Task_1

- Run a container nginx with name my-nginx and attach 2 volumes to the container
- Volume1 for containing static html file
- Volume2 for containing nginx configuration
- Edit the html content
- Remove the container

- Run new 2 containers with the following:
- Attach the 2 volumes that was attached to the previous container

in two different ways (volume mount – bind mount)
- Map port 80 to port 8080 on you host machine
- Access the html file from your browser

Step 1: Run the Initial Nginx Container

1. **Create the required directories on the host machine** to hold the HTML files and Nginx configuration.

```
mkdir html
mkdir nginx-config
```

2. **Create an example HTML file** inside the `html` directory.

```
echo '<html><body><h1>Hello from Nginx!</h1></body></html>' > ./html/index.html
```

3. **Create a basic Nginx configuration file** inside the `nginx-config` directory.

```
cat <<EOF > ./nginx-config/nginx.conf
server {
    listen 80;
    root /usr/share/nginx/html;
    index index.html;
```

```
    location / {
        try_files \\$uri \\$uri/ =404;
    }
}
EOF
```

4. **Run the Nginx container** with the required volumes.

```
docker run -d --name my-nginx \
  -v ./html:/usr/share/nginx/html:ro \
  -v ./nginx-config:/etc/nginx/conf.d:ro \
  -p 8080:80 \
  nginx
```

- `-d`: Run container in detached mode.
- `--name my-nginx`: Assign the name `my-nginx` to the container.
- `-v ./html:/usr/share/nginx/html:ro`: Mount the host directory containing HTML files to the container's HTML directory as read-only.
- `-v ./nginx-config:/etc/nginx/conf.d:ro`: Mount the host directory containing Nginx configuration files to the container's configuration directory as read-only.
- `-p 8080:80`: Map port 8080 on the host to port 80 on the container.

Step 2: Edit the HTML Content

1. **Edit the HTML file** to update its content if necessary.

```
echo '<html><body><h1>Updated Content!</h1></body></html
>' > ./html/index.html
```

Step 3: Remove the Initial Container

1. **Stop and remove the container**.

```
docker stop my-nginx
```

```
docker rm my-nginx
```

Step 4: Run New Containers with Volume Mounts

Option A: Using Bind Mount

1. **Run a new container** using bind mount.

```
docker run -d --name my-nginx-bind-mount \
  -v ./html:/usr/share/nginx/html:ro \
  -v ./nginx-config:/etc/nginx/conf.d:ro \
  -p 8080:80 \
  nginx
```

Option B: Using Docker Volumes

1. **Create Docker volumes** for HTML and Nginx configuration.

```
docker volume create html_volume
docker volume create nginx_config_volume
```

2. **Copy the contents** from the host directories to the Docker volumes.

`docker run --rm -v ./html:/src -v html_volume:/dst alpine sh -c "cp -a /src/. /dst/"`

`docker run --rm -v ./nginx-config:/src -v nginx_config_volume:/dst alpine sh -c "cp -a /src/. /dst/"`

This command copies the contents of the host directory `./html` to the Docker volume `html_volume` by:

- Mounting the host directory (`./html`) inside the container at `/src`.
- Mounting the Docker volume (`html_volume`) inside the container at `/dst`.
- Using the `cp -a` command inside the Alpine container to copy all files from `/src` (host directory) to `/dst` (Docker volume).

After the copy operation, the container is removed due to the `--rm` flag. This way, you ensure the data in `html_volume` is updated with the contents from the host's `./html` folder.

3. **Remove the previous container** to access the another on the same port

```
docker ps
```

```
docker rm -f <container_id>
```

4. **Run another container** using Docker volumes.

```
docker run -d --name my-nginx-docker-volumes \
  -v html_volume:/usr/share/nginx/html:ro \
  -v nginx_config_volume:/etc/nginx/conf.d:ro \
  -p 8080:80 \
  nginx
```

Step 5: Access the HTML File

1. **Open your web browser** and go to `http://localhost:8080`.