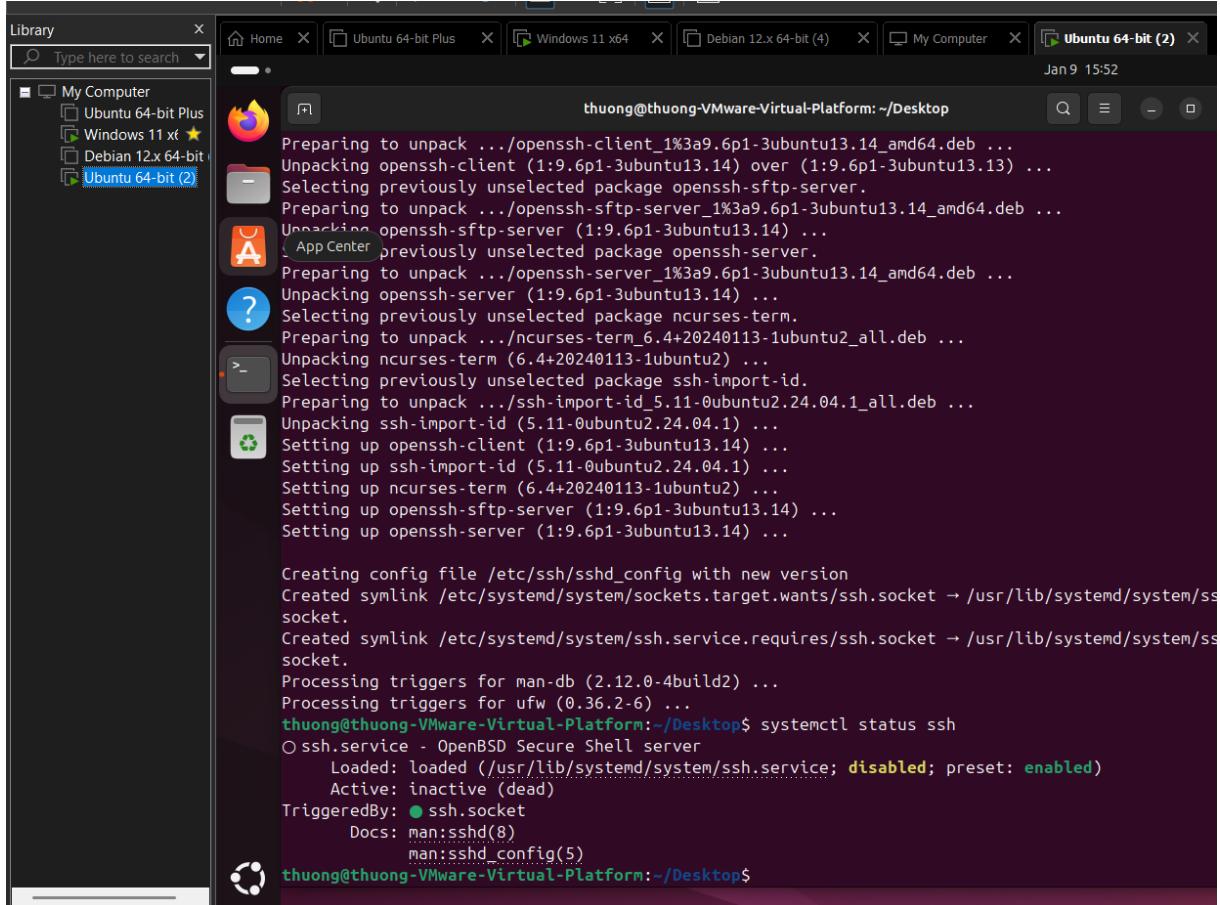


Week 6 – Networking

Student number: 589932

Assignment 6.1: Working from home

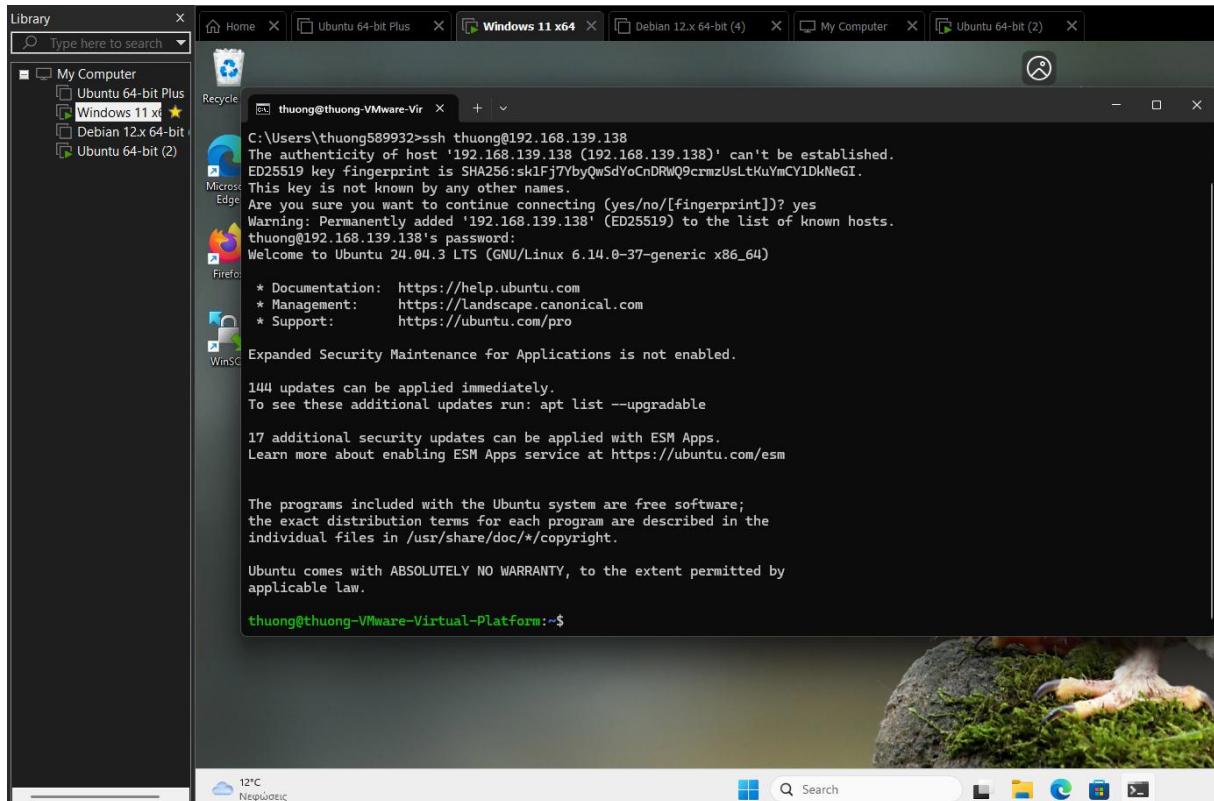
Screenshot installation openssh-server:



```
Preparing to unpack .../openssh-client_1%3a9.6p1-3ubuntu13.14_amd64.deb ...
Unpacking openssh-client (1:9.6p1-3ubuntu13.14) over (1:9.6p1-3ubuntu13.13) ...
Preparing to unpack .../openssh-sftp-server_1%3a9.6p1-3ubuntu13.14_amd64.deb ...
Unpacking openssh-sftp-server (1:9.6p1-3ubuntu13.14) ...
[AppCenter] previously unselected package openssh-server.
Preparing to unpack .../openssh-server_1%3a9.6p1-3ubuntu13.14_amd64.deb ...
Unpacking openssh-server (1:9.6p1-3ubuntu13.14) ...
Selecting previously unselected package ncurses-term.
Preparing to unpack .../ncurses-term_6.4+20240113-1ubuntu2_all.deb ...
Unpacking ncurses-term (6.4+20240113-1ubuntu2) ...
Selecting previously unselected package ssh-import-id.
Preparing to unpack .../ssh-import-id_5.11-0ubuntu2.24.04.1_all.deb ...
Unpacking ssh-import-id (5.11-0ubuntu2.24.04.1) ...
Setting up openssh-client (1:9.6p1-3ubuntu13.14) ...
Setting up ssh-import-id (5.11-0ubuntu2.24.04.1) ...
Setting up ncurses-term (6.4+20240113-1ubuntu2) ...
Setting up openssh-sftp-server (1:9.6p1-3ubuntu13.14) ...
Setting up openssh-server (1:9.6p1-3ubuntu13.14) ...

Creating config file /etc/ssh/sshd_config with new version
Created symlink /etc/systemd/system/sockets.target.wants/ssh.socket → /usr/lib/systemd/system/ssh.socket.
Created symlink /etc/systemd/system/ssh.service.requires/ssh.socket → /usr/lib/systemd/system/ssh.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for ufw (0.36.2-6) ...
thuong@thuong-VMware-Virtual-Platform:~/Desktop$ systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
    Loaded: loaded (/usr/lib/systemd/system/ssh.service; disabled; preset: enabled)
      Active: inactive (dead)
   TriggeredBy: ● ssh.socket
     Docs: man:sshd(8)
           man:sshd_config(5)
thuong@thuong-VMware-Virtual-Platform:~/Desktop$
```

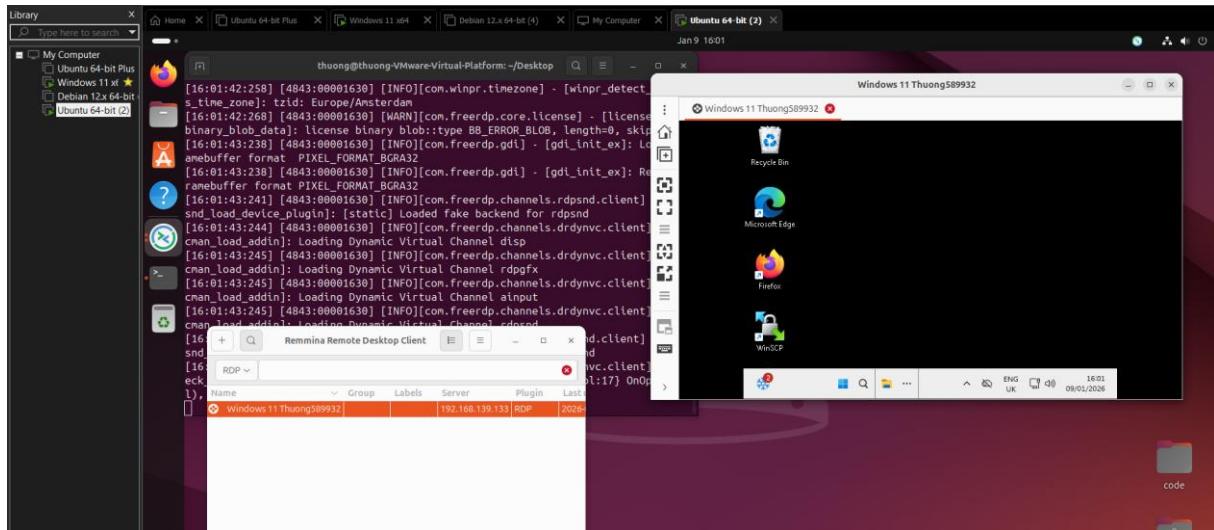
Screenshot successful SSH command execution:



Screenshot successful execution SCP command:

```
thuong@thuong-VMware-Virtual-Platform:~$ echo Week 6 SCP test > week6.txt
thuong@thuong-VMware-Virtual-Platform:~$ scp week6.txt thuong@192.168.139.138:/home/thuong/
The authenticity of host '192.168.139.138 (192.168.139.138)' can't be established.
ED25519 key fingerprint is SHA256:sk1Fj7YbyQwSdYoCnDRWQ9crmzUsLtkUyMcyIDkNeGI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.139.138' (ED25519) to the list of known hosts.
thuong@192.168.139.138's password:
week6.txt                                                 100%   16    15.9KB/s   00:00
thuong@thuong-VMware-Virtual-Platform:~$ |
```

Screenshot remmina:



Assignment 6.2: IP addresses websites

Relevant screenshots nslookup command:

```
C:\WINDOWS\system32\cmd. x + v

Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

C:\Users\LEGION>nslookup
Default Server: RZ-DC01.Personeel.local
Address: 10.171.92.1

> amazon.com
Server: RZ-DC01.Personeel.local
Address: 10.171.92.1

Non-authoritative answer:
Name: amazon.com
Addresses: 98.87.170.71
          98.82.161.185
          98.87.170.74

> google.com
Server: RZ-DC01.Personeel.local
Address: 10.171.92.1

Non-authoritative answer:
Name: google.com
Addresses: 2a00:1450:400e:804::200e
          142.251.39.142

> one.one.one.one
Server: RZ-DC01.Personeel.local
Address: 10.171.92.1

Non-authoritative answer:
Name: one.one.one.one
Addresses: 2606:4700:4700::1111
          2606:4700:4700::1001
          1.1.1.1
          1.0.0.1

> dns.google.com
Server: RZ-DC01.Personeel.local
Address: 10.171.92.1
```



Search

```
C:\WINDOWS\system32\cmd. + ▾

> dns.google.com
Server: RZ-DC01.Personeel.local
Address: 10.171.92.1

Non-authoritative answer:
Name: dns.google.com
Addresses: 2001:4860:4860::8888
           2001:4860:4860::8844
           8.8.8.8
           8.8.4.4

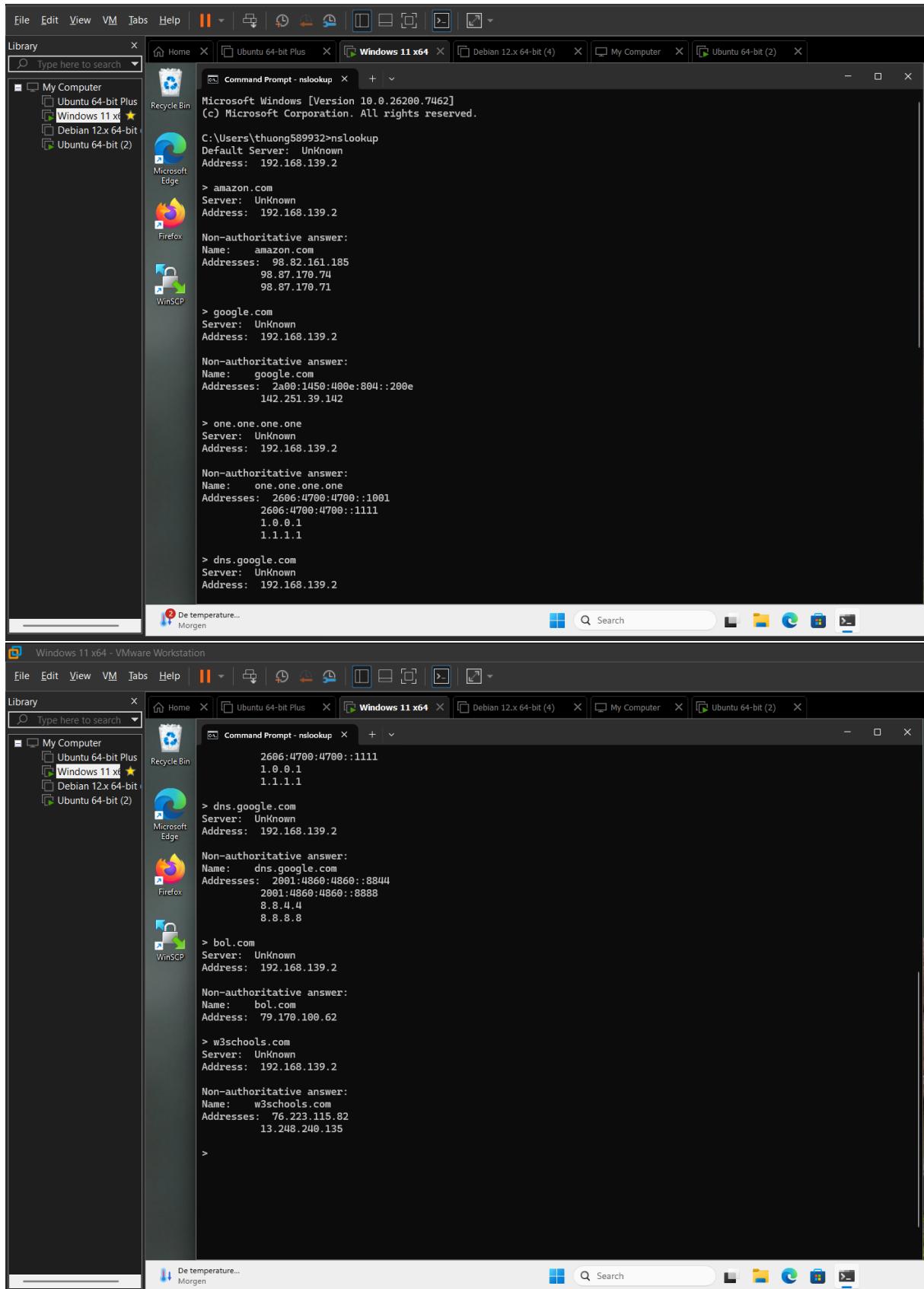
> bol.com
Server: RZ-DC01.Personeel.local
Address: 10.171.92.1

Non-authoritative answer:
Name: bol.com
Address: 79.170.100.62

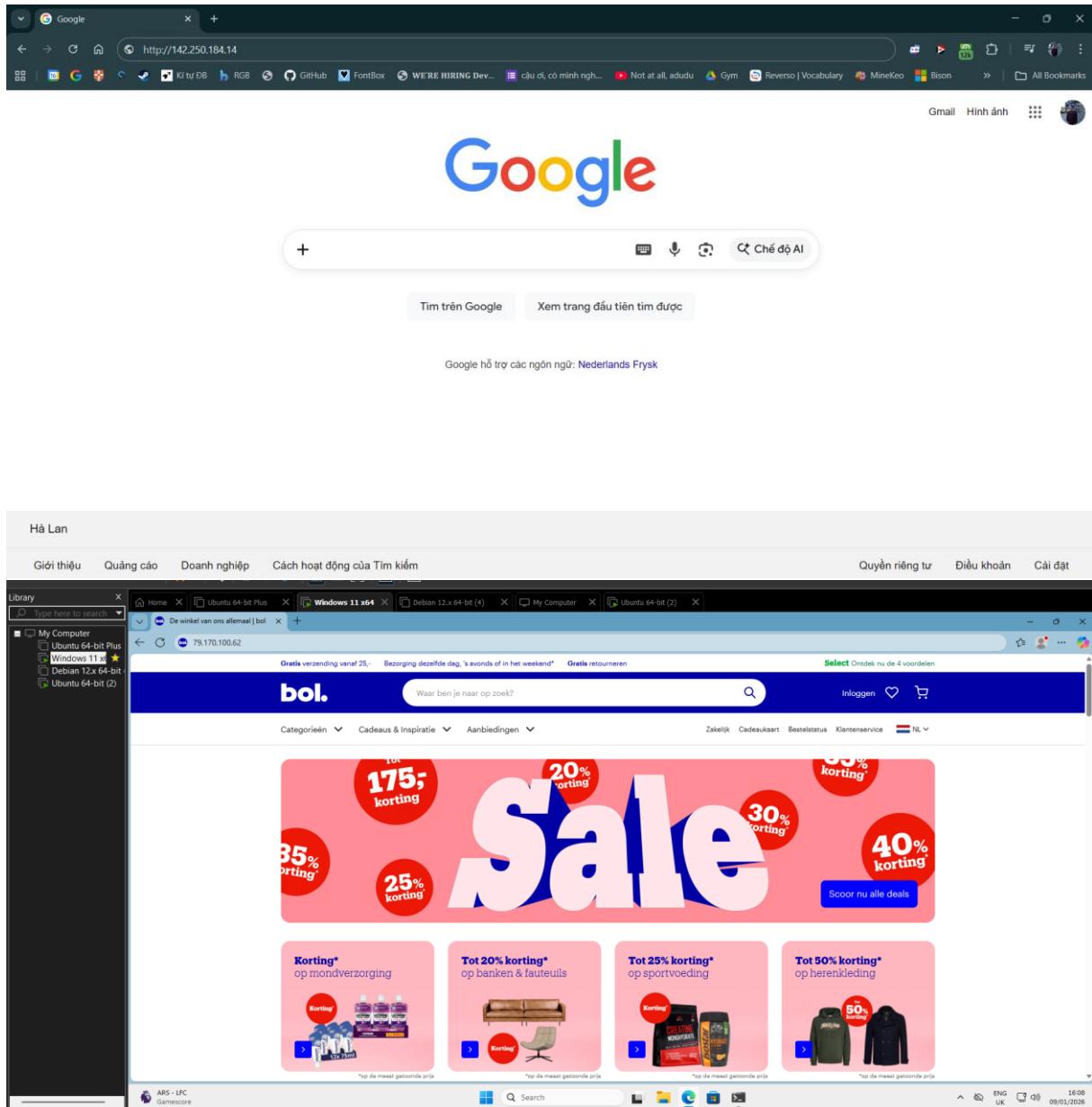
> w3schools.com
Server: RZ-DC01.Personeel.local
Address: 10.171.92.1

Non-authoritative answer:
Name: w3schools.com
Addresses: 76.223.115.82
           13.248.240.135

>
```



Screenshot website visit via IP address:



Assignment 6.3: subnetting

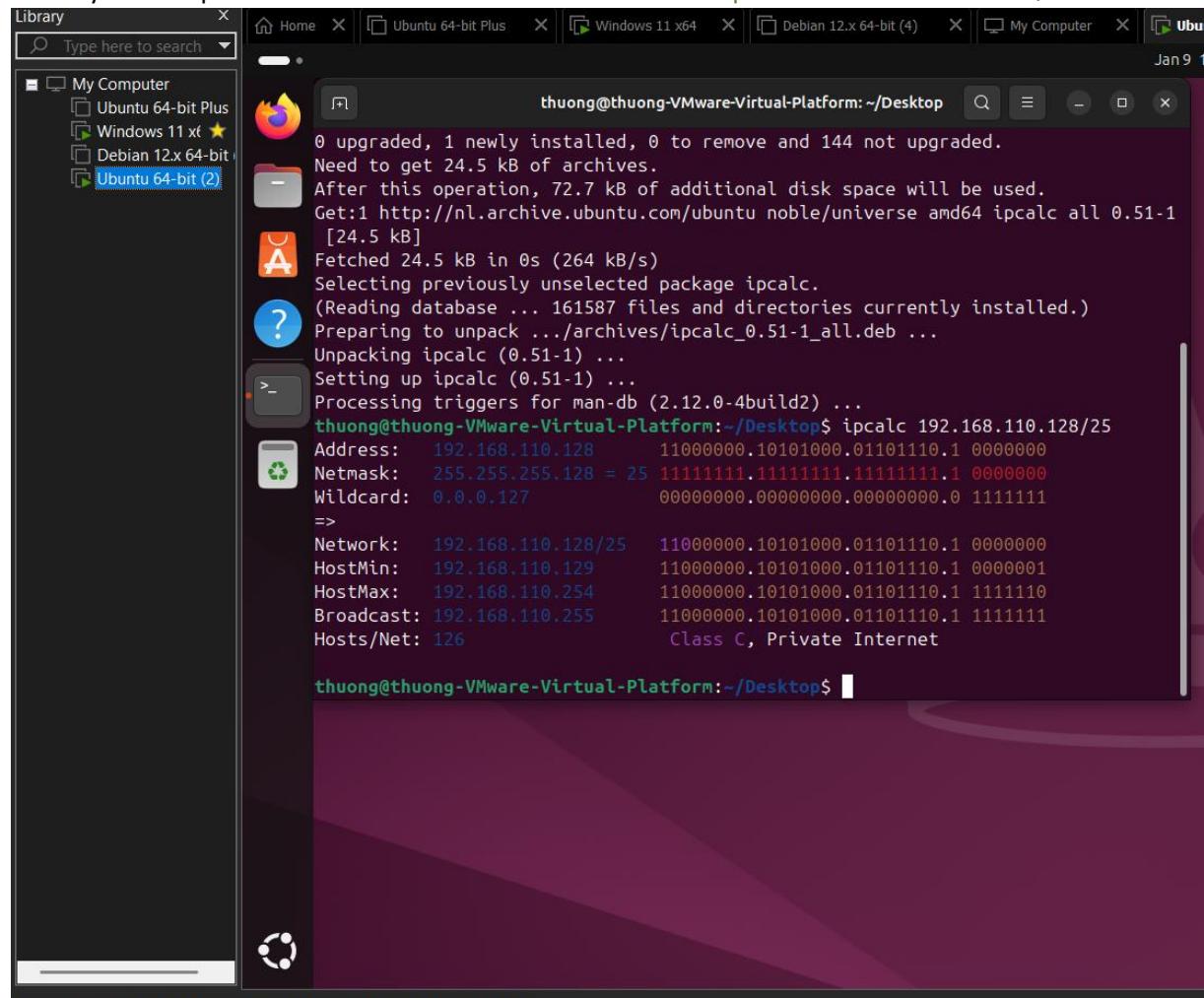
How many IP addresses are in this network configuration 192.168.110.128/25?

There are **128** IP addresses

What is the usable IP range to hand out to the connected computers?

From 192.168.110.129 to 192.168.110.254

Check your two previous answers with this Linux command: `ipcalc 192.168.110.128/25`



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "thuong@thuong-VMware-Virtual-Platform: ~/Desktop". The terminal output shows the results of the ipcalc command:

```
0 upgraded, 1 newly installed, 0 to remove and 144 not upgraded.  
Need to get 24.5 kB of archives.  
After this operation, 72.7 kB of additional disk space will be used.  
Get:1 http://nl.archive.ubuntu.com/ubuntu noble/universe amd64 ipcalc all 0.51.1  
[24.5 kB]  
Fetched 24.5 kB in 0s (264 kB/s)  
Selecting previously unselected package ipcalc.  
(Reading database ... 161587 files and directories currently installed.)  
Preparing to unpack .../archives/ipcalc_0.51.1_all.deb ...  
Unpacking ipcalc (0.51.1) ...  
Setting up ipcalc (0.51.1) ...  
Processing triggers for man-db (2.12.0-4build2) ...  
thuong@thuong-VMware-Virtual-Platform:~/Desktop$ ipcalc 192.168.110.128/25  
Address: 192.168.110.128      11000000.10101000.01101110.1 00000000  
Netmask: 255.255.255.128 = 25 11111111.11111111.11111111.1 00000000  
Wildcard: 0.0.0.127            00000000.00000000.00000000.0 11111111  
=>  
Network: 192.168.110.128/25  11000000.10101000.01101110.1 00000000  
HostMin: 192.168.110.129    11000000.10101000.01101110.1 00000001  
HostMax: 192.168.110.254    11000000.10101000.01101110.1 11111110  
Broadcast: 192.168.110.255  11000000.10101000.01101110.1 11111111  
Hosts/Net: 126                Class C, Private Internet  
thuong@thuong-VMware-Virtual-Platform:~/Desktop$
```

Explain the above calculation in your own words.

A /25 subnet means that 25 bits are used for the network part of the IP address, leaving 7 bits for host addresses.

With 7 host bits, the total number of IP addresses is 2^7 , which equals 128.

The first address of the subnet is reserved as the network address, and the last address is reserved as the

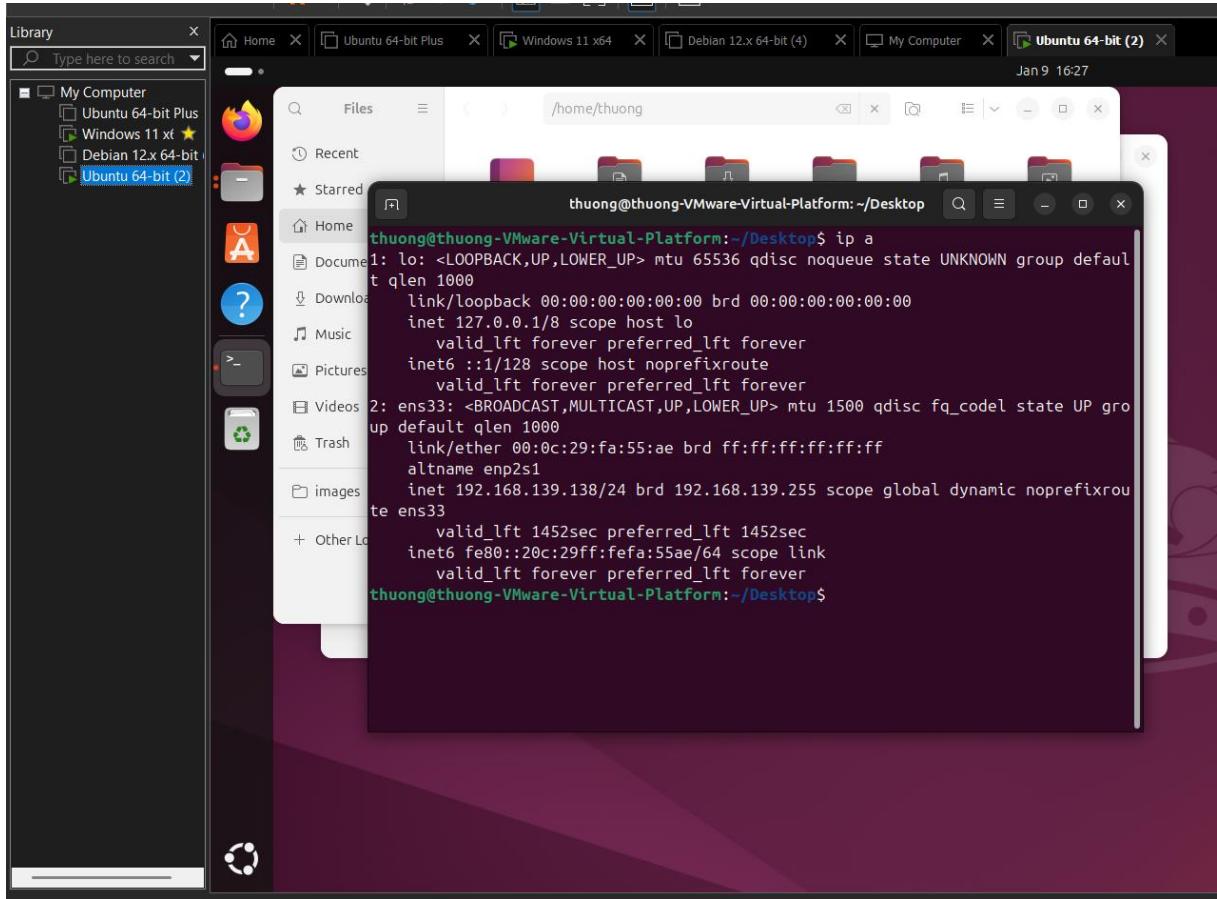
broadcast address.

Because these two addresses cannot be assigned to computers, only 126 IP addresses are usable.

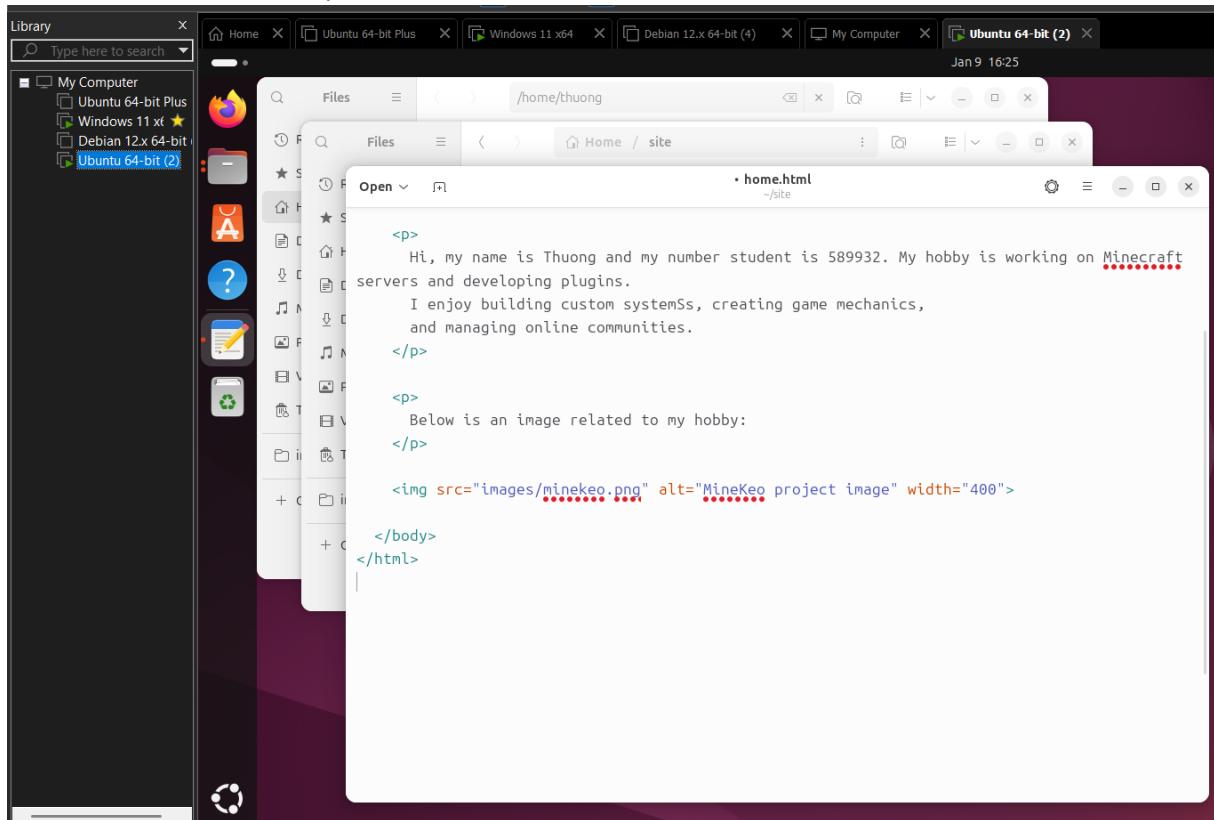
Therefore, the usable IP range for connected computers is from 192.168.110.129 to 192.168.110.254.

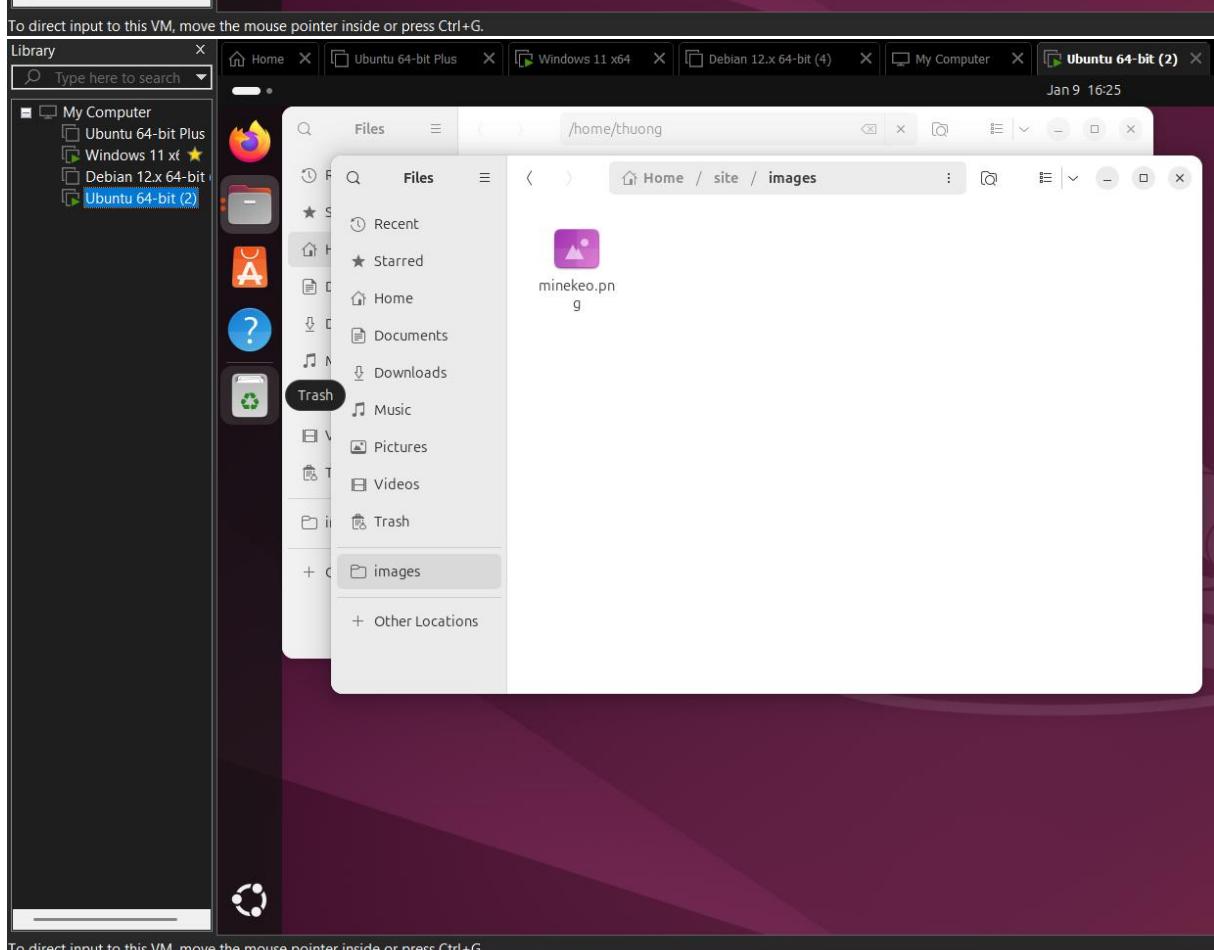
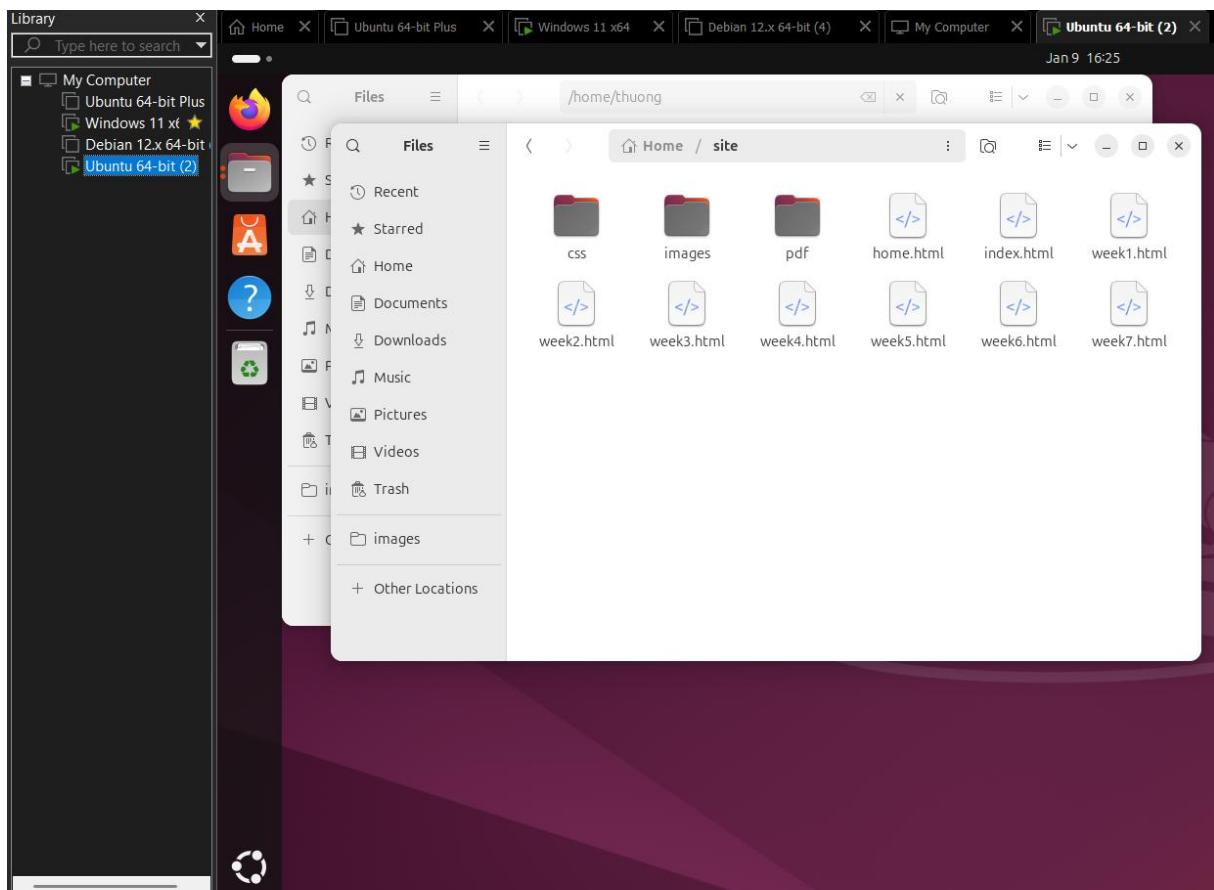
Assignment 6.4: HTML

Screenshot IP address Ubuntu VM:



Screenshot of Site directory contents:



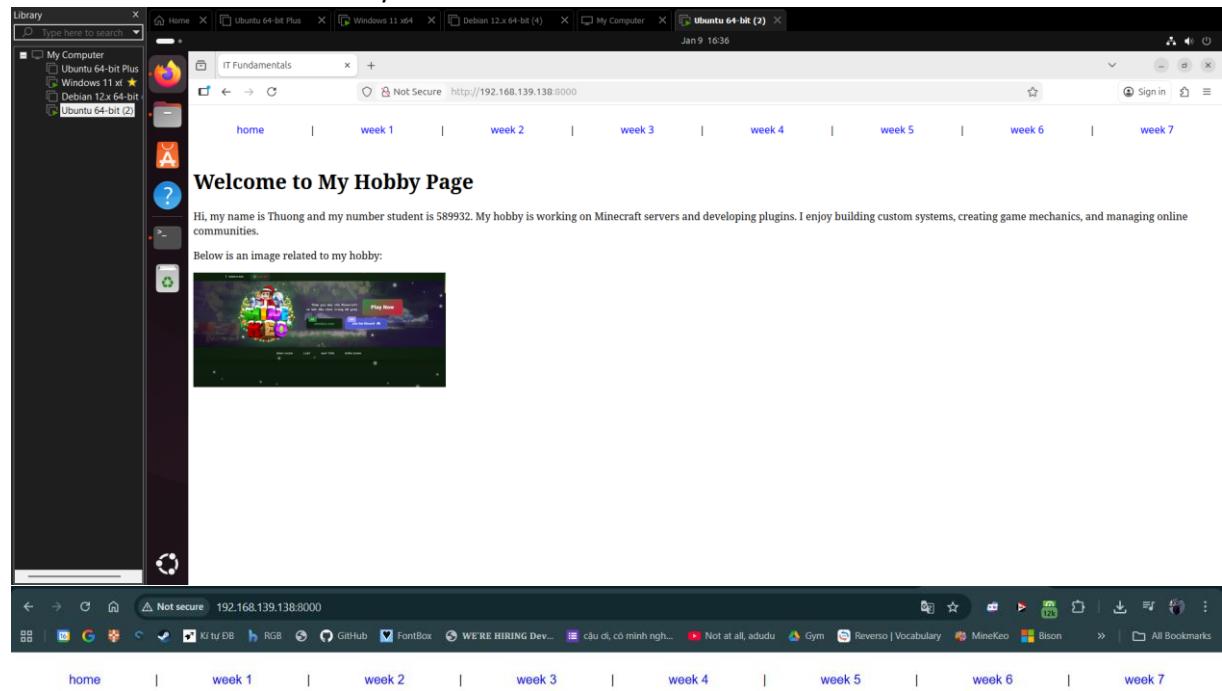


Screenshot python3 webserver command:

The screenshot shows a Linux desktop environment with a dark theme. A terminal window is open in the center, displaying the output of a Python web server command. The terminal window title is "thuong@thuong-VMware-Virtual-Platform: ~/site". The command run was "python3 -m http.server 8000". The output shows the server serving files from the current directory to an IP address of 192.168.139.138 on port 8000. Several requests are listed, including "GET / HTTP/1.1" 200, "GET /home.html HTTP/1.1" 200, "GET /css/mypdfstyle.css HTTP/1.1" 200, and multiple requests for "/css/mypdfstyle.css" and "/images/minekeo.png" which resulted in a 404 error message: "File not found".

```
thuong@thuong-VMware-Virtual-Platform:~/site$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.139.138 - - [09/Jan/2026 16:29:00] "GET / HTTP/1.1" 200 -
192.168.139.138 - - [09/Jan/2026 16:29:00] "GET /home.html HTTP/1.1" 200 -
192.168.139.138 - - [09/Jan/2026 16:29:00] "GET /css/mypdfstyle.css HTTP/1.1" 200 -
192.168.139.138 - - [09/Jan/2026 16:29:00] "GET /css/mypdfstyle.css HTTP/1.1" 200 -
192.168.139.138 - - [09/Jan/2026 16:29:00] code 404, message File not found
192.168.139.138 - - [09/Jan/2026 16:29:00] "GET /favicon.ico HTTP/1.1" 404 -
192.168.139.138 - - [09/Jan/2026 16:29:00] "GET /images/minekeo.png HTTP/1.1" 200 -
```

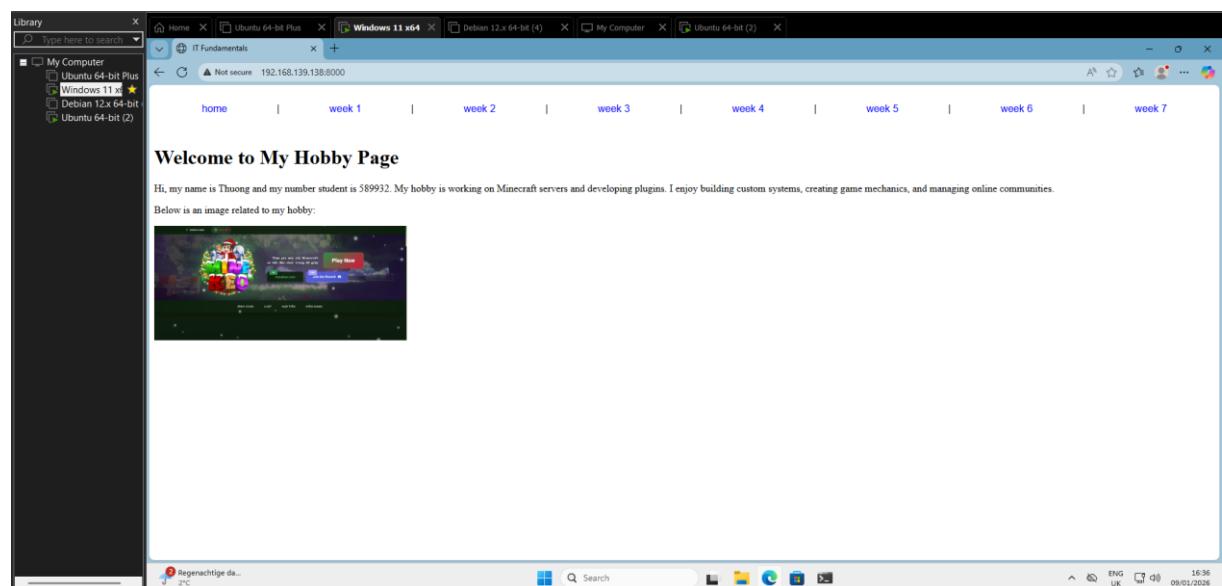
Screenshot web browser visits your site



Welcome to My Hobby Page

Hi, my name is Thuong and my number student is 589932. My hobby is working on Minecraft servers and developing plugins. I enjoy building custom systems, creating game mechanics, and managing online communities.

Below is an image related to my hobby:



Assignment 6.5: Network segment

Remember that bitwise java application you've made in week 2? Expand that application so that you can also calculate a network segment as explained in the PowerPoint slides of week 6. Use the bitwise & AND operator. You need to be able to input two Strings. An IP address and a subnet.

IP: 192.168.1.100 and subnet: 255.255.255.224 for /27

Example: 192.168.1.100/27

Calculate the network segment

IP Address: 11000000.10101000.00000001.01100100

Subnet Mask: 11111111.11111111.11111111.11100000

Network Addr: 11000000.10101000.00000001.01100000

This gives 192.168.1.96 in decimal as the network address.

For a /27 subnet, each segment (or subnet) has 32 IP addresses (2^5).

The range of this network segment is from 192.168.1.96 to 192.168.1.127.

Paste source code here, with a screenshot of a working application.

```
import nl.saxion.app.SaxionApp;

public class Application implements Runnable {

    public static void main(String[] args) {
        SaxionApp.start(new Application(), 600, 500);
    }

    public void run() {
        SaxionApp.printLine("== BITWISE CALCULATOR ==");
        SaxionApp.printLine("Choose an option:");
        SaxionApp.printLine("1. Is number odd?");
        SaxionApp.printLine("2. Is number a power of 2?");
        SaxionApp.printLine("3. Two's complement");
        SaxionApp.printLine("4. Calculate Network Segment");
        int choice = SaxionApp.readInt("Your choice: ");

        switch (choice) {
            case 1:
                SaxionApp.printLine("Enter a number: ");
                int num1 = SaxionApp.readInt();
                if (isOdd(num1)) SaxionApp.printLine("The number is odd.");
        }
    }
}
```

```

        else SaxonApp.printLine("The number is even.");
        break;

    case 2:
        SaxonApp.printLine("Enter a number: ");
        int num2 = SaxonApp.readInt();
        if (isPowerOfTwo(num2)) SaxonApp.printLine("The number IS
a power of 2.");
        else SaxonApp.printLine("The number is NOT a power of
2.");
        break;

    case 3:
        SaxonApp.printLine("Enter a number: ");
        int num3 = SaxonApp.readInt();
        int result = twosComplement(num3);
        SaxonApp.printLine("Two's complement: " + result);
        break;

    case 4:
        calculateNetworkSegment();
        break;

    default:
        SaxonApp.printLine("Invalid option.");
    }
}

boolean isOdd(int number) {
    return (number & 1) == 1;
}

boolean isPowerOfTwo(int number) {
    if (number <= 0) return false;
    return (number & (number - 1)) == 0;
}

int twosComplement(int number) {
    return (~number) + 1;
}

void calculateNetworkSegment() {
    SaxonApp.printLine("\n==== NETWORK SEGMENT CALCULATOR ====");

    SaxonApp.print("Enter IP Address (e.g., 192.168.1.100): ");
    String ipAddress = SaxonApp.readString();

    SaxonApp.print("Enter Subnet Mask (e.g., 255.255.255.224): ");
    String subnetMask = SaxonApp.readString();
}

```

```

        int[] ipParts = parseIP(ipAddress);
        int[] subnetParts = parseIP(subnetMask);

        int[] networkParts = new int[4];
        for (int i = 0; i < 4; i++) {
            networkParts[i] = ipParts[i] & subnetParts[i];
        }

        int cidr = calculateCIDR(subnetParts);

        int hostBits = 32 - cidr;
        int totalAddresses = (int) Math.pow(2, hostBits);

        int[] broadcastParts = new int[4];
        int[] invertedMask = new int[4];
        for (int i = 0; i < 4; i++) {
            invertedMask[i] = 255 - subnetParts[i];
            broadcastParts[i] = networkParts[i] | invertedMask[i];
        }

        SaxionApp.printLine("\n-----");
        SaxionApp.printLine("IP Address:      " + ipAddress);
        SaxionApp.printLine("IP Binary:       " + toBinaryString(ipParts));
        SaxionApp.printLine("-----");
        SaxionApp.printLine("Subnet Mask:     " + subnetMask + " (" + cidr +
")");
        SaxionApp.printLine("Subnet Binary:   " +
toBinaryString(subnetParts));
        SaxionApp.printLine("-----");
        SaxionApp.printLine("          IP:    " + toBinaryString(ipParts));
        SaxionApp.printLine("          & Mask:  " +
toBinaryString(subnetParts));
        SaxionApp.printLine("          =      " +
toBinaryString(networkParts));
        SaxionApp.printLine("-----");
        SaxionApp.printLine("Network Addr:   " + formatIP(networkParts));
        SaxionApp.printLine("Broadcast:      " + formatIP(broadcastParts));
        SaxionApp.printLine("-----");
        SaxionApp.printLine("Network Range:  " + formatIP(networkParts) + " -
" + formatIP(broadcastParts));
        SaxionApp.printLine("Total IPs:      " + totalAddresses + " addresses");
        SaxionApp.printLine("Usable Hosts:   " + (totalAddresses - 2) + " hosts");
    }
}

```

```

int[] parseIP(String ip) {
    String[] parts = ip.split("\\.");
    int[] result = new int[4];
    for (int i = 0; i < 4; i++) {
        result[i] = Integer.parseInt(parts[i]);
    }
    return result;
}

String toBinaryString(int[] parts) {
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < 4; i++) {
        String binary = Integer.toBinaryString(parts[i]);
        while (binary.length() < 8) {
            binary = "0" + binary;
        }
        sb.append(binary);
        if (i < 3) sb.append(".");
    }
    return sb.toString();
}

String formatIP(int[] parts) {
    return parts[0] + "." + parts[1] + "." + parts[2] + "." + parts[3];
}

int calculateCIDR(int[] subnetParts) {
    int cidr = 0;
    for (int i = 0; i < 4; i++) {
        int octet = subnetParts[i];
        while (octet > 0) {
            cidr += (octet & 1);
            octet = octet >> 1;
        }
    }
    return cidr;
}

```

Saxion Drawingboard

```
Enter Subnet Mask (e.g., 255.255.255.224): 255.255.255.224

-----
IP Address: 192.168.1.100
IP Binary: 11000000.10101000.00000001.01100100
-----

Subnet Mask: 255.255.255.224 (/27)
Subnet Binary: 11111111.11111111.11111111.11100000
-----

IP: 11000000.10101000.00000001.01100100
& Mask: 11111111.11111111.11111111.11100000
= 11000000.10101000.00000001.01100000
-----

Network Addr: 192.168.1.96
Broadcast: 192.168.1.127
-----

Network Range: 192.168.1.96 - 192.168.1.127
Total IPs: 32 addresses
Usable Hosts: 30 hosts
```

APPLICATION EXITED NORMALLY

Ready? Save this file and export it as a pdf file with the name: [week6.pdf](#)