**Exercise 1**

For this exercise, I have decided to create 3 subnets, with star topology, and connected between them via a central router, which contains 3 IP addresses, one from each of the subnets. Inside each network, there is also a switch, to make more efficient the communication between the devices.

To prevent infection between the different subnets, firewalls should be used, to prevent malware from reaching the router and being distributed to the other networks.

For the software developers, who need 2 IP addresses each, I have decided to assign 2 devices to each of them, with one IP per device. As an alternative, we could assign 2 IP addresses to each device by using IP dual stack, or we could also assign the second IP to a virtual machine of the device. However, to make the design simpler, I have decided to go for 2 computers per software developer, each of them with only 1 IP address.

To make the network diagram easier to read, only the first and the last IP address of the devices on each network are visible in the diagram. The complete list of the IP addresses can be found in the Part b of the exercise, in the tables.

**PART A**

**A diagram of a network diagram

Description automatically generated**

**PART B**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Subnet** | **Network Address** | **Broadcast Address** | **Mask** | **Number of usable hosts** |
| 1 | 192.168.10.0 | 192.168.10.127 | 255.255.255.128 | 126 |
| 2 | 192.168.10.128 | 192.168.10.191 | 255.255.255.192 | 62 |
| 3 | 192.168.10.192 | 192.168.10.255 | 255.255.255.192 | 62 |

Network 1 (software developers)

|  |  |
| --- | --- |
| **Device** | **IP** |
| Router | 192.168.10.1 |
| Device 1 (software developer 1) | 192.168.10.2 |
| Device 2 (software developer 1) | 192.168.10.3 |
| Device 3 (software developer 2) | 192.168.10.4 |
| Device 4 (software developer 2) | 192.168.10.5 |
| Device 5 (software developer 3) | 192.168.10.6 |
| Device 6 (software developer 3) | 192.168.10.7 |
| Device 7 (software developer 4) | 192.168.10.8 |
| Device 8 (software developer 4) | 192.168.10.9 |
| Device 9 (software developer 5) | 192.168.10.10 |
| Device 10 (software developer 5) | 192.168.10.11 |
| Device 11 (software developer 6) | 192.168.10.12 |
| Device 12 (software developer 6) | 192.168.10.13 |
| Device 13 (software developer 7) | 192.168.10.14 |
| Device 14 (software developer 7) | 192.168.10.15 |
| Device 15 (software developer 8) | 192.168.10.16 |
| Device 16 (software developer 8) | 192.168.10.17 |
| Device 17 (software developer 9) | 192.168.10.18 |
| Device 18 (software developer 9) | 192.168.10.19 |
| Device 19 (software developer 10) | 192.168.10.20 |
| Device 20 (software developer 10) | 192.168.10.21 |
| Device 21 (system administrator 1) | 192.168.10.22 |
| Device 22 (system administrator 2) | 192.168.10.23 |
| Empty IPs range | 192.168.10.24 to 192.168.10.126 |

Network 2 (economics)

|  |  |
| --- | --- |
| **Device** | **IP** |
| Router | 129.168.10.129 |
| Device 1 | 192.168.10.130 |
| Device 2 | 192.168.10.131 |
| Device 3 | 192.168.10.132 |
| Device 4 | 192.168.10.133 |
| Device 5 | 192.168.10.134 |
| Device 6 | 192.168.10.135 |
| Device 7 | 192.168.10.136 |
| Device 8 | 192.168.10.137 |
| Device 9 | 192.168.10.138 |
| Device 10 | 192.168.10.139 |
| Device 11 | 192.168.10.140 |
| Device 12 | 192.168.10.141 |
| Device 13 | 192.168.10.142 |
| Device 14 | 192.168.10.143 |
| Device 15 | 192.168.10.144 |
| Empty IPs range | 192.168.10.145 to 192.168.10.190 |

Network 3 (IoT devices)

|  |  |
| --- | --- |
| **Device** | **IP** |
| Router | 192.168.10.193 |
| Device 1 | 192.168.10.194 |
| Device 2 | 192.168.10.195 |
| Device 3 | 192.168.10.196 |
| Device 4 | 192.168.10.197 |
| Device 5 | 192.168.10.198 |
| Device 6 | 192.168.10.199 |
| Device 7 | 192.168.10.200 |
| Device 8 | 192.168.10.201 |
| Device 9 | 192.168.10.202 |
| Empty IPs range | 192.168.10.203 to 192.168.10.254 |

Central Router

|  |  |
| --- | --- |
| **Device** | **IPs** |
| Router | 192.168.10.1 / 192.168.10.129 / 192.168.10.193 |

**Exercise 2**

**SECTION D**

These are all the commands (written within quotes ””) needed for this section, and all the steps followed:

* *“docker network create private-network”* 🡪 command to create a network for the containers.
* *“docker run --rm -d -p 8001:8001 --name redis --network private-network redis/redis-stack:latest”* 🡪 command to run the Redis container in the private network previously created.
* The Dockerfile is modified with the appropriate Redis\_server value, which is the name assigned to the redis container.
* From the directory where the Dockerfile is saved, execute *“docker build -t flask-image .”* 🡪 command to build the image after modifying the Dockerfile (content details below)
* *“* *cd app/”* 🡪 to change to the app directory in the host machine.
* *“docker run -d -p 5000:5000 -v $(pwd):/app --name flask-server --network private-network flask-image”* 🡪 command to launch the flask server in a container mapped to our app directory, and to our port 5000. The container will be run in the same network as the Redis container, so they can communicate.
* http://localhost:5000 shows the incremental counter.

The final content of the Dockerfile is:

*FROM python:3.8-slim*

*COPY requirements.txt .*

*RUN pip install --no-cache-dir -r /requirements.txt*

*ENV FLASK\_APP=app.py*

*ENV FLASK\_RUN\_HOST=0.0.0.0*

*ENV FLASK\_ENV=development*

*ENV REDIS\_SERVER=redis*

*WORKDIR /app*

*ENTRYPOINT [“flask”, “run”, “--host=0.0.0.0”, “--debug”]*