

# 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.

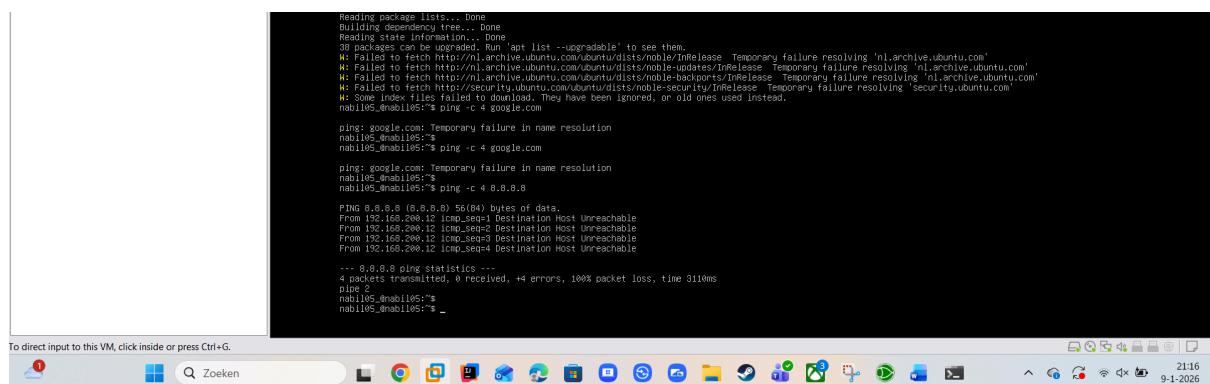
Vervolgens wordt het IP-adres van de server opgezocht, zodat vanaf de eigen laptop een verbinding kan worden gemaakt.

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.

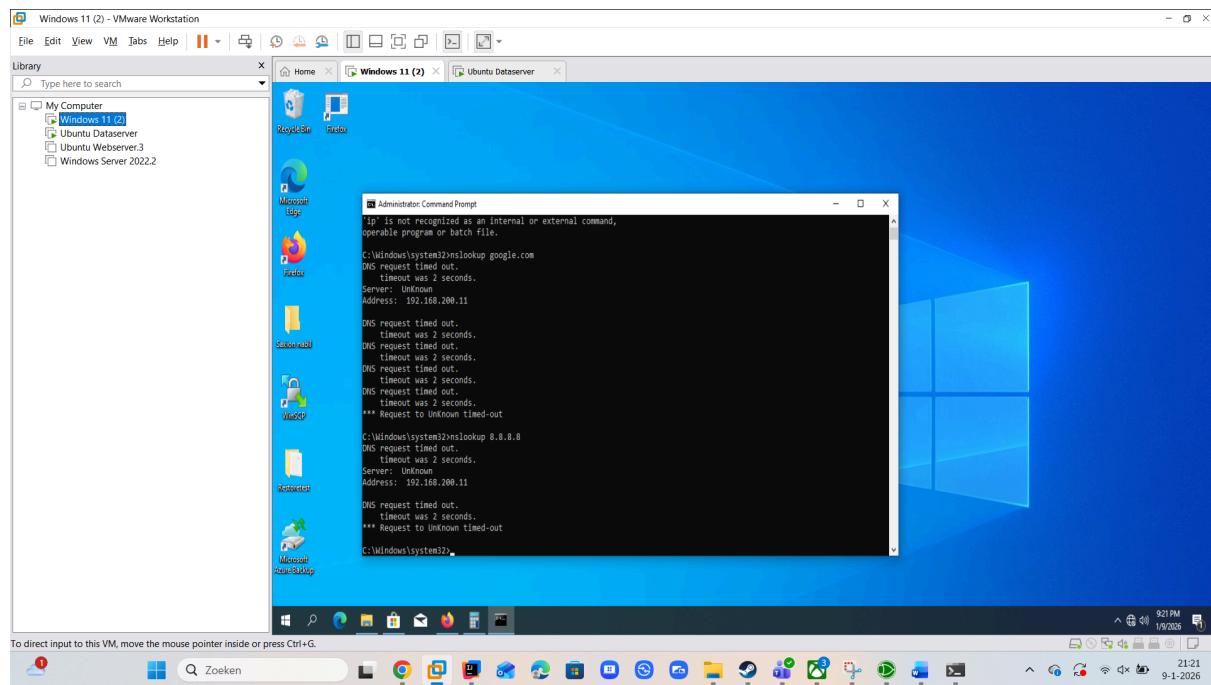
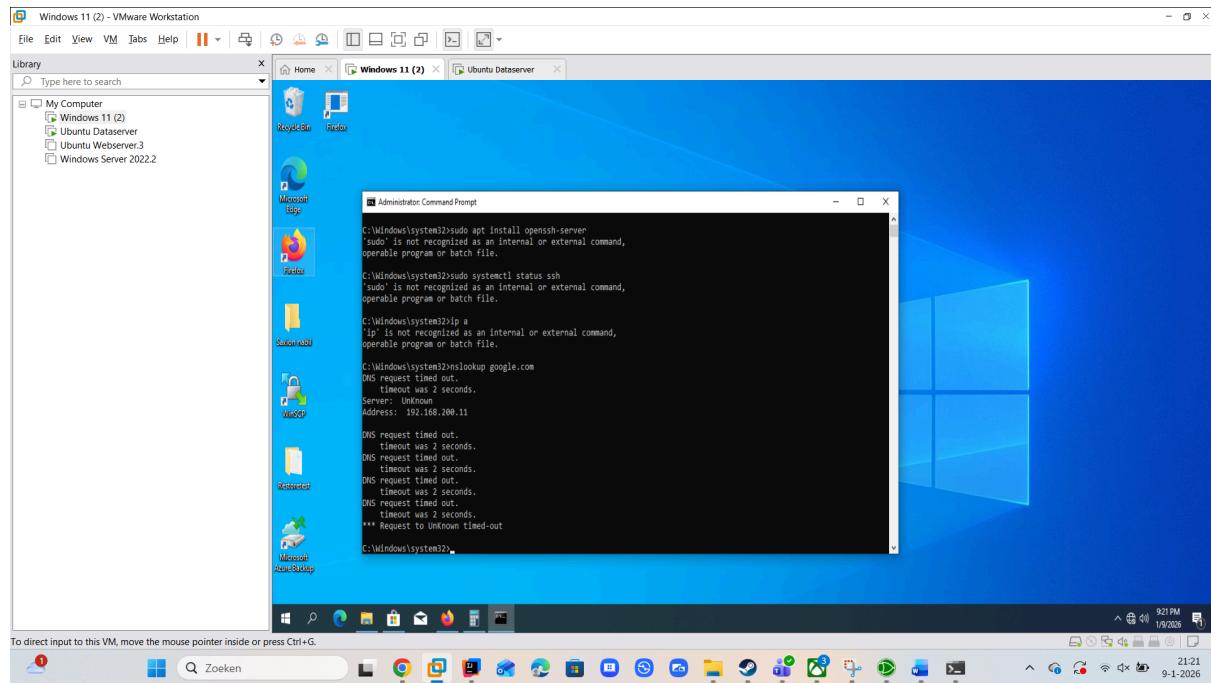
### Week 6 — Assignment 6.2: Ping & connectivity



```
Reading package lists... done
Building dependency tree... done
Reading status information... done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
You might want to run 'apt-get update' first.
W: Failed to fetch http://nl.archive.ubuntu.com/ubuntu/dists/noble/InRelease Temporary failure resolving 'nl.archive.ubuntu.com'
W: Failed to fetch http://nl.archive.ubuntu.com/ubuntu/dists/noble-updates/InRelease Temporary failure resolving 'nl.archive.ubuntu.com'
W: Failed to fetch http://security.ubuntu.com/ubuntu/dists/noble-security/InRelease Temporary failure resolving 'security.ubuntu.com'
W: Some index files failed to download. They have been ignored, or old ones used instead.
nab105.enable05:~$ ping -c 4 google.com
ping: google.com: Temporary failure in name resolution
nab105.enable05:~$ ping -c 4 google.com
ping: google.com: Temporary failure in name resolution
nab105.enable05:~$ nab105.enable05:~$ ping -c 4 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data
From 192.168.200.12 icmp_seq=2 destination Host Unreachable
From 192.168.200.12 icmp_seq=3 destination Host Unreachable
From 192.168.200.12 icmp_seq=4 destination Host Unreachable
...
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 311ms
ping: 8.8.8.8 ping statistics ...
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 311ms
nab105.enable05:~$ _
```

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

```
Ubuntu 24.04.1 LTS ubuntuserver ttys1
ubuntuserver login:
Password:
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.0.0-90-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of vr 19 dec 2025 20:33:53 UTC

System load: 0.0      Processes:          219
Usage of /: 46.1% of 9.75GB  Users logged in:   1
Memory usage: 8%
Swap usage:  0%

Expanded Security Maintenance for Applications is not enabled.

145 updates kunnen onmiddelijk worden toegepast.
Om deze extra updates te zien, kunt u de volgende opdracht gebruiken: apt list --upgradable

6 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

@ubuntuserver:"$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host noprefixroute
            valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:69:80:e0 brd ff:ff:ff:ff:ff:ff
        altname enp2s1
        inet 192.168.222.130/24 metric 100 brd 192.168.222.255 scope global dynamic ens33
            valid_lft 1382sec preferred_lft 1382sec
        inet6 fe80::20c:29ff:fe69:80e0/64 scope link
            valid_lft forever preferred_lft forever
@ubuntuserver:"$
```

## Screenshot python3 webserver command:

```
System load:  0.0      Processes:          219
Usage of /: 46.1% of 9.75GB  Users logged in:   1
Memory usage: 8%
Swap usage:  0%

Expanded Security Maintenance for Applications is not enabled.

145 updates kunnen onmiddelijk worden toegepast.
Om deze extra updates te zien, kunt u de volgende opdracht gebruiken: apt list --upgradable

6 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

@ubuntuserver:"$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host noprefixroute
            valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:69:80:e0 brd ff:ff:ff:ff:ff:ff
        altname enp2s1
        inet 192.168.222.130/24 metric 100 brd 192.168.222.255 scope global dynamic ens33
            valid_lft 1382sec preferred_lft 1382sec
        inet6 fe80::20c:29ff:fe69:80e0/64 scope link
            valid_lft forever preferred_lft forever
@ubuntuserver:"$ cd ~/Site
@ubuntuserver:"$ cd ~/Site
@ubuntuserver:"$ /Sites/python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

## 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.println("Invalid input format.");
        return;
    }

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

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

    String networkDecimal = toDecimal(networkBinary);
    SAXIONAPP.println("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)
    );
}
}

```

The screenshot shows the IntelliJ IDEA interface with a Java application running. On the left, a terminal window titled "Saxion Drawingboard" displays the output of a script asking for IP and subnet mask input. On the right, the code editor shows the Java source code for "Application2.java". The code imports "nl.saxion.app.SaxionApp" and defines a class "Application2" that implements Runnable. It prints prompts for IP and subnet mask, reads them, converts them to binary, and then runs. The application exits normally. Below the code editor is the "Run" tool bar with the "Application2" configuration selected. The status bar at the bottom shows the file path "PracticeExam > src > part2 > Application2", the build configuration "108:1 LF UTF-8 4 spaces", and the date/time "21:50 9-1-2026".

```
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
```

```
1 package part2;
2
3 import nl.saxion.app.SaxionApp;
4
5 public class Application2 implements Runnable {
6
7     public static void main(String[] args) {
8         SaxionApp.start(new Application2(), width: 500, height: 500);
9     }
10
11     @Override
12     public void run() {
13
14         SaxionApp.printLine(text: "Enter IP address (e.g. 192.168.10.75):");
15         String ipInput = SaxionApp.readString();
16
17         SaxionApp.printLine(text: "Enter subnet mask (e.g. 255.255.255.240):");
18         String maskInput = SaxionApp.readString();
19
20         int[] ipBinary = toBinary(ipInput);
21         int[] maskBinary = toBinary(maskInput);
22
23         int[] networkBinary = AND(ipBinary, maskBinary);
24
25         int networkDecimal = toDecimal(networkBinary);
26
27         int startIP = toIP(ipInput);
28         int endIP = toIP(ipInput);
29
30         startIP = networkDecimal;
31         endIP = networkDecimal + 1;
32
33         for (int i = startIP; i <= endIP; i++) {
34             int[] ipOctets = toOctet(i);
35             String ipString = toIP(ipOctets);
36             String subnetString = toIP(maskBinary);
37             String networkString = toIP(networkBinary);
38
39             drawBarGraph(ipString, subnetString, networkString);
40         }
41
42     }
43
44     private int[] toBinary(String ipString) {
45         String[] octets = ipString.split("\\.");
46         int[] binaryOctets = new int[4];
47
48         for (int i = 0; i < 4; i++) {
49             binaryOctets[i] = Integer.parseInt(octets[i], 10);
50         }
51
52         return binaryOctets;
53     }
54
55     private int[] toOctet(int decimalIP) {
56         int[] octet = new int[4];
57
58         for (int i = 0; i < 4; i++) {
59             octet[i] = decimalIP % 256;
60             decimalIP /= 256;
61         }
62
63         return octet;
64     }
65
66     private String toIP(int[] octet) {
67         String ipString = "";
68
69         for (int i = 0; i < 4; i++) {
70             ipString += octet[i] + ".";
71         }
72
73         return ipString.substring(0, ipString.length() - 1);
74     }
75
76     private int[] AND(int[] ip, int[] mask) {
77         int[] result = new int[4];
78
79         for (int i = 0; i < 4; i++) {
80             result[i] = ip[i] & mask[i];
81         }
82
83         return result;
84     }
85
86     private int toDecimal(int[] binaryOctets) {
87         int decimalIP = 0;
88
89         for (int i = 0; i < 4; i++) {
90             decimalIP += binaryOctets[i] * Math.pow(256, i);
91         }
92
93         return decimalIP;
94     }
95
96     private void drawBarGraph(String ip, String subnet, String network) {
97
98     }
```