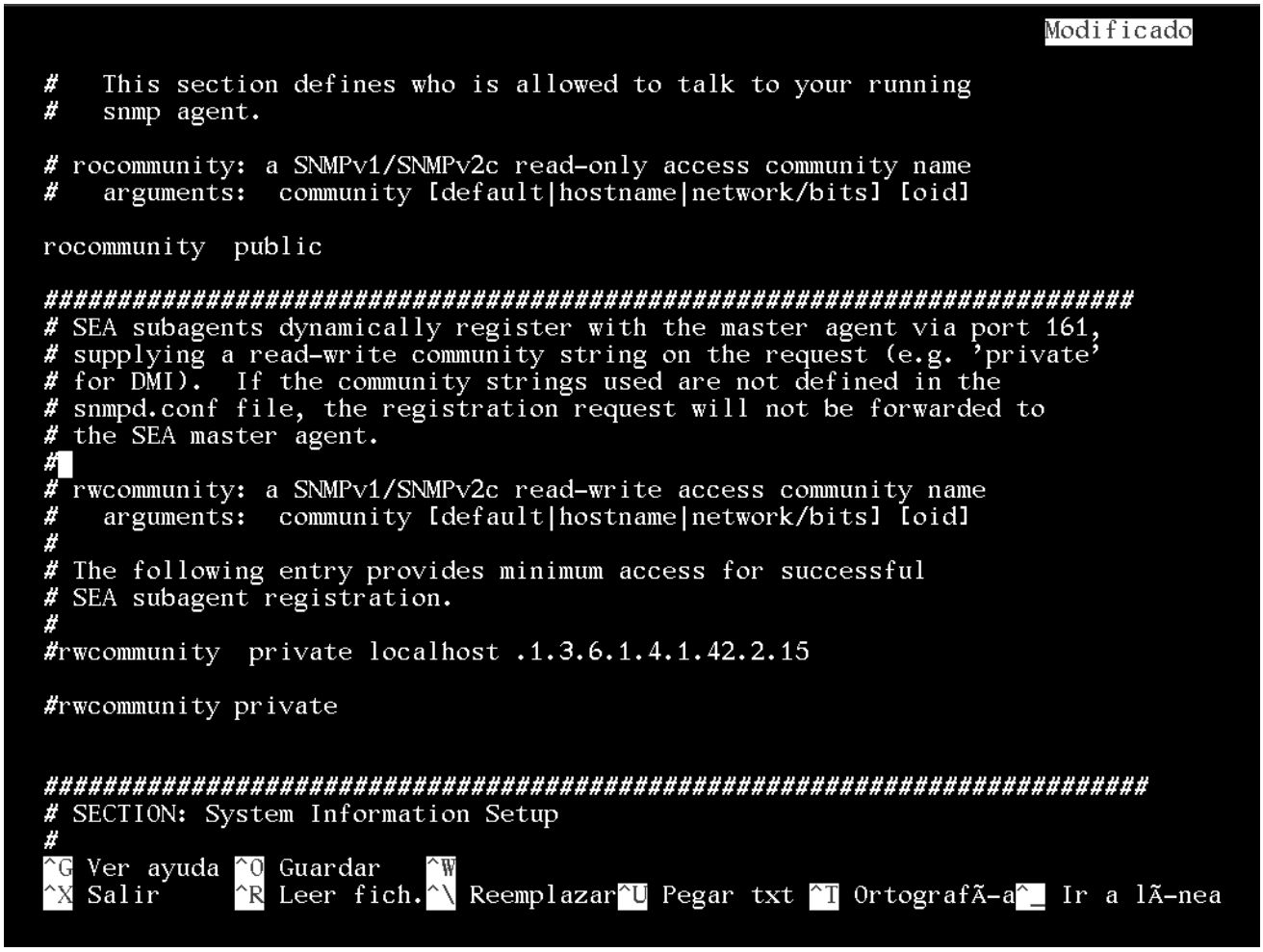


Figure 41. Configuring SNMP

Additionally, we add the following configuration to monitor the disk, eth0 and lo0 interfaces, and allow the connection with Nagios

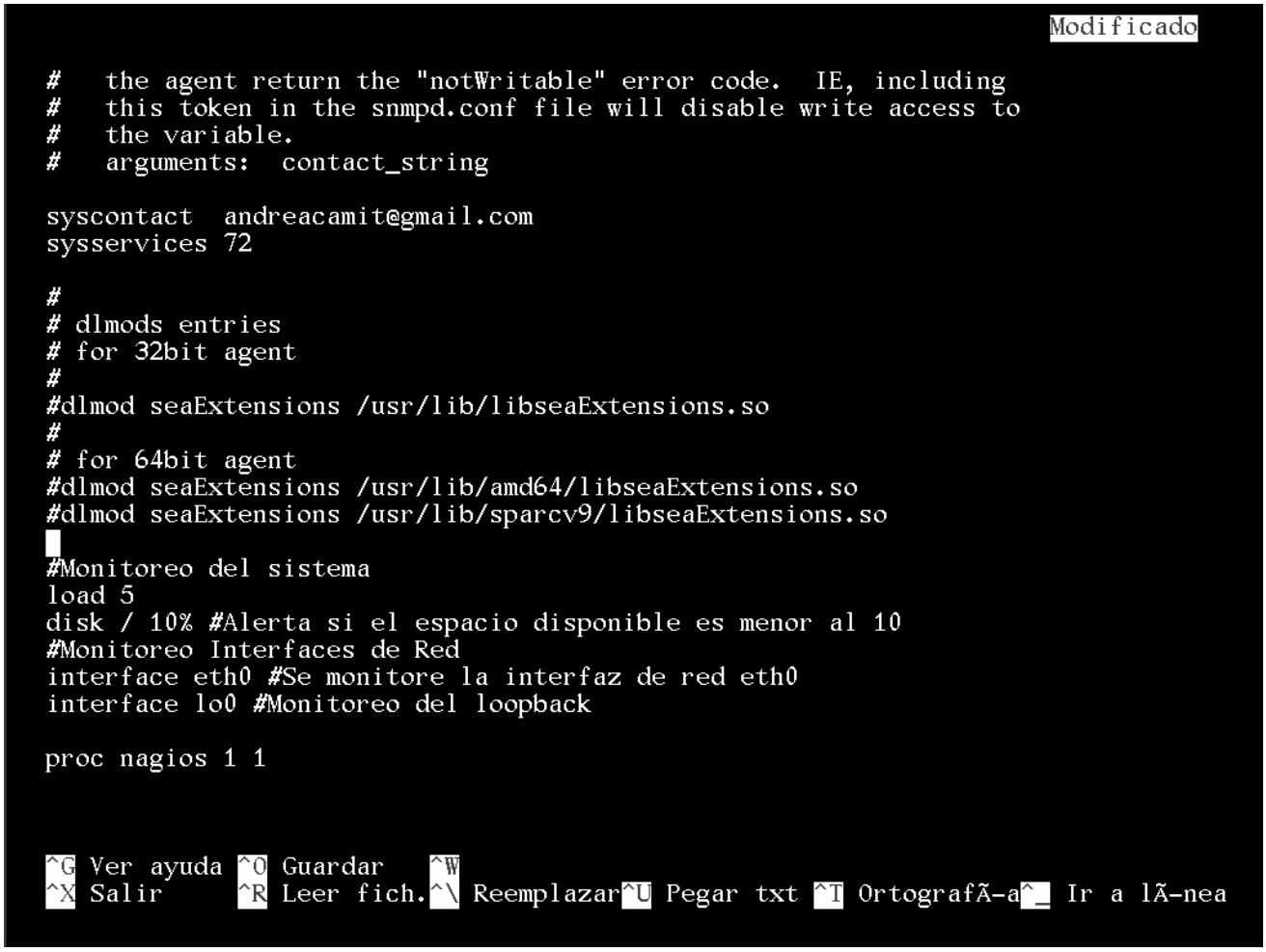


Figure 42. Adding additional information on snmpd.conf

We test the functionality with **snmpwalk -v 2c -c public [ip]**

```
-EVENT-MIB::mteTriggerFired
DISMAN-EVENT-MIB::mteEventNotification."_snmpd".'_mteTriggerRising' = OID: DISMA
N-EVENT-MIB::mteTriggerRising
DISMAN-EVENT-MIB::mteEventNotificationObjectsOwner."_snmpd".'_linkDown' = STRING
: _snmpd
DISMAN-EVENT-MIB::mteEventNotificationObjectsOwner."_snmpd".'_linkUp' = STRING:
_snmpd
DISMAN-EVENT-MIB::mteEventNotificationObjectsOwner."_snmpd".'_mteTriggerFailure'
 = STRING: _snmpd
DISMAN-EVENT-MIB::mteEventNotificationObjectsOwner."_snmpd".'_mteTriggerFalling'
 = STRING: _snmpd
DISMAN-EVENT-MIB::mteEventNotificationObjectsOwner."_snmpd".'_mteTriggerFired' =
 STRING: _snmpd
DISMAN-EVENT-MIB::mteEventNotificationObjectsOwner."_snmpd".'_mteTriggerRising'
= STRING: _snmpd
DISMAN-EVENT-MIB::mteEventNotificationObjects."_snmpd".'_linkDown' = STRING: _li
nkUpDown
DISMAN-EVENT-MIB::mteEventNotificationObjects."_snmpd".'_linkUp' = STRING: _link
UpDown
DISMAN-EVENT-MIB::mteEventNotificationObjects."_snmpd".'_mteTriggerFailure' = ST
RING: _triggerFail
DISMAN-EVENT-MIB::mteEventNotificationObjects."_snmpd".'_mteTriggerFalling' = ST
RING: _triggerFire
DISMAN-EVENT-MIB::mteEventNotificationObjects."_snmpd".'_mteTriggerFired' = STRI
NG: _triggerFire
DISMAN-EVENT-MIB::mteEventNotificationObjects."_snmpd".'_mteTriggerRising' = STR
ING: _triggerFire
NOTIFICATION-LOG-MIB::nlmConfigGlobalEntryLimit.0 = Gauge32: 1000
NOTIFICATION-LOG-MIB::nlmConfigGlobalAgeOut.0 = Gauge32: 1440 minutes
NOTIFICATION-LOG-MIB::nlmStatsGlobalNotificationsLogged.0 = Counter32: 0 notific
ations
NOTIFICATION-LOG-MIB::nlmStatsGlobalNotificationsBumped.0 = Counter32: 0 notific
ations
root@solaris:~#
```

Figure 43. Testing SNMP service on Solaris

We start SNMP service

```
root@solaris:~# svcadm enable svc:/application/management/net-snmp:default
root@solaris:~# svcadm restart svc:/application/management/net-snmp:default
root@solaris:~#
```

Figure 44. Starting SNMP service on Solaris

Now we need to configure Apache to access Nagios services from the web browser. To do this, we open the file **/etc/apache2/2.4/httpd.conf** and add the configuration shown in *image 45*

```
<IfModule alias_module>
        ScriptAlias /nagios/cgi-bin "/usr/local/nagios/sbin/"
        Alias /nagios /usr/local/nagios/share
</IfModule>
<Directory "/usr/local/nagios/sbin/">
        Options +ExecCGI
        AddHandler cgi-script.cgi
        AllowOverride None
        Require all granted
        AuthType Basic
        AuthName "Nagios Access"
        AuthUserFile /usr/local/nagios/etc/htpasswd.users
        Require valid-user
</Directory>
<Directory "/usr/local/nagios/share">
         Options None
         AllowOverride None
         Require all granted
         DirectoryIndex index.php index.html
         AuthType Basic
         AuthName "Nagios Access"
         AuthUserFile /usr/local/nagios/etc/htpasswd.users
         Require valid-user
</Directory>
```

Figure 45. Apache file configuration

We restart the Apache service with the command **svcadm restart apache24**, then we restart the SNMP service with the command **svcadm restart svc:/application/management/net-snmp:default**. Finally, we start Nagios with the command **/usr/local/nagios/bin/nagios /usr/local/nagios/etc/nagios.cfg**

```
root@solaris:~# svcadm restart apache24
root@solaris:~# svcadm -v enable /network/http:apache24
svc:/network/http:apache24 activado.
root@solaris:~# svcadm restart svc:/application/management/net-snmp:default
root@solaris:~# svcadm enable svc:/application/management/net-snmp:default
root@solaris:~# /usr/local/nagios/bin/nagios /usr/local/nagios/etc/nagios.cfg

Nagios Core 4.5.7
Copyright (c) 2009-present Nagios Core Development Team and Community Contributo
rs
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2024-10-24
License: GPL

Website: https://www.nagios.org
Nagios 4.5.7 starting... (PID=1281)
Local time is Thu Nov 14 20:28:36 -05 2024
wproc: Successfully registered manager as @wproc with query handler
wproc: Registry request: name=Core Worker 1283;pid=1283
wproc: Registry request: name=Core Worker 1284;pid=1284
wproc: Registry request: name=Core Worker 1282;pid=1282
wproc: Registry request: name=Core Worker 1285;pid=1285
```

Figure 46. Initializing Nagios, SNMP and Apache

We open the browser and enter the Solaris IP along with the Nagios address **(<solaris_ip>/nagios)**. We type in the previously configured user along with the assigned password

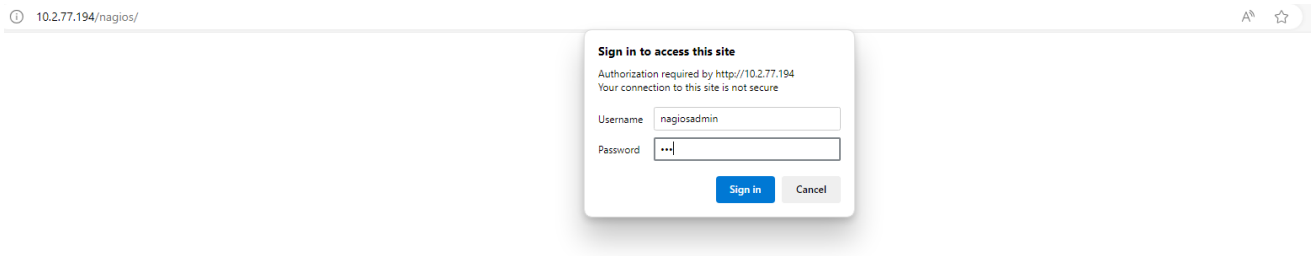


Figure 47. Accessing the Nagios web interface

Once authenticated, we access the Nagios web interface with the services it offers

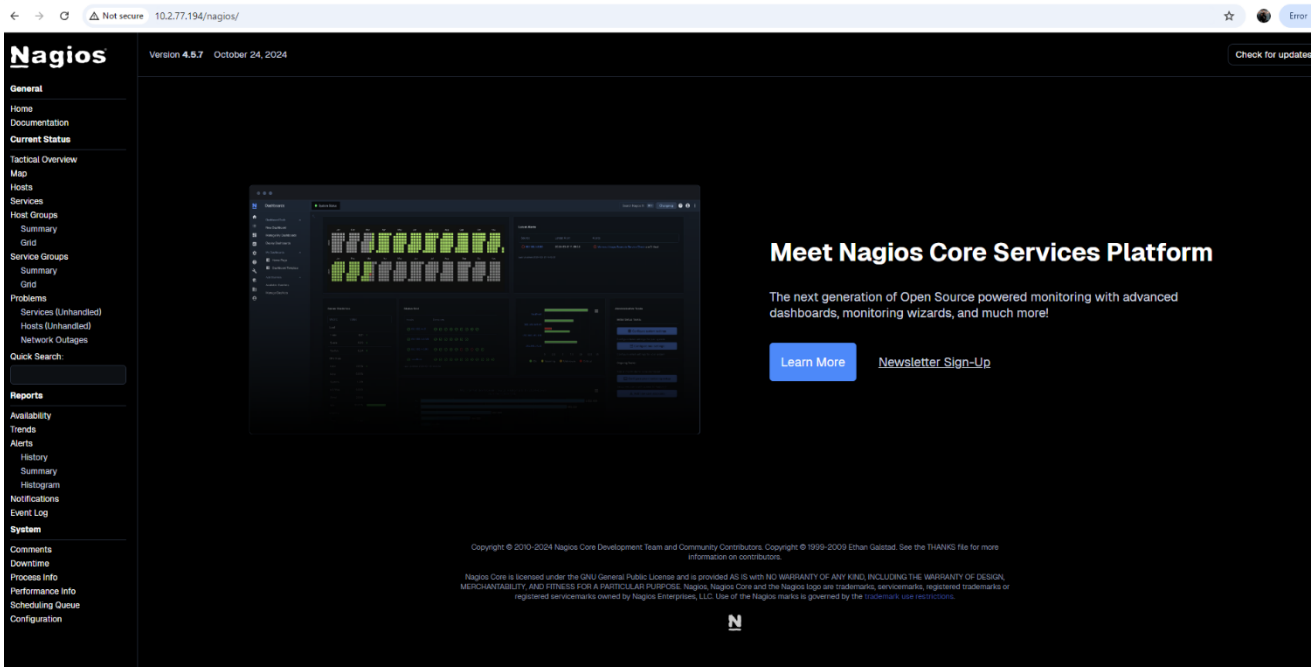


Figure 48. Nagios web interface

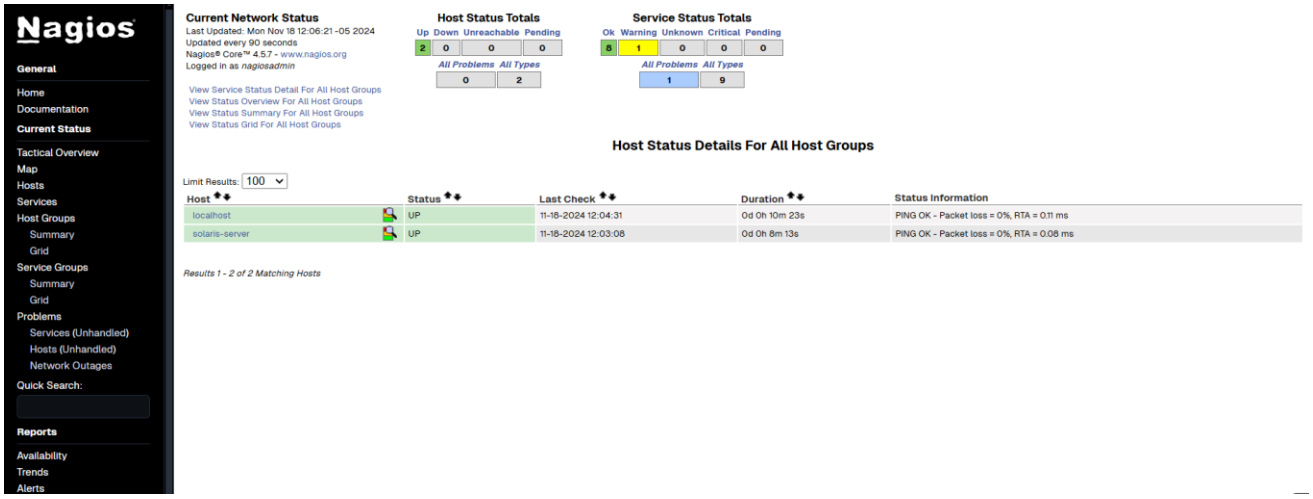


Figure 49. Page where all the hosts are displayed

Figure 50. Page where all the services are displayed

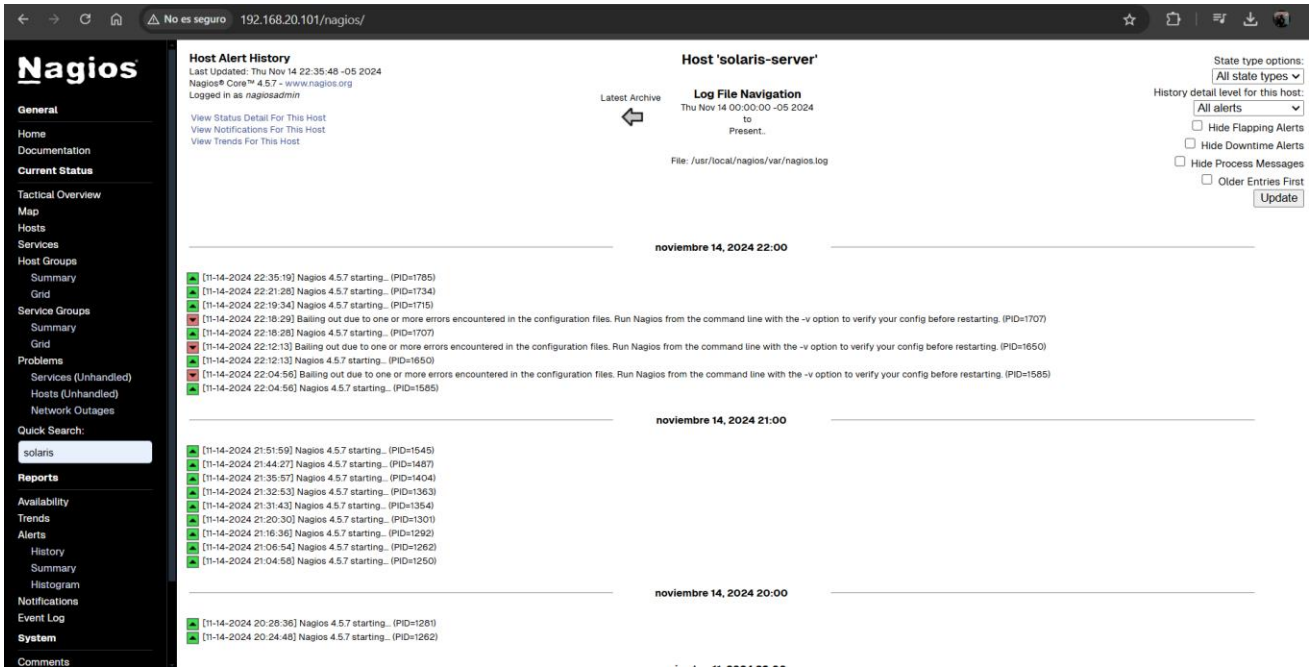


Figure 51. Solaris Host log information into Nagios

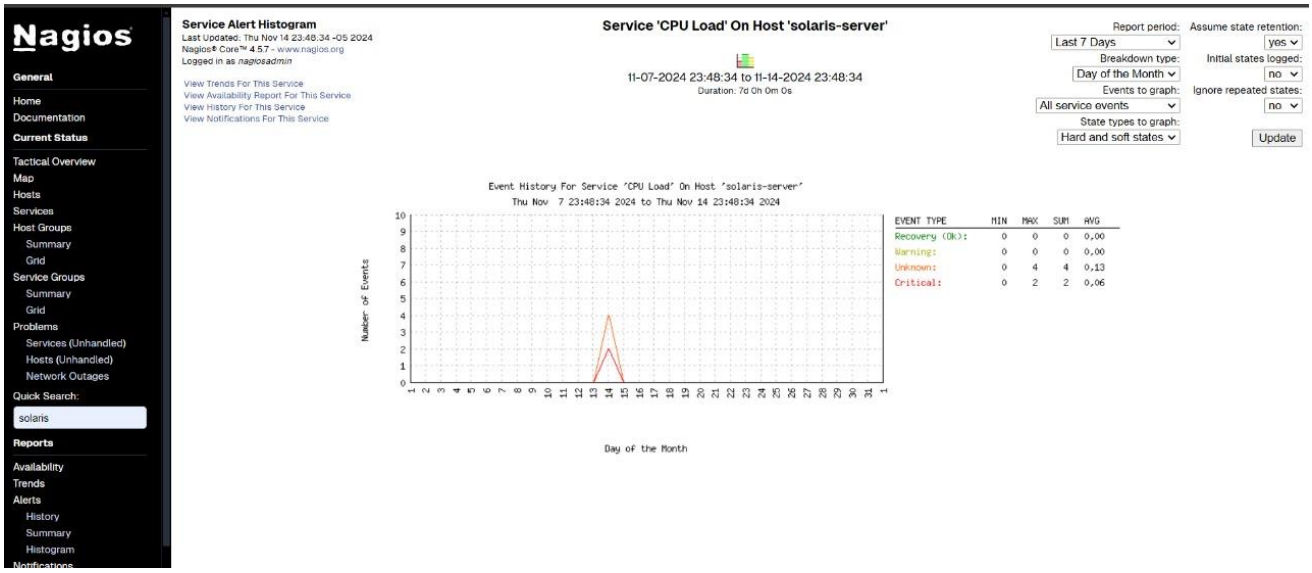


Figure 52. CPU Service of Solaris Graphic

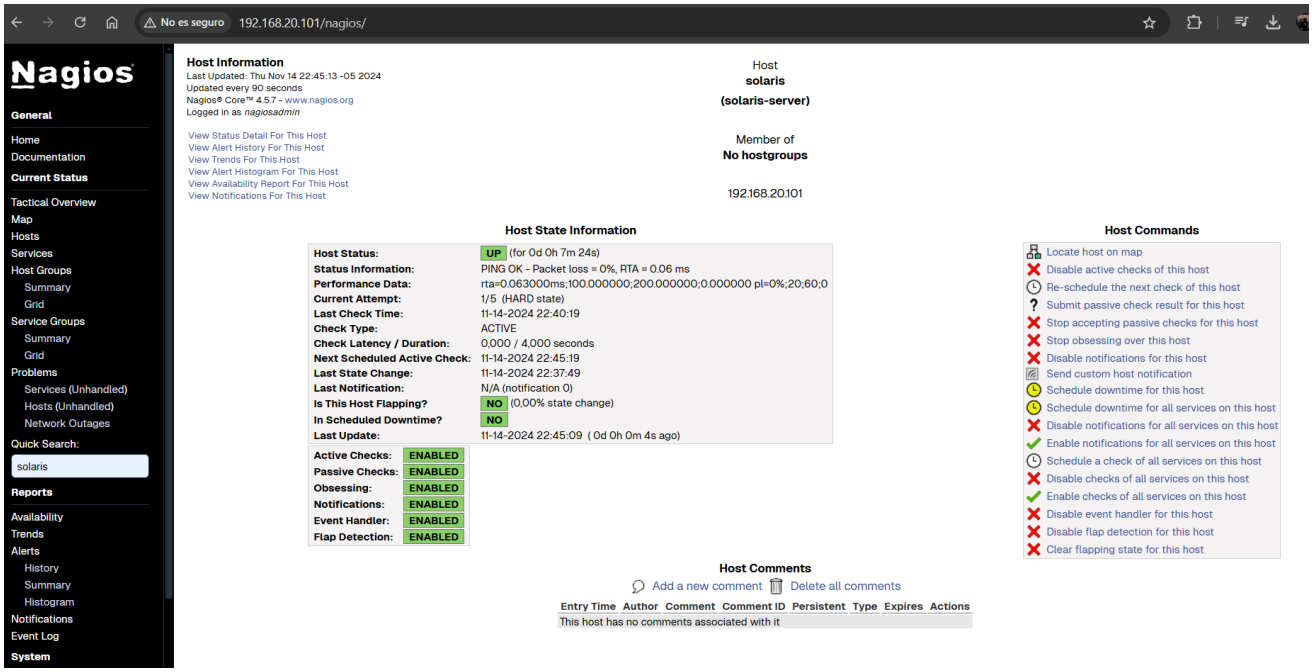


Figure 53. Solaris Host information into Nagios

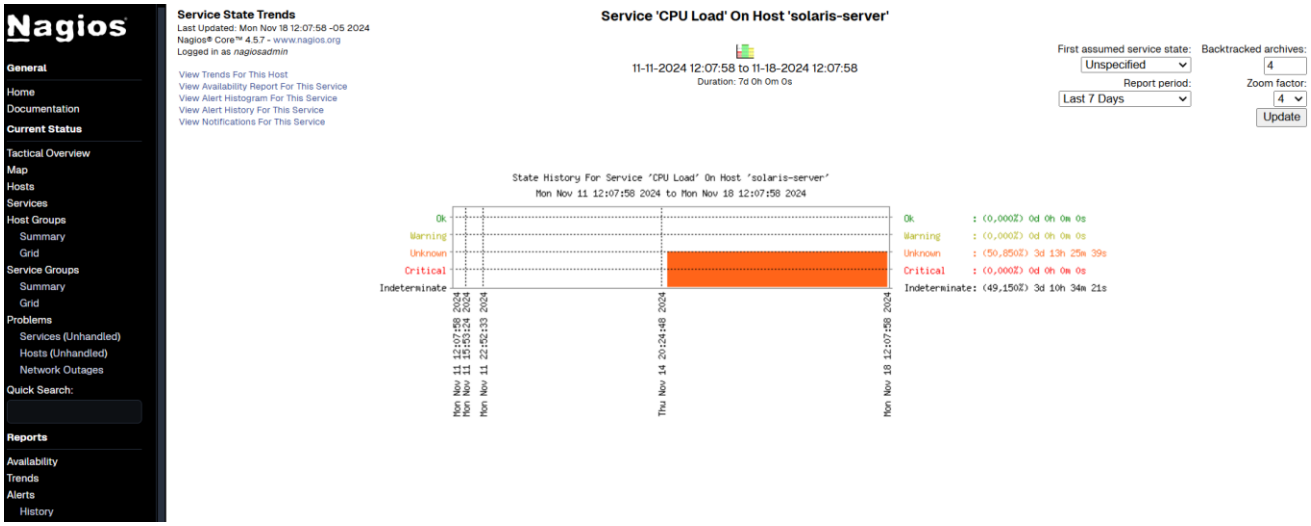


Figure 54. Information about the CPU service in Solaris

2.3. Monitoring Linux Slackware with Nagios

We install SNMP using **slackpkg** command

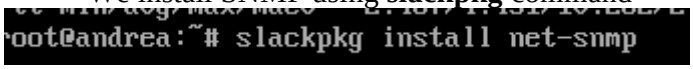


Figure 55. Command to install SNMP

We select the package and wait for it to install

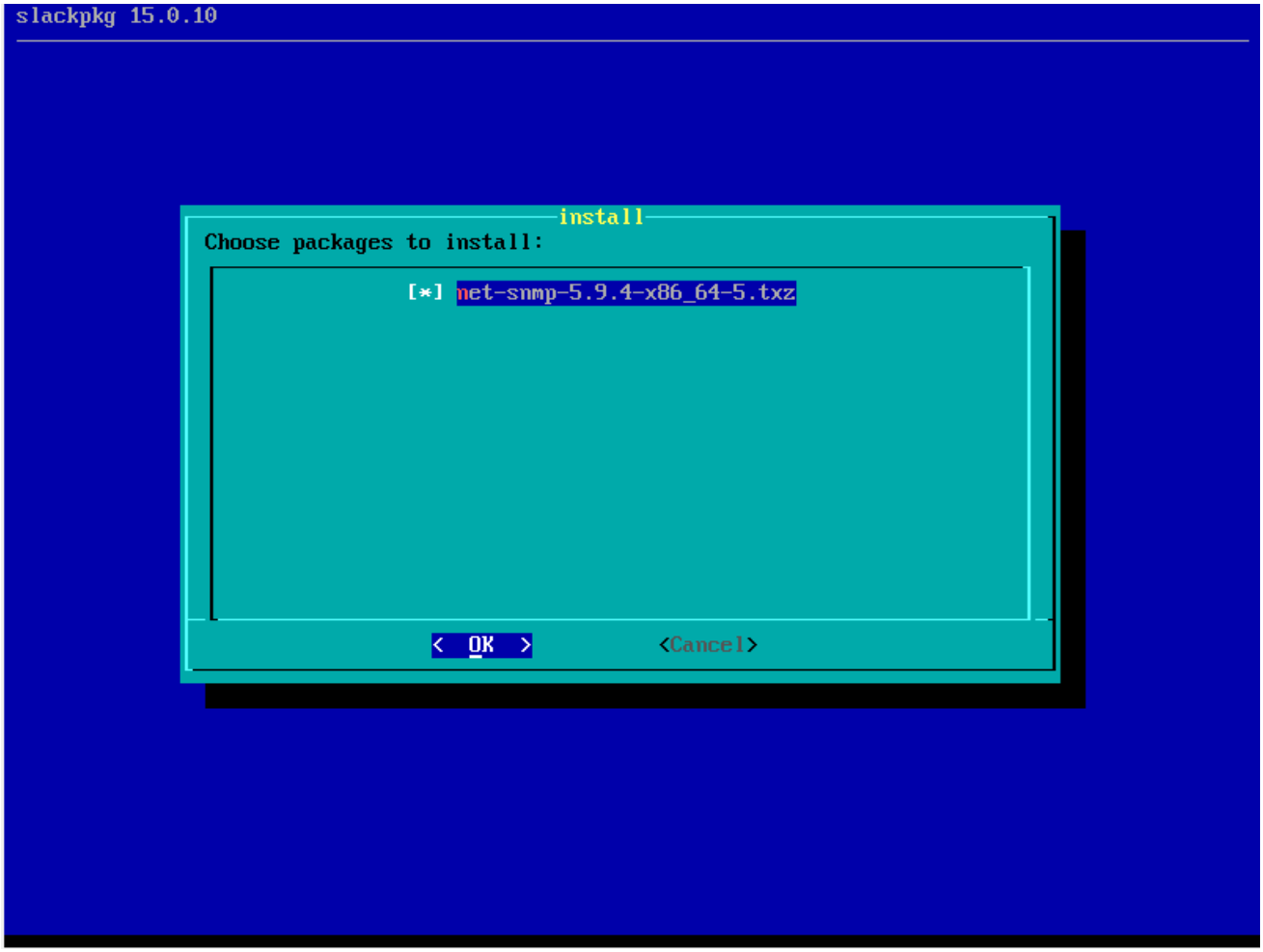


Figure 56. Selecting SNMP package

```
re64/n/net-snmp-5.9.4-x86_64-5.txz.asc...
--2024-11-08 13:39:57--  http://ftp.slackware-brasil.com.br/slackware64-current/slackware64/n/net-sn
mp-5.9.4-x86_64-5.txz.asc
Resolving ftp.slackware-brasil.com.br (ftp.slackware-brasil.com.br)... 200.137.217.134
Connecting to ftp.slackware-brasil.com.br (ftp.slackware-brasil.com.br)|200.137.217.134|:80... conne
cted.
HTTP request sent, awaiting response... 200 OK
Length: 195 [text/plain]
Saving to: ■//var/cache/packages/./slackware64/n/net-snmp-5.9.4-x86_64-5.txz.asc■

//var/cache/packages/./s 100%[===================================>]     195  --.-KB/s    in 0s

2024-11-08 13:39:57 (16.8 MB/s) - ■//var/cache/packages/./slackware64/n/net-snmp-5.9.4-x86_64-5.txz.
asc■ saved [195/195]

        Package net-snmp-5.9.4-x86_64-5.txz is already in cache - not downloading
        Installing net-snmp-5.9.4-x86_64-5...
Verifying package net-snmp-5.9.4-x86_64-5.txz.
Installing package net-snmp-5.9.4-x86_64-5.txz:
PACKAGE DESCRIPTION:
# net-snmp (Simple Network Management Protocol tools)
#
# Various tools relating to the Simple Network Management Protocol:
#
# An extensible agent
# An SNMP library
# Tools to request or set information from SNMP agents
# Tools to generate and handle SNMP traps
# A version of the UNIX 'netstat' command using SNMP
# A graphical Perl/Tk/SNMP based mib browser
#
Executing install script for net-snmp-5.9.4-x86_64-5.txz.
Package net-snmp-5.9.4-x86_64-5.txz installed.
Searching for NEW configuration files...
               No .new files found.

root@andrea:~# _
```

Figure 57. SNMP installation

We open /etc/snmp/snmpd.conf file

```
root@andrea:~# nano /etc/snmp/snmpd.conf
```

Figure 58. Opening snmpd.conf file

At the end of the file, we add the configuration to monitor the devices and, additionally, the port on which it listens

```
#           script in the right location. (its not installed by default))

# % snmpwalk -v 1 localhost -c public .1.3.6.1.4.1.2021.255
# enterprises.ucdavis.255.1 = "life the universe and everything"
# enterprises.ucdavis.255.2.1 = 42
# enterprises.ucdavis.255.2.2 = OID: 42.42.42
# enterprises.ucdavis.255.3 = Timeticks: (363136200) 42 days, 0:42:42
# enterprises.ucdavis.255.4 = IpAddress: 127.0.0.1
# enterprises.ucdavis.255.5 = 42
# enterprises.ucdavis.255.6 = Gauge: 42
#
# % snmpget -v 1 localhost public .1.3.6.1.4.1.2021.255.5
# enterprises.ucdavis.255.5 = 42
#
# % snmpset -v 1 localhost public .1.3.6.1.4.1.2021.255.1 s "New string"
# enterprises.ucdavis.255.1 = "New string"
#

# For specific usage information, see the man/snmpd.conf.5 manual page
# as well as the local/passtest script used in the above example.

# Added for support of bcm5820 cards.
pass .1.3.6.1.4.1.4413.4.1 /usr/bin/ucd5820stat

##############################################################################
# Further Information
#
#  See the snmpd.conf manual page, and the output of "snmpd -H".
rocommunity public
rocommunity public 192.168.20.101
rocommunity public 192.168.20.160
rocommunity public 192.168.20.100
agentAdress udp:161



root@andrea:~# _
```

Figure 59. Configuring SNMP

We start the service, but as we can see, we are missing some libraries

- ✓ Libnsl
- ✓ Libnl.3.0.200
- ✓ Libsensors.so.5
- ✓ pciutils

```
root@andrea:~# /etc/rc.d/rc.snmpd start
Starting snmpd: /usr/sbin/snmpd: error while loading shared libraries: libnsl.so.3: cannot open shar
ed object file: No such file or directory
 /usr/sbin/snmpd -A -p /var/run/snmpd -a -c /etc/snmp/snmpd.conf
root@andrea:~# _
```

Figure 60. SNMP starting failed

```
-2.0.1-x86_64-1.txz.asc
Resolving ftp.slackware-brasil.com.br (ftp.slackware-brasil.com.br)... 200.137.217.134
Connecting to ftp.slackware-brasil.com.br (ftp.slackware-brasil.com.br)|200.137.217.134|:80... conne
cted.
HTTP request sent, awaiting response... 200 OK
Length: 163 [text/plain]
Saving to: ■//var/cache/packages/./slackware64/l/libnsl-2.0.1-x86_64-1.txz.asc■

//var/cache/packages/./s 100%[===================================>]     163  --.-KB/s    in 0s

2024-11-12 23:16:46 (23.9 MB/s) - ■//var/cache/packages/./slackware64/l/libnsl-2.0.1-x86_64-1.txz.as
c■ saved [163/163]

        Package libnsl-2.0.1-x86_64-1.txz is already in cache - not downloading
        Installing libnsl-2.0.1-x86_64-1...
Verifying package libnsl-2.0.1-x86_64-1.txz.
Installing package libnsl-2.0.1-x86_64-1.txz:
PACKAGE DESCRIPTION:
# libnsl (NIS/YP library)
#
# This package contains the libnsl library. This library contains the
# public client interface for NIS(YP). This code was formerly part of
# glibc, but is now standalone to be able to link against TI-RPC for
# IPv6 support.
#
# Homepage: https://github.com/thkukuk/libnsl
#
Executing install script for libnsl-2.0.1-x86_64-1.txz.
Package libnsl-2.0.1-x86_64-1.txz installed.
Searching for NEW configuration files...
               No .new files found.
```

Figure 61. Installing libnsl

```
root@andrea:~# installpkg libnl3-3.5.0-x86_64-3.txz
Verifying package libnl3-3.5.0-x86_64-3.txz.
Installing package libnl3-3.5.0-x86_64-3.txz:
PACKAGE DESCRIPTION:
# libnl3 (Netlink Protocol Library Suite version 3)
#
# The libnl suite is a collection of libraries providing APIs to
# netlink protocol based Linux kernel interfaces. Netlink is a IPC
# mechanism primarily between the kernel and user space processes.
# It was designed to be a more flexible successor to ioctl to provide
# mainly networking related kernel configuration and monitoring
# interfaces.
#
# Homepage: https://github.com/thom311/libnl
#
Executing install script for libnl3-3.5.0-x86_64-3.txz.
Package libnl3-3.5.0-x86_64-3.txz installed.
root@andrea:~#
```

Figure 62. Installing libnl

```
root@andrea:~# installpkg lib64lm_sensors5-3.6.0-2-omv4090.x86_64.tgz
Verifying package lib64lm_sensors5-3.6.0-2-omv4090.x86_64.tgz.
Installing package lib64lm_sensors5-3.6.0-2-omv4090.x86_64.tgz:
PACKAGE DESCRIPTION:
Package lib64lm_sensors5-3.6.0-2-omv4090.x86_64.tgz installed.
root@andrea:~#
```

Figure 63. Installing libsensors

```
.asc■ saved [195/195]

        Package pciutils-3.13.0-x86_64-1.txz is already in cache - not downloading
        Installing pciutils-3.13.0-x86_64-1...
Verifying package pciutils-3.13.0-x86_64-1.txz.
Installing package pciutils-3.13.0-x86_64-1.txz:
PACKAGE DESCRIPTION:
# pciutils (PCI utilities)
#
# lspci displays detailed information about all PCI buses and devices
# in the system, replacing the original /proc/pci interface.
#
# setpci allows reading from and writing to PCI device configuration
# registers. For example, you can adjust the latency timers with it.
#
# See the manual pages for more details.
#
Executing install script for pciutils-3.13.0-x86_64-1.txz.
Package pciutils-3.13.0-x86_64-1.txz installed.
Searching for NEW configuration files...
Some packages had new configuration files installed (1 new files):

/etc/snmp/snmpd.conf.new

What do you want (K/O/R/P)?

        (K)eep the old files and consider .new files later

        (O)verwrite all old files with the new ones. The
           old files will be stored with the suffix .orig

        (R)emove all .new files

        (P)rompt K, O, R selection for every single file
O

root@andrea:~# _
```

Figure 64. Installing pciutils

Now we can start the SNMP service

```
root@andrea:~# /etc/rc.d/rc.snmpd restart
Shutting down snmpd:  DONE
Starting snmpd:  /usr/sbin/snmpd -A -p /var/run/snmpd -a -c /etc/snmp/snmpd.conf
root@andrea:~# /usr/sbin/snmpd -A -p /var/run/snmpd -a -c /etc/snmp/snmpd.conf
root@andrea:~# _
```

Figure 65. Starting SNMP on Slackware

We test the functionality with **snmpwalk -v 2c -c public [ip]**

```
SNMPv2-MIB::sysLocation.0 = STRING: Unknown (edit /etc/snmp/snmpd.conf)
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORID.1 = OID: SNMP-FRAMEWORK-MIB::snmpFrameworkMIBCompliance
SNMPv2-MIB::sysORID.2 = OID: SNMP-MPD-MIB::snmpMPDCompliance
SNMPv2-MIB::sysORID.3 = OID: SNMP-USER-BASED-SM-MIB::usmMIBCompliance
SNMPv2-MIB::sysORID.4 = OID: SNMPv2-MIB::snmpMIB
SNMPv2-MIB::sysORID.5 = OID: SNMP-VIEW-BASED-ACM-MIB::vacmBasicGroup
SNMPv2-MIB::sysORID.6 = OID: TCP-MIB::tcpMIB
SNMPv2-MIB::sysORID.7 = OID: UDP-MIB::udpMIB
SNMPv2-MIB::sysORID.8 = OID: IP-MIB::ip
SNMPv2-MIB::sysORID.9 = OID: SNMP-NOTIFICATION-MIB::snmpNotifyFullCompliance
SNMPv2-MIB::sysORID.10 = OID: NOTIFICATION-LOG-MIB::notificationLogMIB
SNMPv2-MIB::sysORDescr.1 = STRING: The SNMP Management Architecture MIB.
SNMPv2-MIB::sysORDescr.2 = STRING: The MIB for Message Processing and Dispatching.
SNMPv2-MIB::sysORDescr.3 = STRING: The management information definitions for the SNMP User-based Se
curity Model.
SNMPv2-MIB::sysORDescr.4 = STRING: The MIB module for SNMPv2 entities
SNMPv2-MIB::sysORDescr.5 = STRING: View-based Access Control Model for SNMP.
SNMPv2-MIB::sysORDescr.6 = STRING: The MIB module for managing TCP implementations
SNMPv2-MIB::sysORDescr.7 = STRING: The MIB module for managing UDP implementations
SNMPv2-MIB::sysORDescr.8 = STRING: The MIB module for managing IP and ICMP implementations
SNMPv2-MIB::sysORDescr.9 = STRING: The MIB modules for managing SNMP Notification, plus filtering.
SNMPv2-MIB::sysORDescr.10 = STRING: The MIB module for logging SNMP Notifications.
SNMPv2-MIB::sysORUpTime.1 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.2 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.3 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.4 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.5 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.6 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.7 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.8 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.9 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.10 = Timeticks: (0) 0:00:00.00
HOST-RESOURCES-MIB::hrSystemUptime.0 = Timeticks: (148539) 0:24:45.39
HOST-RESOURCES-MIB::hrSystemUptime.0 = No more variables left in this MIB View (It is past the end o
f the MIB tree)
root@andrea:~# _
```

Figure 66. Testing SNMP on slackware

We verify that port 161 is open

```
root@andrea:~# netstat -tuln | grep 161
udp        0      0 0.0.0.0:161             0.0.0.0:*
root@andrea:~# _
```

Figure 67. Verifying port 161 is open

We go back to Solaris and in the file **/usr/local/nagios/etc/objects/hosts.cfg**, we add the Slackware host, specifying its IP address.

```
                                                                Modificado

define host {
        use             linux-server
        host_name       solaris-server
        alias           solaris
        address         192.168.20.101
        check_command   check-host-alive
        max_check_attempts 5
        check_interval 5
        retry_interval 1
        check_period 24x7
        notification_interval 30
        notification_period 24x7
}
define host {
        use             linux-server
        host_name       slackware-server
        alias           Slackware
        address         192.168.20.100
        max_checks_attempts 3
        check_period    24x7
        notification_interval 30
        notification_period 24x7
}

                            [ 13 lÃ-neas leÃ-das ]
^G Ver ayuda ^O Guardar   ^W B
^X Salir     ^R Leer fich.^\ Reemplazar^U Pegar txt ^T OrtografÃ-a^_ Ir a lÃ-nea
```

Figure 68. Slackware host monitoring configuration

In the file **/usr/local/nagios/etc/objects/services.cfg**, we add the CPU Load service for Slackware, specifying its host name

```
define service {
        use                     generic-service
        host_name               slackware-server
        service_description     CPU Load
        check_command           check_cpu_load
        max_check_attempts      3
        check_interval          5
        retry_interval          1
}                           [ 29 lÃ-neas leÃ-das ]
^G Ver ayuda ^O Guardar   ^W B
^X Salir     ^R Leer fich.^\ Reemplazar^U Pegar txt ^T OrtografÃ-a^_ Ir a lÃ-nea
```

Figure 69. Configuration of the CPU service for Slackware

We start the service with the command **/usr/local/nagios/bin/nagios /usr/local/nagios/etc/nagios.cfg** and refresh the page to see the all the hosts that we configured, we can see the slackware host and its CPU service

Host	Service	Last Check	Next Check	Type	Active Checks	Actions
localhost	Current Load	11-23-2024 18:22:35	11-23-2024 18:27:35	Normal	ENABLED	
localhost	PING	11-23-2024 18:22:45	11-23-2024 18:27:45	Normal	ENABLED	
localhost	Root Partition	11-23-2024 18:23:03	11-23-2024 18:28:03	Normal	ENABLED	
slackware-server	CPU Load	11-23-2024 18:23:30	11-23-2024 18:28:30	Normal	ENABLED	
slackware-server		11-23-2024 18:23:57	11-23-2024 18:28:57	Normal	ENABLED	
localhost	Current Users	11-23-2024 18:23:57	11-23-2024 18:28:57	Normal	ENABLED	
localhost		11-23-2024 18:24:08	11-23-2024 18:29:08	Normal	ENABLED	
localhost	SSH	11-23-2024 18:24:24	11-23-2024 18:29:24	Normal	ENABLED	
solaris-server		11-23-2024 18:24:34	11-23-2024 18:29:34	Normal	ENABLED	
solaris-server	CPU Load	11-23-2024 18:24:52	11-23-2024 18:29:52	Normal	ENABLED	
localhost	HTTP	11-23-2024 18:25:19	11-23-2024 18:30:19	Normal	ENABLED	
localhost	Swap Usage	11-23-2024 18:25:46	11-23-2024 18:30:46	Normal	ENABLED	
solaris-server	Disk Space	11-23-2024 18:26:14	11-23-2024 18:31:14	Normal	ENABLED	
localhost	Total Processes	11-23-2024 18:27:08	11-23-2024 18:32:08	Normal	ENABLED	

Figure 70. Slackware monitoring on the website

2.4. Adding new services into Nagios

We create a command in the file **/usr/local/nagios/etc/objects/commands.cfg** where we use the **snmpwalk** command to retrieve information from a specific OID that we specify as a parameter

```
define command {

    command_name    process-host-perfdata
    command_line    /usr/bin/printf "%b" "$LASTHOSTCHECK$\t$HOSTNAME$\t$HOSTSTA$
}


define command {

    command_name    process-service-perfdata
    command_line    /usr/bin/printf "%b" "$LASTSERVICECHECK$\t$HOSTNAME$\t$SERV$
}
define command {
    command_name        check_cpu_load
    command_line        /usr/local/nagios/libexec/check_snmp -H $HOSTADDRESS$ -$
}
define command {
    command_name        check_snmpwalk
    command_line        snmpwalk -v 2c -c public $HOSTADDRESS$ $ARG1$
}


^G Ver ayuda ^O Guardar   ^W
^X Salir     ^R Leer fich.^\ Reemplazar^U Pegar txt ^T OrtografÃ-a^_ Ir a lÃ-nea
```

Figure 71. Adding new command to monitor any OID

We verify that this command is working in the console. We can observe that it returns the disk usage percentage and memory usage

```
root@solaris:~# snmpwalk -v 2c -c public 192.168.20.101 .1.3.6.1.4.1.2021.9.1.9
.1
UCD-SNMP-MIB::dskPercent.1 = INTEGER: 22
```

Figure 72. Testing the OIDs to monitor the disk

```
root@solaris:~# snmpwalk -v 2c -c public 192.168.20.101 .1.3.6.1.4.1.2021.4.6.0
UCD-SNMP-MIB::memAvailReal.0 = INTEGER: 2409228 kB
root@solaris:~#
```

Figure 73. Testing the OIDs to monitor the memory

We add services for disk and memory usage for both hosts

```
define service {
        use                     generic-service
        host_name               solaris-server
        service_description     Disk Space
        check_command           check_snmpwalk! .1.3.6.1.4.1.2021.9.1.9.1
        check_interval          5
        retry_interval          1
        max_check_attempts      3
        notification_interval   60
        contacts                nagiosadmin
}
```

Figure 74. Service to monitor the disk space of Solaris

```
define service {
        use                     generic-service
        host_name               solaris-server
        service_description     Memory Usage
        check_command           check_snmpwalk! .1.3.6.1.4.1.2021.4.6.0
        check_interval          5
        retry_interval          1
        max_check_attempts      3
        notification_interval   60
        contacts                nagiosadmin
}
```

Figure 75. Service to monitor the memory usage of Solaris

```
define service {
        use                     generic-service
        host_name               slackware-server
        service_description     Disk Space
        check_command           check_snmpwalk! .1.3.6.1.4.1.2021.9.1.9.1
        check_interval          5
        retry_interval          1
        max_check_attempts      3
        notification_interval   60
        contacts                nagiosadmin
}
```

Figure 76. Service to monitor the disk space of Slackware

```
define service {
        use                     generic_service
        host_name               slackware-server
        service_description     Memory Usage
        check_command           check_snmpwalk! .1.3.6.1.4.1.2021.4.6.0
        check_interval          5
        retry_interval          1
        max_check_attempts      3
        notification_interval   60
        contacts                nagiosadmin
}
```

Figure 77. Service to monitor the memory usage of Slackware

We verify that the configuration has no syntax errors with the command **/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg**

```
License: GPL

Website: https://www.nagios.org
Reading configuration data...
   Read main config file okay...
   Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
        Checked 11 services.
        Checked 3 hosts.
        Checked 1 host groups.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
        Checked 26 commands.
        Checked 5 time periods.
        Checked 0 host escalations.
        Checked 0 service escalations.
Checking for circular paths...
        Checked 3 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
root@solaris:~#
```

Figure 78. Verifying Nagios configuration for all hosts

We run the Nagios service and refresh the page to see the new services

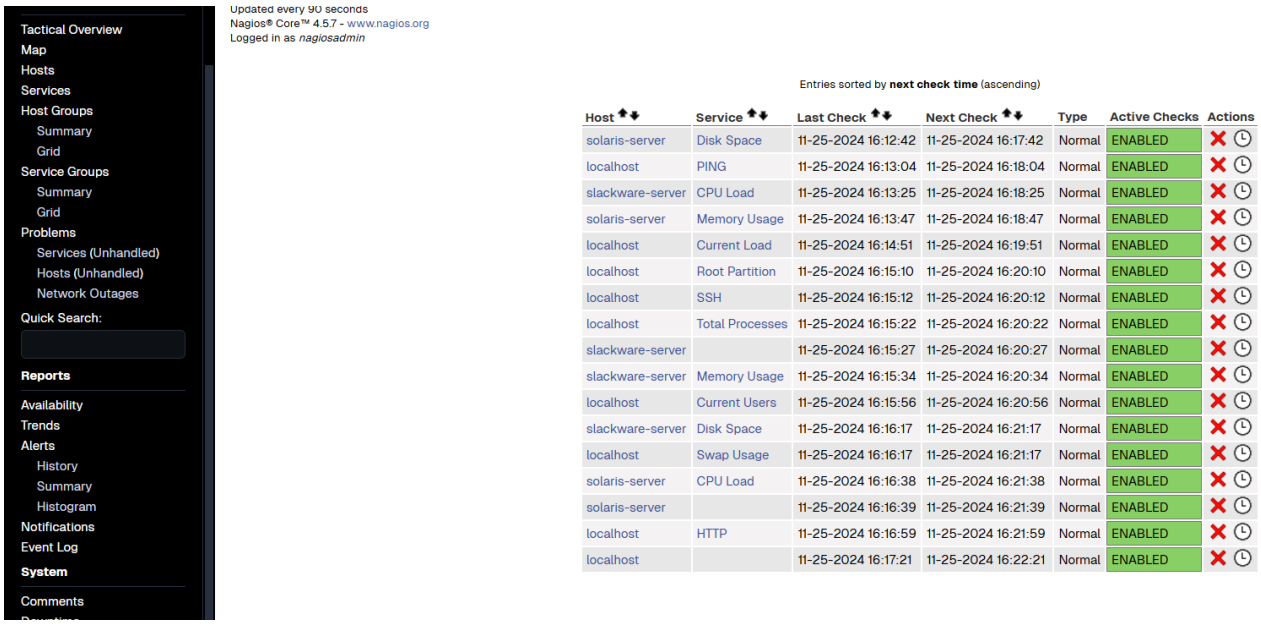


Figure 79. Monitoring the new services (disk and memory) in the Nagios interface for both hosts

2.4. SNMP version and using the snmpwalk command

SNMP version

We are using SNMP version 2 for Solaris and Slackware. Below is a comparative table of the three SNMP versions:

Feature	SNMP v1	SNMP v2	SNMP v3
Security	Community based	Community based	User and Group based (authentication and encryption)
Authentication	Plain text community strings	Plain text community strings	MD5, SHA (authentication), DES, AES (encryption)
Complexity	Simple	Moderate	Complex
Error Reporting	Limited	Enhanced error reporting and exception	Same as SNMP v2
Allowed Operations	GET, SET, TRAP	GET, SET, GETBULK, TRAP, INFORM	GET, SET, GETBULK, TRAP, INFORM, GETNEXT, Response with PDU message format
Typical usage	Used in small networks with basic monitoring needs	Used in medium-sized networks requiring better performance and bulk operations	Used in large or sensitive networks where security is a priority

Snmpwalk command

The snmpwalk command is a tool that leverages the Simple Network Management Protocol (SNMP) to automatically retrieve hierarchical information from a network-enabled device. This tool is especially useful for querying a device's MIB (Management Information Base), which contains data about the device's hardware, software, and network performance.

When snmpwalk is run, it sends multiple GETNEXT requests to the target device. The GETNEXT requests traverse the MIB tree sequentially, collecting all available SNMP objects starting from a specified OID (Object Identifier). This simplifies the process of gathering data, as users do not need to manually query each OID.

For example, let's run this command on Solaris and Slackware, we must write:
snmpwalk -v [SNMP_VERSION] -c [COMMUNITY_NAME] [TARGET_IP]
in our case the command is:

snmpwalk -v 2c -c public [IP]
where:

-v 2c – specifies the SNMP version to use, which in this case is version 2c

-c public – sets the community string to public acting like a password and provides a level of security

[IP] – IP address of the target device we wish to query

```
-EVENT-MIB::mteTriggerFired
DISMAN-EVENT-MIB::mteEventNotification."_snmpd".'_mteTriggerRising' = OID: DISMA
N-EVENT-MIB::mteTriggerRising
DISMAN-EVENT-MIB::mteEventNotificationObjectsOwner."_snmpd".'_linkDown' = STRING
: _snmpd
DISMAN-EVENT-MIB::mteEventNotificationObjectsOwner."_snmpd".'_linkUp' = STRING:
_snmpd
DISMAN-EVENT-MIB::mteEventNotificationObjectsOwner."_snmpd".'_mteTriggerFailure'
 = STRING: _snmpd
DISMAN-EVENT-MIB::mteEventNotificationObjectsOwner."_snmpd".'_mteTriggerFalling'
 = STRING: _snmpd
DISMAN-EVENT-MIB::mteEventNotificationObjectsOwner."_snmpd".'_mteTriggerFired' =
 STRING: _snmpd
DISMAN-EVENT-MIB::mteEventNotificationObjectsOwner."_snmpd".'_mteTriggerRising'
= STRING: _snmpd
DISMAN-EVENT-MIB::mteEventNotificationObjects."_snmpd".'_linkDown' = STRING: _li
nkUpDown
DISMAN-EVENT-MIB::mteEventNotificationObjects."_snmpd".'_linkUp' = STRING: _link
UpDown
DISMAN-EVENT-MIB::mteEventNotificationObjects."_snmpd".'_mteTriggerFailure' = ST
RING: _triggerFail
DISMAN-EVENT-MIB::mteEventNotificationObjects."_snmpd".'_mteTriggerFalling' = ST
RING: _triggerFire
DISMAN-EVENT-MIB::mteEventNotificationObjects."_snmpd".'_mteTriggerFired' = STRI
NG: _triggerFire
DISMAN-EVENT-MIB::mteEventNotificationObjects."_snmpd".'_mteTriggerRising' = STR
ING: _triggerFire
NOTIFICATION-LOG-MIB::nlmConfigGlobalEntryLimit.0 = Gauge32: 1000
NOTIFICATION-LOG-MIB::nlmConfigGlobalAgeOut.0 = Gauge32: 1440 minutes
NOTIFICATION-LOG-MIB::nlmStatsGlobalNotificationsLogged.0 = Counter32: 0 notific
ations
NOTIFICATION-LOG-MIB::nlmStatsGlobalNotificationsBumped.0 = Counter32: 0 notific
ations
root@solaris:~#
```

Figure 80. Running snmpwalk on Solaris

```
SNMPv2-MIB::sysLocation.0 = STRING: Unknown (edit /etc/snmp/snmpd.conf)
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORID.1 = OID: SNMP-FRAMEWORK-MIB::snmpFrameworkMIBCompliance
SNMPv2-MIB::sysORID.2 = OID: SNMP-MPD-MIB::snmpMPDCompliance
SNMPv2-MIB::sysORID.3 = OID: SNMP-USER-BASED-SM-MIB::usmMIBCompliance
SNMPv2-MIB::sysORID.4 = OID: SNMPv2-MIB::snmpMIB
SNMPv2-MIB::sysORID.5 = OID: SNMP-VIEW-BASED-ACM-MIB::vacmBasicGroup
SNMPv2-MIB::sysORID.6 = OID: TCP-MIB::tcpMIB
SNMPv2-MIB::sysORID.7 = OID: UDP-MIB::udpMIB
SNMPv2-MIB::sysORID.8 = OID: IP-MIB::ip
SNMPv2-MIB::sysORID.9 = OID: SNMP-NOTIFICATION-MIB::snmpNotifyFullCompliance
SNMPv2-MIB::sysORID.10 = OID: NOTIFICATION-LOG-MIB::notificationLogMIB
SNMPv2-MIB::sysORDescr.1 = STRING: The SNMP Management Architecture MIB.
SNMPv2-MIB::sysORDescr.2 = STRING: The MIB for Message Processing and Dispatching.
SNMPv2-MIB::sysORDescr.3 = STRING: The management information definitions for the SNMP User-based Se
curity Model.
SNMPv2-MIB::sysORDescr.4 = STRING: The MIB module for SNMPv2 entities
SNMPv2-MIB::sysORDescr.5 = STRING: View-based Access Control Model for SNMP.
SNMPv2-MIB::sysORDescr.6 = STRING: The MIB module for managing TCP implementations
SNMPv2-MIB::sysORDescr.7 = STRING: The MIB module for managing UDP implementations
SNMPv2-MIB::sysORDescr.8 = STRING: The MIB module for managing IP and ICMP implementations
SNMPv2-MIB::sysORDescr.9 = STRING: The MIB modules for managing SNMP Notification, plus filtering.
SNMPv2-MIB::sysORDescr.10 = STRING: The MIB module for logging SNMP Notifications.
SNMPv2-MIB::sysORUpTime.1 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.2 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.3 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.4 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.5 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.6 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.7 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.8 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.9 = Timeticks: (0) 0:00:00.00
SNMPv2-MIB::sysORUpTime.10 = Timeticks: (0) 0:00:00.00
HOST-RESOURCES-MIB::hrSystemUptime.0 = Timeticks: (148539) 0:24:45.39
HOST-RESOURCES-MIB::hrSystemUptime.0 = No more variables left in this MIB View (It is past the end o
f the MIB tree)
root@andrea:~# _
```

Figure 81. Running snmpwalk on Slackware

3. Network Administration – Azure

3.1. Configuration of the web application service

We navigate to the “Education Section”

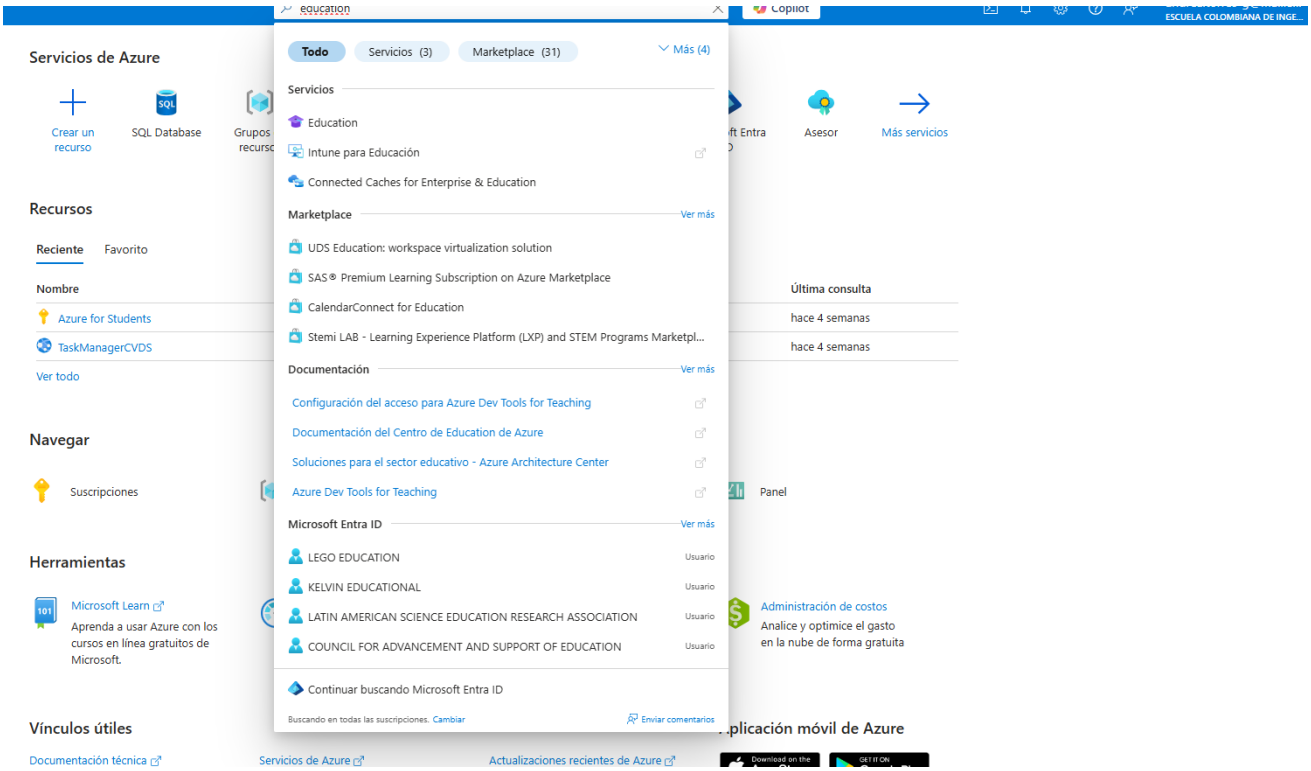


Figure 82. Searching "Education" Section

We navigate to the "**Templates**" Section located in the left menu

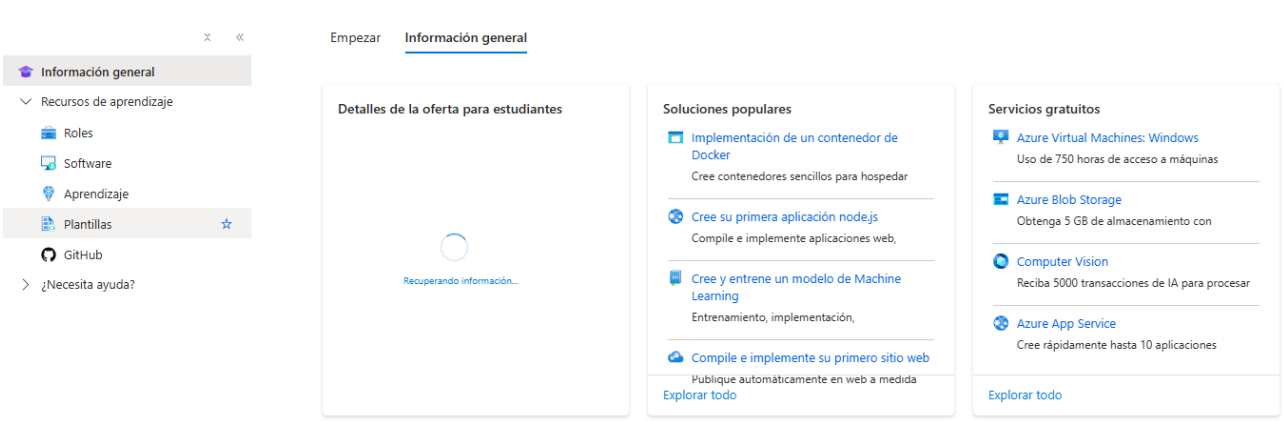


Figure 83. Navigating to "Templates" Section

We search for "**Web app deployment from GitHub**" and click on it

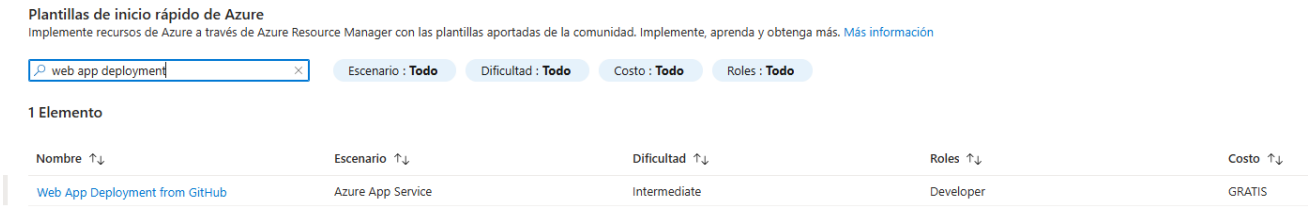


Figure 84. Selecting "Web App Deployment from GitHub" template

We create a resource group and leave the default values as they are

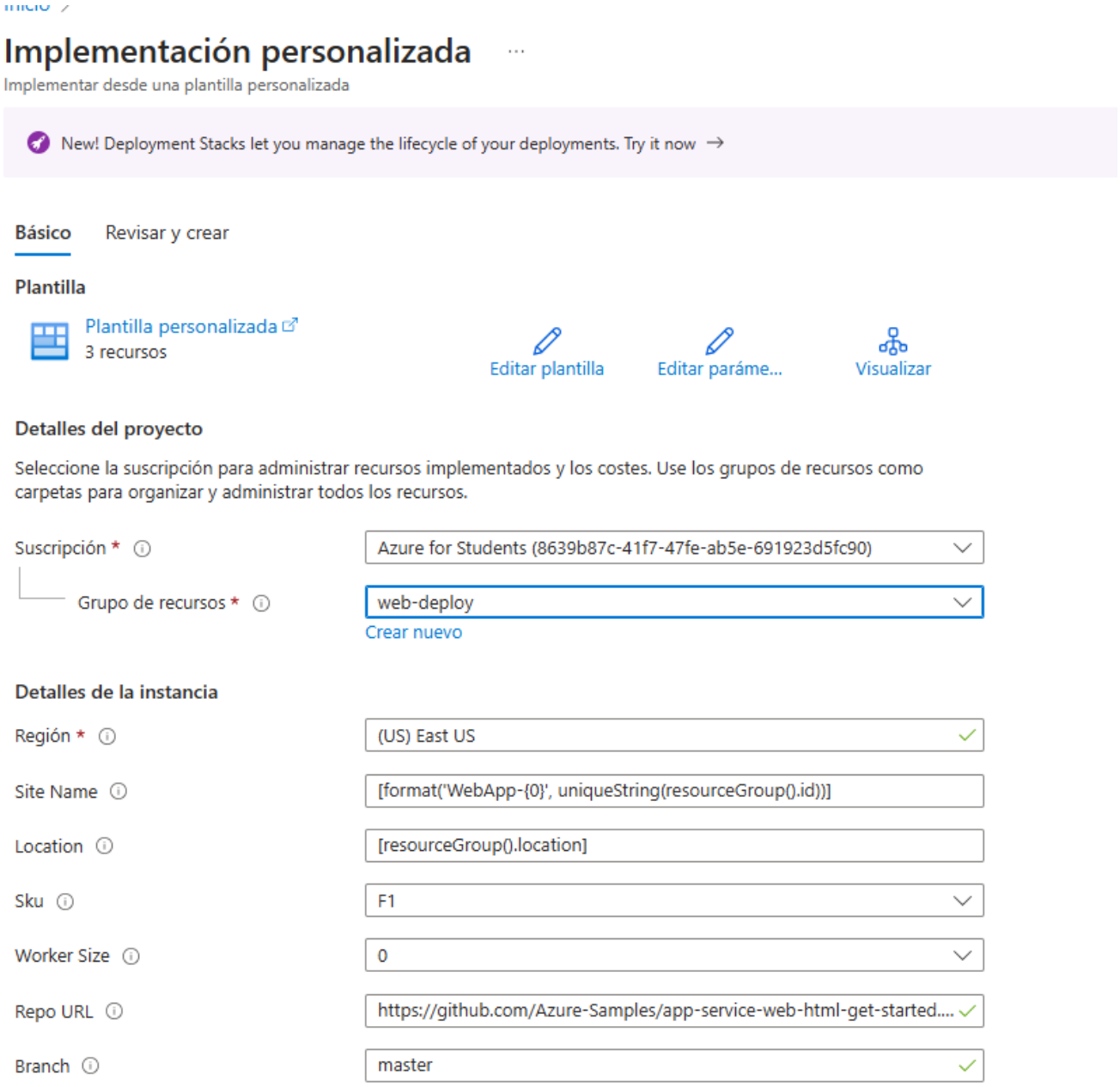


Figure 85. Deploying web service

Click on "**Review + create**" and subsequently "**Create**."

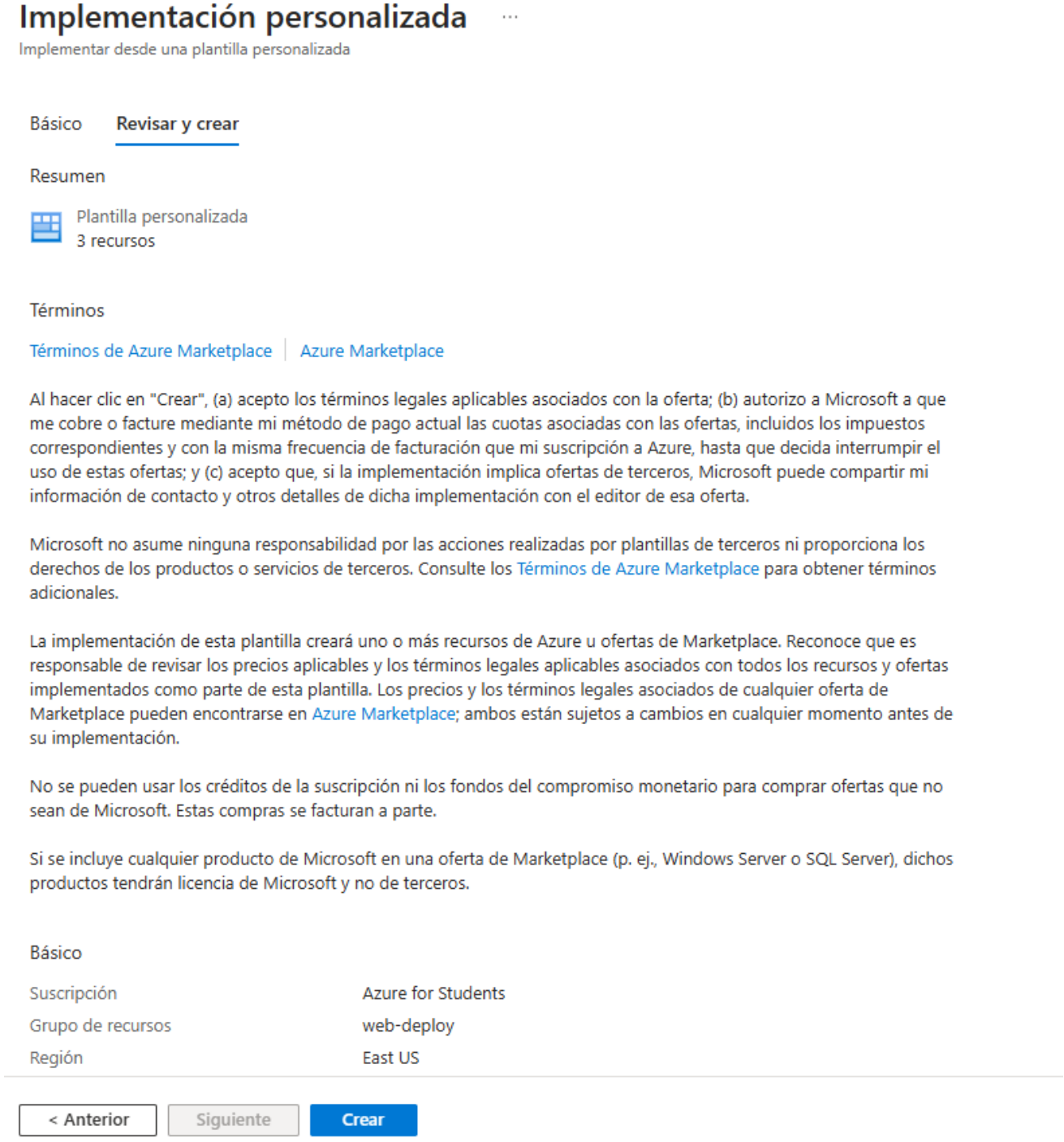


Figure 86. Verifying deployment

We wait for the deployment to be completed

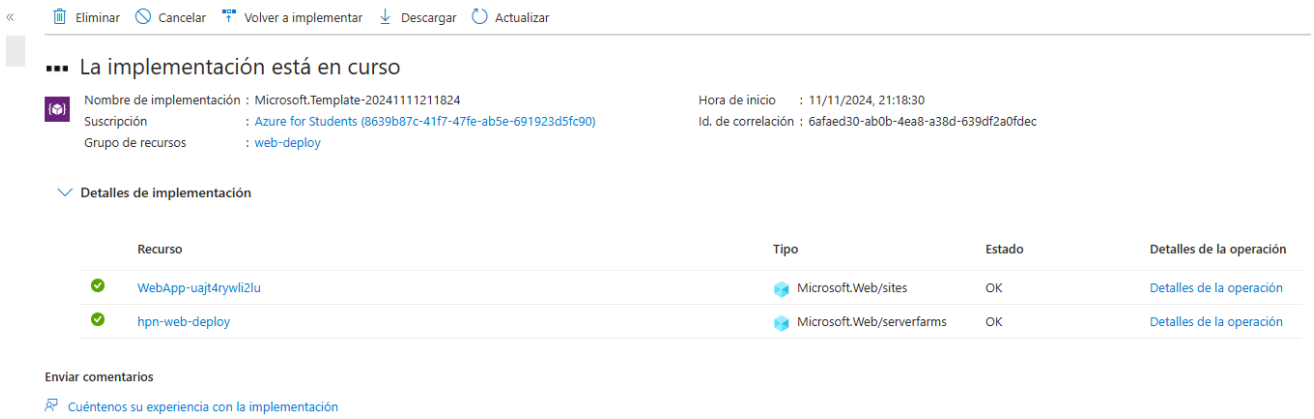


Figure 87. Finalizing deployment

We navigate to the created resource and access the "**Web App**"

Figure 88. Navigating to Web App resource

We explore the website by clicking the "**Default domain**" at the "**Overview**" section

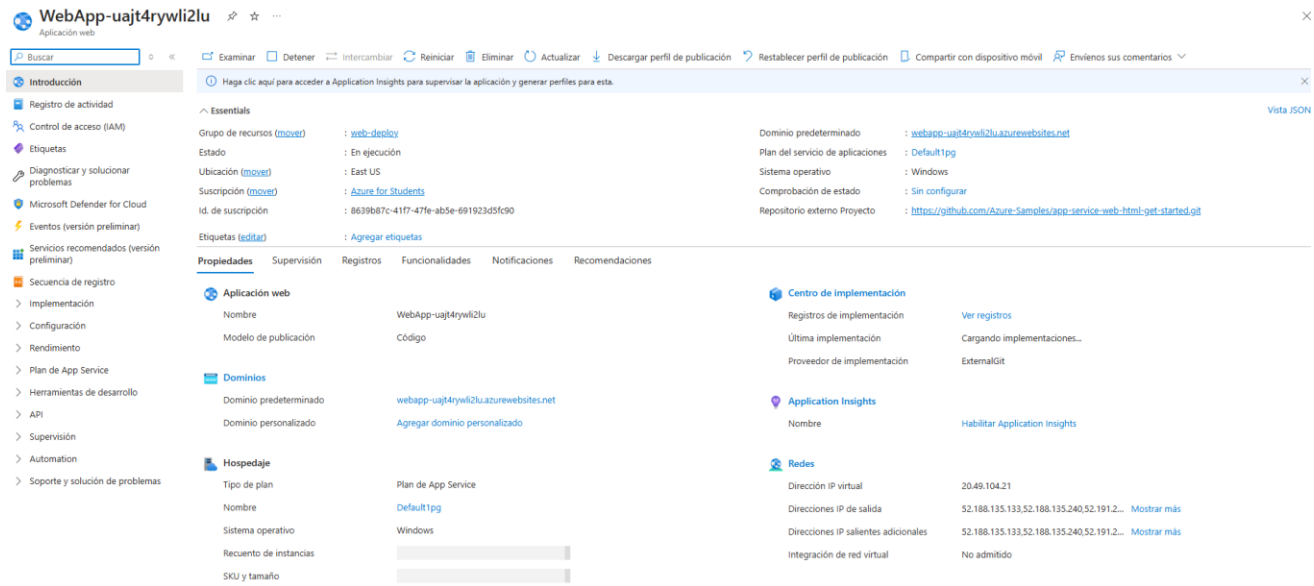


Figure 89. Exploring the web application

We can see the website that was deployed

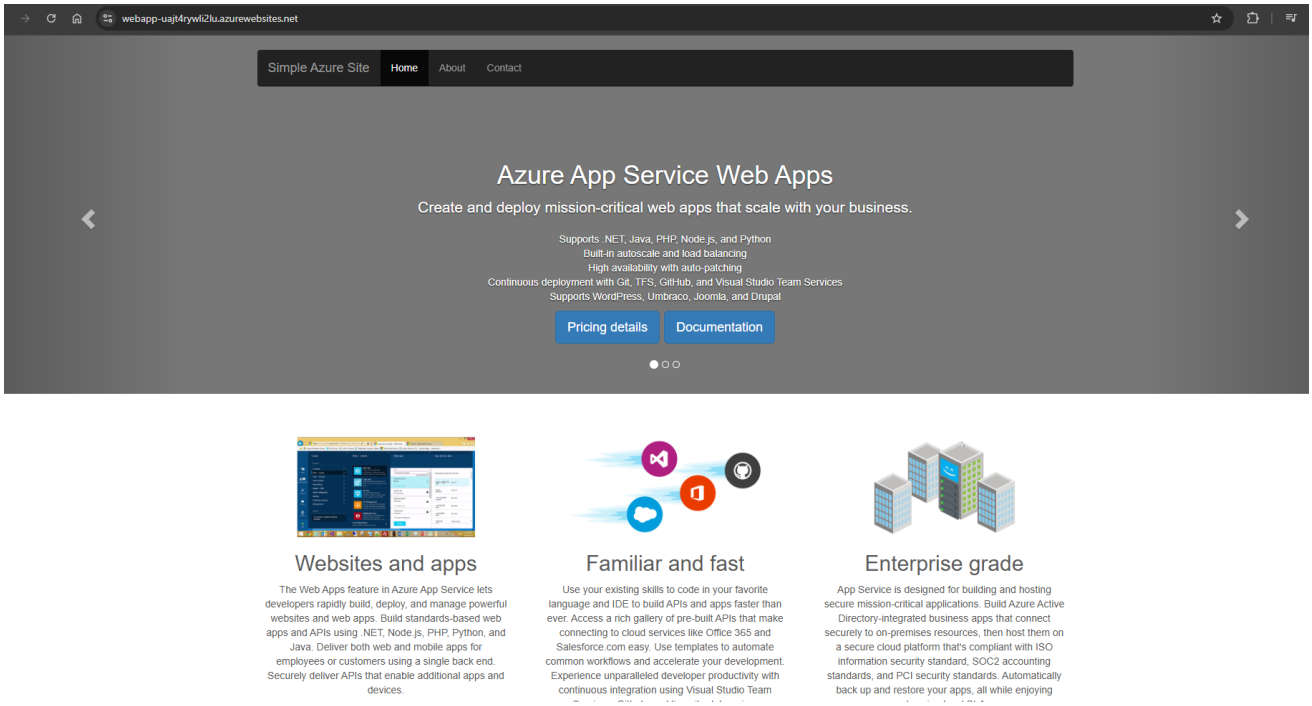


Figure 90. Verifying that Web Service is working

Now, we go to the "**Monitoring**" section located in the left menu, then navigate to "**Application Insights**"

Figure 91. Navigating to Application Insights

We enable the service clicking on "Enable Application Insights"

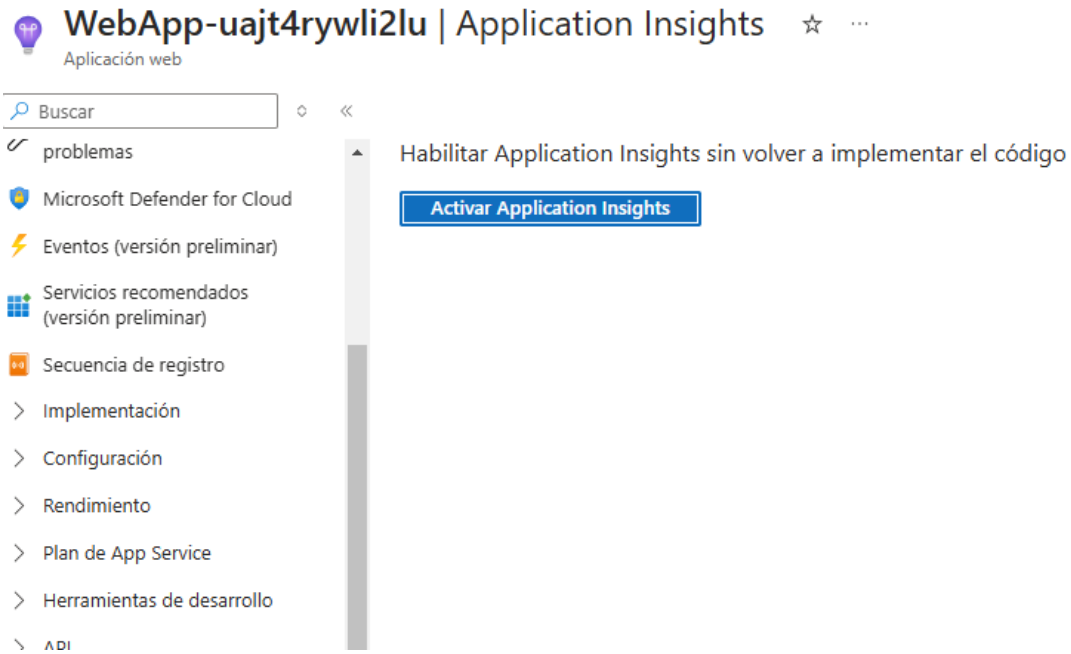


Figure 92. Activating the "Application Insights" service

Once there, click on "**Enable**" and then click on "**Apply**"

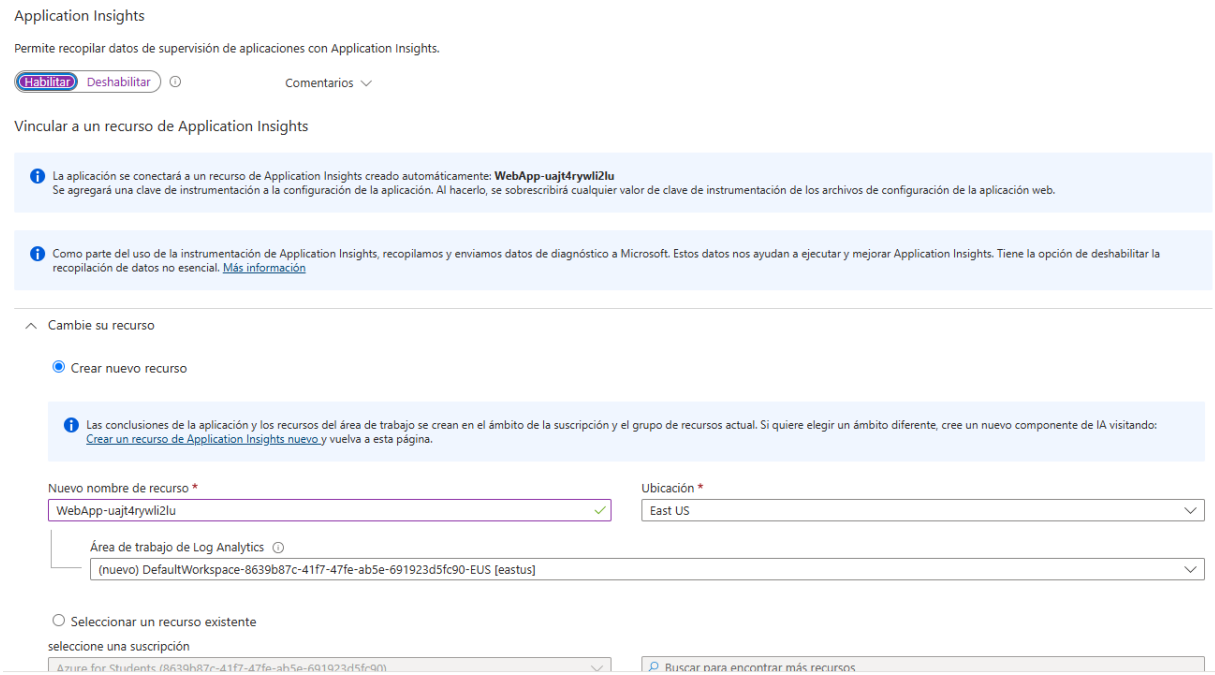


Figure 93. Enabling the "Application Insights" service

When we click on "**Apply**", a confirmation window appears, click on "Yes"

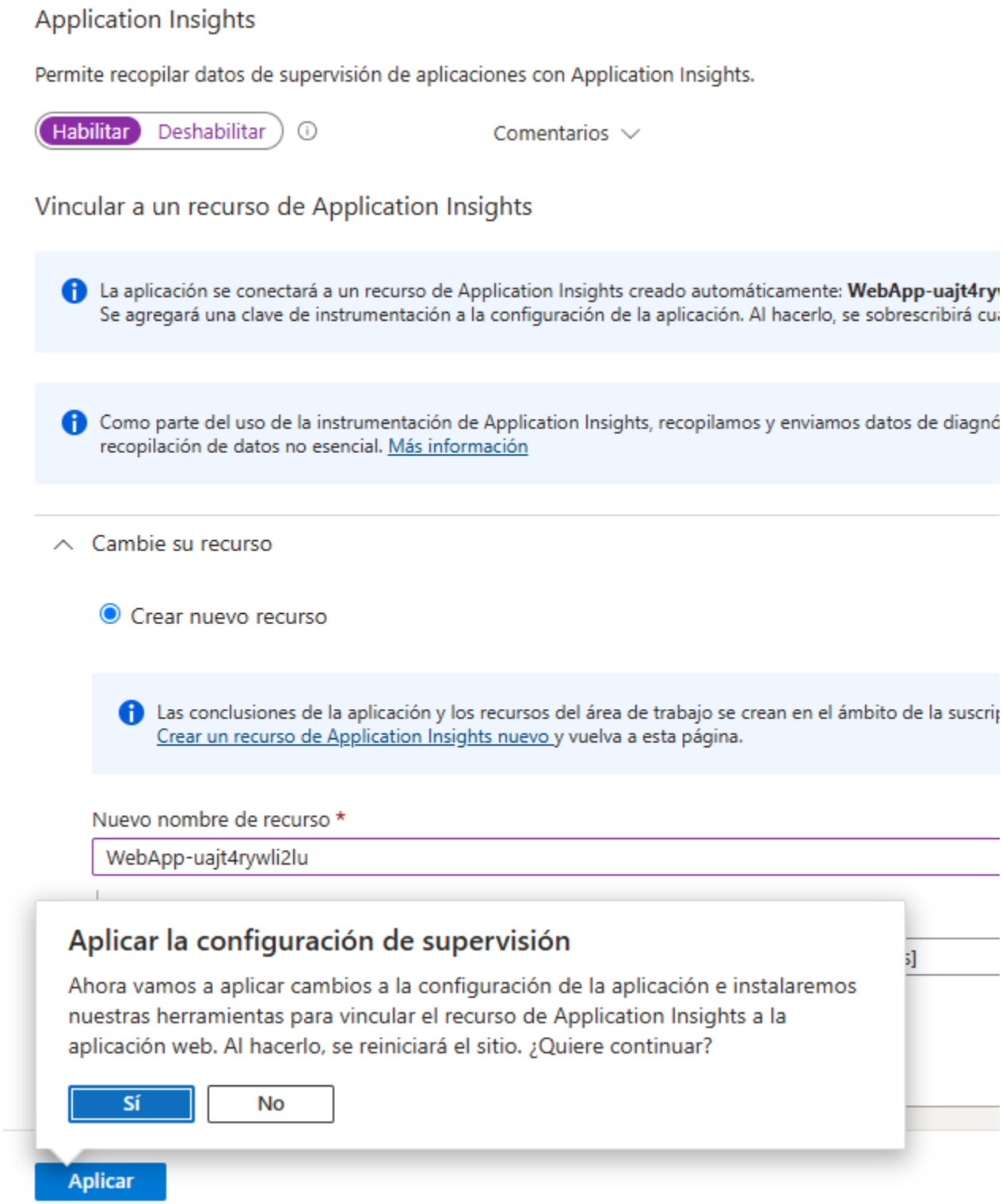


Figure 94. Verifying configuration of "Application Insights"

Once there, click on “View Application Insights data”

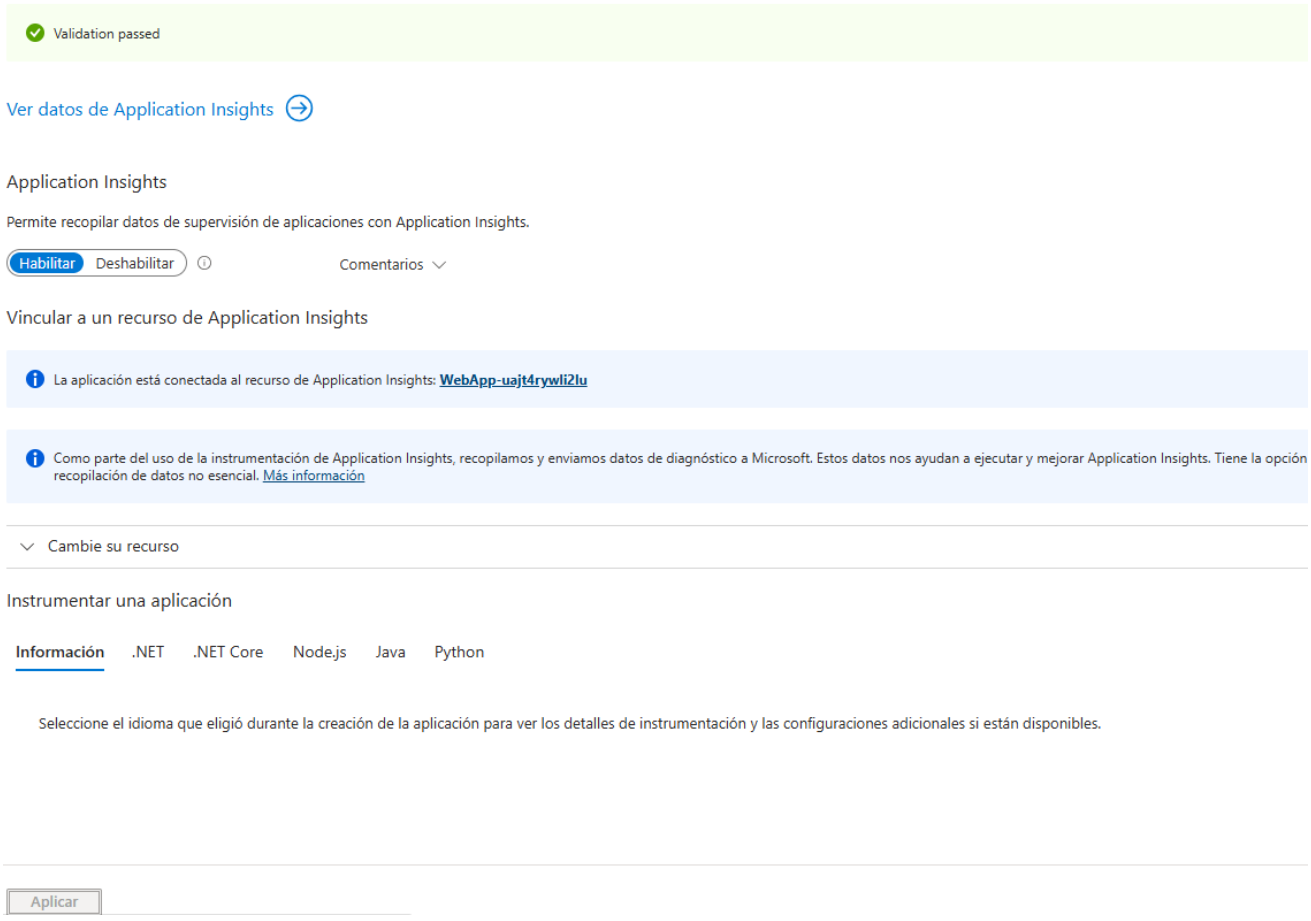


Figure 95. Navigating to Application Insights data

We can see the overview of the service

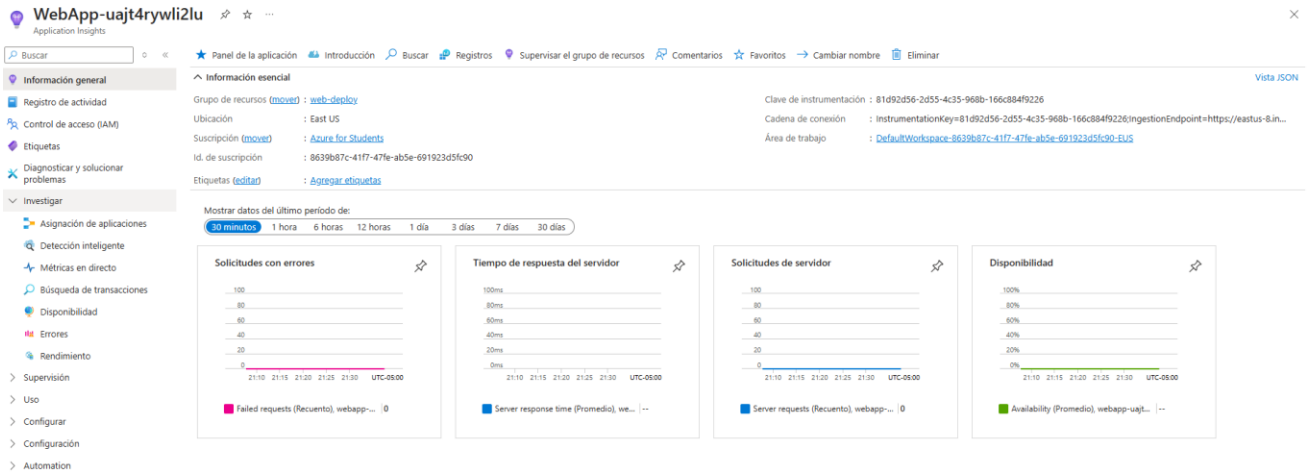


Figure 96. Overview of Application Insights

In the left menu, we go to "**Investigate**" and select "**Live Metric**". We can observe a real-time view of the performance and behavior of our application

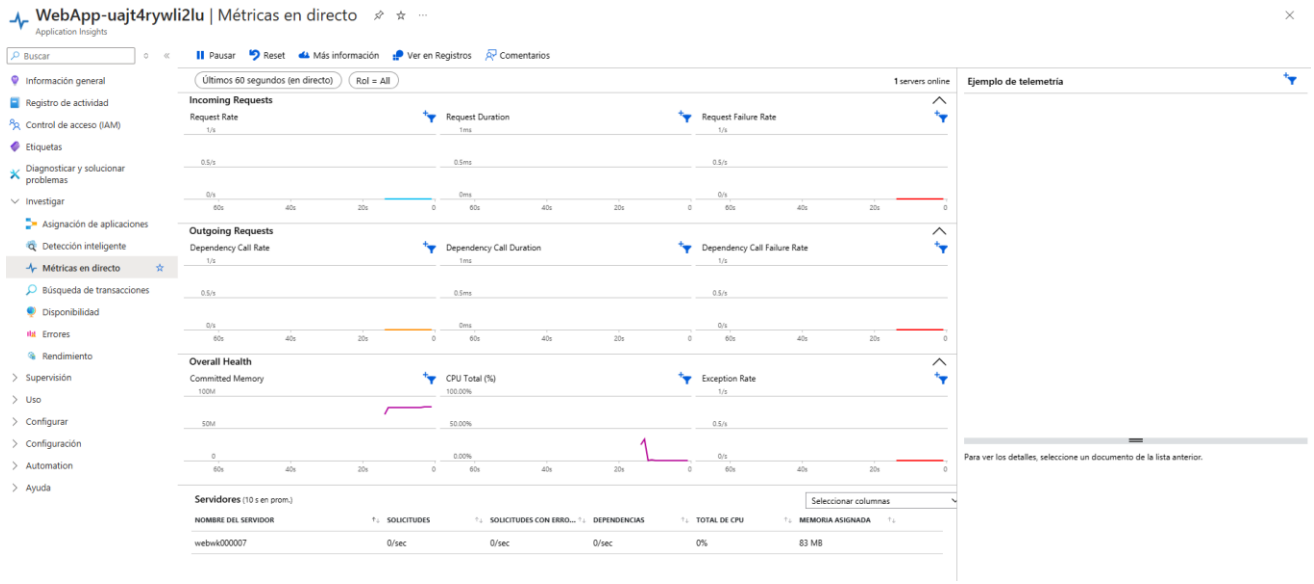


Figure 97. Real-time view of the performance of the web service application

3.2. Query to monitor the IP's accessing the application

In Application Insights, we go to the '**Automation**' section and then to '**Export Template**'

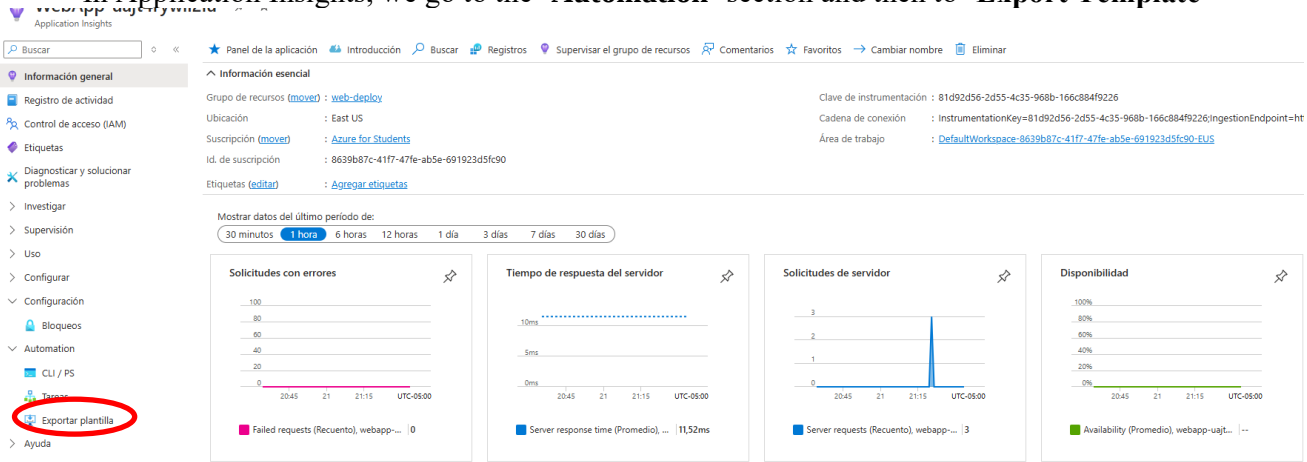


Figure 98. Navigating to 'Export Template'

At the top, click on '**Deploy**'

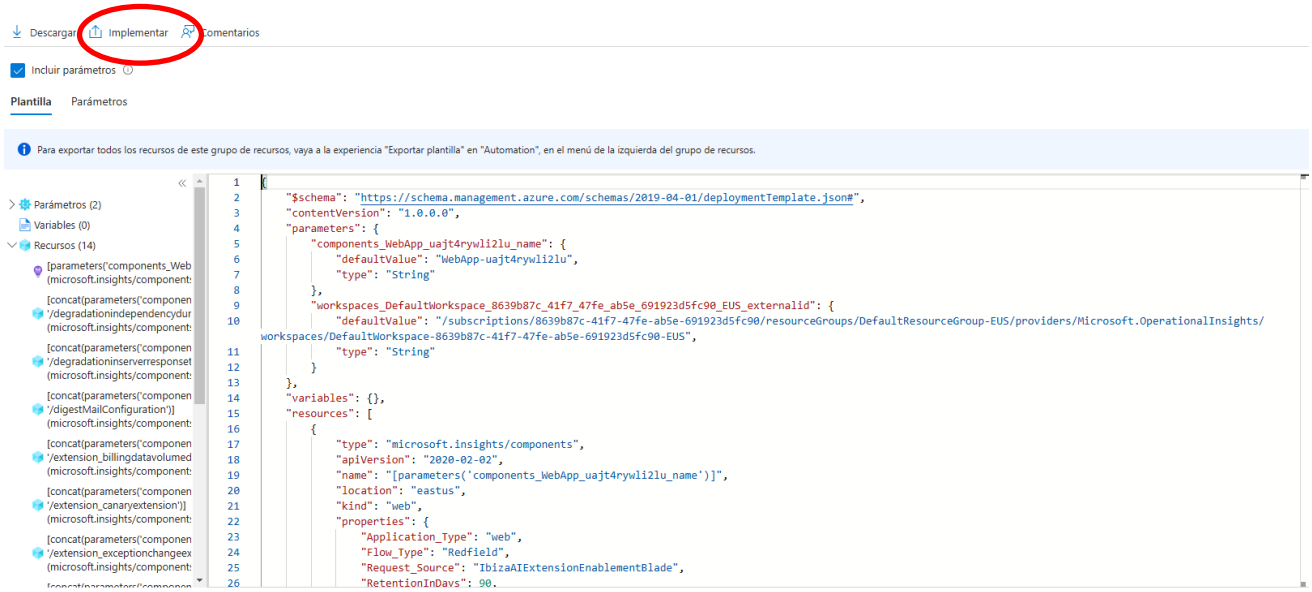


Figure 99. Deploying template

Click on “Edit template”

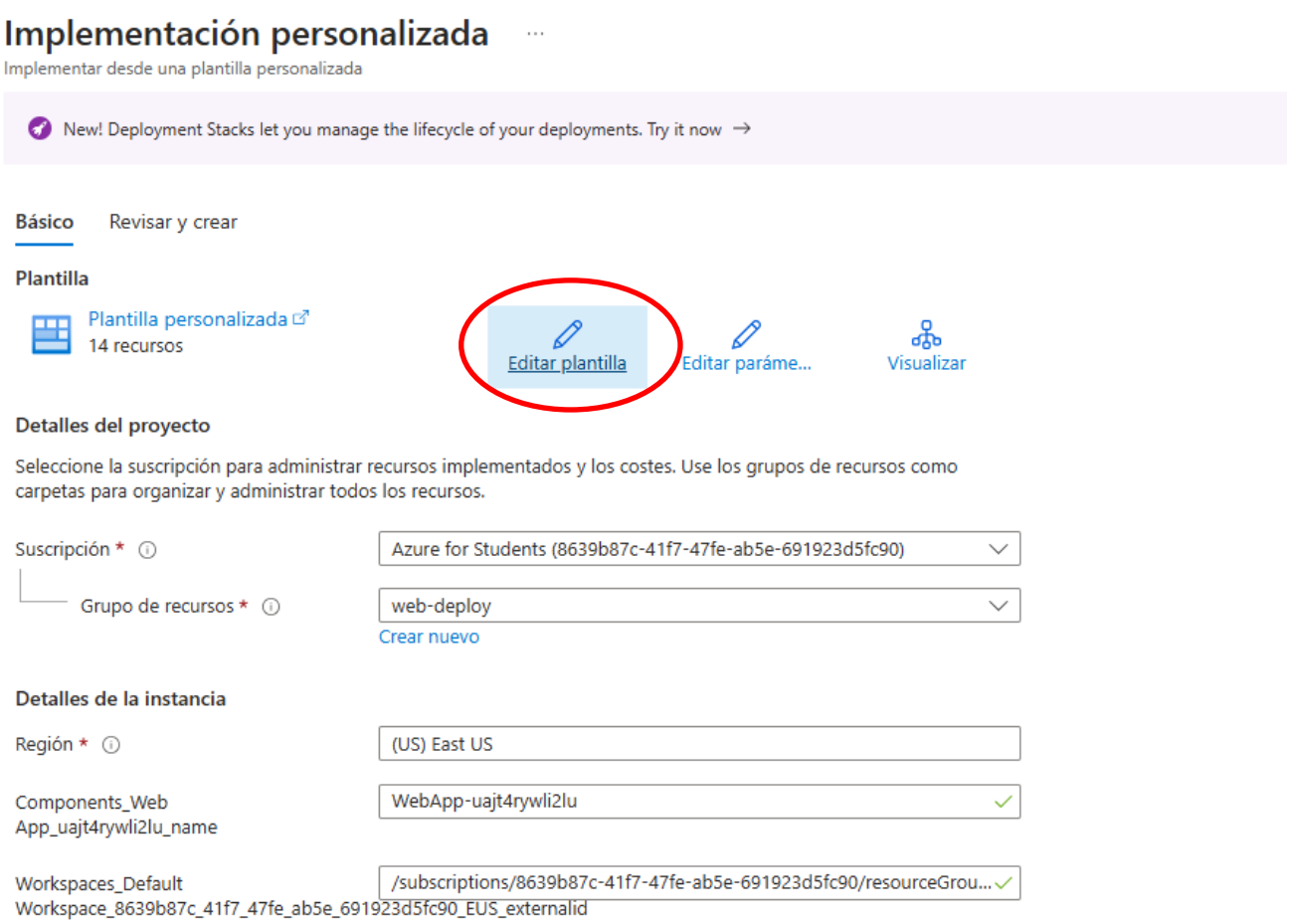


Figure 100. Editing template

Search for "microsoft.insights/components" and in the properties section, add "**DisableIpMasking: true**" at the end. This will disable the masking of IPs that access our application, allowing them to be shown without anonymity.

Figure 101. Adding command to remove IP anonymization in the template

Save the changes and wait for the changes to be applied to the template

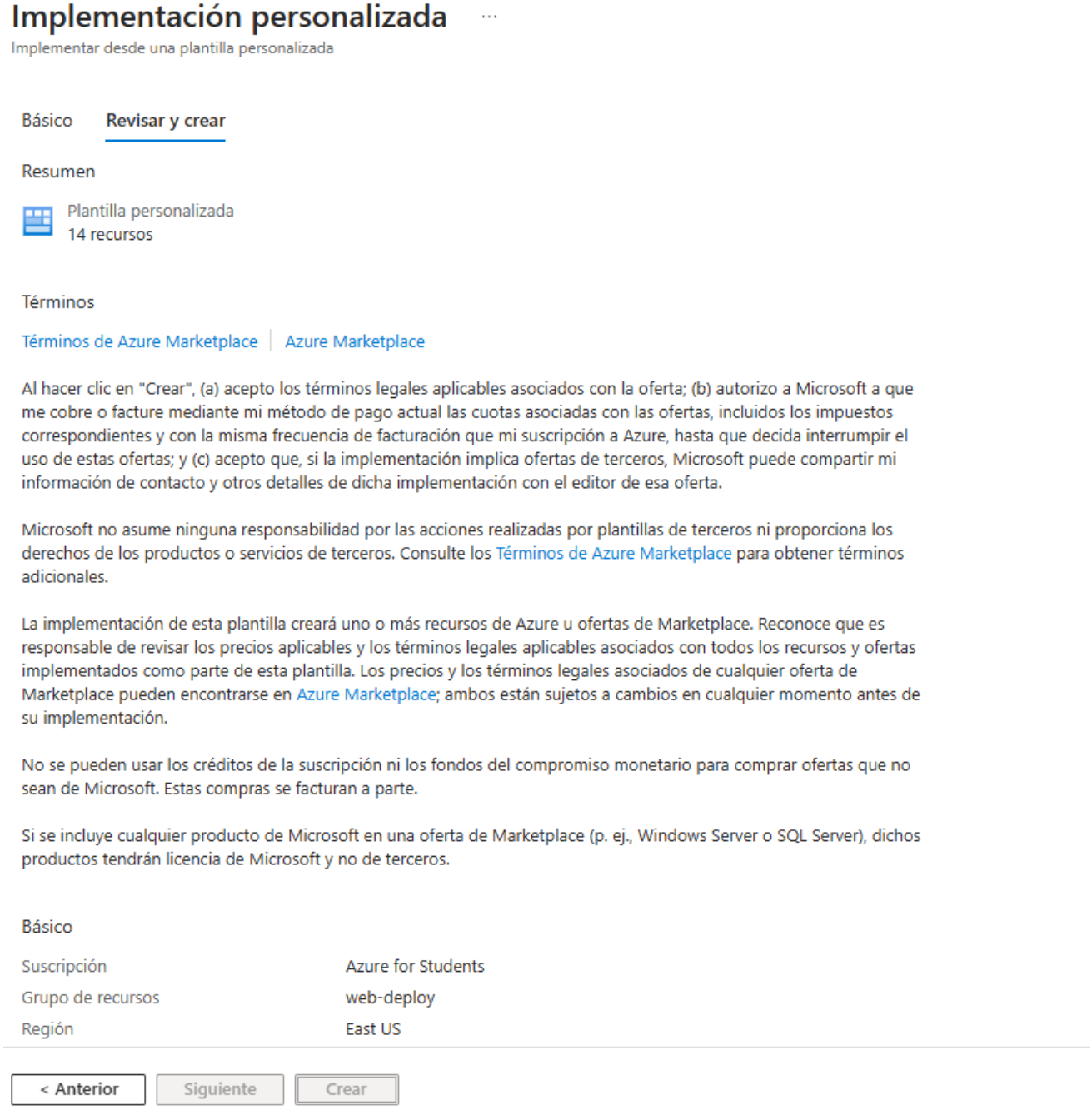


Figure 102. Verifying template changes

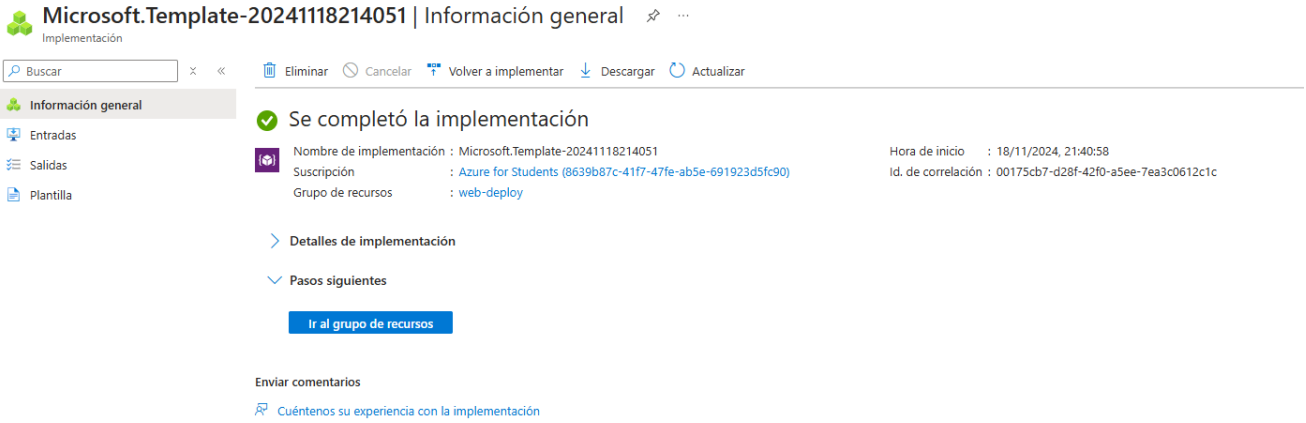


Figure 103. Saving template changes

Go back to Application Insights, then go to Log Analytics, and add the following query. This query will allow us to retrieve the IPs that access our application without anonymity

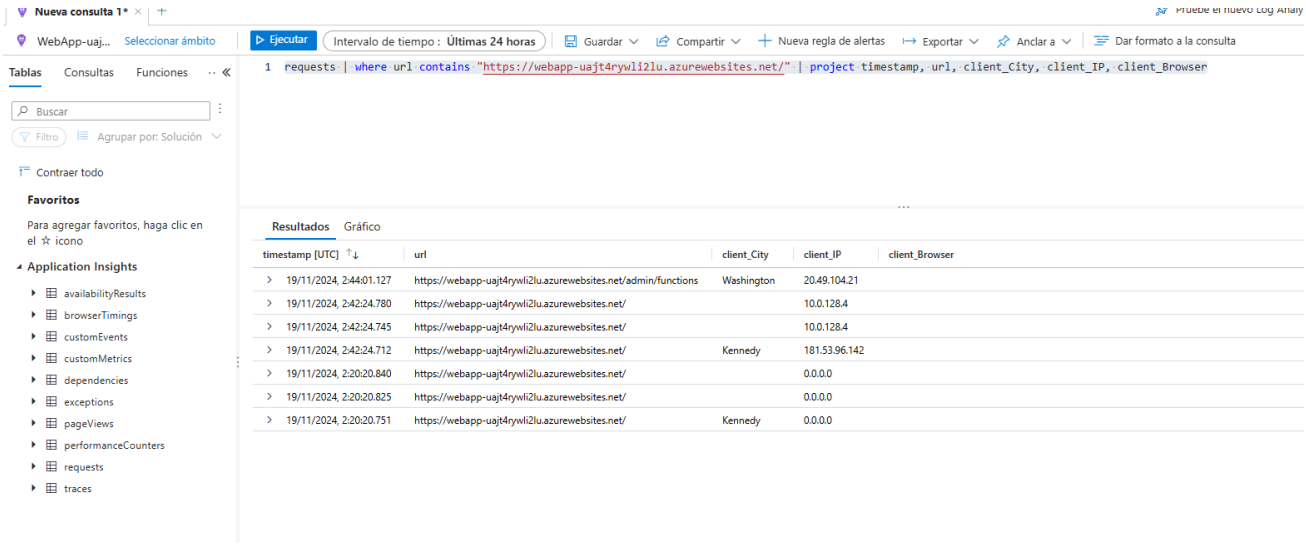


Figure 104. Query to retrieve the IPs that access the web application

3.3. Questionnaire

1. Explore the ”Overview” and ”Live Metric” tabs in the ”Investigate” section.
   - What do you observe when you refresh the website repeatedly?

Each refresh generates new data, updating metrics in real time, like request rates, request duration, committed memory and CPU total.

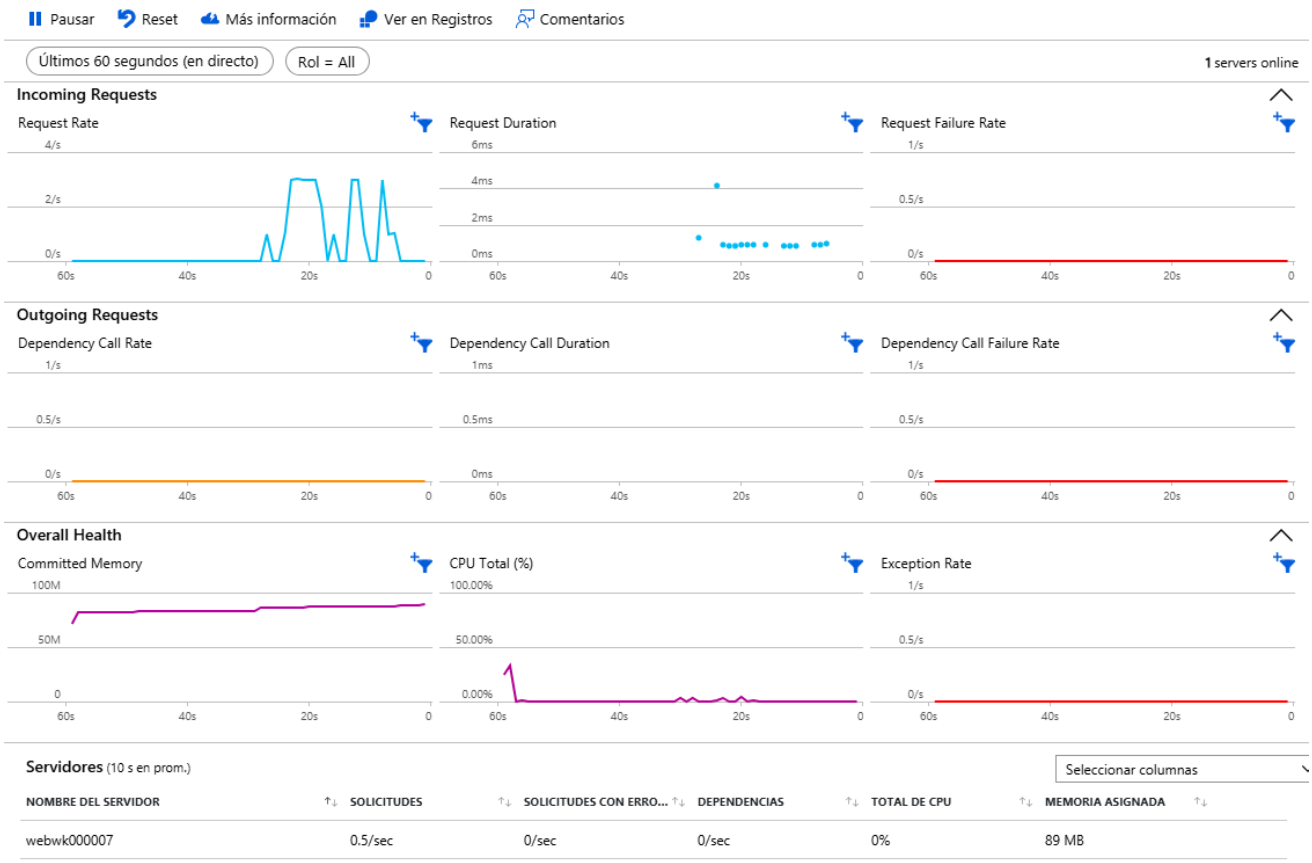


Figure 105. Metrics after refreshing the website repeatedly

– What does each of the items displayed there represent?

Incoming Request

Request Rate: Number of requests made to the app

Request Duration: Represents the average time the application takes to process each request

Request Failure Rate: Measures the percentage of requests that did not complete successfully

Outgoing Request

- Dependency Call Rate: Indicates the number of requests the application makes to other external services, such as databases or third-party APIs
- Dependency Call Duration: It is the average time taken for requests sent to external services to complete
- Dependency Call Failure Rate: Shows the percentage of requests to external services that failed

Overall Health

- Committed Memory: Represents the amount of memory that is allocated or committed to the application at a given moment
- CPU Total (%): Indicates the percentage of CPU usage on the server or resource hosting the application
- Exception Rate: It is the percentage of exceptions or unexpected errors that occur in the application

– What other functionalities does the ”Application Insights” system offer for web services, databases,and other cloud-deployed systems?

Application Insights provides many experiences to enhance the performance, reliability, and quality of our applications.

Investigate

- Application dashboard: An at-a-glance assessment of your application's health and performance.
- Application map: A visual overview of application architecture and components' interactions.
- Live metrics: A real-time analytics dashboard for insight into application activity and performance.
- Transaction search: Trace and diagnose transactions to identify issues and optimize performance.
- Availability view: Proactively monitor and test the availability and responsiveness of application endpoints.
- Failures view: Identify and analyze failures in your application to minimize downtime.
- Performance view: Review application performance metrics and potential bottlenecks.

Monitoring

- Alerts: Monitor a wide range of aspects of your application and trigger various actions.
- Metrics: Dive deep into metrics data to understand usage patterns and trends.
- Diagnostic settings: Configure streaming export of platform logs and metrics to the destination of your choice.
- Logs: Retrieve, consolidate, and analyze all data collected into Azure Monitoring Logs.
- Workbooks: Create interactive reports and dashboards that visualize application monitoring data.

Usage

- Users, sessions, and events: Determine when, where, and how users interact with your web app.
- Funnels: Analyze conversion rates to identify where users progress or drop off in the funnel.
- Flows: Visualize user paths on your site to identify high engagement areas and exit points.
- Cohorts: Group users by shared characteristics to simplify trend identification, segmentation, and performance troubleshooting.

Code analysis

- Profiler: Capture, identify, and view performance traces for the application.
- Code optimizations: Harness AI to create better and more efficient applications.
- Snapshot debugger: Automatically collect debug snapshots when exceptions occur in .NET application

– How might this functionality be beneficial in a corporate environment?

It helps detect issues early, optimize resource usage, enhance user experience, and support proactive monitoring, which minimizes downtime and improves overall performance.

– How does the network layer contribute to the transmission of data between the web app andclients in different geographic locations?

The network layer handles IP addressing and routing, ensuring data packets are transmitted to the correct locations across various networks, optimizing speed and reliability based on geographic

distance and network conditions.

- In what ways do the application and transport layers ensure reliable data transfer and correctinterpretation of web app responses?
  - **Transport Layer:** Uses protocols like TCP to ensure packets are delivered reliably and in the correct order. Error checking and retransmission requests are included.
  - **Application Layer:** Converts data to and from formats understandable by the application and user, ensuring data consistency and correct interpretation across devices.

Experiments

Perform the following tests in groups or individually, as instructed by the professor, and document the experience.

1. Use of ICMP Messages

We go to https://traceroute-online.com/ and search for the Computer Science Laboratory page and theStanford University page. Show the results:

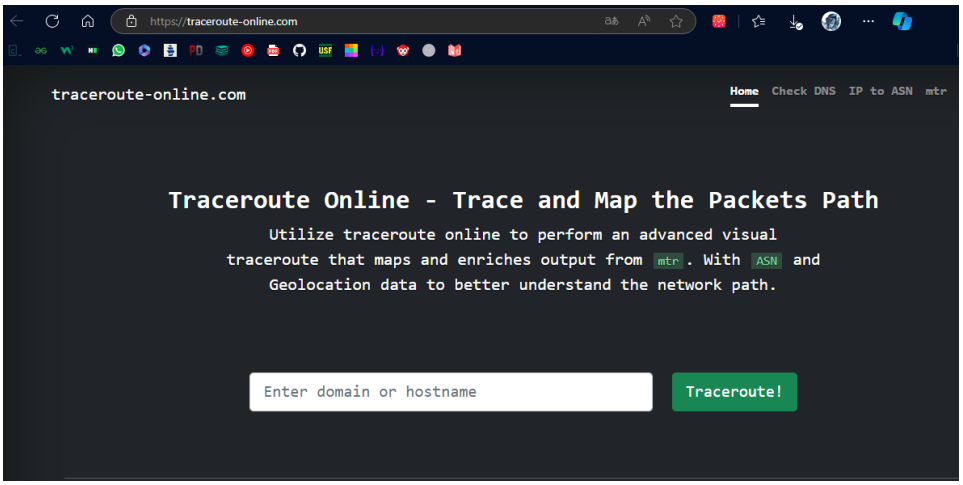


Figure 106 traceroute-online page

Then, we searched the Computer Science Laboratory page

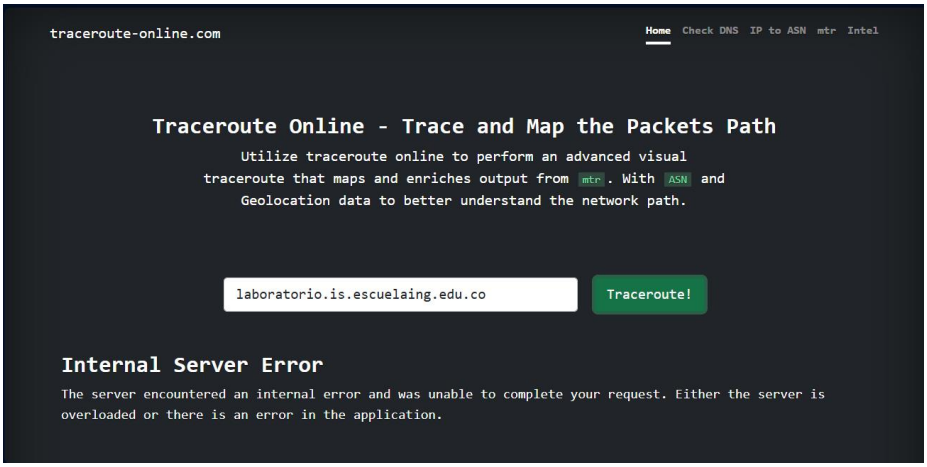
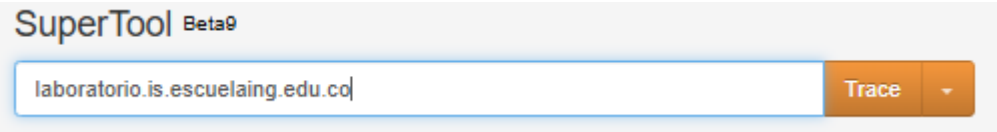


Figure 107 Search for the laboratory page

We can observe an internal server error, so we use the following page: MXToolbox.



HopCount	IP Address	HostName
1	10.140.10.147	* * *
2	3.236.63.11 Amazon.com, Inc. (AS14618)	ec2-3-236-63-11.compute-1.amazonaws.com
3	* * *	* * *
4	* * *	* * *
5	* * *	* * *
6	* * *	* * *
7	99.83.66.22	* * *
8	99.83.69.241	* * *
9	69.79.100.53 Columbus Networks USA, Inc. (AS23520)	* * *
10	69.79.100.5 Columbus Networks USA, Inc. (AS23520)	ae2.brx-mx2020-2.boca-raton.fl.usa.cwc.com
11	190.131.207.5 LIBERTY NETWORKS DE COLOMBIA S.A.S (AS262191)	* * *
12	190.131.207.183 LIBERTY NETWORKS DE COLOMBIA S.A.S (AS262191)	* * *
13	* * *	* * *
14	45.239.88.78 Unknown (AS266862)	* * *
15	* * *	* * *
16	45.239.88.78 ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO (AS266862)	* * *
17	45.239.88.88 ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO (AS266862)	* * *

From the information provided on the page, we can see the following:

Hop 1: The packet leaves your local network, passing through a private IP address (10.140.10.147), suggesting it is your router or a device within your local network.

Hop 2: The packet enters the Amazon Web Services (AWS) infrastructure with the IP 3.236.63.11, belonging to Amazon.com, Inc., indicating that AWS is handling part of the traffic.

Hops 3 to 6: These hops show no response from the routers. This could be due to routers configured not to respond to traceroute or packet loss, which is not unusual.

Hop 7: The packet continues within AWS infrastructure, reaching IP 99.83.66.22, without a hostname available, indicating it is still within Amazon's network.

Hop 8: Still within AWS, the packet reaches IP 99.83.69.241, again without a hostname available.

Hop 9: The packet leaves AWS and enters the network of Columbus Networks USA, Inc. with IP 69.79.100.53, showing it has reached a service provider in the United States.

Hop 10: Within Columbus Networks, the packet reaches IP 69.79.100.5, with the hostname ae2.brx-mx2020-2.boca-raton.fl.usa.cwc.com, indicating its passage through Boca Raton, Florida.

Hop 11: The packet enters the network of Liberty Networks of Colombia with IP 190.131.207.5, showing it has reached a service provider in Colombia.

Hop 12: Continuing within Liberty Networks, the packet reaches IP 190.131.207.183, confirming its presence within the Colombian network.

Hop 13: There is no response at this hop, which could be due to packet filtering or routers configured not to respond to traceroute requests.

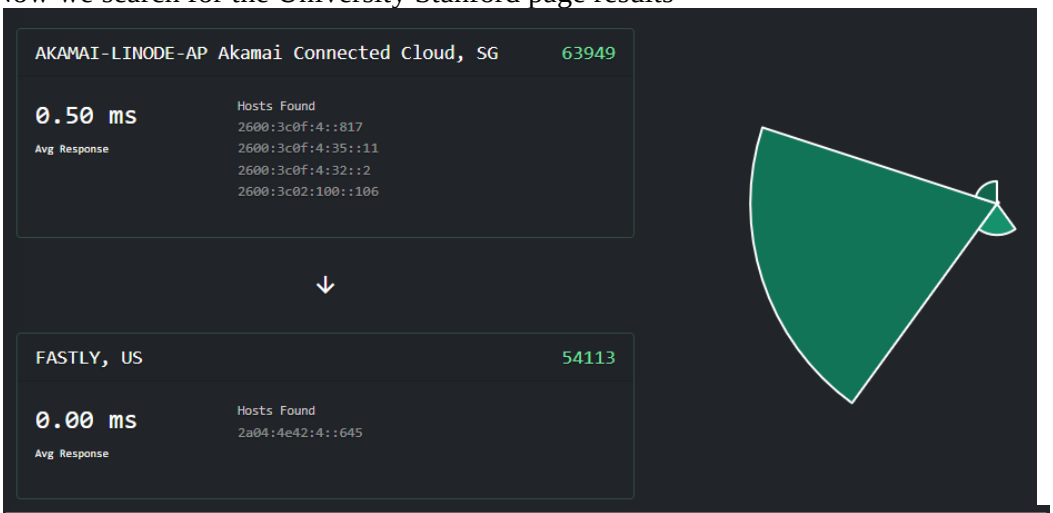
Hop 14: The packet reaches IP 45.239.88.78, registered to an unknown ASN, but continues its way to the destination.

Hop 15: No response at this hop, similar to previous hops where routers do not respond.

Hop 16: The packet reaches IP 45.239.88.78 again, now identified as part of the network of Escuela Colombiana de Ingeniería Julio Garavito.

Hop 17: Finally, the packet reaches IP 45.239.88.88, belonging to the network of Escuela Colombiana de Ingeniería Julio Garavito, thus reaching its destination.

Now we search for the University Stanford page results



Hop	IP / Host Name	ISP	Netblock	Country	Loss	Response
1	2600:3c0f:4::817	AKAMAI-LINODE-AP Akamai Connected Cloud, SG	2600:3c0f:4::/48		0.0%	0.46ms
2	2600:3c0f:4:35::11	AKAMAI-LINODE-AP Akamai Connected Cloud, SG	2600:3c0f:4::/48		0.0%	0.58ms
3	2600:3c0f:4:32::2	AKAMAI-LINODE-AP Akamai Connected Cloud, SG	2600:3c0f:4::/48		0.0%	2.91ms
4	2600:3c02:100::106	AKAMAI-LINODE-AP Akamai Connected Cloud, SG	2600:3c02::/32		0.0%	0.69ms
5	2001:504:17:110::215				0.0%	0.59ms
6	2a04:4e42:4::645	FASTLY, US	2a04:4e42:4::/48		0.0%	0.48ms

The information we can obtain is the following:

Hop 1: Device managed by Akamai Connected Cloud in Singapore (IP: 2600:3c0f:4::817), with a response time of 0.50 ms, indicating a fast connection.

Hop 2: Another Akamai Connected Cloud device in Singapore (IP: 2600:3c0f:4:35::11), with a response time of 0.57 ms, similar to the first hop, suggesting physical proximity.

Hop 3: Akamai Connected Cloud node (IP: 2600:3c0f:4:32::2) in Singapore, with a response time of 1.43 ms, showing a slight increase in latency.

Hop 4: Another Akamai Connected Cloud node (IP: 2600:3c02:100::106) in Singapore, with a response time of 9.05 ms, indicating higher latency, likely due to greater distance or network congestion.

Hop 5: Intermediate node with no ISP information (IP: 2001:504:17:110::215), with a response time of 0.72 ms, suggesting an efficient connection, though the ISP is unspecified.

Hop 6: Node managed by Fastly in the US (IP: 2a04:4e42:4::645), with a response time of 0.65 ms, indicating a fast connection on the final stretch before reaching the destination.

The only point on the map corresponds to a Fastly node, a content delivery network (CDN) that optimizes content delivery from servers located close to the destination. The IP address 2a04:4e42:4::645 represents the last hop before reaching the University of Stanford server, which explains the display of a single point on the map due to the proximity and efficiency of the node. CDNs like Fastly hide intermediate hops, resulting in only the final node being shown.

Using the tracert or traceroute command, search for a page in France and check the route.

```
jgamb  System32  tracert www.info.gouv.fr

Traza a la dirección cs964.wpc.upsiloncdn.net [152.199.55.123]
sobre un máximo de 30 saltos:

  1     5 ms     5 ms     5 ms  192.168.1.254
  2     *        *        *     Tiempo de espera agotado para esta solicitud.
  3    19 ms    35 ms    16 ms  10.166.12.58
  4    22 ms    21 ms    19 ms  10.166.12.57
  5    39 ms    40 ms    33 ms  static-adsl200-24-35-183.epm.net.co [200.24.35.183]
  6    81 ms    81 ms    82 ms  static-adsl200-24-33-90.epm.net.co [200.24.33.90]
  7    82 ms    81 ms    92 ms  ae-112.border1.min.edgecastcdn.net [152.195.89.204]
  8    82 ms    73 ms    74 ms  po-67.core1.mid.edgecastcdn.net [152.195.89.145]
  9    69 ms    69 ms    68 ms  152.199.55.123

Traza completa.
```

Hop 1: The first hop is the local network gateway (192.168.1.254), which responds quickly at 5 ms, indicating that the packet hasn't left the local network yet.

Hop 2: The second hop does not respond, which suggests that the device is configured not to reply to traceroute requests or that it is blocking this type of traffic.

Hop 3: The third hop shows an internal IP address of the ISP network (10.166.12.58), with response times ranging from 16 ms to 35 ms, indicating that the packet is still within the ISP's infrastructure.

Hop 4: The fourth hop shows another internal IP address of the ISP (10.166.12.57), with similar response times, indicating that the packet is still within the ISP's network.

Hop 5: The fifth hop shows a public IP address (200.24.35.183) from the ISP, with response times of 39 ms to 40 ms, suggesting that the packet has exited the ISP's internal network and is now on the public internet.

Hop 6: The sixth hop shows another public IP address (200.24.33.90) from the ISP, with response times of 81 ms to 82 ms, indicating that the packet is traveling through a network with greater distance or congestion.

Hop 7: The seventh hop corresponds to a node in the Edgecast CDN (152.195.89.204), with response times ranging from 81 ms to 92 ms, suggesting that the packet is passing through a Content Delivery Network (CDN).

Hop 8: The eighth hop remains in the Edgecast CDN network (152.195.89.145), with slightly lower response times (73 ms to 74 ms), bringing the packet closer to the final destination.

Hop 9: The ninth hop reaches the destination server (152.199.55.123), with response times of 69 ms, indicating that the packet has successfully arrived at the target website, www.info.gouv.fr.

Download and install software such as VisualRoute, Open Visual Traceroute, or similar. They can befree tools or demos.

We go to Open Visual Traceroute and install the application

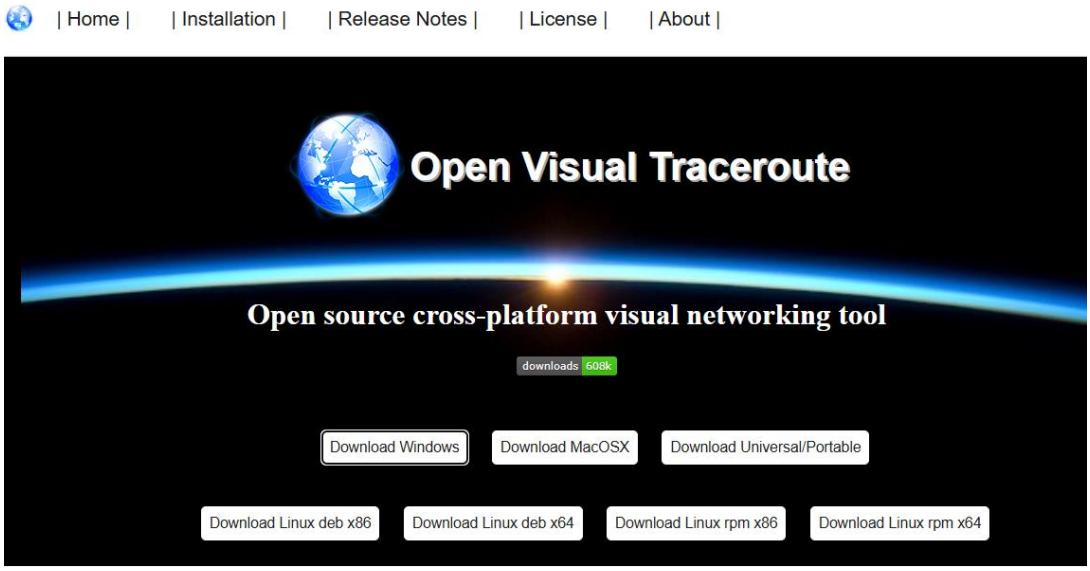


Figure 108 Open Visual Traceroute page

We configure the installation features and click on finish.

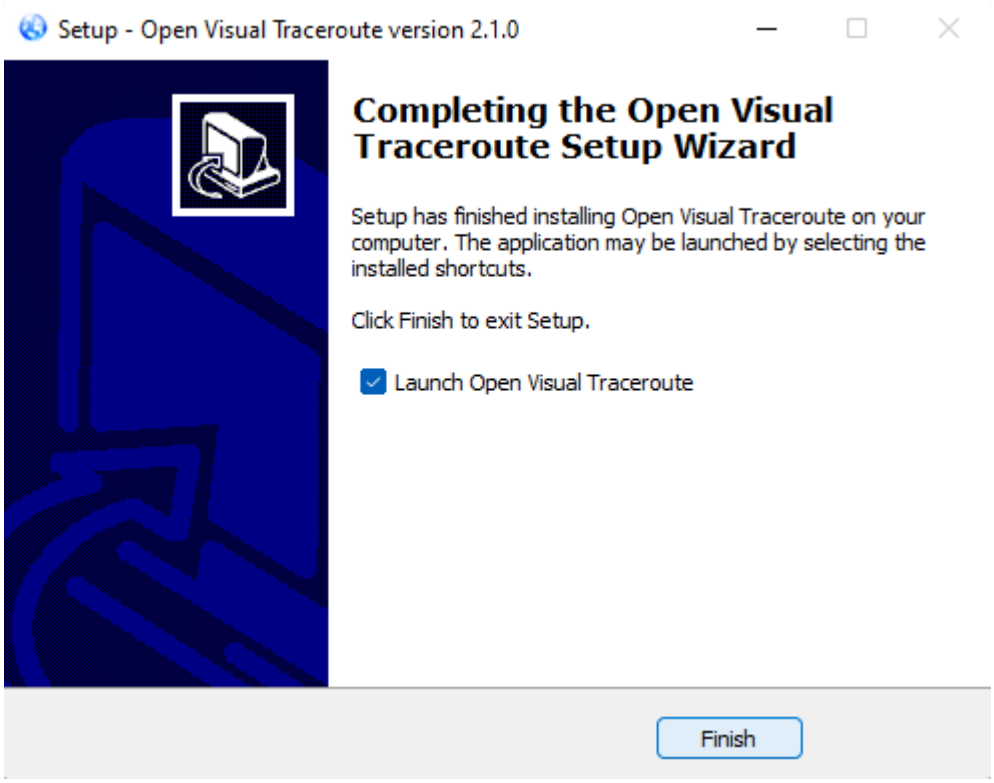


Figure 109 Completion of the program installation

Now we open the program, and the following interface will appear.

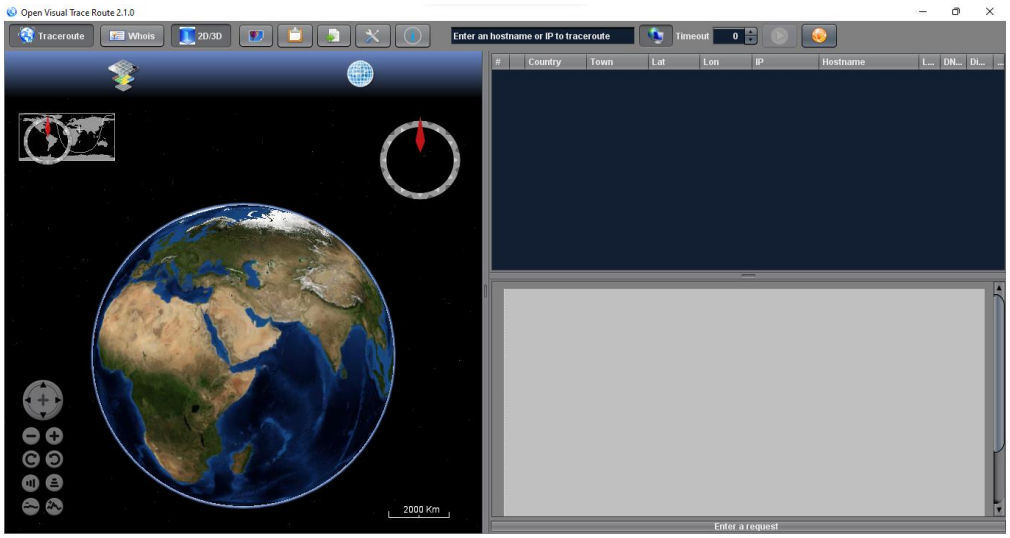


Figure 110 Initial interface of Open Visual Traceroute

Now we will make 5 queries to different places in the world to see how it works, with the queries being to car manufacturers' websites

- In Australia
  - awww.volkswagen-group.com/en

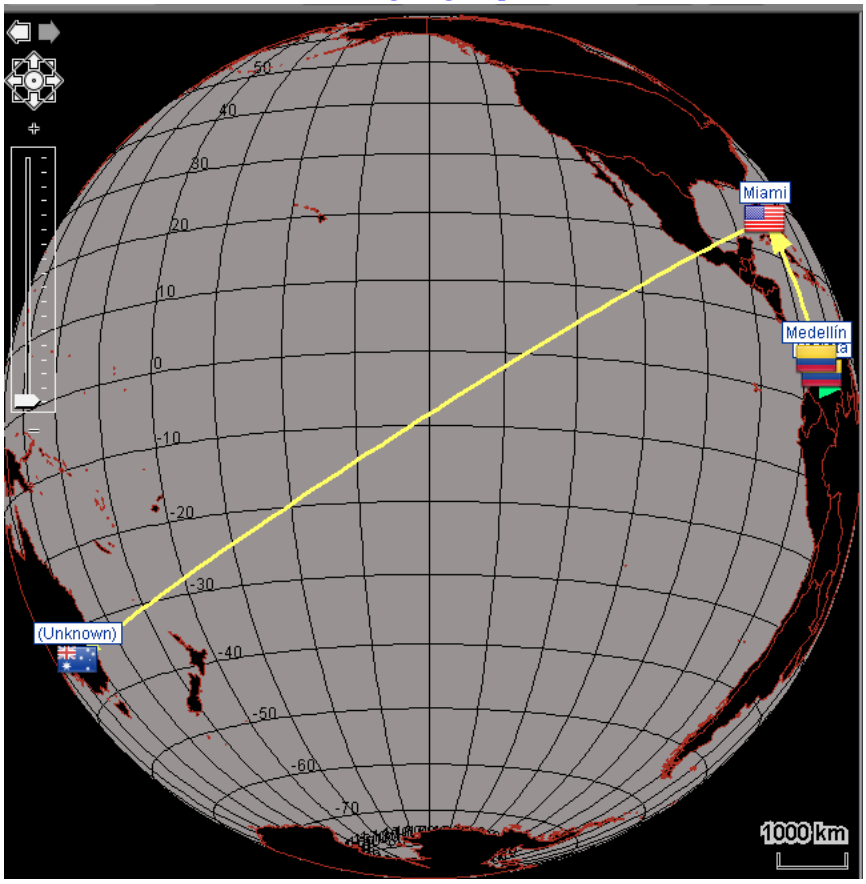


Figure 111 Route from Bogotá to Australia

#		Country	Town	Lat	Lon	IP	Hostname	L...	DN...	Di...	...
1		Colombia	Bogotá	4.6115	-74.0833	192.168.1.254	(None)	10	21	0	
2		*	*	4.6115	-74.0833	*	*	0	<1	0	
3		*	*	4.6115	-74.0833	10.166.12.58	(None)	16	21	0	
4		*	*	4.6115	-74.0833	10.166.12.57	(None)	19	20	0	
5		Colombia	Medellín	6.2529	-75.5646	200.24.35.183	static-adsl200-24-35...	32	19	245	
6		Colombia	Medellín	6.2529	-75.5646	200.24.33.90	static-adsl200-24-33...	77	23	0	
7		United Stat...	Miami	25.7634	-80.1886	152.195.89.2...	ae-112.border1.min....	83	66	22...	
8		United Stat...	Miami	25.7634	-80.1886	152.195.89.1...	po-67.core1.mid.edg...	75	38	0	
9		Australia	(Unknown)	-33.494	143.2104	117.18.238.2...	(None)	73	132	15...	

Figure 112 Packet route to Australia

- In Central America
    - www.grupoq.com

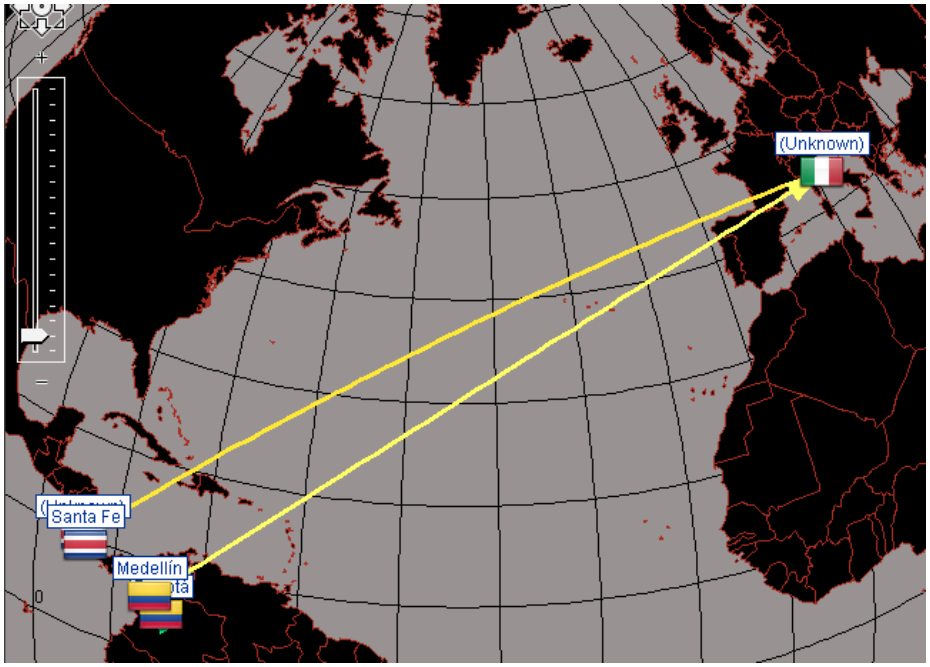


Figure 113 Route from Bogotá to Costa Rica

#		Country	Town	Lat	Lon	IP	Hostname	L...	DN...	Di...	...
2		*	*	4.6115	-74.0833	*	*	0	<1	0	
3		*	*	4.6115	-74.0833	10.166.12.58	(None)	15	20	0	
4		*	*	4.6115	-74.0833	10.166.12.57	(None)	19	20	0	
8		*	*	43.1479	12.1097	*	*	0	<1	0	
9		*	*	43.1479	12.1097	*	*	0	<1	0	
10		*	*	43.1479	12.1097	*	*	0	<1	0	
1		Colombia	Bogotá	4.6115	-74.0833	192.168.1.254	(None)	5	23	0	
5		Colombia	Medellín	6.2529	-75.5646	200.24.35.183	static-adsl200-24-35...	33	21	245	
6		Colombia	Medellín	6.2529	-75.5646	200.24.33.99	static-adsl200-24-33...	74	21	0	
11		Costa Rica	(Unknown)	10.0029	-84.0	200.107.83.1...	(None)	1...	20	97...	
12		Costa Rica	Santa Fe	9.0932	-83.5572	190.0.224.34	(None)	1...	121	112	
13		Costa Rica	Santa Fe	9.0932	-83.5572	190.0.224.62	(None)	1...	333	0	
14		Costa Rica	Santa Fe	9.0932	-83.5572	190.0.230.180	sitios.grupoq.co.cr	1...	27	0	
7		Italy	(Unknown)	43.1479	12.1097	79.140.83.82	(None)	76	609	93...	

Figure 114 Packet route to Costa Rica

- In North America

- www.renaultgroup.com

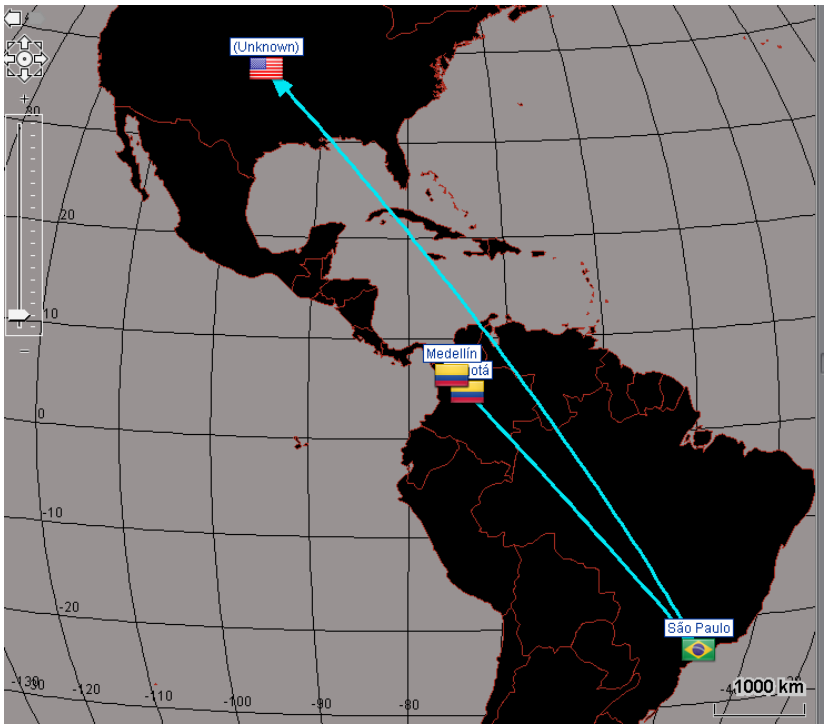


Figure 115 Route from Bogotá to the United States

..		Country	Town	Lat	Lon	IP	Hostname	L...	D...	Di...	...
1		Colombia	Bogotá	4.6115	-74.0833	192.168.1....	(None)	5	19	0	
2		*	*	4.6115	-74.0833	*	*	0	<1	0	
3		*	*	4.6115	-74.0833	*	*	0	<1	0	
4		*	*	4.6115	-74.0833	10.166.12....	(None)	16	19	0	
5		Colombia	Medellín	6.2529	-75.5646	200.24.35....	static-adsl200-2...	15	19	245	
6		*	*	6.2529	-75.5646	*	*	0	<1	0	
7		*	*	6.2529	-75.5646	*	*	0	<1	0	
8		*	*	6.2529	-75.5646	*	*	0	<1	0	
9		*	*	6.2529	-75.5646	*	*	0	<1	0	
...		Brazil	São Paulo	-23.5335	-46.6359	15.230.0.8	(None)	16	92	4...	
...		United St...	(Unknown)	37.751	-97.822	18.155.25...	server-18-155-2...	14	93	8...	

Figure 116 Packet route to United States

- In Asia
  - www.cheryinternational.com

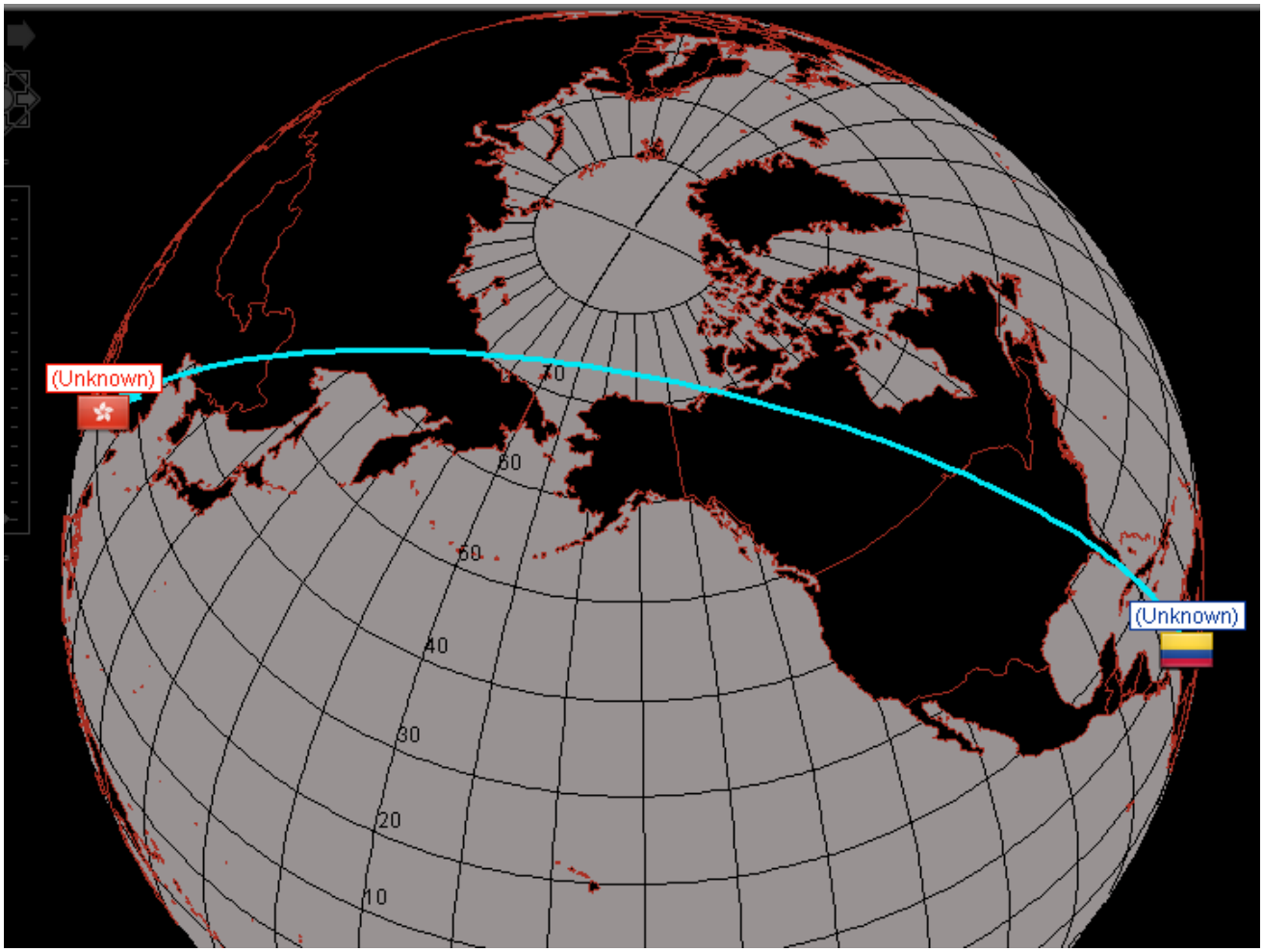


Figure 117 Route from Bogotá to Hong Kong

#		Country	Town	Lat	Lon	IP	Hostname	L...	DN...	Di...	...
1		Colombia	Bogotá	4.6115	-74.0833	192.168.1.254	(None)	11	31	0	
2		*	*	4.6115	-74.0833	*	*	0	<1	0	
3		*	*	4.6115	-74.0833	*	*	0	<1	0	
4		*	*	4.6115	-74.0833	10.166.12.57	(None)	23	35	0	
5		Colombia	(Unknown)	4.5981	-74.0799	200.25.30.146	ae3.0.edge1.bog3.as...	34	162	1	
6		Colombia	(Unknown)	4.5981	-74.0799	200.25.30.161	(None)	15	99	0	
7		Colombia	(Unknown)	4.5981	-74.0799	10.64.251.164	(None)	22	21	0	
8		*	*	4.5981	-74.0799	*	*	0	<1	0	
9		*	*	4.5981	-74.0799	10.64.57.102	(None)	24	26	0	
10		Hong Kong	(Unknown)	22.2578	114.1657	154.94.93.13	(None)	18	320	16...	

Figure 118 Packet route to Honk Kong

- In Europe
  - www.koenigsegg.com

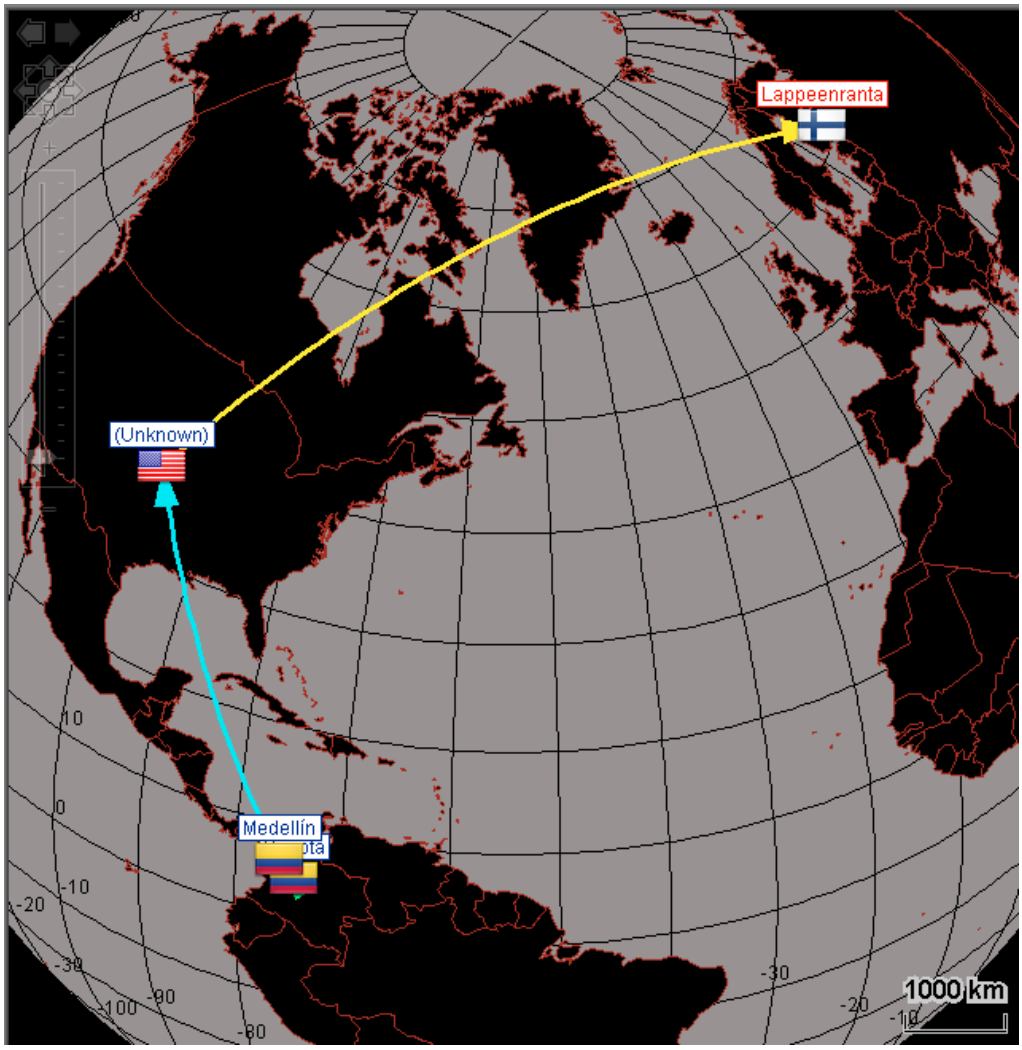


Figure 119 Route from Bogotá to Hong Kong

#		Country	Town	Lat	Lon	IP	Hostname	L...	DN...	Di...	...
1		Colombia	Bogotá	4.6115	-74.0833	192.168.1.254	(None)	12	26	0	
2		*	*	4.6115	-74.0833	*	*	0	<1	0	
3		*	*	4.6115	-74.0833	10.166.12.58	(None)	18	24	0	
4		*	*	4.6115	-74.0833	10.166.12.57	(None)	18	24	0	
5		Colombia	Medellín	6.2529	-75.5646	200.24.35.179	static-adsl200-24-35...	20	22	245	
6		United Stat...	(Unknown)	37.751	-97.822	209.85.168.1...	(None)	18	20	41...	
7		Finland	Lappeenra...	61.0636	28.189	34.88.92.18	18.92.88.34.bc.googl...	1...	107	80...	

Figure 120 Packet route to Finland

2. Some Questions About Router Commands

1. What is the difference between enable password and enable secret? If both are configured, whichone takes precedence?

    - **Enable Password**: This command sets a password to access privileged mode on a Cisco router. The password configured with enable password is stored in plaintext in

the configuration file, making it less secure.

- **Enable Secret**: This command also sets a password to access privileged mode, but the password configured with enable secret is stored encrypted in the configuration file, providing greater security.

- **Precedence**: If both commands are configured, enable secret takes precedence over enable password. This means that the router will use the password configured with enable secret for privileged mode authentication.

2. What is the difference between console and VTY?
   - **Console:** This is a direct physical connection to the router through the console port. This connection is used for initial device configuration, password recovery, and other administrative tasks requiring direct access. It does not require a network and is always available.

   - **VTY (Virtual Teletype):** These are virtual connections that allow remote access to the router through a network using protocols like Telnet or SSH. VTY lines are mainly used for remote administration. A configured and accessible network is required to use VTY connections.

3. What is the boot process of the routers in the Network Laboratory?

The boot process of a Cisco router generally follows these steps:

1. **Power-On Self-Test (POST)**: When the router is powered on, it runs a diagnostic test known as the POST. This test checks the hardware components, such as the CPU, memory, and interfaces, to ensure they are functioning correctly. The POST program is stored in the router's ROM (Read-Only Memory).

2. **Loading the Bootstrap Program**: After the POST, the router runs the bootstrap program, also stored in ROM. The bootstrap program's first task is to check the value of the configuration register, which dictates where to load the IOS (Internetwork Operating System). The default configuration register value (0x2102) instructs the router to load the IOS from the flash memory.

3. **Loading the IOS**: The bootstrap program locates the IOS image in the flash memory and loads it into the router's RAM (Random Access Memory). The IOS is the core operating system that provides the router's functionality.

4. **Initialization of Hardware and IOS**: Once the IOS is loaded, it initializes the router's hardware and prepares the system for operation. This includes setting up interfaces and applying initial settings.

5. **Loading the Startup Configuration**: The IOS then searches for the startup configuration file stored in NVRAM (Non-Volatile RAM). This file, known as the startup-config, contains the saved configuration settings for the router. If the startup-config file is found, it is loaded into RAM and becomes the running configuration (running-config). If the file is not present in NVRAM, the router attempts to load a configuration file from a TFTP (Trivial File Transfer Protocol) server. If no TFTP server is found, the router enters Setup mode, allowing basic

configuration.

6. **Entering User Mode**: After the startup configuration is loaded, the router finishes its boot process and provides access to the Command-Line Interface (CLI). The router starts in user mode, where basic monitoring commands can be executed. From here, administrators can enter privileged mode to perform more advanced configurations and management tasks.

4. What types of memory do the routers in the Network Laboratory have?

- **ROM (Read-Only Memory)**: ROM in the Cisco 1841 router stores the bootstrap program, POST (Power-On Self-Test) routines, and the ROM monitor. This non-volatile memory retains its contents even when the router is powered off, ensuring critical startup instructions are always available.

- **Flash Memory**: Flash memory holds the Cisco IOS (Internetwork Operating System) image and other system files. It is non-volatile, meaning it preserves data without power, and can be erased and reprogrammed to allow IOS updates.

- **NVRAM (Non-Volatile RAM)**: NVRAM stores the startup configuration file (startup-config). This type of memory is non-volatile, so it keeps the configuration settings intact even when the router is powered off or restarted.

- **RAM (Random Access Memory)**: RAM is used for the running configuration file (running-config), routing tables, and other operational data while the router is on. Unlike the other types, RAM is volatile, meaning it loses its data when the router is powered off or restarted.

5. What is the difference between startup-configuration and running-configuration files?

- **Startup-Configuration**: The startup-configuration file is stored in NVRAM (Non-Volatile RAM). This file contains the saved configuration of the router that is used when the device boots up or restarts. It retains its settings even when the router is powered off, ensuring that the router starts with the same configuration each time.
- **Running-Configuration**: The running-configuration file resides in the router's RAM (Random Access Memory). This file holds the current active configuration of the router, which is used during its operation. Any changes made to the router's configuration are first applied to the running-configuration, making them effective immediately. However, these changes are temporary and will be lost if the router is rebooted unless they are saved to the startup-configuration.

3. Setup: Access and Basic Configuration of Routers

Review and document the different routers available in the Laboratory and the network interfacesthey have.

We use the RS-232 serial cable extension and connect it to a computer, then connect it to the router

Figure 121 RS-232 serial cable extension

We open the Device Manager of the system and look for the port to which we are connected with router 10, in this case it is COM4. We select the serial connection option and change the serial line to the router's port.

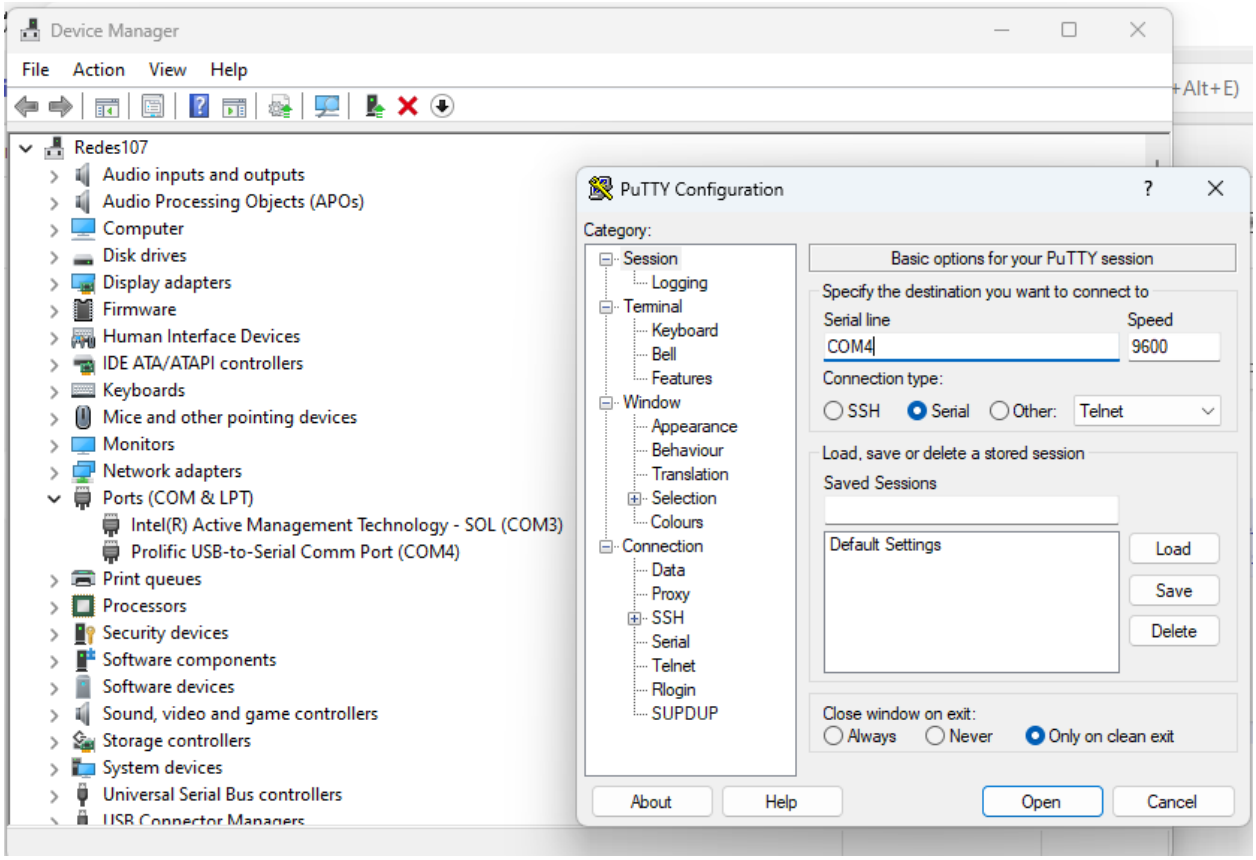


Figure 122 PuTTY configuration

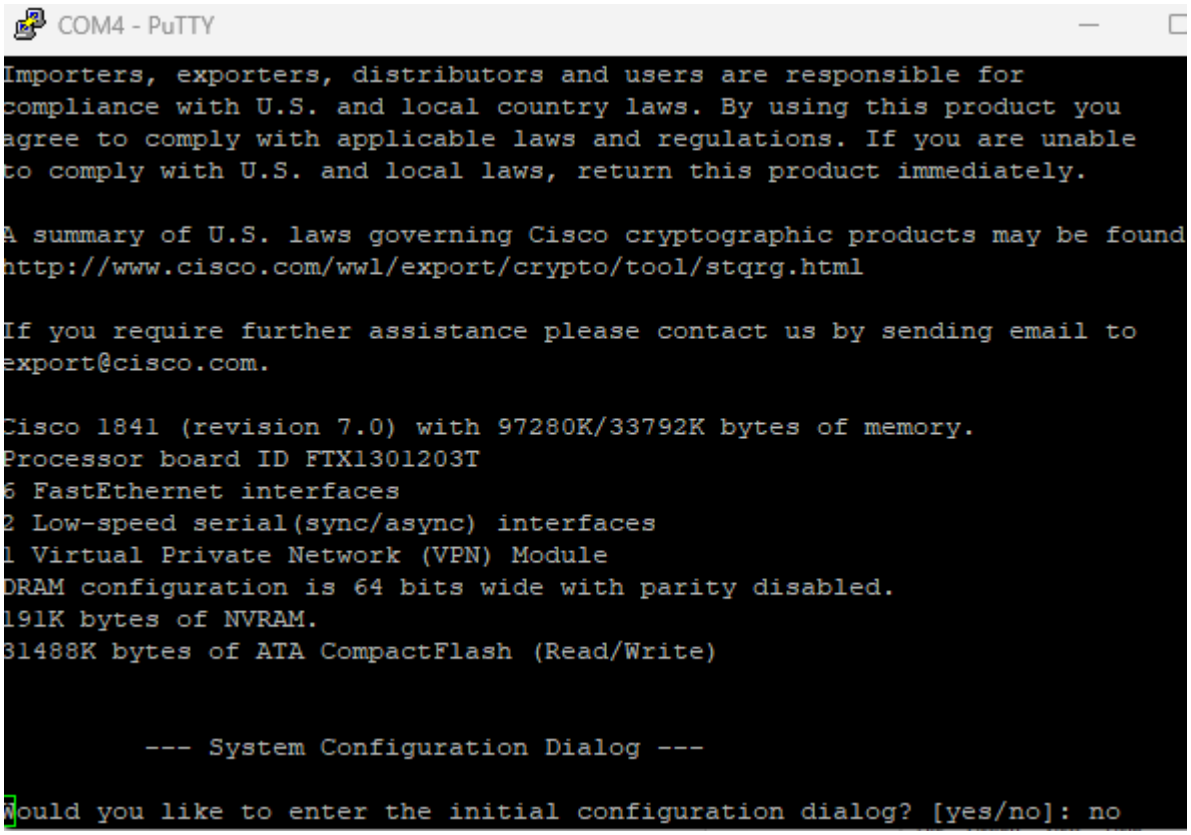


Figure 123 initial router configuration

We observe that there are no restrictions when accessing the router or entering privileged mode, so we configure it according to the specifications.

Now we are going to explain the process the router goes through on boot when configured in modes0x2142 and 0x2102.

Configuration Register 0x2142

- **Purpose**: Password recovery and troubleshooting.
- **Process**:
    1. Performs the POST to check hardware components.
    2. Loads the bootstrap program from ROM.
    3. Check the configuration register and detects the value 0x2142.
    4. Bypasses the startup-config in NVRAM, loading default settings.
    5. Loads the IOS from flash memory into RAM.
    6. Enters setup mode or CLI with factory default settings.

Configuration Register 0x2102

- **Purpose**: Normal router operation.
- **Process**:
    1. Performs the POST to check hardware components.
    2. Loads the bootstrap program from ROM.
    3. Checks the configuration register and detects the value 0x2102.
    4. Loads the IOS from flash memory into RAM.
    5. Loads the startup-config from NVRAM.
    6. Completes the boot process and presents the CLI in user mode with the saved configuration.

We perform the following configuration using physical devices. Which include:

- Access keys for privileged mode, console, and remote access. The privileged mode key shouldbe ”cisco”, the console key ”claveC”, and the remote access (telnet) key ”claveT”.
- Router name. Assign the router the last name of one of the students in the group.
- Console and remote access screen synchronization.
- Description of the interfaces used.
- Disable remote command server lookup.
- Message of the day.

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#hostname Torres
Torres(config)#banner motd "configuracion router8"
Torres(config)#line console 0
Torres(config-line)#logging synchronous
Torres(config-line)#password claveC
Torres(config-line)#login
Torres(config-line)#exit
Torres(config)#line vty 0 15
```

Figure 124 basic configuration

We configure interfaces. First, we set the IPs according to the given conditions. We need a network for 4000 hosts and another for 600 hosts. Based on the classroom distribution, we will manage the network 88.0.0.0/10.

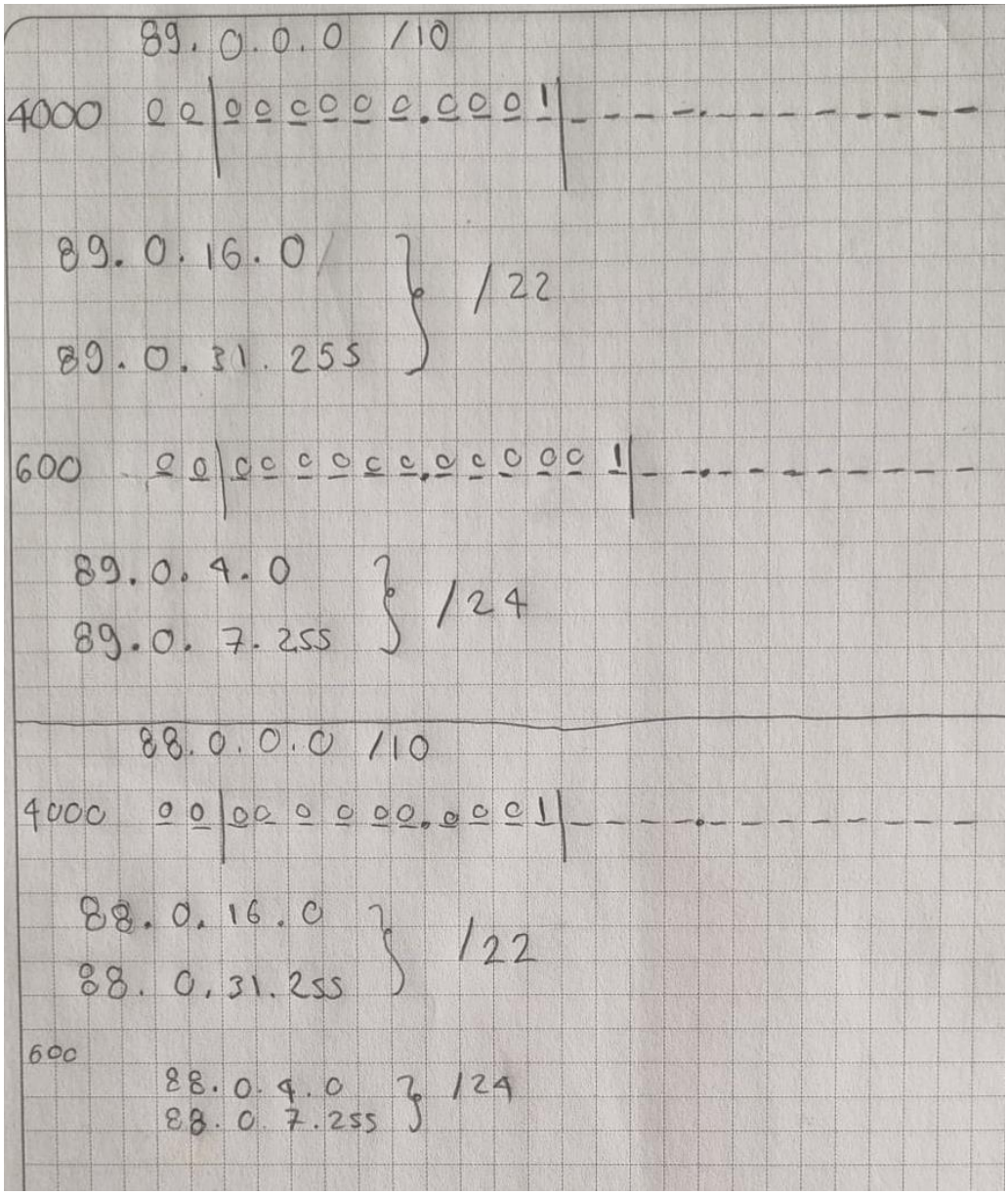
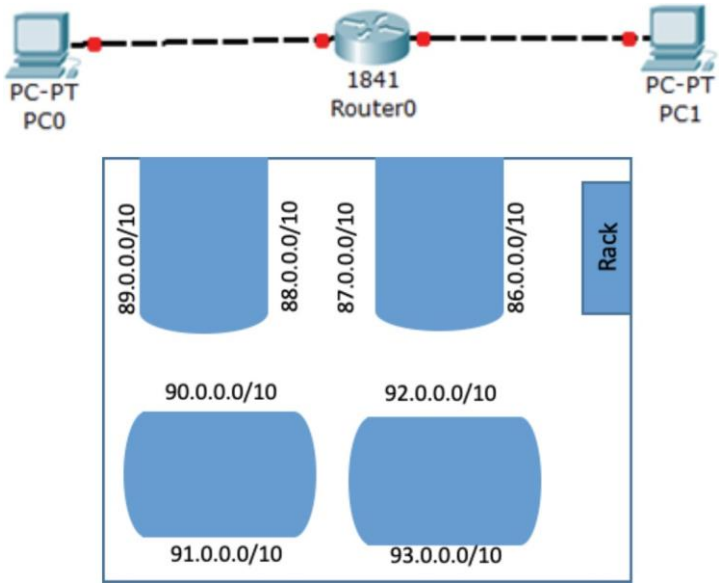


Figure 125 subnetting configuration

When performing the subnetting process, the subnets we will manage are as follows: 88.0.4.1/22 y 88.0.16.1/24



Now we configure the physical settings on the router to enable communication between the two computers.

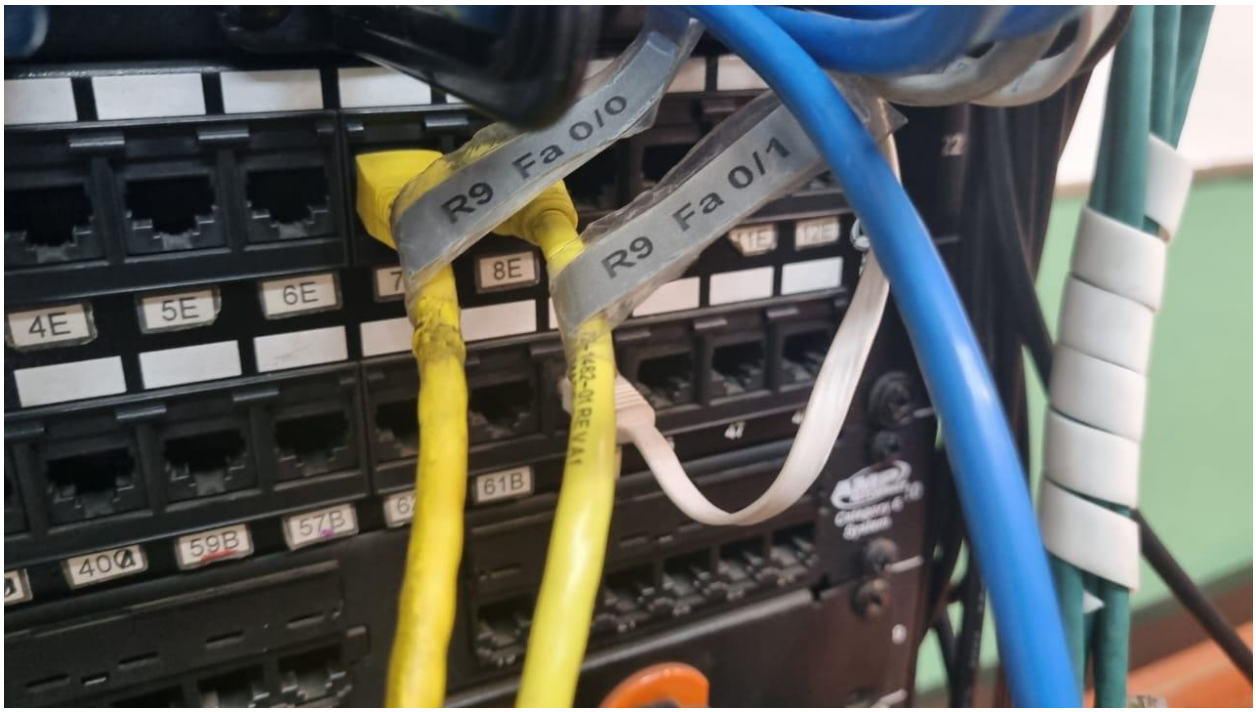


Figure 126 connection of devices

Figure 127 connection between devices and router

Figure 128 connection between devices and router

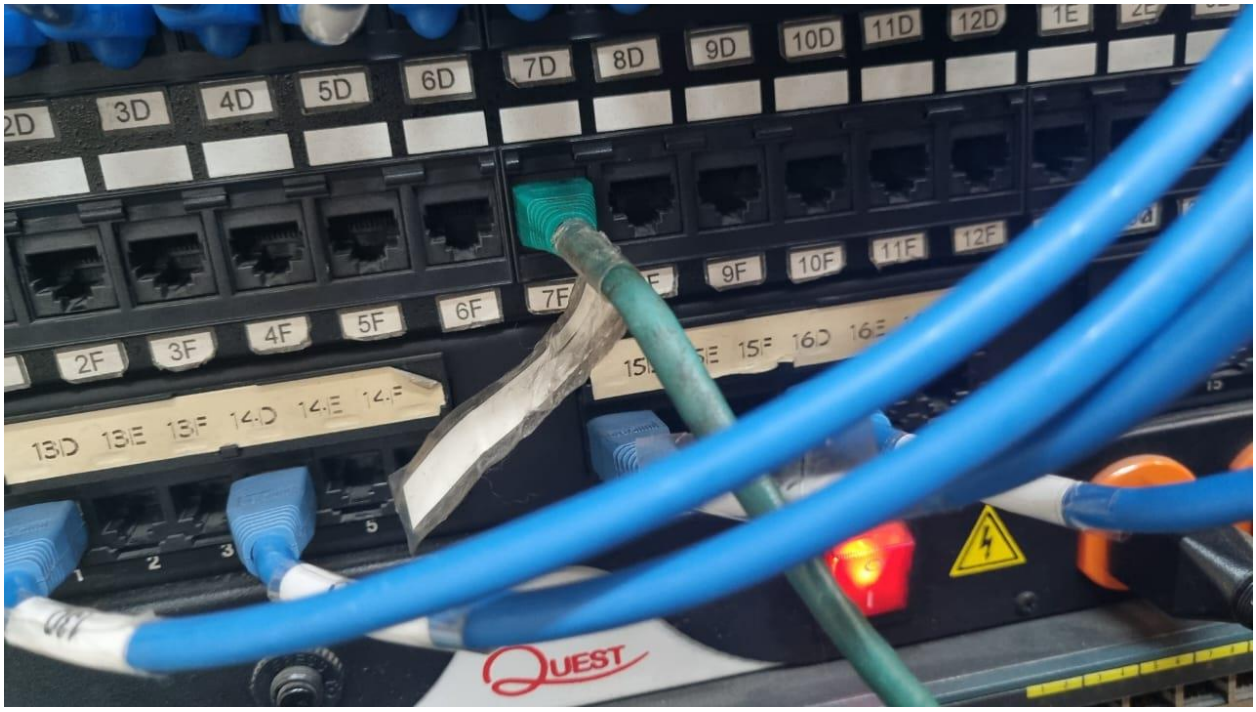


Figure 129 connection to console cable

Now we configure the logical settings on the router to enable communication between the two computers.

```
Torres(config)#interface fa0/0
Torres(config-if)#ip address 88.0.16.1 255.255.252.0
Torres(config-if)#description "router 8 fa0/0"
Torres(config-if)#no shutdown
Torres(config-if)#exit
Torres(config)#exit
```

Figure 130 configuration interface Fa0/

```
Torres(config)#interface fa0/1
Torres(config-if)#ip address 88.0.4.1 255.255.255.0
Torres(config-if)#description "router 8 fa0/1"
Torres(config-if)#no shutdown
Torres(config-if)#
```

Figure 131 configuration interface fa0/1

We see that the interfaces have been configured correctly with the command show ip interface brief.

```
Torres(config)#exit
Torres#confirgure terminal
*Jan  1 00:09:04.975: %SYS-5-CONFIG_I: Configured from console by console
Torres#show ip interface brief
Interface                  IP-Address      OK? Method Status                Protocol
FastEthernet0/0            88.0.16.1       YES manual up                    up
FastEthernet0/1            88.0.4.1        YES manual up                    up
Serial0/0/0                unassigned      YES unset  administratively down down
Serial0/0/1                unassigned      YES unset  administratively down down
Serial0/1/0                unassigned      YES unset  administratively down down
Serial0/1/1                unassigned      YES unset  administratively down down
Torres#
```

Figure 132 Verification of the configuration of the interfaces

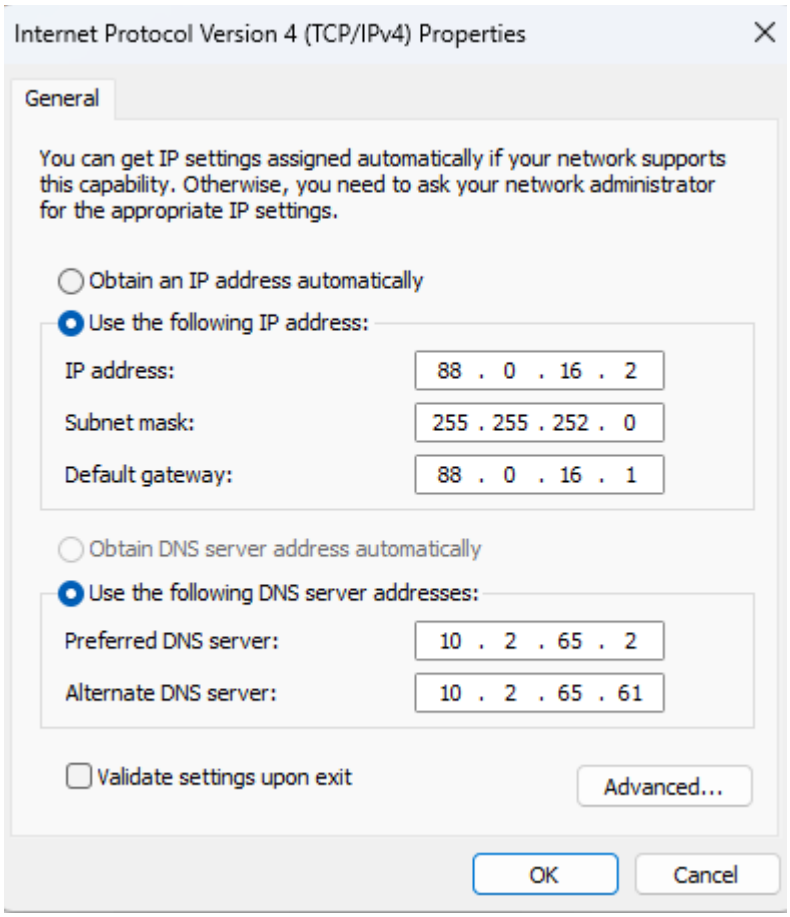


Figure 133 Configuration of IP and gateway on one of the devices

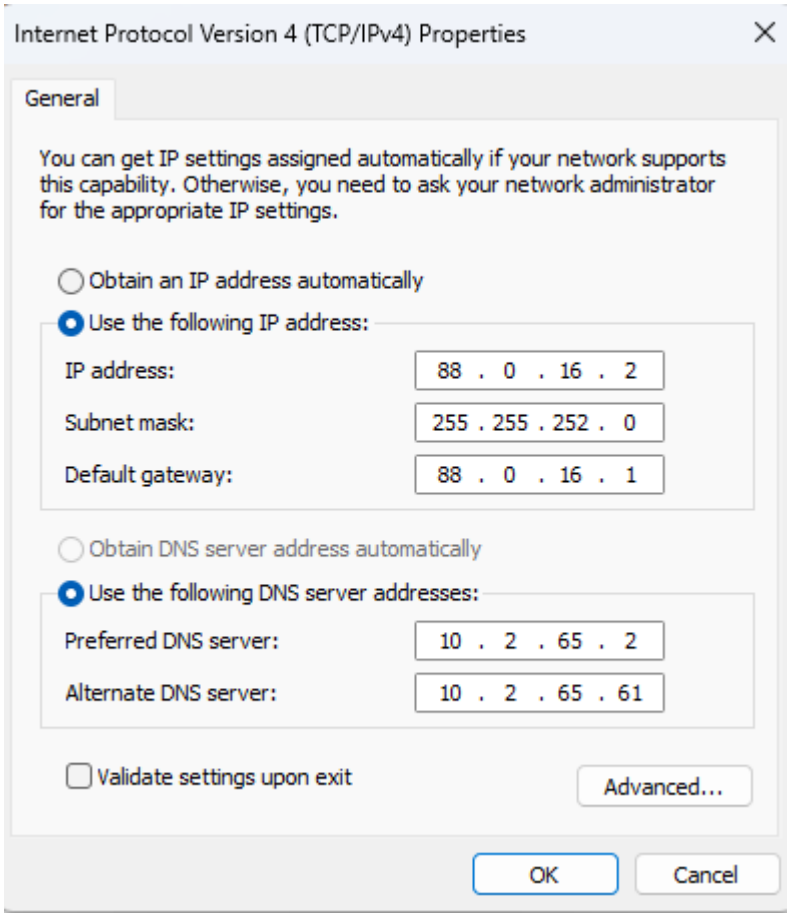


Figure 134 Configuration of IP and gateway on one of the devices

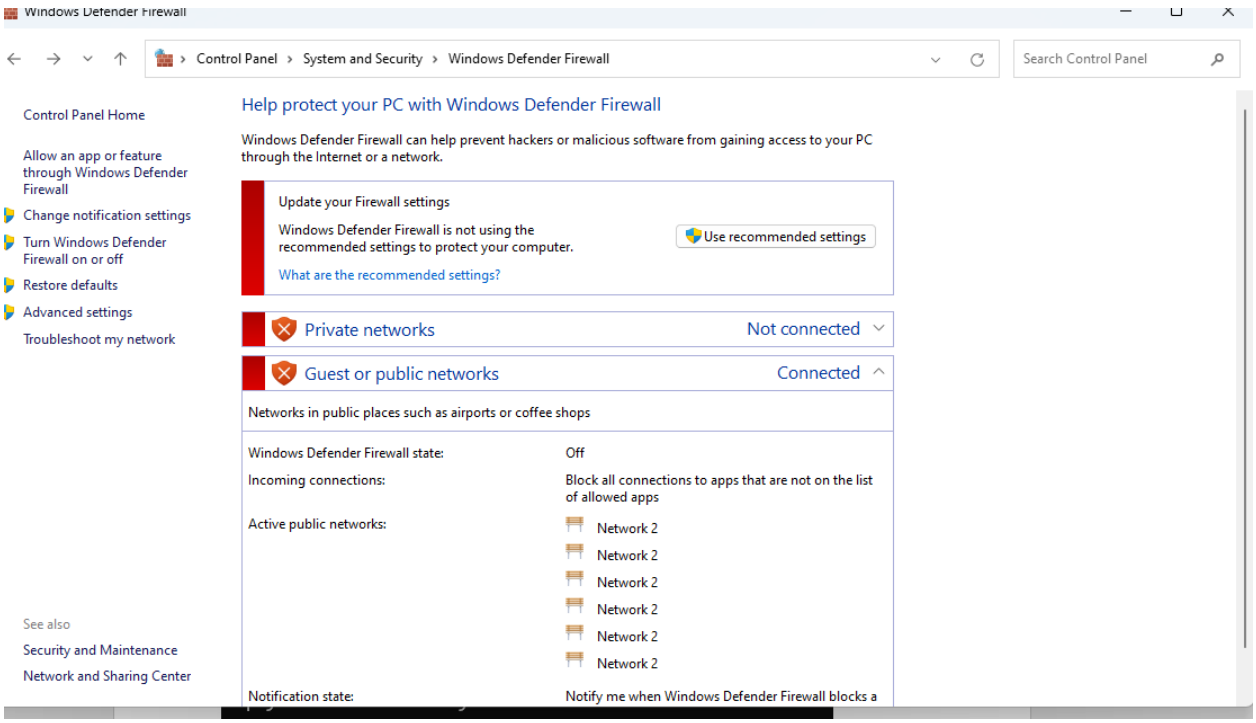


Figure 135 Disabling the firewall

We perform ping tests between the computers and the router's network interfaces.

```
Pinging 88.0.4.1 with 32 bytes of data:
Reply from 88.0.4.1: bytes=32 time=1ms TTL=255
Reply from 88.0.4.1: bytes=32 time=1ms TTL=255
Reply from 88.0.4.1: bytes=32 time=1ms TTL=255

Ping statistics for 88.0.4.1:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms
```

Figure 136 test to interface 88.0.4.1

```
C:\Users\Redes>ping 88.0.4.2

Pinging 88.0.4.2 with 32 bytes of data:
Reply from 88.0.4.2: bytes=32 time=1ms TTL=127
Reply from 88.0.4.2: bytes=32 time=1ms TTL=127
Reply from 88.0.4.2: bytes=32 time=1ms TTL=127
Reply from 88.0.4.2: bytes=32 time=1ms TTL=127

Ping statistics for 88.0.4.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms
```

Figure 137 test to device with Ip 88.0.4.2

```
C:\Users\Redes>ping 88.0.16.2

Pinging 88.0.16.2 with 32 bytes of data:
Reply from 88.0.16.2: bytes=32 time<1ms TTL=128
Reply from 88.0.16.2: bytes=32 time<1ms TTL=128
Reply from 88.0.16.2: bytes=32 time<1ms TTL=128
Reply from 88.0.16.2: bytes=32 time<1ms TTL=128

Ping statistics for 88.0.16.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Figure 138 test to device with Ip 88.0.16.0.2

4. Setup - Serial Interconnection

What is a null modem?

A **null modem** is a cable that allows direct communication between two devices, such as two computers or routers, without the need for a traditional modem. It is used to connect two serial ports by crossing the transmission and reception pins, allowing bidirectional data transmission between the devices. This type of connection is commonly used in networking to connect routers or devices that do not have Ethernet network ports available.

- **What is the clock rate command used for in routers, and why is it needed?**

  The **clock rate** command is used in routers to set the clock speed on serial interfaces, especially for **DCE (Data Communications Equipment)** connections. This command is necessary because, in a serial connection between two devices, one of them must generate the "clock" that controls the data transmission rate. The device configured as DCE is responsible for generating this clock, while the DTE (Data Terminal Equipment) device simply receives it. Without a clock rate, data transmission cannot occur properly.

- **What does DTE and DCE mean? What is the relationship with the routers in the Network Labo-ratory?**

  - **DTE (Data Terminal Equipment)**: Refers to a device that receives data or connects to the equipment generating the data. In terms of routers, the **DTE** device would be the router or computer that does not generate the clock signal and connects to a DCE device.
  - **DCE (Data Communications Equipment)**: The device that generates the clock signal in a serial connection. In the network lab, the **DCE** router is responsible for providing the clock signal via the serial interface.

We connect our router with the network device 89.0.0.0/10. We ensure that all devices connected to the router are on the 89.0.0.0 network.

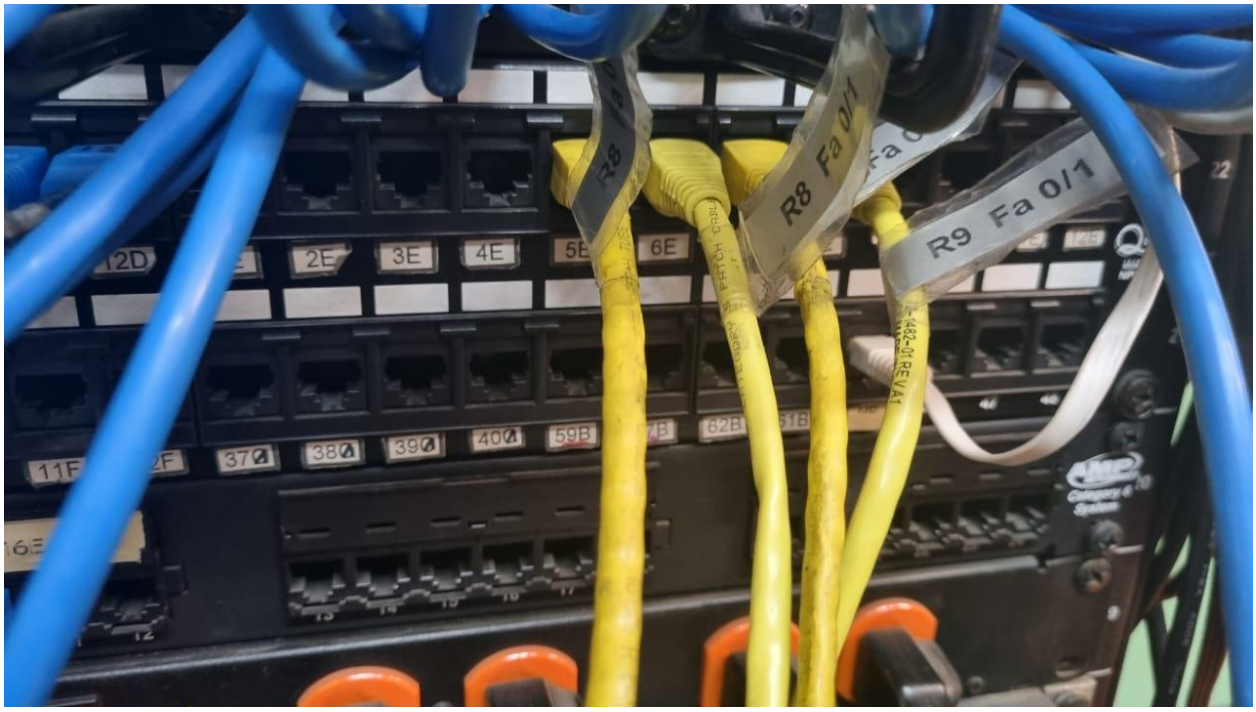


Figure 139 connection between devices

Figure 140 connection between devices an router

Now we connect the two routers using a serial cable.

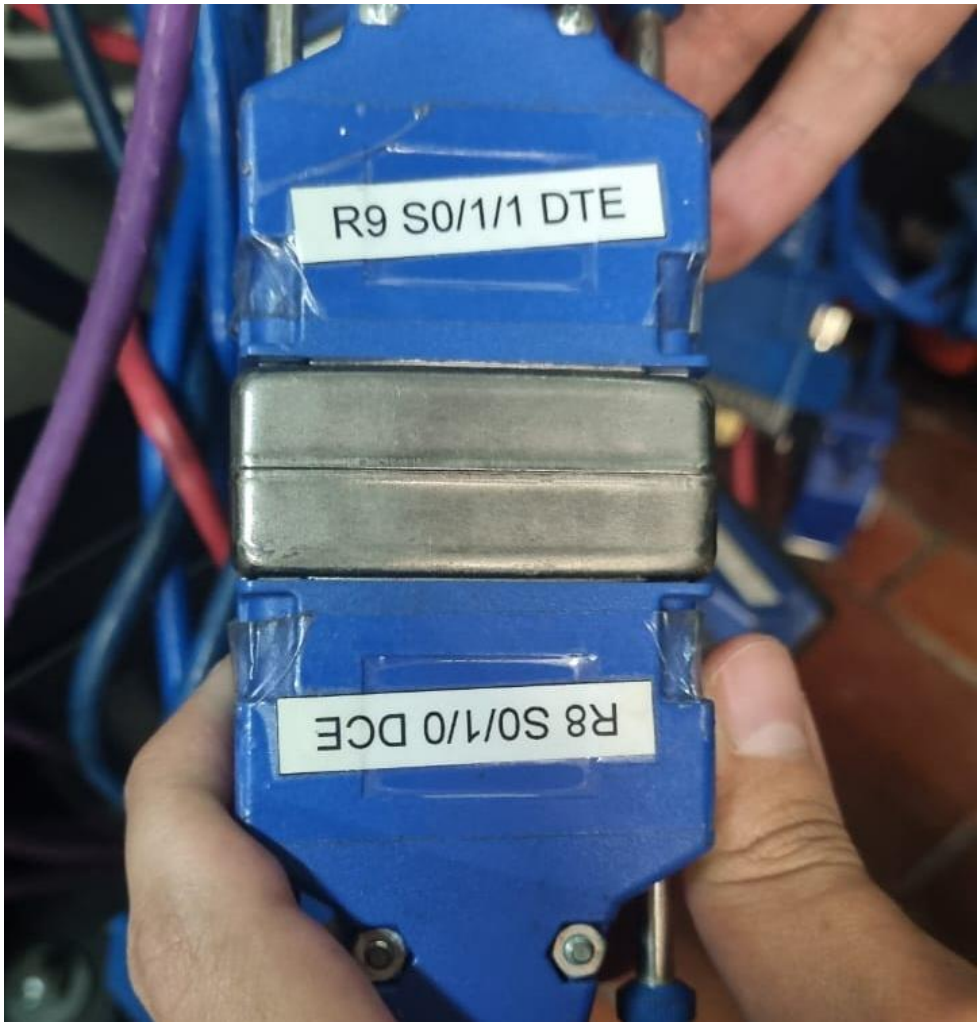


Figure 141 serial connection

We configure the network interfaces on each router.

Configuration of Router 8.

```
Torres#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Torres(config)#interface serial0/1/0
Torres(config-if)#ip address 100.0.0.1 255.255.255.0
Torres(config-if)#description "configuracion Router8"
Torres(config-if)#no shutdown
Torres(config-if)#
```

Configuration of Router 9

```
vargas(config)#interface s0/0/1
vargas(config-if)#ip address 100.0.0.2 255.255.255.0
vargas(config-if)#description "Configuration between Router5 and Router4"
vargas(config-if)#no shutdown
```

Figure 142 configuration interface s0/0/1

```
vargas#show ip interface brief

Serial0/0/1                100.0.0.2       YES SLARP  up                    up
```

Figure 143 verification connection interfaces

```
vargas#ping 100.0.0.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 100.0.0.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
```

Figure 144 test connection between routers Static Routing

We configure static routes on each router so that when sending packets, the router knows which network to send them to.

```
Torres(config)#ip route 89.0.16.0 255.255.252.0 100.0.0.2
Torres(config)#ip route 89.0.4.0 255.255.255.0 100.0.0.2
```

Figure 145 static routes router 8

```
jorge(config)#ip route 88.0.4.0 255.255.255.0 100.0.0.1
jorge(config)#ip route 88.0.16.0 255.255.255.0 100.0.0.1
```

Figure 146 static routes router2

```
C:\Users\Redes>tracert 88.0.16.2

Tracing route to 88.0.16.2 over a maximum of 30 hops

  1    <1 ms    <1 ms    <1 ms  89.0.4.1
  2     1 ms     1 ms     1 ms  100.0.0.1
  3     2 ms     2 ms     2 ms  88.0.16.2

Trace complete.
```

Figure 147 trace route between devices

5. Dynamic Routing

Now we will proceed with dynamic routing. To configure the routers for dynamic routing, we need to set up the routers and their respective interfaces, like what we did in the static routing section. However, instead of applying static configuration, we will use the configuration for the RIP routing algorithm first, and then OSPF.

Initial configuration

We will start by selecting Router Two for configuration and then enter the initial settings for the router.

```
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#hostname Torres
Torres(config)#line console 0
Torres(config-line)#logging synchronous
Torres(config-line)#password claveC
Torres(config-line)#login
Torres(config-line)#exit
Torres(config)#line vty 0 15
```

Figure 148 Initial configuration

Now we configure the router interfaces using the commands mentioned earlier. We will use the following IP addresses: 104.0.0.2/8 and 101.0.0.2/8 for the connection between the routers, and 88.0.0.1/8 for the connection to a computer to test the network using ICMP commands.

```
Torres(config)#interface serial0/2/0
Torres(config-if)#ip address 104.0.0.2 255.0.0.0
Torres(config-if)#no shutdown
```

Figure 149 interface configuration

```
Interface                  IP-Address      OK? Method Status                Protocol
FastEthernet0/0            88.0.0.1        YES manual up                    up

Serial0/2/0                104.0.0.2       YES manual up                    up

Serial0/2/1                unassigned      YES unset  administratively down down

Serial0/3/0                unassigned      YES unset  administratively down down

Serial0/3/1                101.0.0.2       YES manual up                    up
```

Figure 150 configuration of interfaces of router2

Then, we verify that the physical connections are properly connected and check that the firewall is disabled.

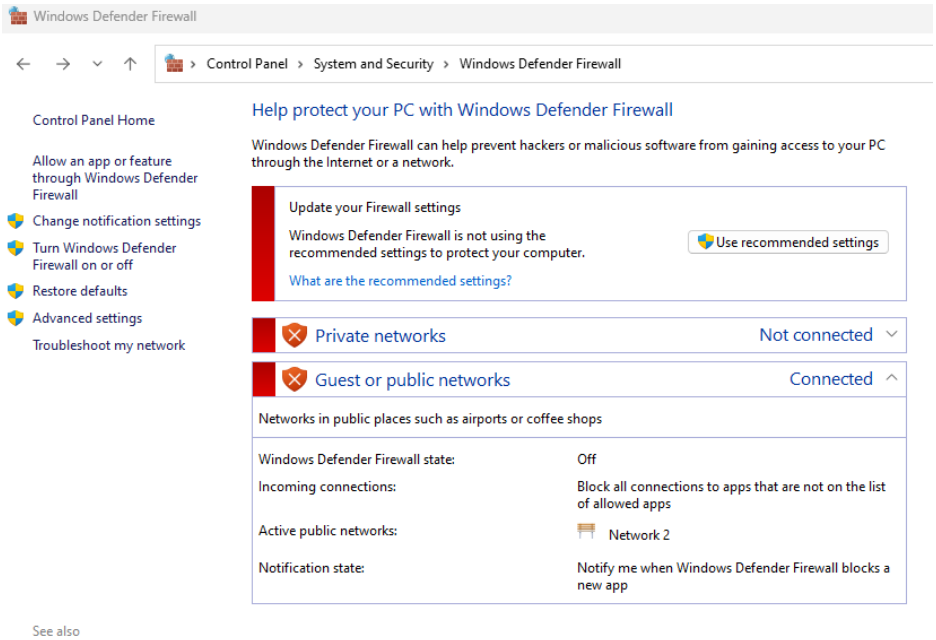


Figure 151 Firewall deactivation

Configuration of the RIP Algorithm

We use the command router rip to start the configuration of the routing protocol. Then, we specify the version with command version 2. Next, we define the local networks, or networks that are directly connected to the router, which should be advertised by RIP. This is done with one line for each network. In this case, the networks are 101.0.0.0 and 88.0.0.0.

```
Torres(config)#router rip
Torres(config-router)#version 2
Torres(config-router)#network 101.0.0.0
Torres(config-router)#network 88.0.0.0
Torres(config-router)#exit
Torres(config)#exit
```

Figure 152 RIP configuration

Now we verify that it has been configured correctly with the command ip route. As we can see in the following image, the configuration of the RIP algorithm was done correctly.

```
Torres#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

R    100.0.0.0/8 [120/1] via 101.0.0.1, 00:00:18, Serial0/3/1
C    101.0.0.0/8 is directly connected, Serial0/3/1
R    86.0.0.0/8 [120/1] via 101.0.0.1, 00:00:18, Serial0/3/1
     88.0.0.0/22 is subnetted, 1 subnets
C       88.0.0.0 is directly connected, FastEthernet0/0
R    91.0.0.0/8 [120/2] via 101.0.0.1, 00:00:18, Serial0/3/1
```

Figure 153 Ip route configuration

We run tests using the ping and tracert commands to check the connectivity between different devices on the network.

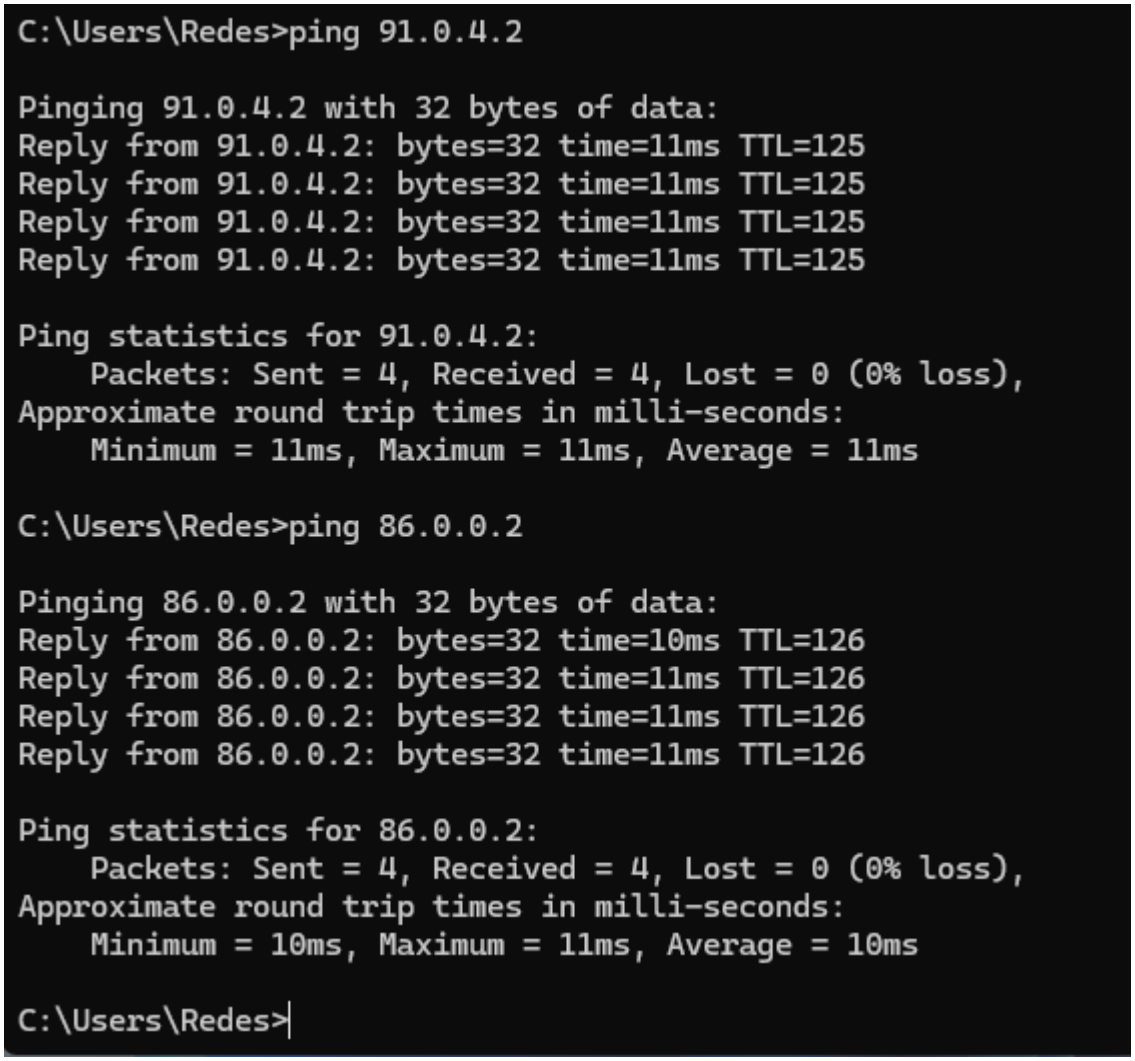


Figure 154 test ping

```
C:\Users\Redes>tracert 86.0.0.2

Tracing route to 86.0.0.2 over a maximum of 30 hops

  1     1 ms     1 ms     1 ms  88.0.0.1
  2    12 ms    12 ms    11 ms  101.0.0.1
  3    14 ms    15 ms    14 ms  86.0.0.2

Trace complete.

C:\Users\Redes>tracert 91.0.4.2

Tracing route to 91.0.4.2 over a maximum of 30 hops

  1     1 ms     1 ms     1 ms  88.0.0.1
  2    11 ms    11 ms    11 ms  101.0.0.1
  3    13 ms    12 ms    12 ms  100.0.0.2
  4    15 ms    15 ms    15 ms  91.0.4.2

Trace complete.
```

Figure 155 test tracert

We capture the information of the packets sent in the previous tests for analysis using command rip. Then we save the capture in file rip.pcapng

rip

No.	Time	Source	Destination	Protocol	Length	Info
17	11.269992	88.0.0.1	224.0.0.9	RIPv2	126	Response
91	38.486874	88.0.0.1	224.0.0.9	RIPv2	126	Response
156	68.047367	88.0.0.1	224.0.0.9	RIPv2	126	Response
234	95.935979	88.0.0.1	224.0.0.9	RIPv2	126	Response
297	125.676258	88.0.0.1	224.0.0.9	RIPv2	126	Response
348	153.088765	88.0.0.1	224.0.0.9	RIPv2	126	Response
388	181.929713	88.0.0.1	224.0.0.9	RIPv2	126	Response
419	207.841959	88.0.0.1	224.0.0.9	RIPv2	126	Response
452	234.794514	88.0.0.1	224.0.0.9	RIPv2	126	Response
481	261.287055	88.0.0.1	224.0.0.9	RIPv2	126	Response
517	290.923985	88.0.0.1	224.0.0.9	RIPv2	126	Response

Figure 156 RIP packets

When executing the RIP command, we can observe several packets that are response messages from RIPv2. These packets are sent by router with IP address 88.0.0.1 to the multicast address 224.0.0.9. The purpose of these messages is to share the router's routing table with other routers on the network. This process allows routers to dynamically update and maintain their routing tables, ensuring that the routes are efficient and always up to date.

Ping

Filter: icmp.type in {8, 0} && ip.ttl > 30 && !(icmp.type == 11)

- icmp.type in {8, 0}: Filters the request (ping) and reply (pong) packets.
- ip.ttl > 30: Filters packets with a TTL greater than 30, since ping packets usually have a higher TTL. This helps filter out packets that have made few hops and arrived correctly at their destination.
- !(icmp.type == 11): Excludes packets with type 11 (Time Exceeded), which indicate routing errors.

icmp.type in {8, 0} && ip.ttl > 30 && !(icmp.type == 11)

No.	Time	Source	Destination	Protocol	Length	Info
128	52.098800	91.0.4.2	88.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=139/35584, ttl=125 (request in 127)
130	52.115382	91.0.4.2	88.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=140/35840, ttl=125 (request in 129)
132	52.131109	91.0.4.2	88.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=141/36096, ttl=125 (request in 131)
255	103.913581	86.0.0.2	88.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=148/37888, ttl=126 (request in 254)
257	103.929079	86.0.0.2	88.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=149/38144, ttl=126 (request in 256)
259	103.946056	86.0.0.2	88.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=150/38400, ttl=126 (request in 258)
835	508.655222	86.0.0.2	88.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=80/20480, ttl=126 (reply in 836)
836	508.655491	88.0.0.2	86.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=80/20480, ttl=128 (request in 835)
839	509.658516	86.0.0.2	88.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=81/20736, ttl=126 (reply in 840)
840	509.658660	88.0.0.2	86.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=81/20736, ttl=128 (request in 839)
841	510.675444	86.0.0.2	88.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=82/20992, ttl=126 (reply in 842)
842	510.675596	88.0.0.2	86.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=82/20992, ttl=128 (request in 841)
848	511.693018	86.0.0.2	88.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=83/21248, ttl=126 (reply in 849)
849	511.693174	88.0.0.2	86.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=83/21248, ttl=128 (request in 848)
893	548.099347	91.0.4.2	88.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=162/41472, ttl=125 (reply in 894)
894	548.099498	88.0.0.2	91.0.4.2	ICMP	74	Echo (ping) reply id=0x0001, seq=162/41472, ttl=128 (request in 893)
895	549.113640	91.0.4.2	88.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=163/41728, ttl=125 (reply in 896)
896	549.113774	88.0.0.2	91.0.4.2	ICMP	74	Echo (ping) reply id=0x0001, seq=163/41728, ttl=128 (request in 895)
900	550.123500	91.0.4.2	88.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=164/41984, ttl=125 (reply in 901)
901	550.123683	88.0.0.2	91.0.4.2	ICMP	74	Echo (ping) reply id=0x0001, seq=164/41984, ttl=128 (request in 900)
902	551.138111	91.0.4.2	88.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=165/42240, ttl=125 (reply in 903)
903	551.138279	88.0.0.2	91.0.4.2	ICMP	74	Echo (ping) reply id=0x0001, seq=165/42240, ttl=128 (request in 902)
962	600.684537	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=90/23040, ttl=128 (request in 961)
964	600.700374	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=91/23296, ttl=128 (request in 963)
966	600.717081	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=92/23552, ttl=128 (request in 965)
1157	705.966601	86.0.0.2	88.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=93/23808, ttl=126 (reply in 1158)
1158	705.966758	88.0.0.2	86.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=93/23808, ttl=128 (request in 1157)
1159	706.977799	86.0.0.2	88.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=94/24064, ttl=126 (reply in 1160)
1160	706.977939	88.0.0.2	86.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=94/24064, ttl=128 (request in 1159)
1164	707.994132	86.0.0.2	88.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=95/24320, ttl=126 (reply in 1165)
1165	707.994316	88.0.0.2	86.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=95/24320, ttl=128 (request in 1164)
1166	709.011352	86.0.0.2	88.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=96/24576, ttl=126 (reply in 1167)
1167	709.011515	88.0.0.2	86.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=96/24576, ttl=128 (request in 1166)
1250	773.588608	88.0.0.2	91.0.4.2	ICMP	106	Echo (ping) reply id=0x0001, seq=184/47104, ttl=128 (request in 1249)
1252	773.604620	88.0.0.2	91.0.4.2	ICMP	106	Echo (ping) reply id=0x0001, seq=185/47360, ttl=128 (request in 1251)
1254	773.620369	88.0.0.2	91.0.4.2	ICMP	106	Echo (ping) reply id=0x0001, seq=186/47616, ttl=128 (request in 1253)

Figure 157 Packets sent during ping execution

Tracert

Filter: icmp.type in {8, 0} && ip.ttl <= 30 && !(icmp.type == 11)

- icmp.type in {8, 0}: Filters the request (ping) and reply (pong) packets.
- ip.ttl <= 30: Filters packets with a TTL (Time to Live) less than or equal to 30, as tracert sends packets with TTL that increase with each hop, starting from a low value. This filters tracert packets that have not made too many hops.
- !(icmp.type == 11): Excludes packets with type 11 (Time Exceeded), which indicate routing errors, such as when a packet has not reached its destination due to an expired TTL.

icmp.type in {8, 0} && ip.ttl<=30 && !(icmp.type == 11)

No.	Time	Source	Destination	Protocol	Length	Info
53	31.232913	88.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=130/33280, ttl=1 (no response found!)
55	31.234920	88.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=131/33536, ttl=1 (no response found!)
57	31.236167	88.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=132/33792, ttl=1 (no response found!)
80	37.166476	88.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=133/34048, ttl=2 (no response found!)
82	37.179467	88.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=134/34304, ttl=2 (no response found!)
84	37.194570	88.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=135/34560, ttl=2 (no response found!)
114	50.628745	88.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=136/34816, ttl=3 (no response found!)
116	50.643781	88.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=137/35072, ttl=3 (no response found!)
118	50.657868	88.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=138/35328, ttl=3 (no response found!)
127	52.083460	88.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=139/35584, ttl=4 (reply in 128)
129	52.100563	88.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=140/35840, ttl=4 (reply in 130)
131	52.116380	88.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=141/36096, ttl=4 (reply in 132)
184	84.500250	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=142/36352, ttl=1 (no response found!)
186	84.502089	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=143/36608, ttl=1 (no response found!)
188	84.503718	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=144/36864, ttl=1 (no response found!)
211	90.432929	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=145/37120, ttl=2 (no response found!)
213	90.444884	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=146/37376, ttl=2 (no response found!)
215	90.456598	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=147/37632, ttl=2 (no response found!)
254	103.898599	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=148/37888, ttl=3 (reply in 255)
256	103.914464	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=149/38144, ttl=3 (reply in 257)
258	103.931460	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=150/38400, ttl=3 (reply in 259)
961	600.684344	86.0.0.2	88.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=90/23040, ttl=1 (reply in 962)
963	600.700160	86.0.0.2	88.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=91/23296, ttl=1 (reply in 964)
965	600.716836	86.0.0.2	88.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=92/23552, ttl=1 (reply in 966)
1249	773.588453	91.0.4.2	88.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=184/47104, ttl=1 (reply in 1250)
1251	773.604463	91.0.4.2	88.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=185/47360, ttl=1 (reply in 1252)
1253	773.620227	91.0.4.2	88.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=186/47616, ttl=1 (reply in 1254)
3007	3592.822437	91.0.4.2	88.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=227/58112, ttl=1 (reply in 3008)
3009	3592.839975	91.0.4.2	88.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=228/58368, ttl=1 (reply in 3010)
3011	3592.857030	91.0.4.2	88.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=229/58624, ttl=1 (reply in 3012)
3044	3602.783079	86.0.0.2	88.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=132/33792, ttl=1 (reply in 3045)
3046	3602.798877	86.0.0.2	88.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=133/34048, ttl=1 (reply in 3047)

Figure 158 Packets sent during tracert execution

Configuration of the OSPF Algorithm

We deleted the previously set RIP configuration with the command no ip router rip.

```
Torres(config)#no router rip
```

Figure 159 Deletion of RIP algorithm

```
Torres(config)#router ospf 1
Torres(config-router)#network 88.0.0.0 0.0.15.255 area 0
Torres(config-router)#network 101.0.0.0 0.255.255.255 area 0
Torres(config-router)#network 101.0.0.0 0.255.255.255 area 0
```

Figure 160 OSPF configuration

- **ospf 1**: This command starts the OSPF process on the router with process ID 1, allowing it to manage routes using OSPF. The number 1 is arbitrary and can be replaced with any value between 1 and 65535, but 1 is commonly used by default.
- **network 88.0.0.0 0.0.15.255 area 0:** This command advertises the network 88.0.0.0/20 in OSPF. The wildcard mask 0.0.15.255 indicates that any IP address in the range 88.0.0.0 - 88.15.255.255 will be included. This network belongs to area 0, which is the backbone area in OSPF.
- **network 101.0.0.0 0.255.255.255 area 0**: This command advertises the network 101.0.0.0/8 in OSPF. The wildcard mask 0.255.255.255 allows including all addresses in the range 101.0.0.0 - 101.255.255.255. It also belongs to area 0.

We perform the test by executing the ping and tracert commands to several devices on the network.

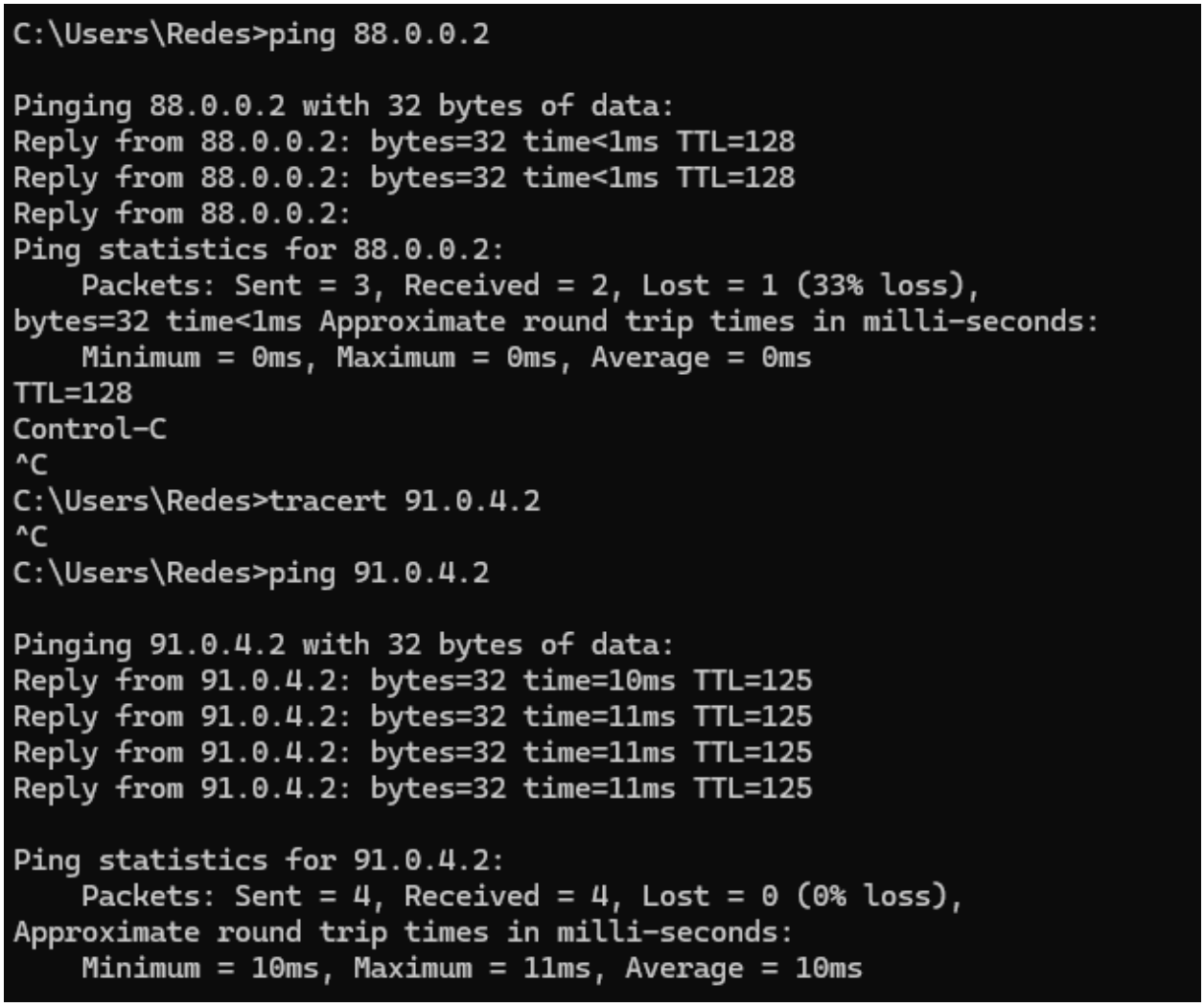


Figure 161 test ping

```
C:\Users\Redes>tracert 86.0.0.2

Tracing route to 86.0.0.2 over a maximum of 30 hops

  1    <1 ms    <1 ms    <1 ms  88.0.0.1
  2    11 ms    11 ms    11 ms  101.0.0.1
  3    14 ms    14 ms    14 ms  86.0.0.2

Trace complete.

C:\Users\Redes>tracert 86.0.0.2

Tracing route to 86.0.0.2 over a maximum of 30 hops

  1    <1 ms    <1 ms    <1 ms  88.0.0.1
  2    11 ms    11 ms    11 ms  101.0.0.1
  3    14 ms    14 ms    14 ms  86.0.0.2

Trace complete.
```

Figure 162 test tracert

We capture the information of the packets sent in the previous tests for analysis using command rip. Then we save the capture in files ospf1.pcapng and ospf2.pcapng

ospf

No.	Time	Source	Destination	Protocol	Length	Info
11	5.620105	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
31	15.620305	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
42	25.620511	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
55	35.620426	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
71	45.620891	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
87	55.620976	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
98	65.621331	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
111	75.621563	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
121	85.621761	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
136	95.621662	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
155	105.622144	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
183	115.622135	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
205	125.622417	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
217	135.622492	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
233	145.622974	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
244	155.623132	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
257	165.623367	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
266	175.623576	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
282	185.623806	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
293	195.623973	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
306	205.623814	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
316	215.624394	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
330	225.624568	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet
373	235.624817	88.0.0.1	224.0.0.5	OSPF	90	Hello Packet

Figure 163 OSPF packets

The packets shown by the filter are OSPF "Hello" messages sent from router 2 with IP address

88.0.0.1 to the multicast address 224.0.0.5, used by OSPF to communicate with all OSPF routers on the same network. These messages are used to discover and maintain relationships with other routers, identify neighbors on the same network link, verify if they are active and operational, and facilitate the exchange of information to build and update the routing table.

ospf

No.	Time	Source	Destination	Protocol	Length	Info
50	8.349285	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
98	18.349226	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
162	28.349553	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
204	38.349458	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
243	48.349539	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
275	58.349935	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
322	68.349753	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
344	78.349925	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
370	88.349924	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
395	98.350126	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
441	108.350434	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
489	118.350578	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
547	128.350763	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
567	138.350618	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
623	148.350712	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
666	158.351255	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
739	168.351254	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
767	178.351241	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
796	188.351168	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
829	198.351485	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
870	208.351407	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
914	218.351522	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
936	228.351826	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
975	238.351744	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
1027	248.351878	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
1073	258.351955	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
1122	268.352096	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
1177	278.352205	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
1235	288.352368	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
1300	298.352416	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
1355	308.352905	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
1375	318.352673	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet
1400	328.352937	92.0.0.1	224.0.0.5	OSPF	90	Hello Packet

Figure 164 OSPF packets

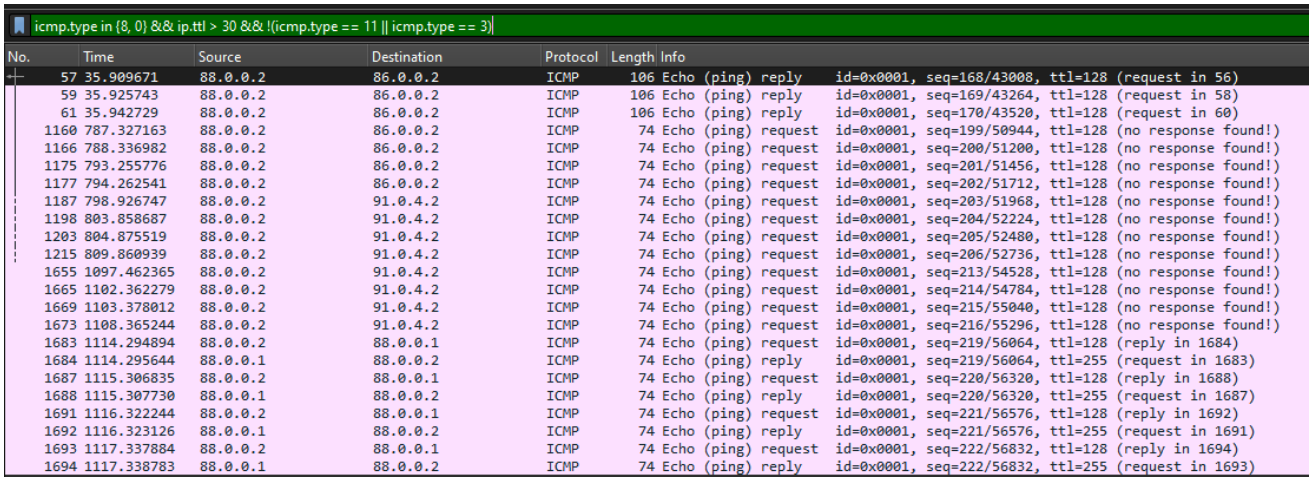
92.0.0.1 to the multicast address 224.0.0.5, used by OSPF to communicate with all OSPF routers on the same network. These messages are used to discover and maintain relationships with other routers, identify neighbors on the same network link, verify if they are active and operational, and facilitate the exchange of information to build and update the routing table

Ping

Filter: icmp.type in {8, 0} && ip.ttl > 30 && !(icmp.type == 11 || icmp.type == 3)

- **icmp.type in {8, 0}:** Filters ICMP packets of type 8 (echo request, ping) and type 0 (echo reply, pong).
- **ip.ttl > 30**: Filters packets with a TTL (Time To Live) greater than 30. This is done because most pings usually exceed this TTL, while traceroute packets typically have a low TTL (below 30), helping to separate pings from traceroutes.
- **!(icmp.type == 11 || icmp.type == 3**): Excludes ICMP packets of type 11 (Time Exceeded, which indicates a packet has expired) and type 3 (Destination Unreachable,

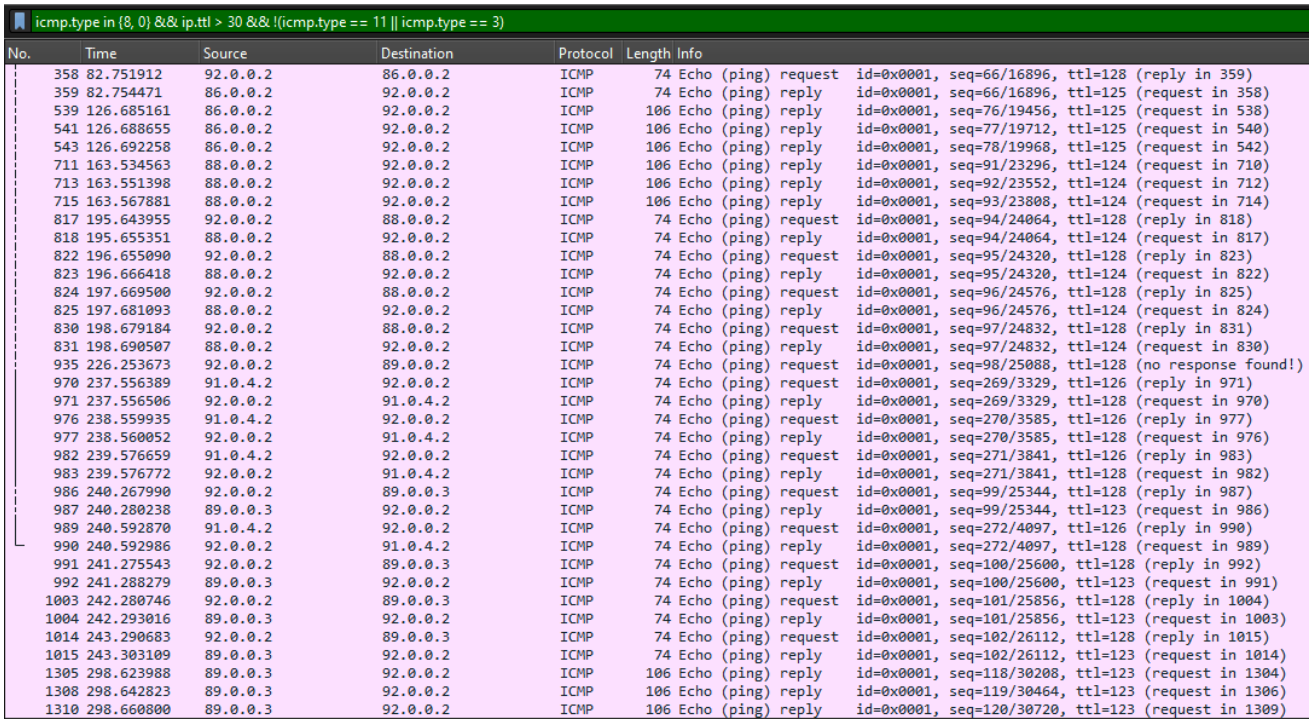
which indicates that the destination cannot be reached), which are network errors. This ensures that the filter only focuses on successful pings and not on errors.



icmp.type in {8, 0} && ip.ttl > 30 && !(icmp.type == 11 || icmp.type == 3)

No.	Time	Source	Destination	Protocol	Length	Info
57	35.909671	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=168/43008, ttl=128 (request in 56)
59	35.925743	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=169/43264, ttl=128 (request in 58)
61	35.942729	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=170/43520, ttl=128 (request in 60)
1160	787.327163	88.0.0.2	86.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=199/50944, ttl=128 (no response found!)
1166	788.336982	88.0.0.2	86.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=200/51200, ttl=128 (no response found!)
1175	793.255776	88.0.0.2	86.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=201/51456, ttl=128 (no response found!)
1177	794.262541	88.0.0.2	86.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=202/51712, ttl=128 (no response found!)
1187	798.926747	88.0.0.2	91.0.4.2	ICMP	74	Echo (ping) request id=0x0001, seq=203/51968, ttl=128 (no response found!)
1198	803.858687	88.0.0.2	91.0.4.2	ICMP	74	Echo (ping) request id=0x0001, seq=204/52224, ttl=128 (no response found!)
1203	804.875519	88.0.0.2	91.0.4.2	ICMP	74	Echo (ping) request id=0x0001, seq=205/52480, ttl=128 (no response found!)
1215	809.860939	88.0.0.2	91.0.4.2	ICMP	74	Echo (ping) request id=0x0001, seq=206/52736, ttl=128 (no response found!)
1655	1097.462365	88.0.0.2	91.0.4.2	ICMP	74	Echo (ping) request id=0x0001, seq=213/54528, ttl=128 (no response found!)
1665	1102.362279	88.0.0.2	91.0.4.2	ICMP	74	Echo (ping) request id=0x0001, seq=214/54784, ttl=128 (no response found!)
1669	1103.378012	88.0.0.2	91.0.4.2	ICMP	74	Echo (ping) request id=0x0001, seq=215/55040, ttl=128 (no response found!)
1673	1108.365244	88.0.0.2	91.0.4.2	ICMP	74	Echo (ping) request id=0x0001, seq=216/55296, ttl=128 (no response found!)
1683	1114.294894	88.0.0.2	88.0.0.1	ICMP	74	Echo (ping) request id=0x0001, seq=219/56064, ttl=128 (reply in 1684)
1684	1114.295644	88.0.0.1	88.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=219/56064, ttl=255 (request in 1683)
1687	1115.306835	88.0.0.2	88.0.0.1	ICMP	74	Echo (ping) request id=0x0001, seq=220/56320, ttl=128 (reply in 1688)
1688	1115.307730	88.0.0.1	88.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=220/56320, ttl=255 (request in 1687)
1691	1116.322244	88.0.0.2	88.0.0.1	ICMP	74	Echo (ping) request id=0x0001, seq=221/56576, ttl=128 (reply in 1692)
1692	1116.323126	88.0.0.1	88.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=221/56576, ttl=255 (request in 1691)
1693	1117.337884	88.0.0.2	88.0.0.1	ICMP	74	Echo (ping) request id=0x0001, seq=222/56832, ttl=128 (reply in 1694)
1694	1117.338783	88.0.0.1	88.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=222/56832, ttl=255 (request in 1693)

Figure 165 ping OSPF file ospf1



icmp.type in {8, 0} && ip.ttl > 30 && !(icmp.type == 11 || icmp.type == 3)

No.	Time	Source	Destination	Protocol	Length	Info
358	82.751912	92.0.0.2	86.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=66/16896, ttl=128 (reply in 359)
359	82.754471	86.0.0.2	92.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=66/16896, ttl=125 (request in 358)
539	126.685161	86.0.0.2	92.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=76/19456, ttl=125 (request in 538)
541	126.688655	86.0.0.2	92.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=77/19712, ttl=125 (request in 540)
543	126.692258	86.0.0.2	92.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=78/19968, ttl=125 (request in 542)
711	163.534563	88.0.0.2	92.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=91/23296, ttl=124 (request in 710)
713	163.551398	88.0.0.2	92.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=92/23552, ttl=124 (request in 712)
715	163.567881	88.0.0.2	92.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=93/23808, ttl=124 (request in 714)
817	195.643955	92.0.0.2	88.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=94/24064, ttl=128 (reply in 818)
818	195.655351	88.0.0.2	92.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=94/24064, ttl=124 (request in 817)
822	196.655090	92.0.0.2	88.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=95/24320, ttl=128 (reply in 823)
823	196.666418	88.0.0.2	92.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=95/24320, ttl=124 (request in 822)
824	197.669500	92.0.0.2	88.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=96/24576, ttl=128 (reply in 825)
825	197.681093	88.0.0.2	92.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=96/24576, ttl=124 (request in 824)
830	198.679184	92.0.0.2	88.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=97/24832, ttl=128 (reply in 831)
831	198.690507	88.0.0.2	92.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=97/24832, ttl=124 (request in 830)
935	226.253673	92.0.0.2	89.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=98/25088, ttl=128 (no response found!)
970	237.556389	91.0.4.2	92.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=269/3329, ttl=126 (reply in 971)
971	237.556506	92.0.0.2	91.0.4.2	ICMP	74	Echo (ping) reply id=0x0001, seq=269/3329, ttl=128 (request in 970)
976	238.559935	91.0.4.2	92.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=270/3585, ttl=126 (reply in 977)
977	238.560052	92.0.0.2	91.0.4.2	ICMP	74	Echo (ping) reply id=0x0001, seq=270/3585, ttl=128 (request in 976)
982	239.576659	91.0.4.2	92.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=271/3841, ttl=126 (reply in 983)
983	239.576772	92.0.0.2	91.0.4.2	ICMP	74	Echo (ping) reply id=0x0001, seq=271/3841, ttl=128 (request in 982)
986	240.267990	92.0.0.2	89.0.0.3	ICMP	74	Echo (ping) request id=0x0001, seq=99/25344, ttl=128 (reply in 987)
987	240.280238	89.0.0.3	92.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=99/25344, ttl=123 (request in 986)
989	240.592870	91.0.4.2	92.0.0.2	ICMP	74	Echo (ping) request id=0x0001, seq=272/4097, ttl=126 (reply in 990)
990	240.592986	92.0.0.2	91.0.4.2	ICMP	74	Echo (ping) reply id=0x0001, seq=272/4097, ttl=128 (request in 989)
991	241.275543	92.0.0.2	89.0.0.3	ICMP	74	Echo (ping) request id=0x0001, seq=100/25600, ttl=128 (reply in 992)
992	241.288279	89.0.0.3	92.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=100/25600, ttl=123 (request in 991)
1003	242.280746	92.0.0.2	89.0.0.3	ICMP	74	Echo (ping) request id=0x0001, seq=101/25856, ttl=128 (reply in 1004)
1004	242.293016	89.0.0.3	92.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=101/25856, ttl=123 (request in 1003)
1014	243.290683	92.0.0.2	89.0.0.3	ICMP	74	Echo (ping) request id=0x0001, seq=102/26112, ttl=128 (reply in 1015)
1015	243.303109	89.0.0.3	92.0.0.2	ICMP	74	Echo (ping) reply id=0x0001, seq=102/26112, ttl=123 (request in 1014)
1305	298.623988	89.0.0.3	92.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=118/30208, ttl=123 (request in 1304)
1308	298.642823	89.0.0.3	92.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=119/30464, ttl=123 (request in 1306)
1310	298.660800	89.0.0.3	92.0.0.2	ICMP	106	Echo (ping) reply id=0x0001, seq=120/30720, ttl=123 (request in 1309)

Figure 166 ping OSPF file ospf2

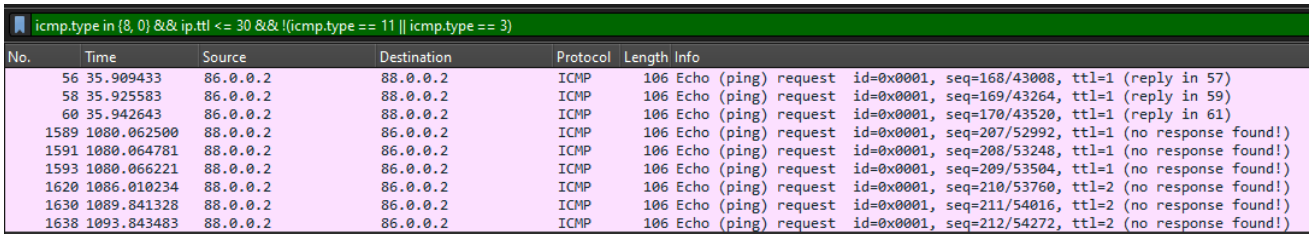
Tracert

Filter: icmp.type in {8, 0} && ip.ttl > 30 && !(icmp.type == 11 || icmp.type == 3)

icmp.type in {8, 0}: Captures ICMP packets of type 8 (echo request, ping) and type 0 (echo reply, pong), which are the packets used by the ping tool in traceroute.

ip.ttl > 30: This filter selects only packets with a TTL (Time To Live) greater than 30. This is because in a traceroute, each hop has an incremented TTL for each packet. Normally, traceroute packets travel with low TTLs, but as the TTL increases with each hop, those with a TTL greater than 30 are associated with successful replies from each hop.

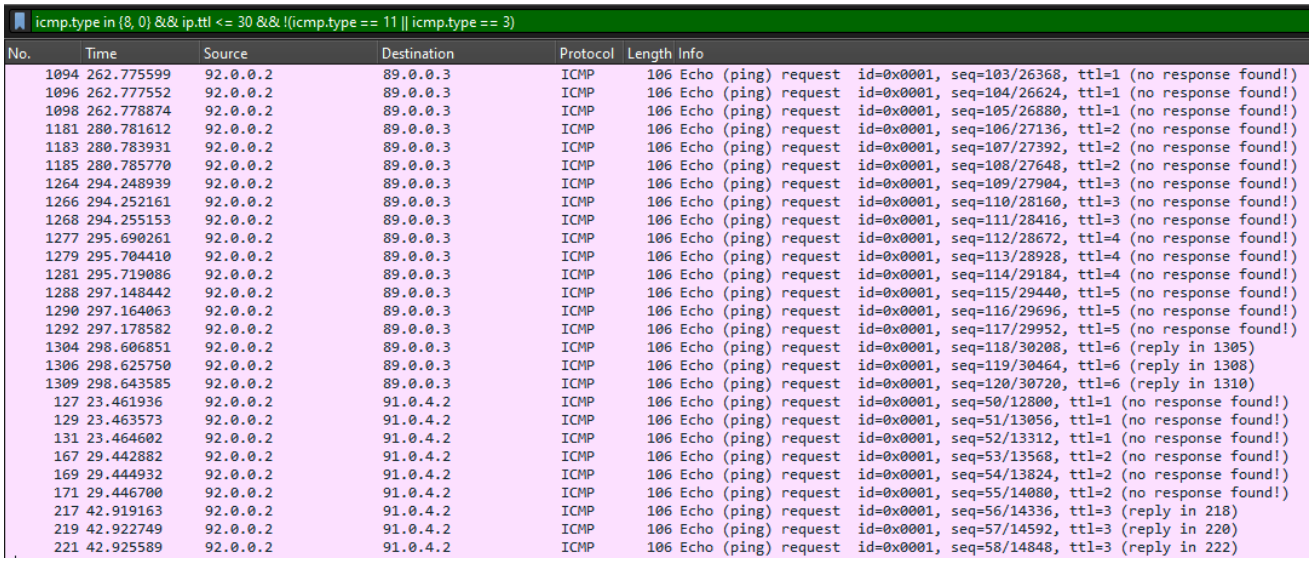
!(icmp.type == 11 || icmp.type == 3): Excludes ICMP packets of type 11 (Time Exceeded) and type 3 (Destination Unreachable). Type 11 refers to TTL expiration, which is common in a traceroute, but this filter ensures that those error packets are not included, capturing only relevant packets for the traceroute hops that are not network errors.



icmp.type in {8, 0} && ip.ttl <= 30 && !(icmp.type == 11 || icmp.type == 3)

No.	Time	Source	Destination	Protocol	Length	Info
56	35.909433	86.0.0.2	88.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=168/43008, ttl=1 (reply in 57)
58	35.925583	86.0.0.2	88.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=169/43264, ttl=1 (reply in 59)
60	35.942643	86.0.0.2	88.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=170/43520, ttl=1 (reply in 61)
1589	1080.062500	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=207/52992, ttl=1 (no response found!)
1591	1080.064781	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=208/53248, ttl=1 (no response found!)
1593	1080.066221	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=209/53504, ttl=1 (no response found!)
1620	1086.010234	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=210/53760, ttl=2 (no response found!)
1630	1089.841328	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=211/54016, ttl=2 (no response found!)
1638	1093.843483	88.0.0.2	86.0.0.2	ICMP	106	Echo (ping) request id=0x0001, seq=212/54272, ttl=2 (no response found!)

Figure 167 tracert OPFS file opfs1



icmp.type in {8, 0} && ip.ttl <= 30 && !(icmp.type == 11 || icmp.type == 3)

No.	Time	Source	Destination	Protocol	Length	Info
1094	262.775599	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=103/26368, ttl=1 (no response found!)
1096	262.777552	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=104/26624, ttl=1 (no response found!)
1098	262.778874	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=105/26880, ttl=1 (no response found!)
1181	280.781612	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=106/27136, ttl=2 (no response found!)
1183	280.783931	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=107/27392, ttl=2 (no response found!)
1185	280.785770	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=108/27648, ttl=2 (no response found!)
1264	294.248939	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=109/27904, ttl=3 (no response found!)
1266	294.252161	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=110/28160, ttl=3 (no response found!)
1268	294.255153	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=111/28416, ttl=3 (no response found!)
1277	295.690261	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=112/28672, ttl=4 (no response found!)
1279	295.704410	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=113/28928, ttl=4 (no response found!)
1281	295.719086	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=114/29184, ttl=4 (no response found!)
1288	297.148442	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=115/29440, ttl=5 (no response found!)
1290	297.164063	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=116/29696, ttl=5 (no response found!)
1292	297.178582	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=117/29952, ttl=5 (no response found!)
1304	298.606851	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=118/30208, ttl=6 (reply in 1305)
1306	298.625750	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=119/30464, ttl=6 (reply in 1308)
1309	298.643585	92.0.0.2	89.0.0.3	ICMP	106	Echo (ping) request id=0x0001, seq=120/30720, ttl=6 (reply in 1310)
127	23.461936	92.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=50/12800, ttl=1 (no response found!)
129	23.463573	92.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=51/13056, ttl=1 (no response found!)
131	23.464602	92.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=52/13312, ttl=1 (no response found!)
167	29.442882	92.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=53/13568, ttl=2 (no response found!)
169	29.444932	92.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=54/13824, ttl=2 (no response found!)
171	29.446700	92.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=55/14080, ttl=2 (no response found!)
217	42.919163	92.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=56/14336, ttl=3 (reply in 218)
219	42.922749	92.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=57/14592, ttl=3 (reply in 220)
221	42.925589	92.0.0.2	91.0.4.2	ICMP	106	Echo (ping) request id=0x0001, seq=58/14848, ttl=3 (reply in 222)

Figure 168 tracert OSPF file ospf2

Comparison between RIP and OSPF

RIP (Routing Information Protocol):

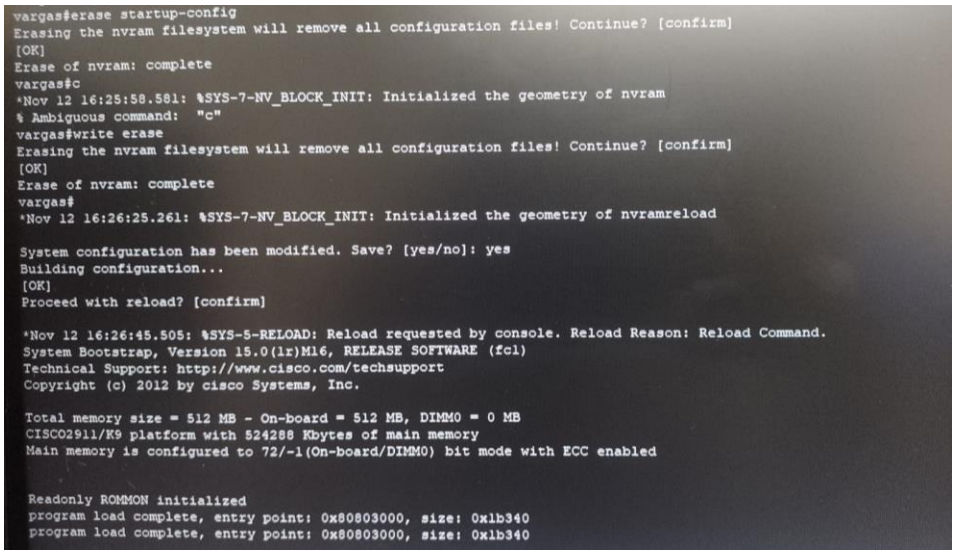
RIP is a distance-vector routing protocol that determines the best route based on hop count. In the context of the ping and tracert tests, it became evident that ICMP packets passed through more routers compared to OSPF, highlighting the inefficiency of RIP's routing method. The minimum response time observed for RIP was approximately 120 ms, but the maximum could reach as high as 3000 ms, indicating that RIP suffers from higher latency and slower convergence. This is primarily due to RIP's periodic updates and its reliance on hop count as the sole metric for route selection, which doesn't

account for network changes or link quality. The protocol's slow adaptation to network changes further contributes to its higher response times. Additionally, the results could have been influenced by the differences in the network setup, with RIP's simplicity potentially struggling to optimize larger, more complex networks, especially when compared to more dynamic protocols like OSPF.

OSPF (Open Shortest Path First):

OSPF, on the other hand, is a more sophisticated link-state routing protocol. It builds a detailed map of the network and uses the SPF (Shortest Path First) algorithm to calculate the most efficient routes, making it significantly more efficient than RIP. In the same ping and tracert tests, ICMP packets traveled through fewer hops in the OSPF network, which indicates the protocol's ability to select more direct and optimized paths. Despite OSPF's network having three additional routers compared to RIP, it still outperformed RIP in terms of response times. The minimum response time for OSPF was around 60 ms, while the maximum was 1200 ms, reflecting much lower latency and faster convergence. This is because OSPF adapts dynamically to network changes and converges more quickly, even in larger or more complex network topologies. Although the increased number of routers in the OSPF network could have slightly influenced the results, the overall performance of OSPF showed clear superiority over RIP, especially in terms of lower response times and more optimized routing. This highlights OSPF's ability to efficiently scale and adapt to larger networks while maintaining faster and more stable performance compared to RIP.

6. Closure



We used the command erase startup-config to clear the router's startup configuration, effectively removing any previous configurations stored in the startup configuration file. After executing this command, we performed a reset on the router to apply the changes and restart it with a clean configuration. This allowed us to verify that the router would boot up with default settings, without any pre-existing configuration that could affect its behavior.

Conclusions

- **Network Management and Monitoring**: During this lab, we learned the importance of network monitoring in a corporate environment. We installed monitoring tools such as SNMP and configured network devices to track performance metrics such as CPU usage, disk space, and memory. Through this, we understood how these tools allow us to maintain network health and ensure efficient resource usage.
- **Router Configuration and Static Routing**: Configuring routers and implementing static routes was one of the key points. By doing this, we understood how networks that are not directly connected can communicate with each other. We used commands like ICMP and traceroute to check connectivity, which helped us visualize how data moves between devices and networks and how to ensure that the configured routes are working correctly.
- **ICMP Messages and Traceroute**: By experimenting with tools like traceroute and ICMP, we learned how to diagnose network routes and troubleshoot connectivity issues. These tools are very useful for network administrators, as they allow tracing the path of packets across the internet and local networks, making it easier to detect connectivity issues.
- **Router and Network Device Configuration**: By configuring routers and connecting network devices, we were able to understand the router boot process, the different types of memory, and the distinction between startup and running configuration files. This gave us a deeper understanding of how routers manage their configurations and how to maintain an optimally functioning network.
- **Serial Interconnection and Troubleshooting**: In this part of the lab, we learned how to interconnect routers using serial connections. We became familiar with commands like clock rate and understood the difference between DTE and DCE, which is essential for establishing network connections between distant devices. Additionally, we learned how to troubleshoot common issues that arise in these links.
- **Application Insights and Cloud Monitoring**: We worked with Azure and the Application Insights tool, which allowed us to explore how cloud monitoring platforms can be used to observe the performance of web applications and services. By enabling real-time monitoring, we were able to see how usage metrics and data can help identify potential issues and maintain good performance for cloud applications.
- **Dynamic Routing Protocols (RIP and OSPF):** During this lab, we explored the practical application of dynamic routing protocols, specifically RIP and OSPF. Through tools like **ping** and **tracert**, we compared the performance of both protocols. We observed that RIP, with its higher number of hops, exhibited longer response times, with a maximum delay of around **3000 ms** and a minimum of **120 ms**. In contrast, OSPF demonstrated superior efficiency, with response times ranging from a minimum of **60 ms** to a maximum of **1200 ms**, even in a network with more routers. These results highlighted how OSPF's ability to optimize routing paths leads to better performance, particularly in larger and more complex networks.
- **Routing Path Optimization and Latency:** The difference in response times between RIP and OSPF further illustrated the advantages of routing path optimization in OSPF. Despite OSPF having more routers in the network, it consistently outperformed RIP in terms of latency, with significantly lower response times. The **ping** and **tracert** tests demonstrated how OSPF adapts to changes in the network more efficiently than RIP, ensuring quicker and more reliable routing decisions, which is crucial for maintaining optimal network performance.

References

- 3 installation from sources. (n.d.). Zabbix.com. Retrieved November 13, 2024, from *https://www.zabbix.com/documentation/current/en/manual/installation/install*

- AaronMaxwell. (n.d.-a). Application Insights overview. Microsoft.com. Retrieved November 13, 2024, from *https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview*

- AaronMaxwell. (n.d.-b). Diagnose with live metrics - Application Insights - Azure Monitor. Microsoft.com. Retrieved November 13, 2024, from *https://learn.microsoft.com/en-us/azure/azure-monitor/app/live-stream?tabs=otel*

- Al equipo, C. (2024, June 24). ¿Qué es un cable de módem nulo? Assured Systems. *https://www.assured-systems.com/es/faq/what-is-a-null-modem-cable/*

- Building and deploying Nagios on Oracle Solaris 11. (n.d.). Oracle.com. Retrieved November 13, 2024, from *https://www.oracle.com/technical-resources/articles/solaris/build-deploy-nagios-s11.html*

- freeCodeCamp. (2020, December 21). The OSI model – the 7 layers of networking explained in plain English. Freecodecamp.org. *https://www.freecodecamp.org/news/osi-model-networking-layers-explained-in-plain-english/*

- Raza, M. (n.d.). OSI Model: The 7 layers of network architecture. BMC Blogs. Retrieved November 13, 2024, from *https://www.bmc.com/blogs/osi-model-7-layers/*

(N.d.). Baeldung.com. Retrieved November 25, 2024, from https://www.baeldung.com/linux/snmp-no-more-variables-left-this-mib-view

- (N.d.-b). Auvik.com. Retrieved November 25, 2024, from https://www.auvik.com/franklyit/blog/difference-between-snmp-v2-v3/