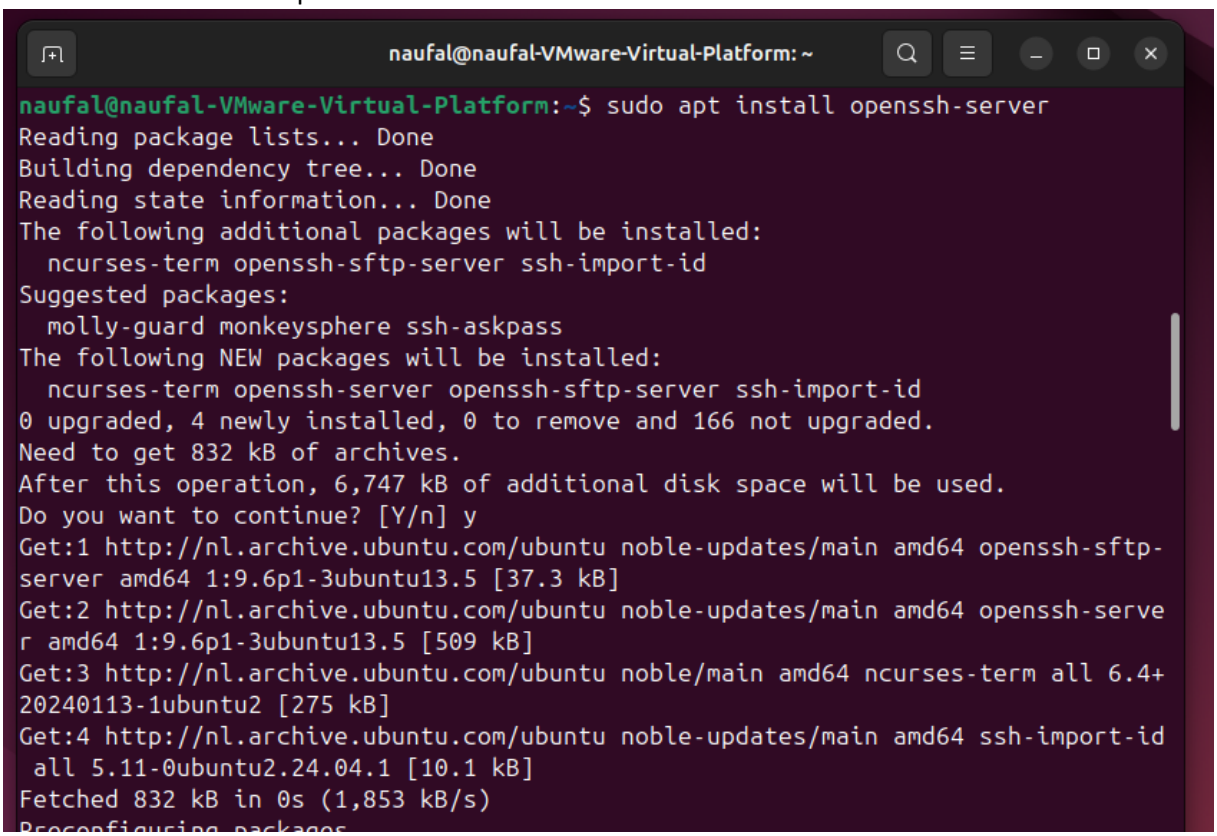
Template Week 6 – Networking

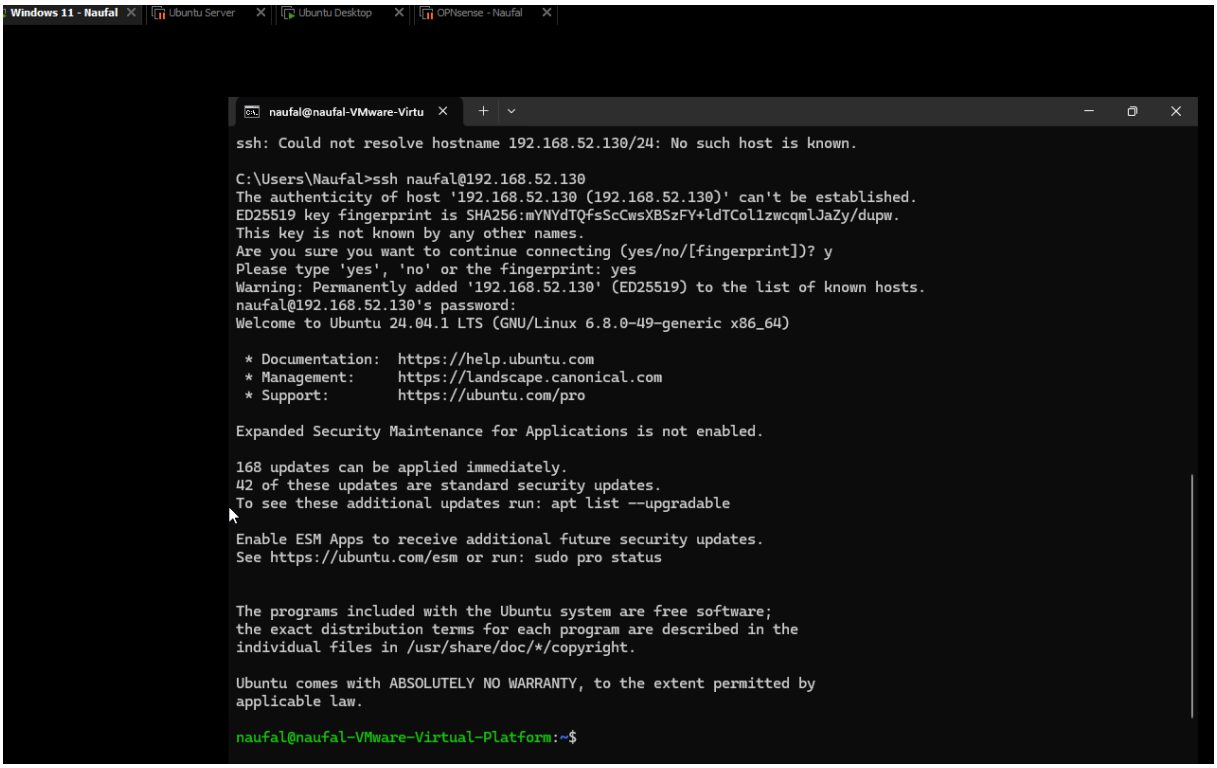
Student number: 569508

Assignment 6.1: Working from home

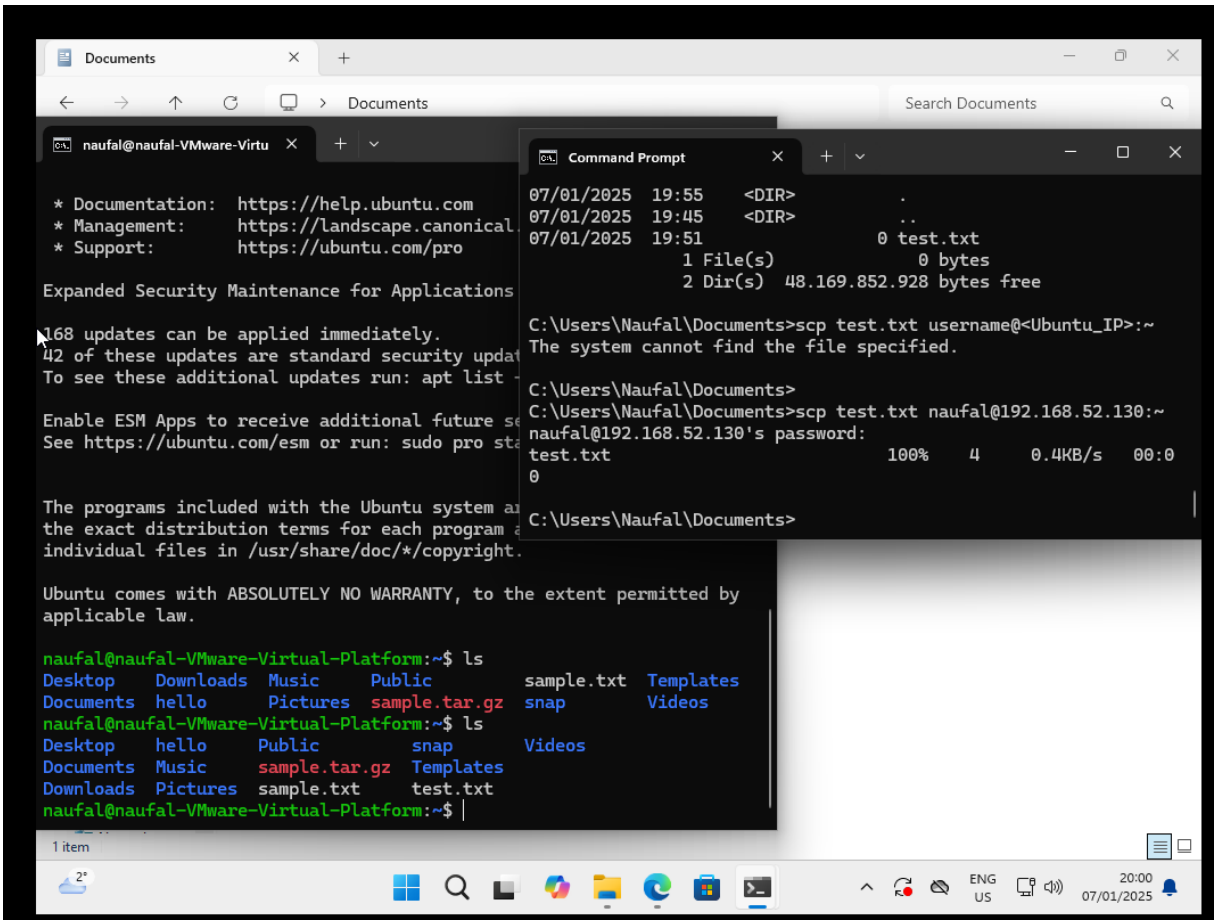
Screenshot installation openssh-server:



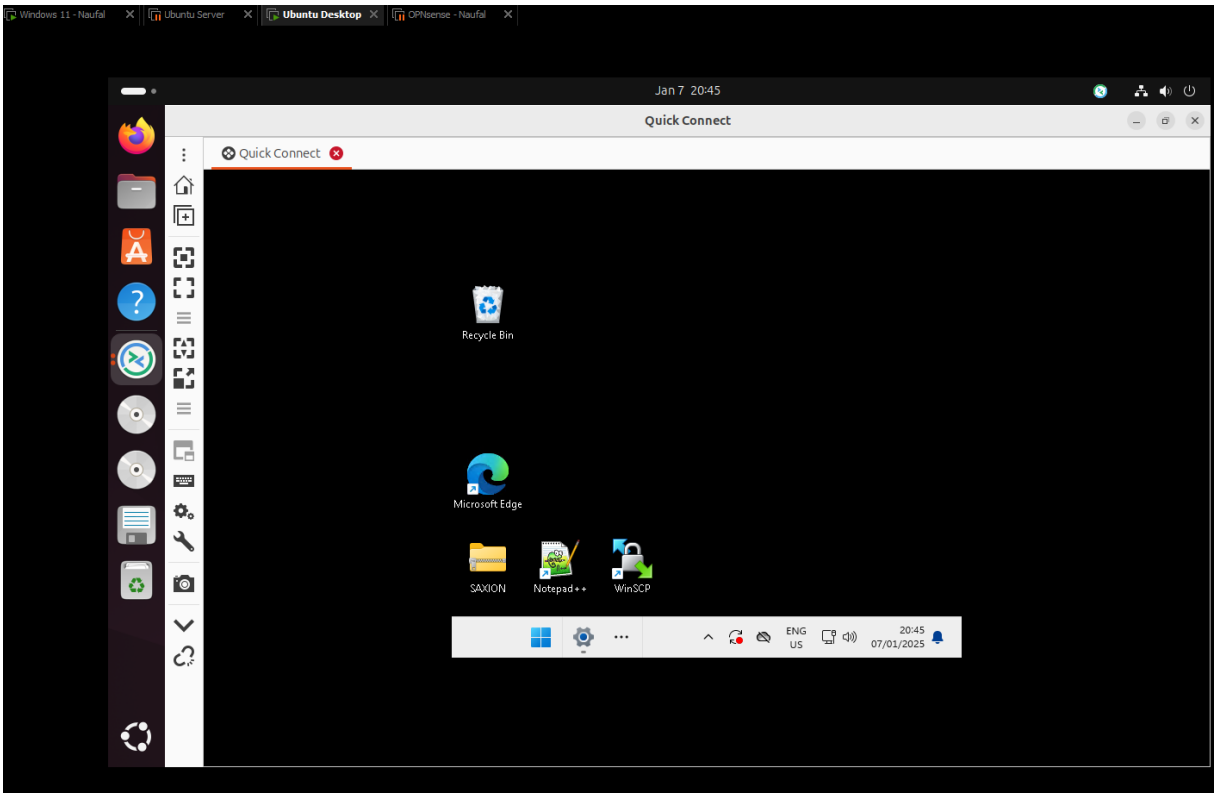
Screenshot successful SSH command execution:



Screenshot successful execution SCP command:

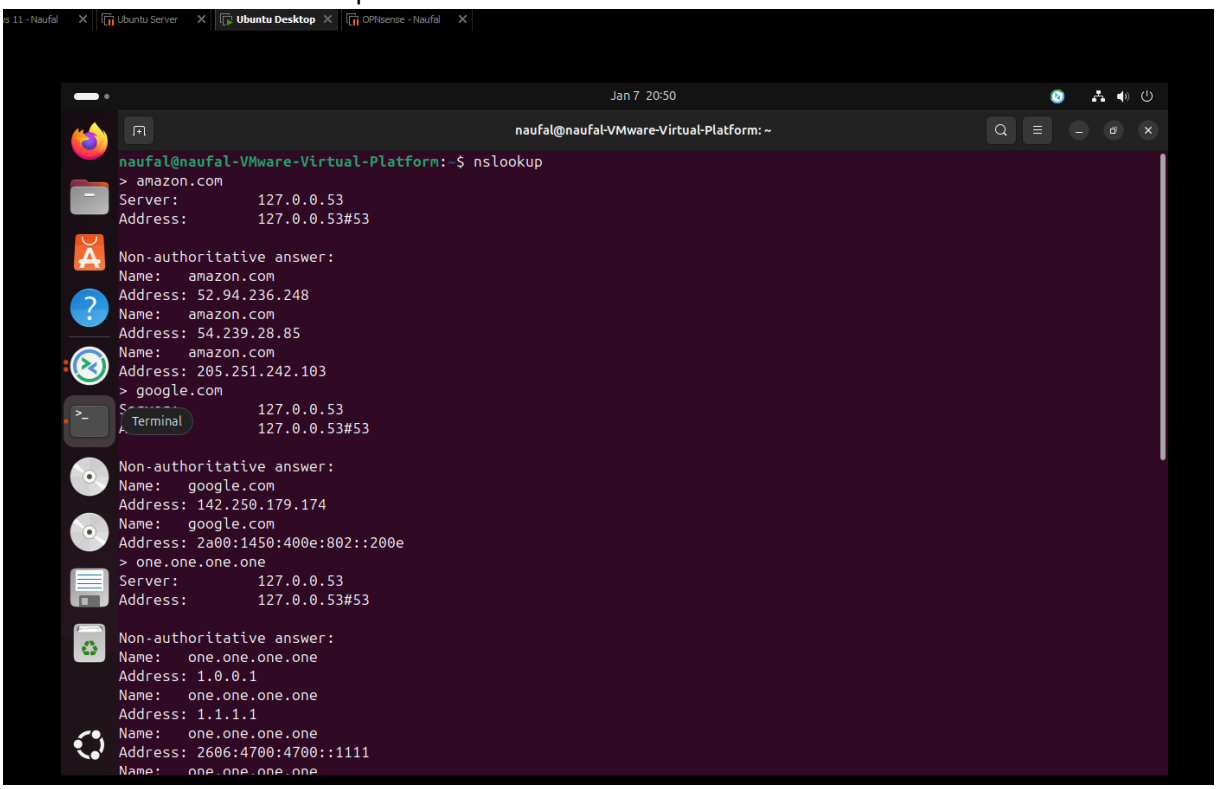


Screenshot remmina:

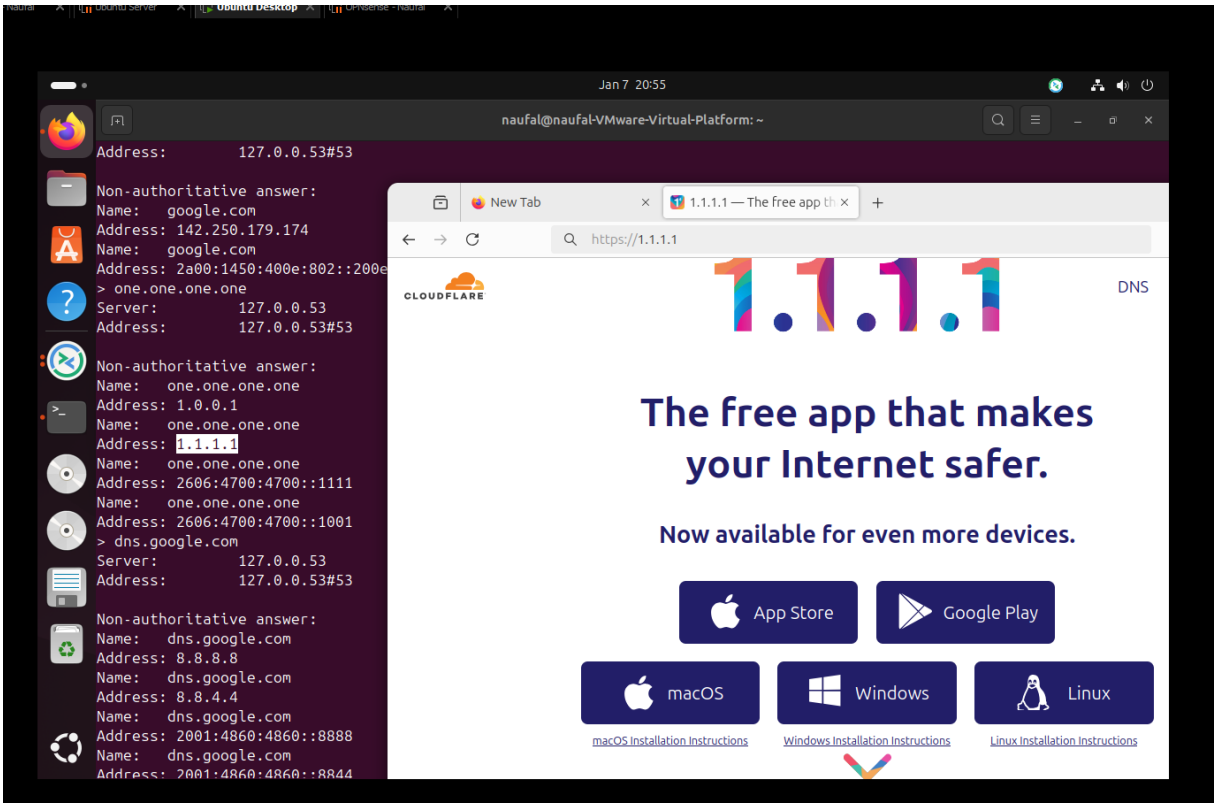


Assignment 6.2: IP addresses websites

Relevant screenshots nslookup command:



Screenshot website visit via IP address:



Assignment 6.3: subnetting

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

128 adresses

This calculator returns a variety of information regarding Internet Protocol version 4 (IPv4) and IPv6 subnets including possible network addresses, usable host ranges, subnet mask, and IP class, am others.

IPv4 Subnet Calculator

Result

IP Address:	192.168.110.128
Network Address:	192.168.110.128
Usable Host IP Range:	192.168.110.129 - 192.168.110.254
Broadcast Address:	192.168.110.255
Total Number of Hosts:	128
Number of Usable Hosts:	126
Subnet Mask:	255.255.255.128
Wildcard Mask:	0.0.0.127
Binary Subnet Mask:	11111111.11111111.11111111.10000000
IP Class:	C
CIDR Notation:	/25
IP Type:	Private
Short:	192.168.110.128 /25
Binary ID:	11000000101010000110111010000000
Integer ID:	3232263808
Hex ID:	0xc0a86e80
in-addr.arpa:	128.110.168.192.in-addr.arpa
IPv4 Mapped Address:	::ffff:c0a8.6e80
6to4 Prefix:	2002:c0a8.6e80::/48

All 2 of the Possible /25 Networks for 192.168.110.*

Network Address	Usable Host Range	Broadcast Address:
192.168.110.0	192.168.110.1 - 192.168.110.126	192.168.110.127
192.168.110.128	192.168.110.129 - 192.168.110.254	192.168.110.255

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

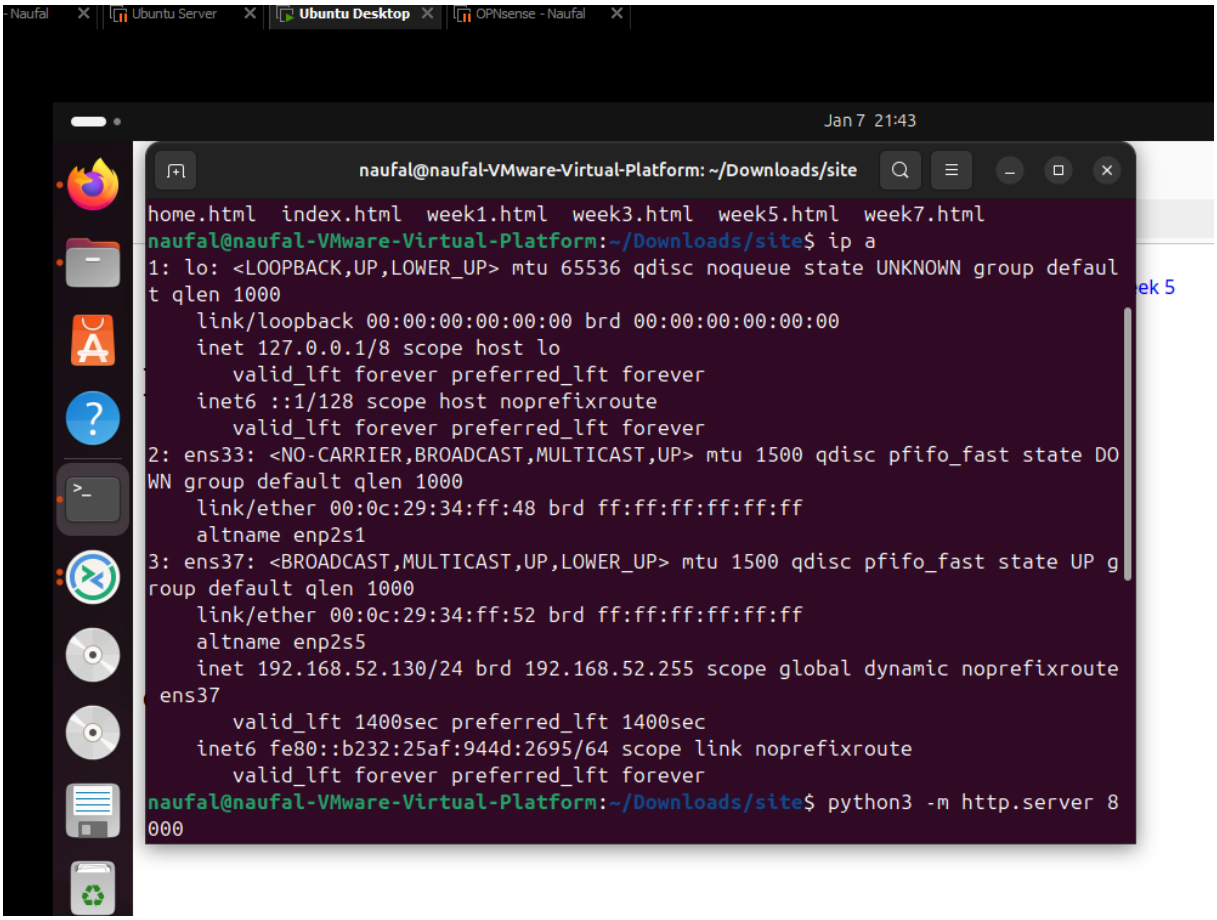
Usable range is 192.168.110.129 to 192.168.110.254

Check your two previous answers with this calculator:
https://www.calculator.net/ip-subnet-calculator.html

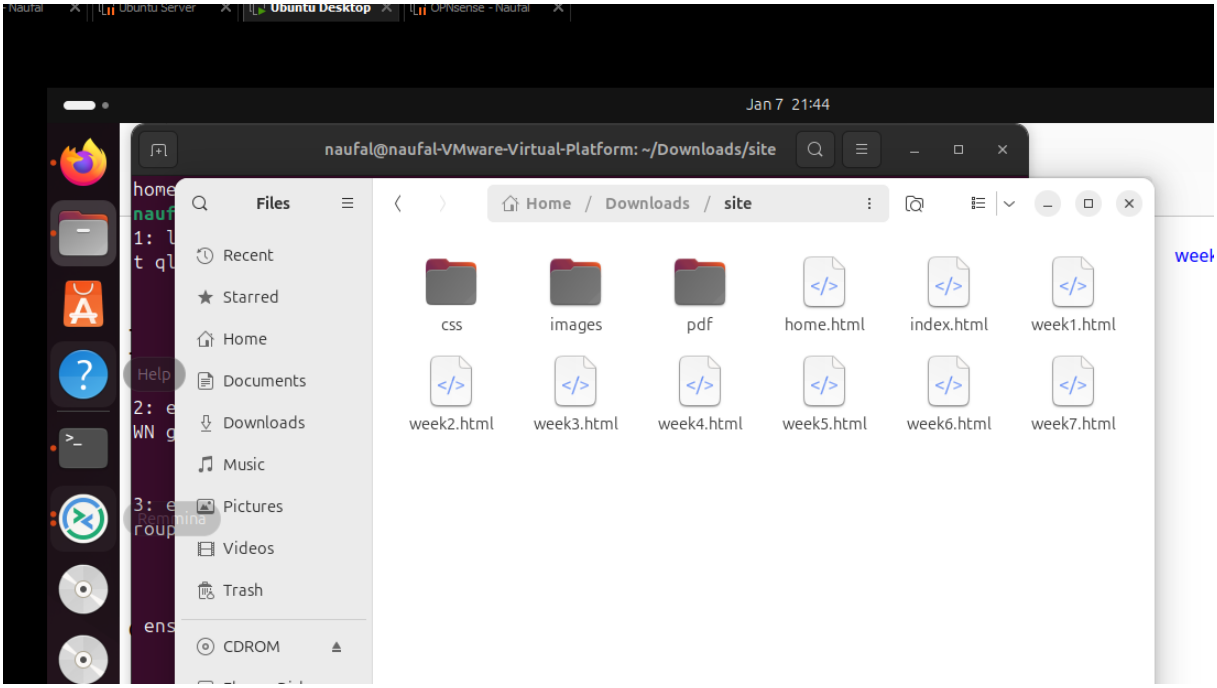
The network 192.168.110.128 includes 128 addresses in total. However, two of these addresses are reserved one for the network itself 192.168.110.128 and one for broadcasting messages to all devices 192.168.110.255. This means you can use 126 addresses for devices, range from 192.168.110.129 to 192.168.110.254

Assignment 6.4: HTML

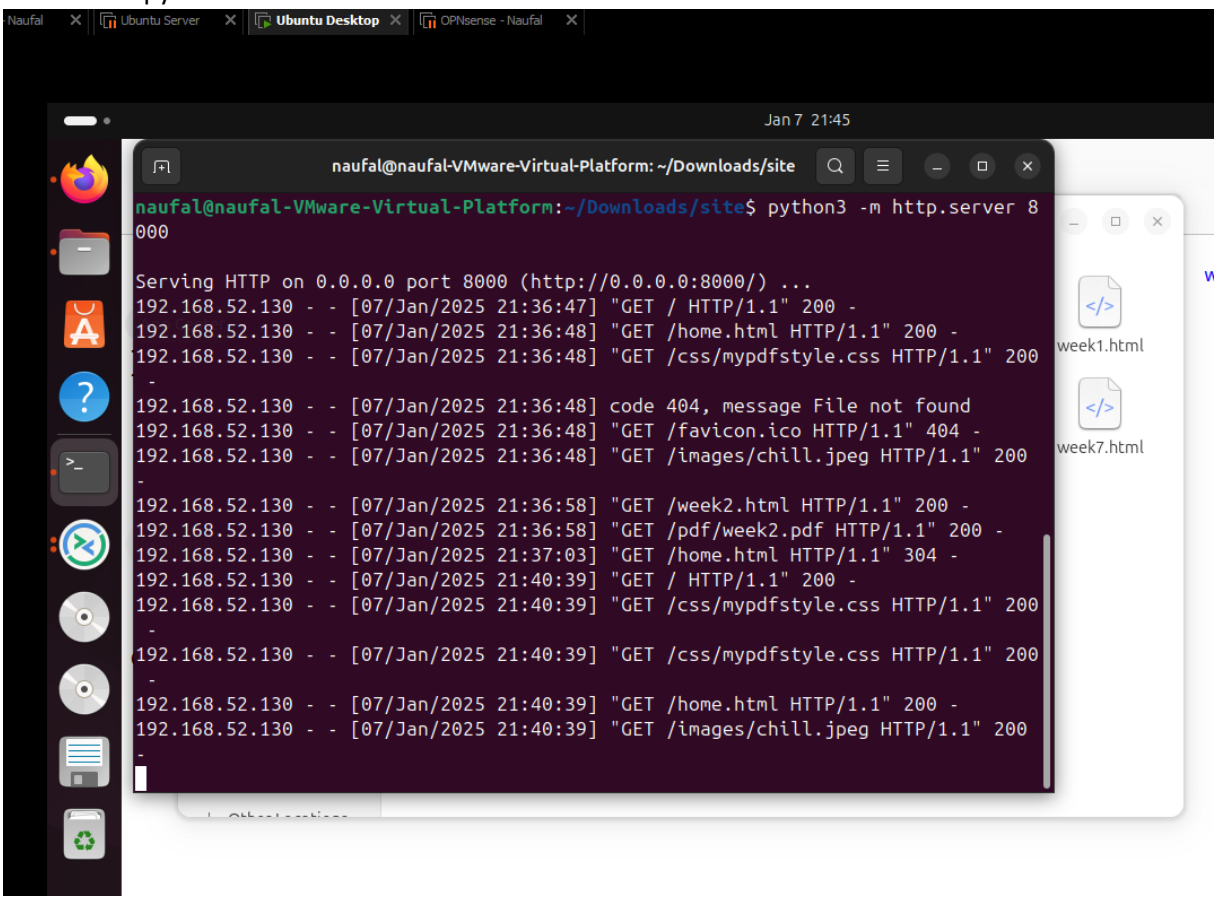
Screenshot IP address Ubuntu VM:



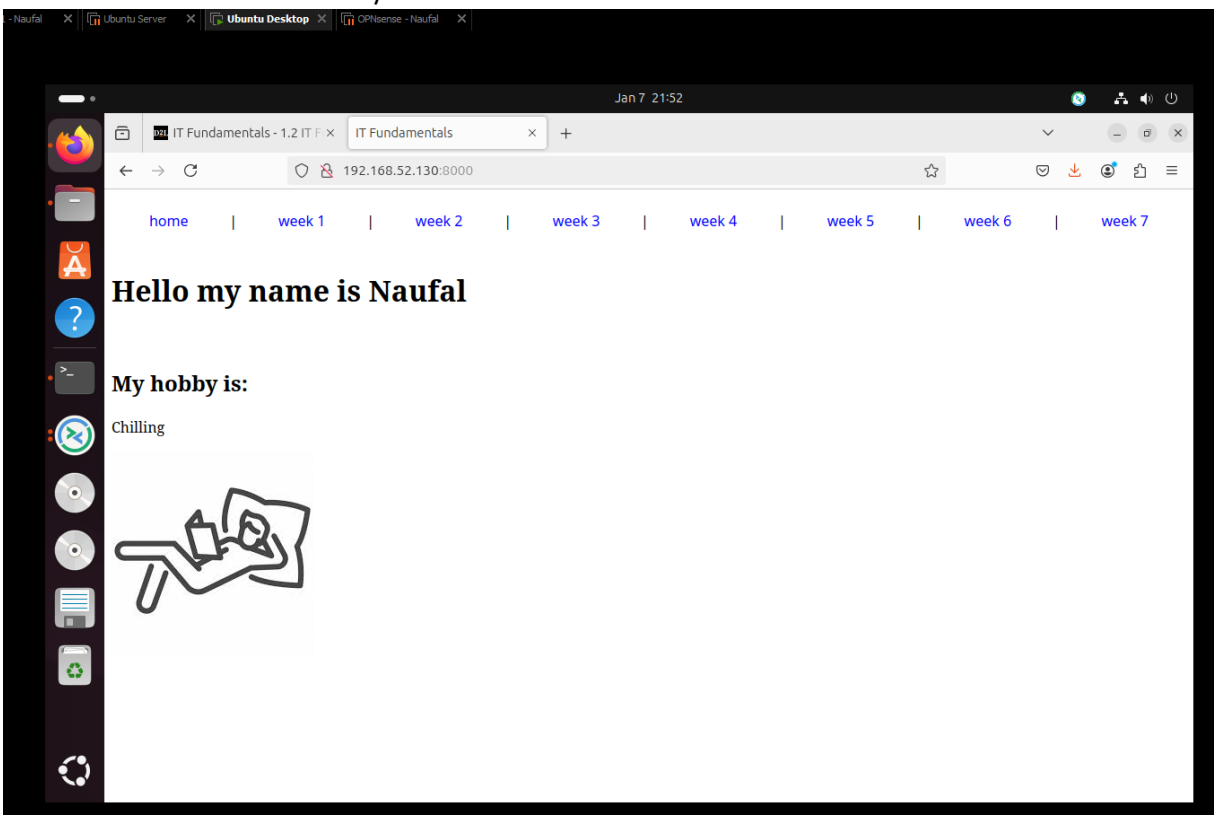
Screenshot of Site directory contents:



Screenshot python3 webserver command:



Screenshot web browser visits your site



Bonus point assignment – week 6

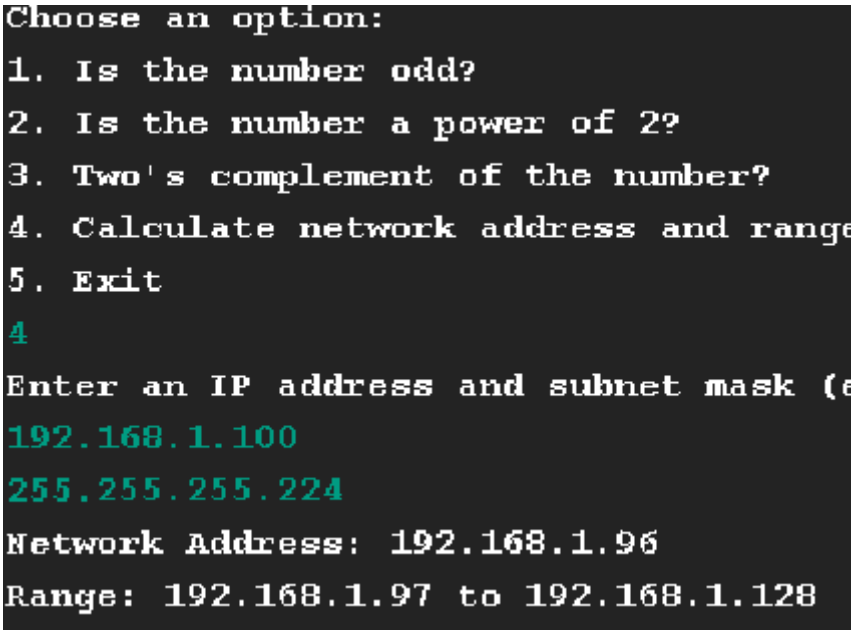
Remember that bitwise java application you've made in week 2? Expand that application so that you can also calculate a network segment as explained in the PowerPoint slides of week 6. Use the bitwise & AND operator. You need to be able to input two Strings. An IP address and a subnet.

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 (2⁵).
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(), 350, 500);
    }

    public void run() {
        while (true) {
            SaxionApp.printLine("Choose an option:");
            SaxionApp.printLine("1. Is the number odd?");
            SaxionApp.printLine("2. Is the number a power of 2?");
            SaxionApp.printLine("3. Two's complement of the number?");
            SaxionApp.printLine("4. Calculate network address and range");
            SaxionApp.printLine("5. Exit");

            int choice = SaxionApp.readInt();

            if (choice == 5) {
                SaxionApp.printLine("DOEI!");
                break;
            }

            int number = 0;
            if (choice != 4) {
                number = SaxionApp.readInt("Enter a number: ");
            }

            switch (choice) {
                case 1 -> SaxionApp.printLine("Is number odd? " + isOdd(number));
```

```
            case 2 -> SaxionApp.printLine("Is number a power of 2? " + isPowerOfTwo(number));
            case 3 -> SaxionApp.printLine("Two's complement: " + twosComplement(number));
            case 4 -> calculateNetworkSegment();
            default -> SaxionApp.printLine("Invalid choice. Try again!!");
          }
        }
    }

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

    private boolean isPowerOfTwo(int number) {
        return number > 0 && (number & (number - 1)) == 0;
    }

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

    private void calculateNetworkSegment() {
        SaxionApp.printLine("Enter an IP address and subnet mask: ");

        String ip = SaxionApp.readString();
        String subnet = SaxionApp.readString();

        String networkAddress = calculateNetworkAddress(ip, subnet);
        SaxionApp.printLine("Network Address: " + networkAddress);

        String[] networkParts = networkAddress.split("\\.");
        int networkStart = Integer.parseInt(networkParts[3]) + 1;
        int networkEnd = networkStart + 31;

        SaxionApp.printLine("Range: " + networkParts[0] + "." + networkParts[1] + "." + networkParts[2]
+ "." + networkStart
                + " to " + networkParts[0] + "." + networkParts[1] + "." + networkParts[2] + "." +
networkEnd);
    }

    private String calculateNetworkAddress(String ip, String subnet) {
        String[] ipParts = ip.split("\\.");
        String[] subnetParts = subnet.split("\\.");

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

```
        return networkParts[0] + "." + networkParts[1] + "." + networkParts[2] + "." + networkParts[3];
    }
}
```

Ready? Save this file and export it as a pdf file with the name: **week6.pdf**