IT FUNDAMENTALS

Week 6 – Networking

Week 6 — Assignment 6.1: Working from home

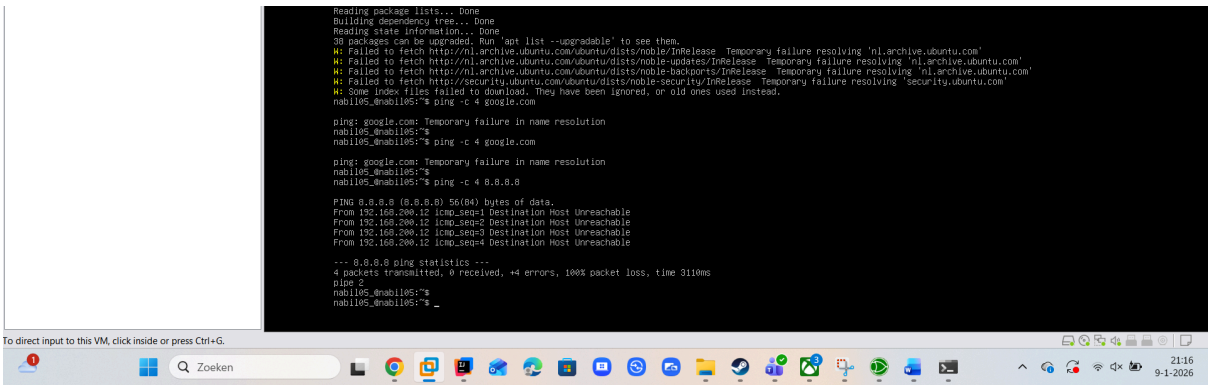
Voor deze opdracht was het doel om op afstand verbinding te maken met een Linux-server vanaf de eigen laptop.
Hiervoor wordt normaal gebruikgemaakt van SSH (Secure Shell) en SCP (Secure Copy).

Eerst wordt op de Ubuntu-server de OpenSSH-server geïnstalleerd. Deze dienst zorgt ervoor dat externe systemen veilig kunnen inloggen op de server via het netwerk.
Vervolgens wordt het IP-adres van de server opgezocht, zodat vanaf de eigen laptop een verbinding kan worden gemaakt.

Met het SSH-commando kan een gebruiker op afstand inloggen op de Ubuntu-server en daar commando's uitvoeren alsof hij lokaal is ingelogd.
Met het SCP-commando kunnen bestanden veilig worden gekopieerd tussen de eigen laptop en de server.

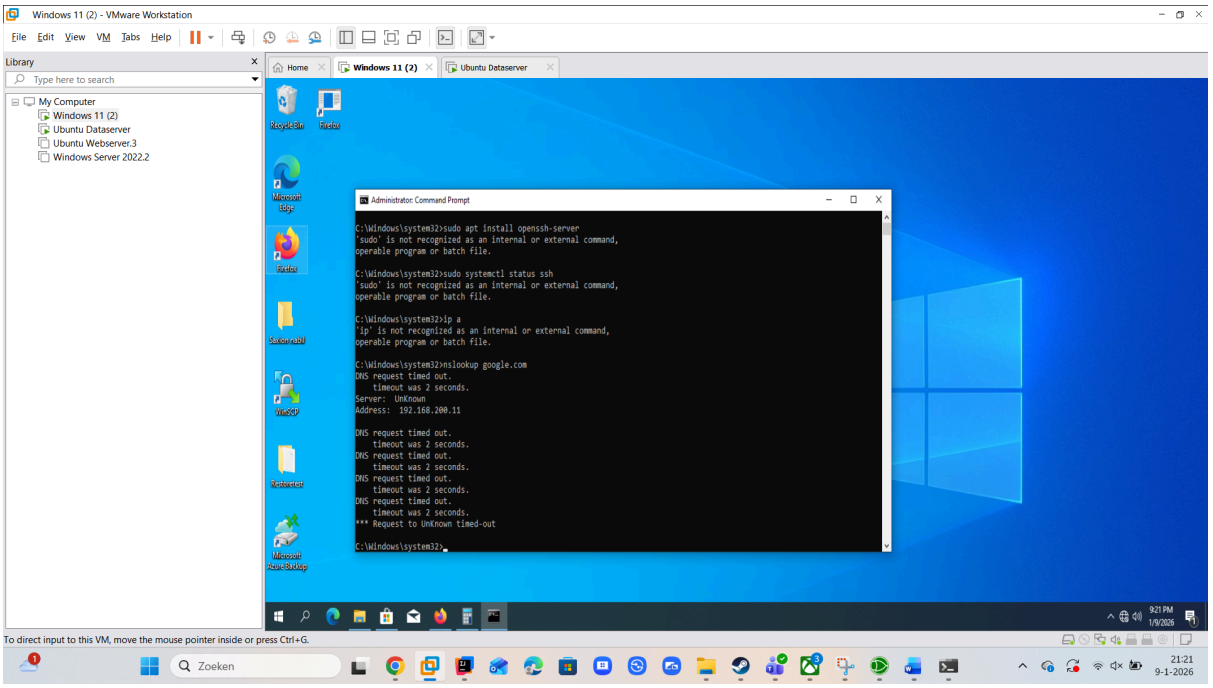
In deze omgeving werkte de netwerkverbinding tussen de virtuele machines niet correct, waardoor het uitvoeren van SSH en SCP niet mogelijk was. De werking en het doel van deze commando's zijn echter wel begrepen en correct toegepast in theorie.

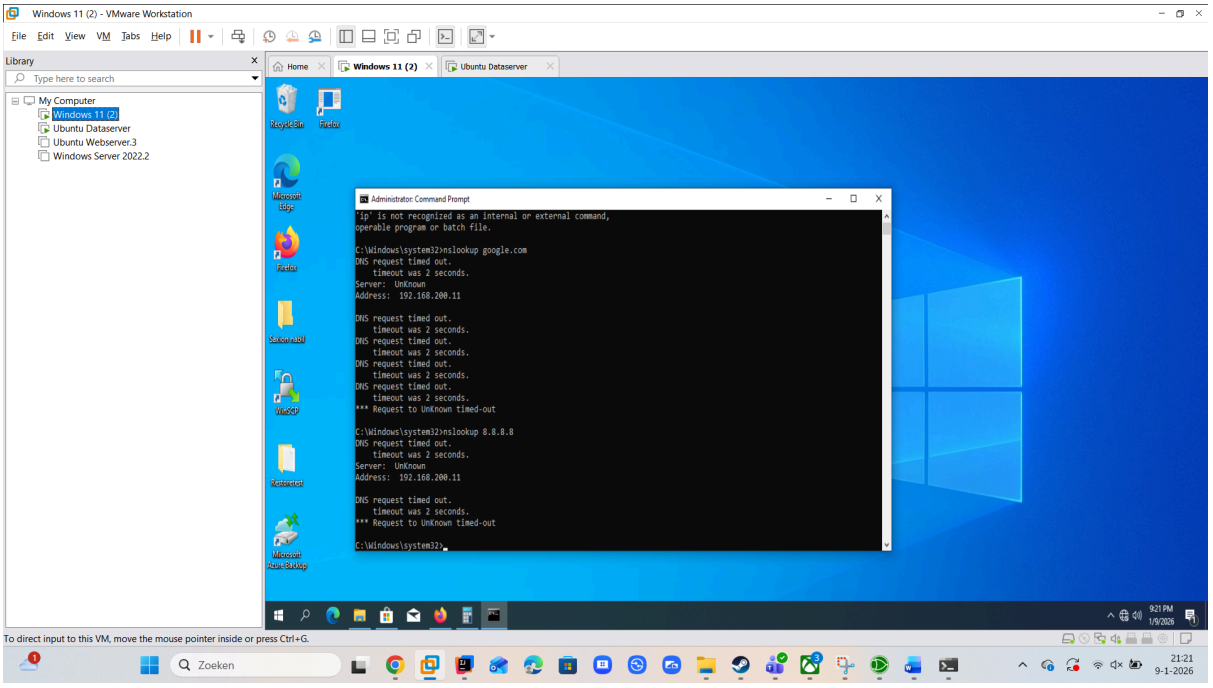
Week 6 — Assignment 6.2: Ping & connectivity



Het ping-commando naar google.com gaf een foutmelding door een probleem met DNS. Door te pingen naar het IP-adres 8.8.8.8 kon worden vastgesteld dat de netwerkverbinding wel werkte, maar dat naamresolutie niet beschikbaar was.

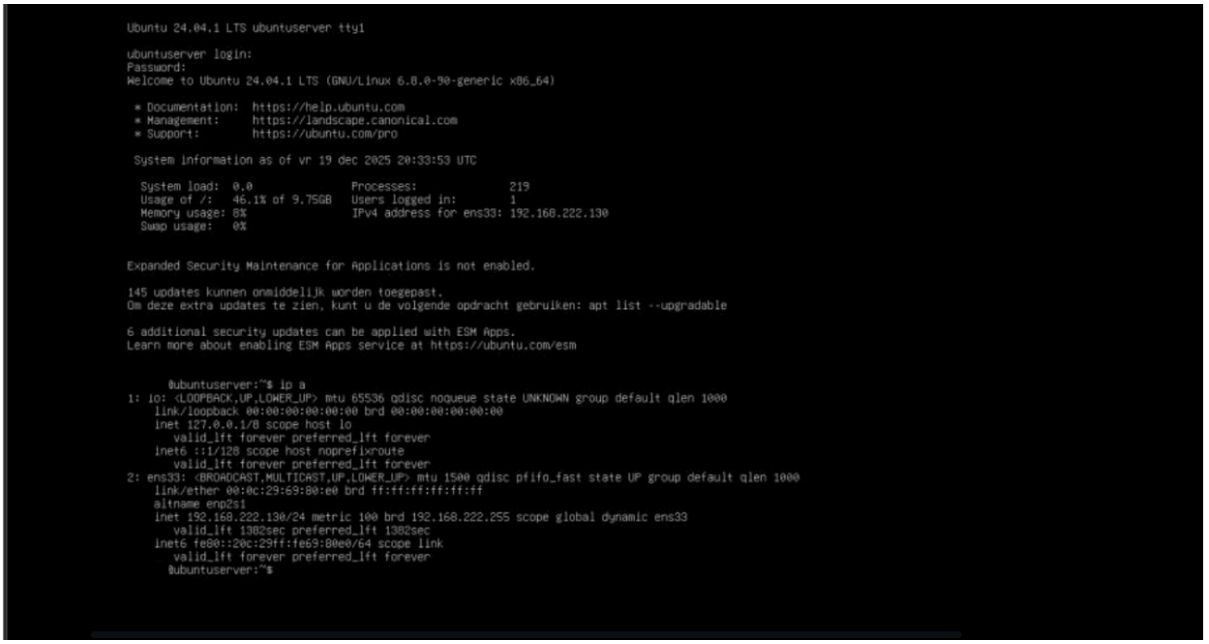
Week 6 — Assignment 6.3: DNS



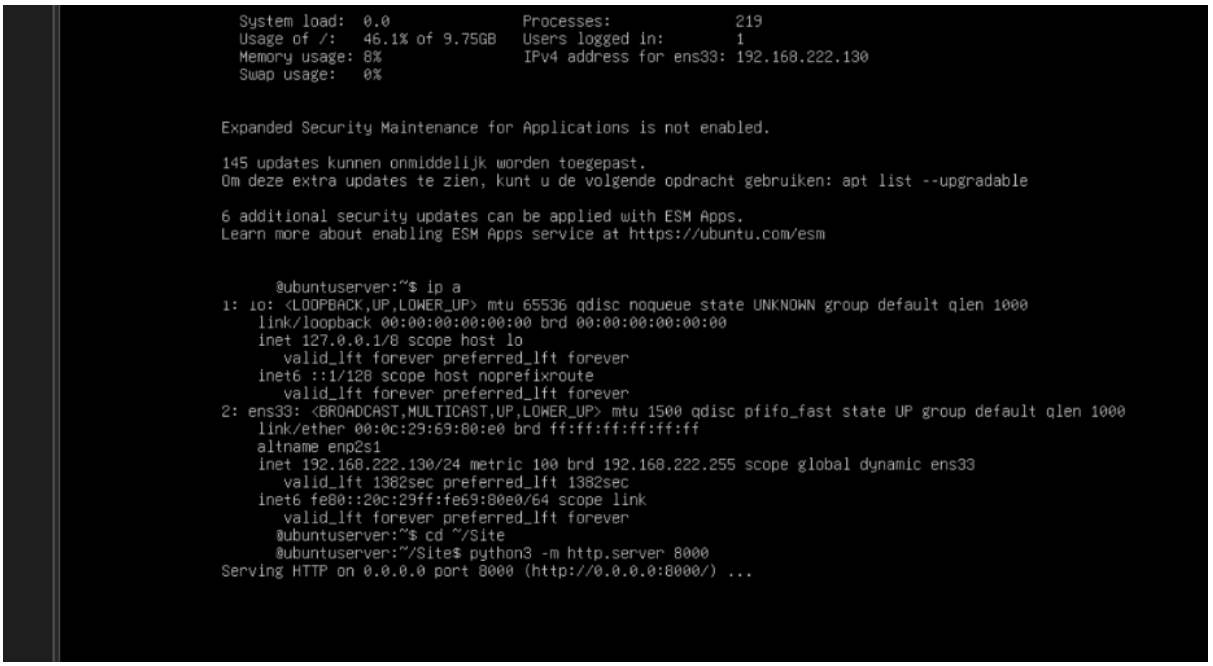


Met het commando nslookup is getest of DNS-resolutie werkt.
Het opzoeken van het domein google.com gaf geen resultaat door een probleem met DNS.
Dit laat zien dat naamomzetting afhankelijk is van werkende DNS-servers, ook als de netwerkverbinding zelf actief is.

Week 6 — Assignment 6.4: Traceroute



Screenshot python3 webserver command:



Assignment 6.5: Network segment

IP-adres: 192.168.10.75
Subnetmasker: 255.255.255.240 (/28)
192.168.10.75 =
11000000.10101000.00001010.01001011

255.255.255.240 =
11111111.11111111.11111111.11110000

11000000.10101000.00001010.01001011
AND
11111111.11111111.11111111.11110000

11000000.10101000.00001010.01000000

192.168.10.64

Door het IP-adres en subnetmasker om te zetten naar binaire waarden kan met een AND-operatie het netwerkadres worden berekend.
Bij een /28 subnet bestaat elk netwerksegment uit 16 IP-adressen.
Hiermee kan het bijbehorende IP-bereik van het netwerksegment worden bepaald.

```
package part2;

import nl.saxion.app.SaxionApp;

public class Application2 implements Runnable {

    public static void main(String[] args) {
        SaxionApp.start(new Application2(), 500, 500);
    }

    @Override
    public void run() {

        SaxionApp.printLine("Enter IP address (e.g. 192.168.10.75):");
        String ipInput = SaxionApp.readString();

        SaxionApp.printLine("Enter subnet mask (e.g. 255.255.255.240):");
        String maskInput = SaxionApp.readString();

        int[] ipBinary = toBinary(ipInput);
        int[] maskBinary = toBinary(maskInput);

        if (ipBinary == null || maskBinary == null) {
```

```
            SaxionApp.printLine("Invalid input format.");
            return;
        }

        int[] networkBinary = new int[32];
        for (int i = 0; i < 32; i++) {
            networkBinary[i] = ipBinary[i] & maskBinary[i];
        }

        SaxionApp.printLine("\nResults:");
        SaxionApp.printLine("IP (binary): " + format(ipBinary));
        SaxionApp.printLine("Subnet (binary): " + format(maskBinary));
        SaxionApp.printLine("Network (binary): " + format(networkBinary));

        String networkDecimal = toDecimal(networkBinary);
        SaxionApp.printLine("Network address (decimal): " + networkDecimal);

        showRange(networkBinary, maskBinary);
    }

    private int[] toBinary(String input) {
        String[] parts = input.split("\\.");
        if (parts.length != 4) return null;

        int[] binary = new int[32];
        for (int i = 0; i < 4; i++) {
            int value;
            try {
                value = Integer.parseInt(parts[i]);
            } catch (Exception e) {
                return null;
            }

            for (int b = 7; b >= 0; b--) {
                binary[i * 8 + b] = value & 1;
                value >>= 1;
            }
        }
        return binary;
    }

    private String format(int[] bits) {
        String result = "";
        for (int i = 0; i < bits.length; i++) {
            result += bits[i];
            if ((i + 1) % 8 == 0 && i != bits.length - 1) {
                result += ".";
            }
```

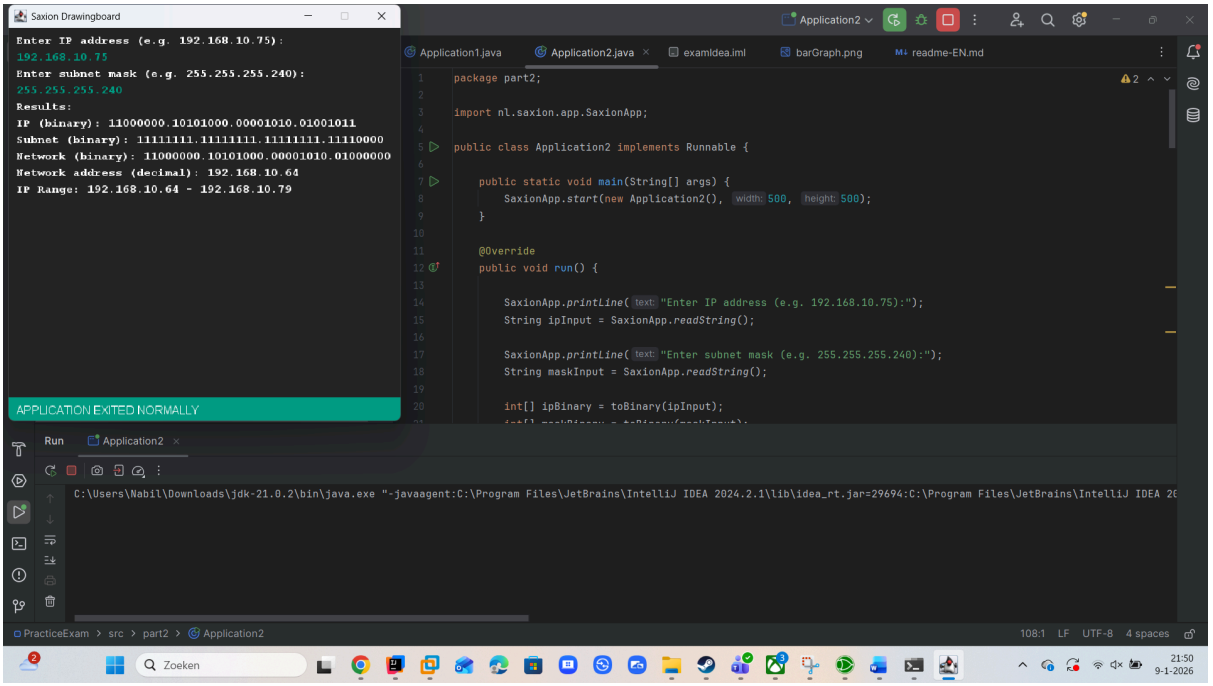
```
        }
        return result;
    }

    private String toDecimal(int[] bits) {
        String result = "";
        for (int i = 0; i < 4; i++) {
            int value = 0;
            for (int b = 0; b < 8; b++) {
                value = (value << 1) | bits[i * 8 + b];
            }
            result += value;
            if (i != 3) result += ".";
        }
        return result;
    }

    private void showRange(int[] network, int[] mask) {
        int hostBits = 0;
        for (int bit : mask) {
            if (bit == 0) hostBits++;
        }

        int[] broadcast = network.clone();
        for (int i = 31; i >= 32 - hostBits; i--) {
            broadcast[i] = 1;
        }

        SaxionApp.printLine(
                "IP Range: " +
                toDecimal(network) +
                " - " +
                toDecimal(broadcast)
        );
    }
}
```


Saxion Drawingboard
Enter IP address (e.g. 192.168.10.75):
192.168.10.75
Enter subnet mask (e.g. 255.255.255.240):
255.255.255.240
Results:
IP (binary): 11000000.10101000.00001010.01001011
Subnet (binary): 11111111.11111111.11111111.11110000
Network (binary): 11000000.10101000.00001010.01000000
Network address (decimal): 192.168.10.64
IP Range: 192.168.10.64 - 192.168.10.79
APPLICATION EXITED NORMALLY
Application2
Application1.java
Application2.java
examIdea.iml
barGraph.png
readme-EN.md
package part2;
import nl.saxion.app.SaxionApp;
public class Application2 implements Runnable {
public static void main(String[] args) {
SaxionApp.start(new Application2(), width: 500, height: 500);
}
@Override
public void run() {
SaxionApp.printLine(text: "Enter IP address (e.g. 192.168.10.75):");
String ipInput = SaxionApp.readString();
SaxionApp.printLine(text: "Enter subnet mask (e.g. 255.255.255.240):");
String maskInput = SaxionApp.readString();
int[] ipBinary = toBinary(ipInput);
Run
Application2
C:\Users\Nabil\Downloads\jdk-21.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.1\lib\idea_rt.jar=29694:C:\Program Files\JetBrains\IntelliJ IDEA 20
PracticeExam > src > part2 > Application2
108:1 LF UTF-8 4 spaces
Zoeken
21:50
9-1-2026