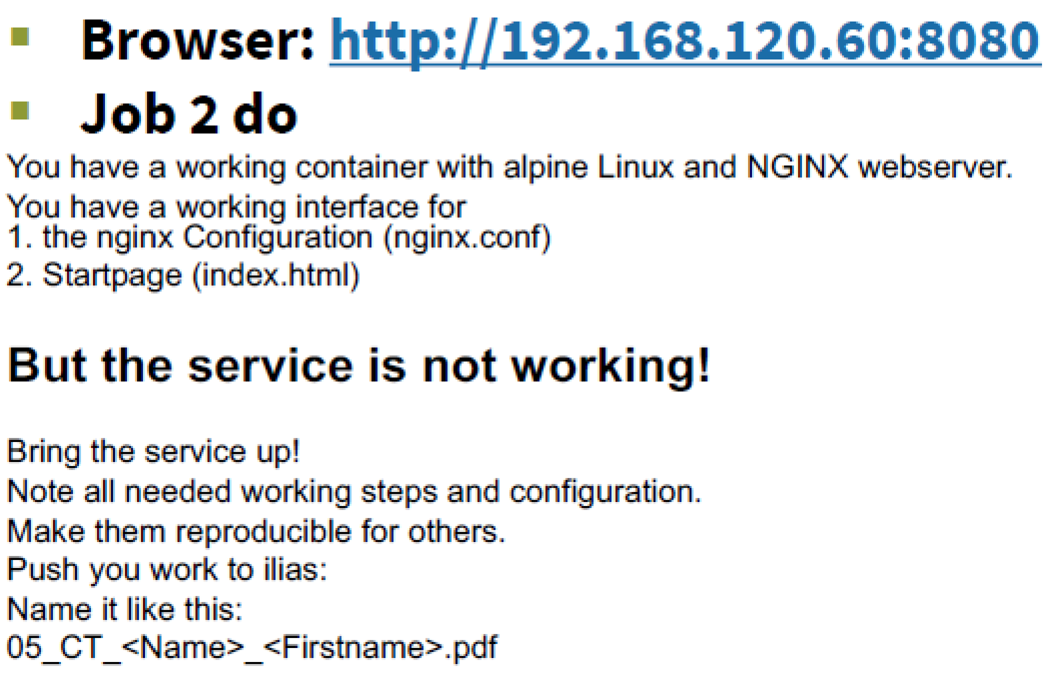
JOB 2

HF Informatik & Telekommunikation
Bern JUNI 2024
24-C

Mehmet Ali Gür
mgu153457@stud.gibb.ch

HF Informatik & Telekommunikation
Höhere Fachschule



- **Browser: http://192.168.120.60:8080**
- **Job 2 do**

You have a working container with alpine Linux and NGINX webserver.
You have a working interface for
1. the nginx Configuration (nginx.conf)
2. Startpage (index.html)

But the service is not working!

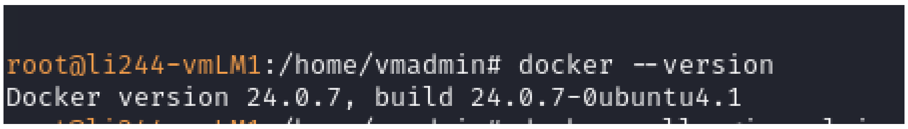
Bring the service up!
Note all needed working steps and configuration.
Make them reproducible for others.
Push you work to ilias:
Name it like this:
05_CT_<Name>_<Firstname>.pdf

Project: Running an NGINX Web Server in an Alpine Linux Container

Explanation: This project aims to install an NGINX web server inside an Alpine Linux Docker container running on Ubuntu operating system. Within the scope of the project, Docker will be installed, the firewall will be configured to allow incoming traffic over port 8080, and the NGINX server will be deployed using an Alpine-based Docker image. After the container installation, the NGINX configuration will be reviewed and adjustments will be made where necessary. The project will be completed by verifying the functionality of the installed server via a web browser.

Step 1: Verify Docker Installation: First, check if Docker is correctly installed on your Ubuntu machine:

docker –version

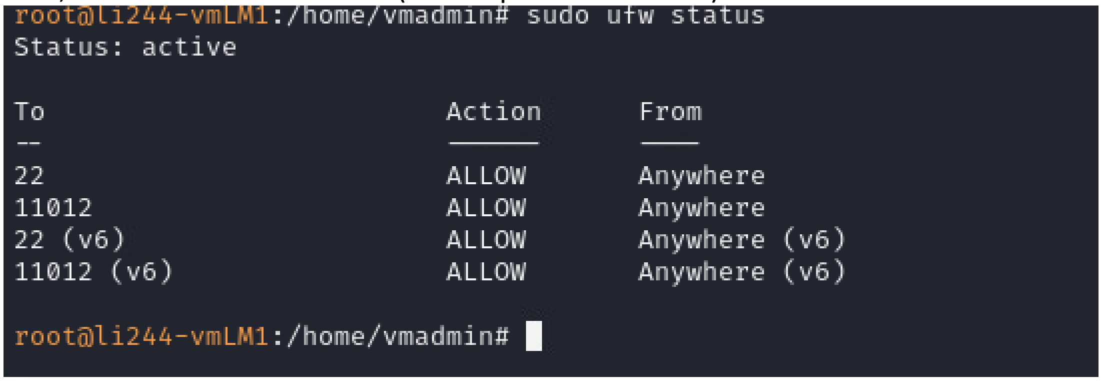


```
root@li244-vmLM1:/home/vmadmin# docker --version
Docker version 24.0.7, build 24.0.7-0ubuntu4.1
```

If Docker is not installed, you can install it using the following commands:
sudo apt-get update
sudo apt-get install -y docker.io
sudo systemctl start docker
sudo systemctl enable docker

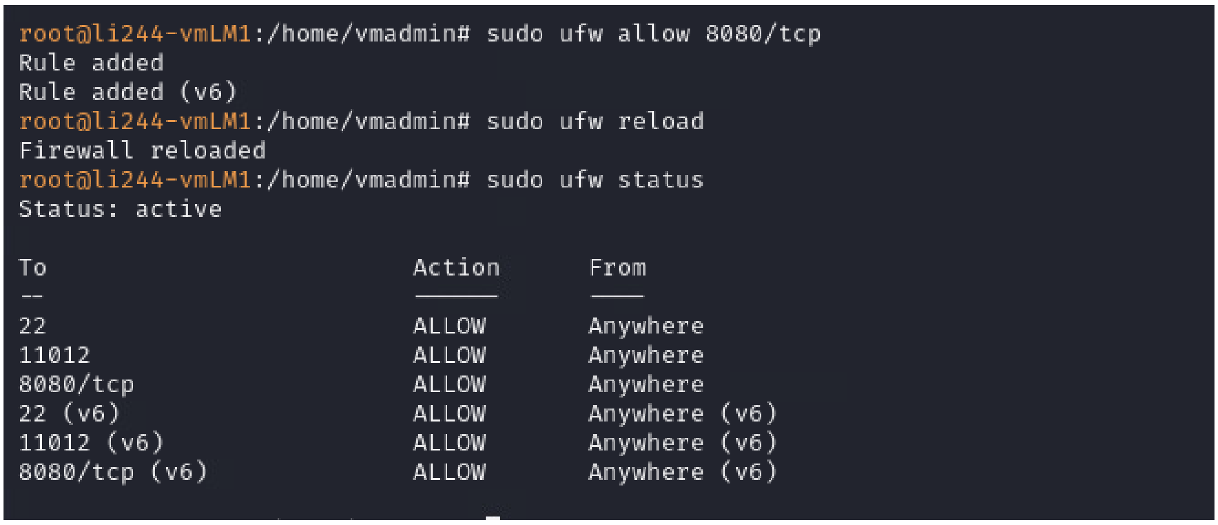
These commands will install Docker, start the service, and ensure Docker starts automatically on system boot.

Step 2: Set Up Firewall Rules
Before proceeding, it's important to configure the firewall to allow traffic on the necessary ports. In this case, we need to allow traffic on port 8080 (which we'll map to port 80 inside the container).
First, check the status of UFW (Uncomplicated Firewall):



```
root@li244-vmLM1:/home/vmadmin# sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
11012                      ALLOW       Anywhere
22 (v6)                    ALLOW       Anywhere (v6)
11012 (v6)                 ALLOW       Anywhere (v6)

root@li244-vmLM1:/home/vmadmin#
```

If it's not enabled, you can enable it:
sudo ufw enable
Now, allow traffic on port 8080:
sudo ufw allow 8080/tcp
Reload the firewall to apply the changes:
sudo ufw reload



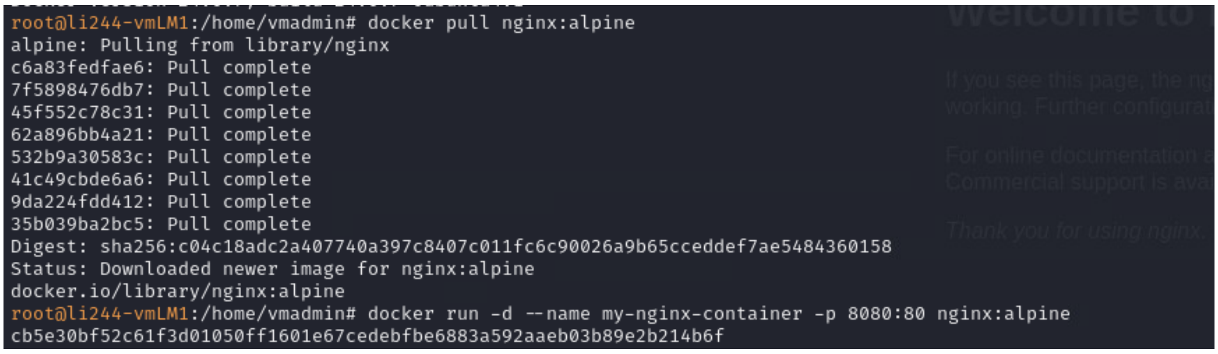
```
root@li244-vmLM1:/home/vmadmin# sudo ufw allow 8080/tcp
Rule added
Rule added (v6)
root@li244-vmLM1:/home/vmadmin# sudo ufw reload
Firewall reloaded
root@li244-vmLM1:/home/vmadmin# sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
11012                      ALLOW       Anywhere
8080/tcp                   ALLOW       Anywhere
22 (v6)                    ALLOW       Anywhere (v6)
11012 (v6)                 ALLOW       Anywhere (v6)
8080/tcp (v6)              ALLOW       Anywhere (v6)
```

Step 3: Create an Alpine Linux and NGINX Container

Now that Docker is installed and the firewall is configured, we can create a container based on Alpine Linux that runs the NGINX web server.

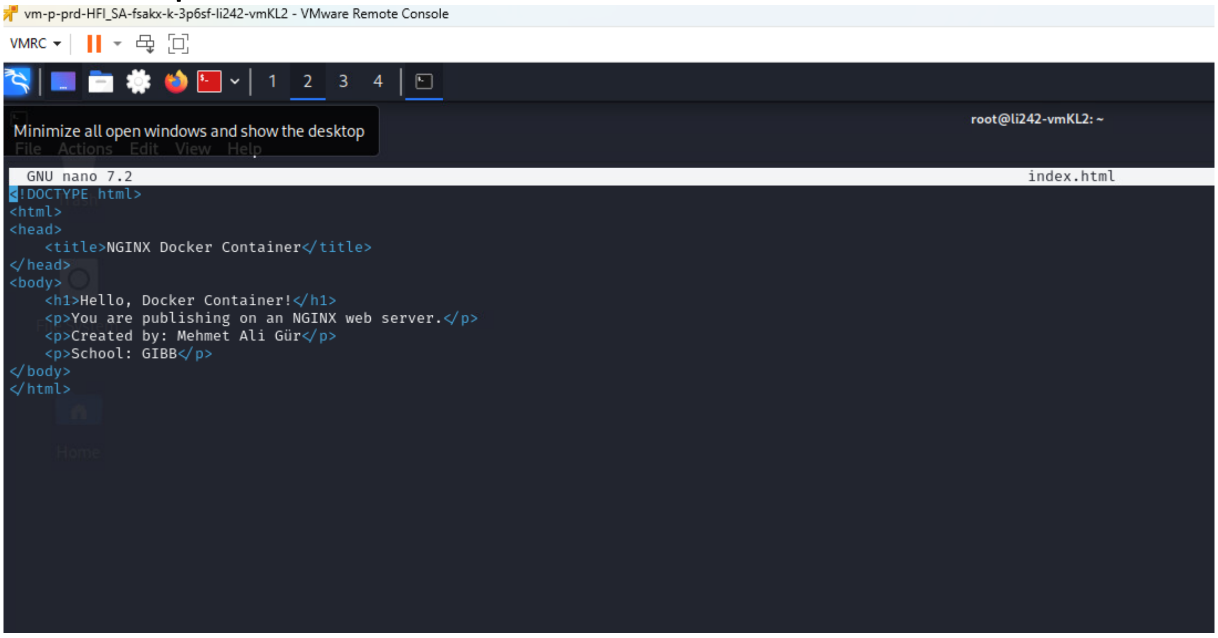
Pull the Alpine-Based NGINX Image
Download the Alpine-based NGINX image from Docker Hub:
docker pull nginx:alpine
After pulling the image, you can start a container using this image:
docker run -d --name my-nginx-container -p 8080:80 nginx:alpine
Explanation of the parameters:

- -d: Runs the container in the background (detached mode).
- --name my-nginx-container: Assigns a name to the container.
- -p 8080:80: Maps port 8080 on the host machine to port 80 inside the container.
- nginx:alpine: Specifies the use of the Alpine-based NGINX image



```
root@li244-vmLM1:/home/vmadmin# docker pull nginx:alpine
alpine: Pulling from library/nginx
c6a83fedfae6: Pull complete
7f5898476db7: Pull complete
45f552c78c31: Pull complete
62a896bb4a21: Pull complete
532b9a30583c: Pull complete
41c49cbde6a6: Pull complete
9da224fdd412: Pull complete
35b039ba2bc5: Pull complete
Digest: sha256:c04c18adc2a407740a397c8407c011fc6c90026a9b65cceddef7ae5484360158
Status: Downloaded newer image for nginx:alpine
docker.io/library/nginx:alpine
root@li244-vmLM1:/home/vmadmin# docker run -d --name my-nginx-container -p 8080:80 nginx:alpine
cb5e30bf52c61f3d01050ff1601e67cedebfbe6883a592aaeb03b89e2b214b6f
```

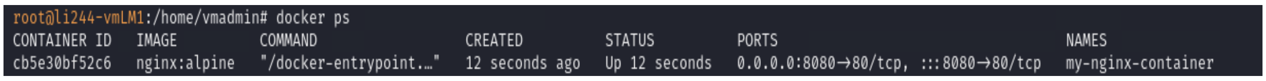
Create a sample HTML content:



vm-p-prd-HFI_SA-fsakx-k-3p6sf-li242-vmKL2 - VMware Remote Console
VMRC
Minimize all open windows and show the desktop
root@li242-vmKL2: ~

```
  GNU nano 7.2                                                                    index.html
<!DOCTYPE html>
<html>
<head>
    <title>NGINX Docker Container</title>
</head>
<body>
    <h1>Hello, Docker Container!</h1>
    <p>You are publishing on an NGINX web server.</p>
    <p>Created by: Mehmet Ali Gür</p>
    <p>School: GIBB</p>
</body>
</html>
```

Step 4: Check the Status of the Container
Ensure that the container is running correctly:
docker ps



```
root@li244-vmLM1:/home/vmadmin# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                                   NAMES
cb5e30bf52c6   nginx:alpine   "/docker-entrypoint.…"   12 seconds ago   Up 12 seconds   0.0.0.0:8080->80/tcp, :::8080->80/tcp   my-nginx-container
```

The container my-nginx-container should appear in the list with the status Up if everything is working properly.

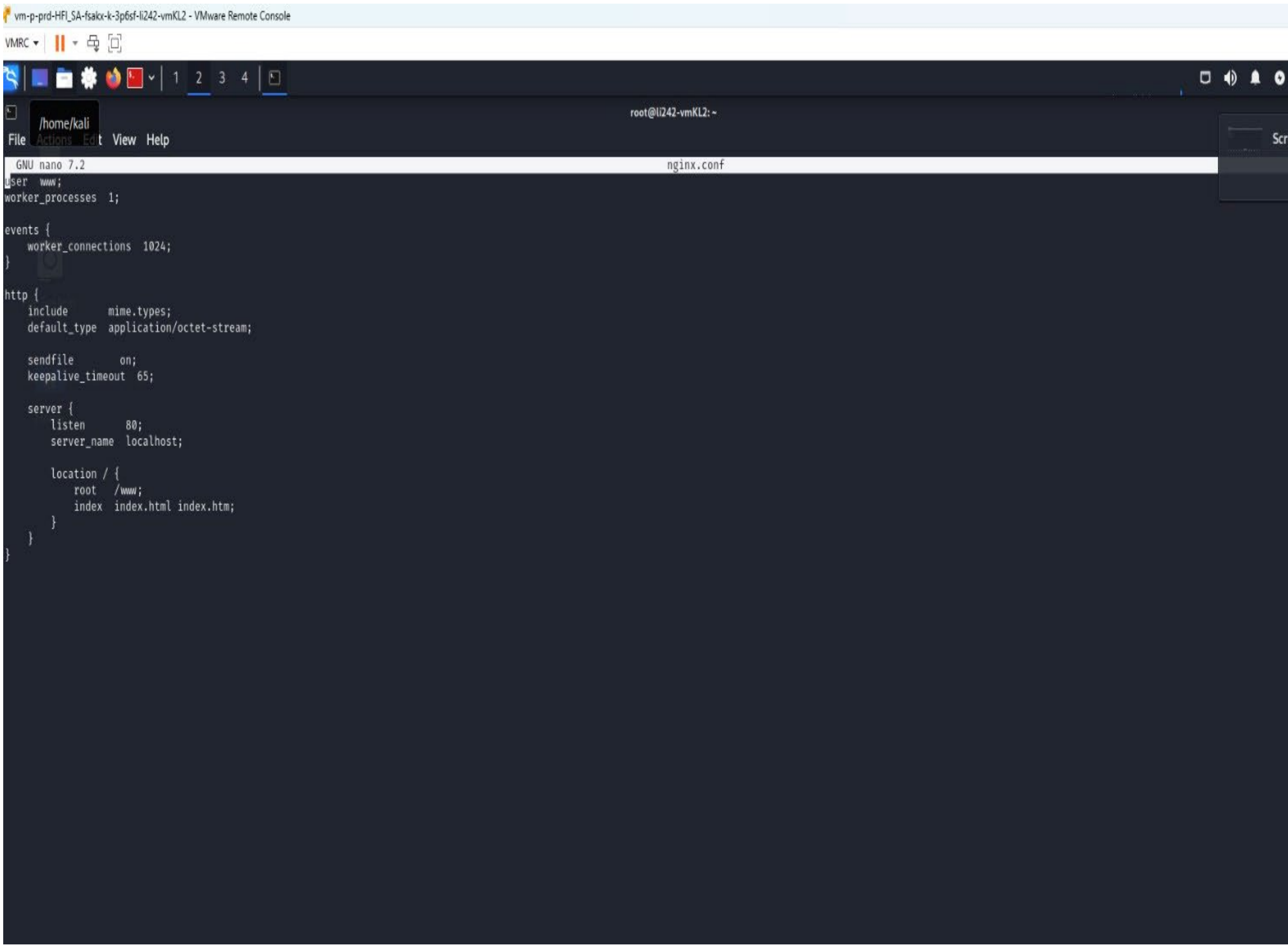
Step 5: Edit the NGINX Configuration File

If you need to customize the NGINX configuration, you can enter the container and edit the configuration file:

docker exec -it my-nginx-container sh
vi /etc/nginx/nginx.conf

After editing the configuration file, reload the NGINX service to apply the changes:

```
user  www;
worker_processes  1;

events {
  worker_connections  1024;
}

http {
  include       mime.types;
  default_type  application/octet-stream;

  sendfile        on;
  keepalive_timeout  65;

  server {
    listen       80;
    server_name  localhost;

    location / {
      root   /www;
      index  index.html index.htm;
    }
  }
}
```


vm-p-prd-HFI_SA-fsakx-k-3p6sf-li242-vmKL2 - VMware Remote Console
VMRC
1 2 3 4
/home/kali
File Actions Edit View Help
root@li242-vmKL2: ~
Scr

```
  GNU nano 7.2                                              nginx.conf
user  www;
worker_processes  1;

events {
    worker_connections  1024;
}

http {
    include       mime.types;
    default_type  application/octet-stream;

    sendfile        on;
    keepalive_timeout  65;

    server {
        listen       80;
        server_name  localhost;

        location / {
            root   /www;
            index  index.html index.htm;
        }
    }
}
```

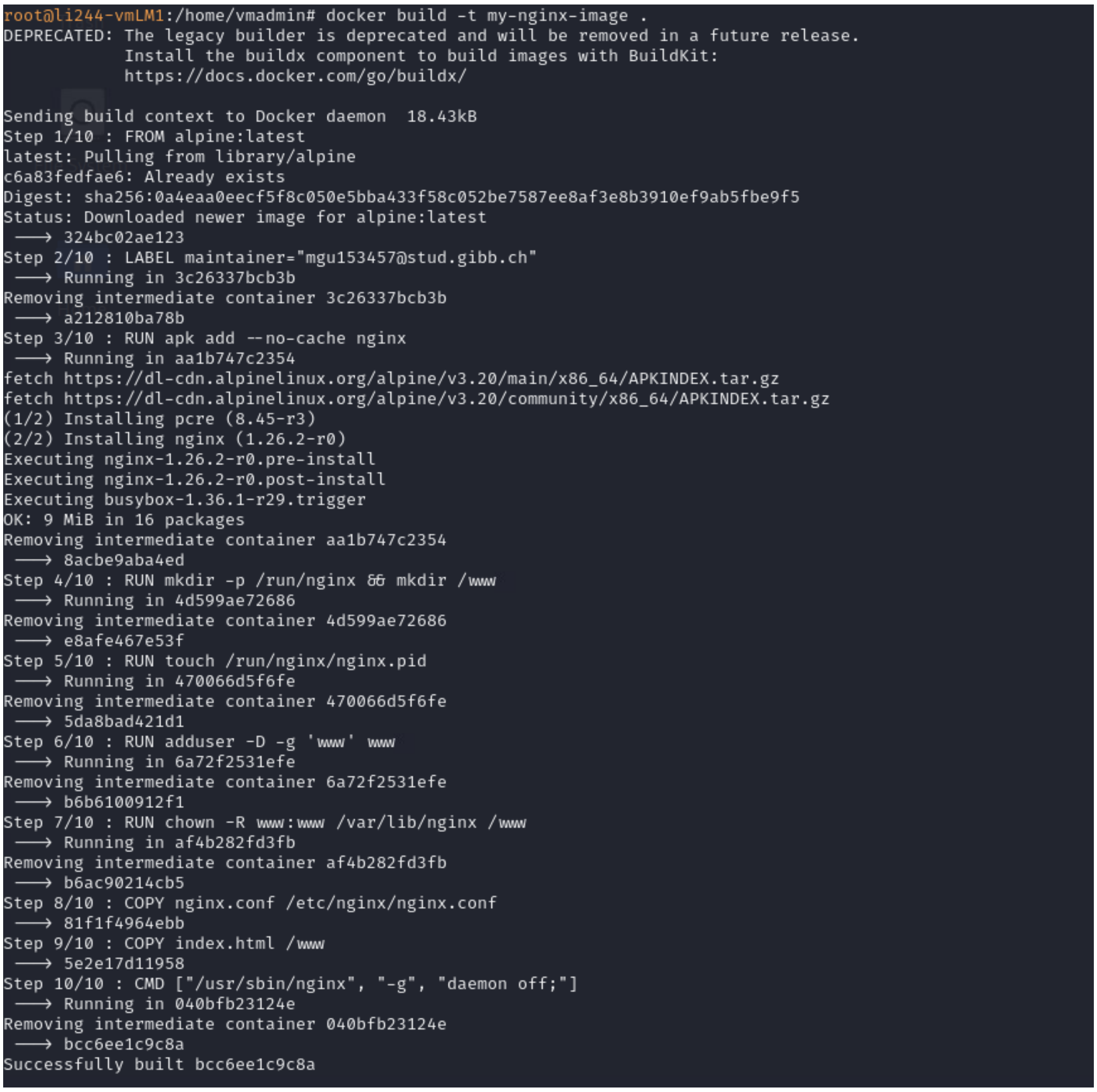
Dockerfile Overview
This Dockerfile sets up an NGINX web server using Alpine Linux as the base image. Below is a summary of the key steps:

1. **Base Image Selection:**
   - FROM alpine:latest: Uses the latest version of Alpine Linux, a lightweight and efficient distribution.
2. **Metadata:**
   - LABEL maintainer="mgu153457@stud-gibb.ch": Specifies the maintainer of the image.
3. **NGINX Installation:**
   - RUN apk add nginx: Installs the NGINX web server using Alpine's package manager, apk.
4. **Directory and File Setup:**
   - Creates necessary directories and files for NGINX to run:
     - /run/nginx directory.
     - nginx.pid file.
5. **User and Group Creation:**
   - RUN adduser -D -g 'www' www: Adds a new user and group named www.
6. **Directory Ownership:**

- Creates the /www directory and sets ownership of /var/lib/nginx and /www to the www user.

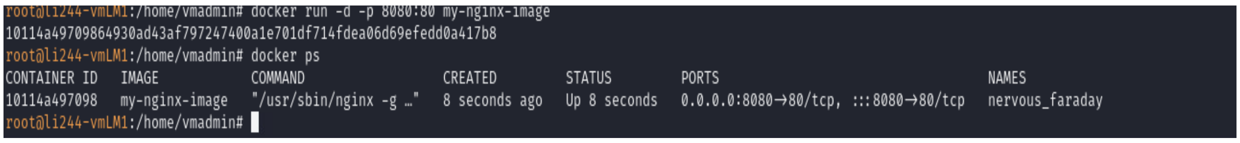
7. **Configuration and Web Content:**
   - COPY nginx.conf /etc/nginx/nginx.conf: Copies the NGINX configuration file into the container.
   - COPY index.html /www: Copies the index.html file to the /www directory.
8. **NGINX Startup:**
   - RUN ["/usr/sbin/nginx"]: Starts the NGINX web server.

Creating the Docker Image
docker build -t my-nginx-image



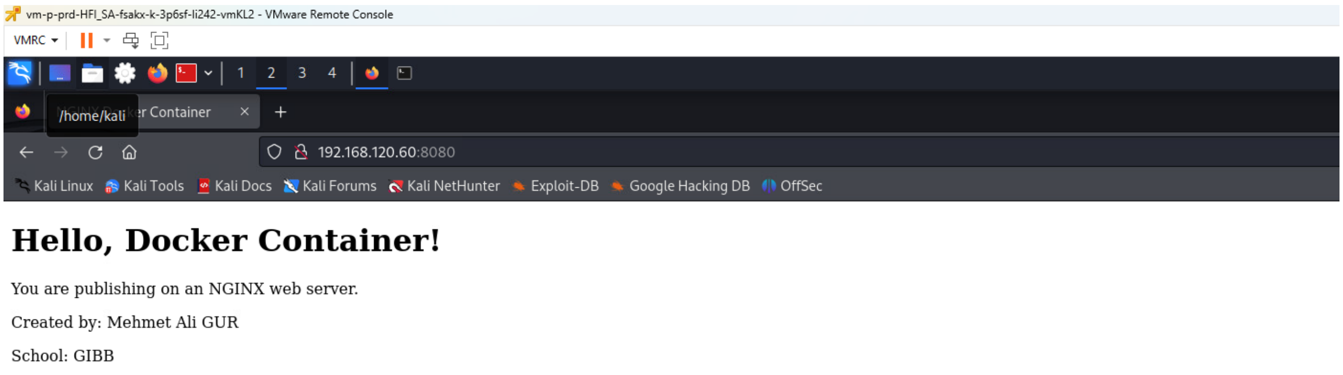
```
root@li244-vmLM1:/home/vmadmin# docker build -t my-nginx-image .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  18.43kB
Step 1/10 : FROM alpine:latest
latest: Pulling from library/alpine
c6a83fedfae6: Already exists
Digest: sha256:0a4eaa0eecf5f8c050e5bba433f58c052be7587ee8af3e8b3910ef9ab5fbe9f5
Status: Downloaded newer image for alpine:latest
 ---> 324bc02ae123
Step 2/10 : LABEL maintainer="mgu153457@stud.gibb.ch"
 ---> Running in 3c26337bcb3b
Removing intermediate container 3c26337bcb3b
 ---> a212810ba78b
Step 3/10 : RUN apk add --no-cache nginx
 ---> Running in aa1b747c2354
fetch https://dl-cdn.alpinelinux.org/alpine/v3.20/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.20/community/x86_64/APKINDEX.tar.gz
(1/2) Installing pcre (8.45-r3)
(2/2) Installing nginx (1.26.2-r0)
Executing nginx-1.26.2-r0.pre-install
Executing nginx-1.26.2-r0.post-install
Executing busybox-1.36.1-r29.trigger
OK: 9 MiB in 16 packages
Removing intermediate container aa1b747c2354
 ---> 8acbe9aba4ed
Step 4/10 : RUN mkdir -p /run/nginx && mkdir /www
 ---> Running in 4d599ae72686
Removing intermediate container 4d599ae72686
 ---> e8afe467e53f
Step 5/10 : RUN touch /run/nginx/nginx.pid
 ---> Running in 470066d5f6fe
Removing intermediate container 470066d5f6fe
 ---> 5da8bad421d1
Step 6/10 : RUN adduser -D -g 'www' www
 ---> Running in 6a72f2531efe
Removing intermediate container 6a72f2531efe
 ---> b6b6100912f1
Step 7/10 : RUN chown -R www:www /var/lib/nginx /www
 ---> Running in af4b282fd3fb
Removing intermediate container af4b282fd3fb
 ---> b6ac90214cb5
Step 8/10 : COPY nginx.conf /etc/nginx/nginx.conf
 ---> 81f1f4964ebb
Step 9/10 : COPY index.html /www
 ---> 5e2e17d11958
Step 10/10 : CMD ["/usr/sbin/nginx", "-g", "daemon off;"]
 ---> Running in 040bfb23124e
Removing intermediate container 040bfb23124e
 ---> bcc6ee1c9c8a
Successfully built bcc6ee1c9c8a
```

Running a Docker Container
After creating the Docker image, you can run a container using this image. Use the following command to run the container on port 8080
docker run -d -p 8080:80 my-nginx-image



```
root@li244-vmLM1:/home/vmadmin# docker run -d -p 8080:80 my-nginx-image
10114a49709864930ad43af797247400a1e701df714fdea06d69efedd0a417b8
root@li244-vmLM1:/home/vmadmin# docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED         STATUS         PORTS                                   NAMES
10114a497098   my-nginx-image   "/usr/sbin/nginx -g …"   8 seconds ago   Up 8 seconds   0.0.0.0:8080->80/tcp, :::8080->80/tcp   nervous_faraday
root@li244-vmLM1:/home/vmadmin# 
```

Testing the Web Server
Once the Docker container is running, you can test the web server by typing 192.168.120.60:8080 into your browser.



Hello, Docker Container!

You are publishing on an NGINX web server.

Created by: Mehmet Ali GUR

School: GIBB