Docker Assignment:- 2

:-Take an existing Docker file and optimize it for build speed and image size.

Using a base image like eclipse-temurin:17-jre-slim or eclipse-temurin:17-jre-alpine instead of eclipse-temurin:17-jre can significantly reduce the image size. The Alpine version is built on a minimal Linux distribution, making it much smaller.

:- Use multi-stage builds where appropriate.

- # Stage 1: Build the application using Maven
- FROM maven:3.8.8-eclipse-temurin-17 AS build
- WORKDIR /app
- # Copy the Maven project files
- COPY pom.xml .
- RUN mvn dependency:go-offline -B
- # Copy the source code
- COPY src ./src
- # Build the application
- RUN mvn clean package -DskipTests
- # Stage 2: Create the runtime image
- FROM eclipse-temurin:17-jre-alpine
- # Create a non-root user myuser
- RUN groupadd -r myuser && useradd -r -g myuser myuser
- WORKDIR /app
- # Copy the JAR file from the build stage
- COPY --from=build /app/target/student-management-0.0.1-SNAPSHOT.jar app.jar
- # Switch to the non-root user
- USER appuser
- # Set entry point to run the application
  ENTRYPOINT ["java", "-jar", "app.jar"]

How It Works

1. **Build Stage**:
   - Uses the maven image to build the application.
   - Copies the source code and pom.xml file.
   - Runs maven clean package to build the JAR file.
2. **Runtime Stage**:
   - Uses a minimal OpenJDK image to run the application.
   - Copies the JAR file from the build stage into the final image.
   - Sets the command to run the JAR file.

:-Set up a custom Docker network and connect multiple containers to it.

:-Use Docker commands to inspect and manage the network.

networks: my_custom_network:

```
anil@IN-PG02P670:~$ docker network ls
NETWORK ID     NAME                        DRIVER    SCOPE
b88eb62e8f32   bridge                      bridge    local
6728753408b9   host                        host      local
728a44f5f058   none                        null      local
8d8756c4e877   student_my_custom_network   bridge    local
```

connect multiple containers with the help of docker-compose.yml

:-Use named volumes and bind mounts and demonstrate how they can be used in a multi-container setup.

There are two volumes define in docker-compose .yml file one is mongo-data and second is app data.

- version: '3'
- services:
- mongo:
- image: mongo:5.0
- container_name: mongo-container
- ports:
- - 27017:27017
- networks:
- - my_custom_network
- volumes:
- - mongo-data:/data/db
- - ./db-backups:/var/lib/mongodata/backups
- springboot-app:
- build: .
- container_name: spring-boot-container
- ports:
- - 8085:8085
- networks:
- - my_custom_network
- depends_on:
- - mongo
- environment:
- SPRING_DATA_MONGODB_URI: mongodb://mongo:27017/student-management
- volumes:
- - app-data:/data/app

- - ./logs:/var/log/spring-boot-app/logs
- networks:
- my_custom_network:
- volumes:
- mongo-data:
  app-data:

:-Backup and restore data from Docker volumes.

Backup

docker run --rm -v student_mongo-data:/data -v $(pwd):/backup student-springboot-app tar cvf /backup/app_data_backup.tar /data

- `docker run --rm`: This runs a new container and removes it after the command has finished. The `--rm` flag ensures that the container is cleaned up and removed once it completes its task.
- `-v student_mongo-data:/data`: This mounts the Docker volume named `student_mongo-data` to the `/data` directory inside the container. This volume should contain your MongoDB data.
- `-v $(pwd):/backup`: This mounts your current working directory (obtained using `$(pwd)`) to the `/backup` directory inside the container. This is where you will store the backup file.
- `student-springboot-app`: This specifies the Docker image to use for running the container. In this case, it's assumed to be an image that has `tar` installed.
- `tar cvf /backup/app_data_backup.tar /data`: This is the command executed inside the container. It uses `tar` to create a backup of the `/data` directory (which is your MongoDB data) and saves it as `app_data_backup.tar` in the `/backup` directory (which maps to your current directory on the host).

Restore

docker run --rm -v student_mongo-data:/data -v $(pwd):/backup student-springboot-app -c "cd /data && tar xvf /backup/app_data_backup.tar --strip 1"

- `ocker run --rm`: Runs a new container and removes it after the command finishes. The `--rm` flag ensures that the container is cleaned up and removed once it completes its task.
- `-v student_mongo-data:/data`: Mounts the Docker volume named `student_mongo-data` to the `/data` directory inside the container. This is where the backup data will be restored.
- `-v $(pwd):/backup`: Mounts your current working directory (using `$(pwd)`) to the `/backup` directory inside the container. This is where the backup file is located.
- `student-springboot-app`: Specifies the Docker image to use for running the container. This image should have the `tar` utility installed.
- `-c "cd /data && tar xvf /backup/app_data_backup.tar --strip 1"`: This is the command executed inside the container

:-Implement security best practices in your Docker setup, including user permissions, image vulnerability scanning, and secret management.

In docker file create and use non-root user.

```
# Create a non-root user
RUN groupadd -r myuser && useradd -r -g myuser myuser

# Switch to the non-root user
USER myuser
```

image vulnerability scanning

```
anil@IN-PG02P670:~$ docker run aquasec/trivy image student-springboot-app
2024-09-12T01:23:13Z    INFO    [db] Need to update DB
2024-09-12T01:23:13Z    INFO    [db] Downloading DB...  repository="ghcr.io/aquasecurity/trivy-db:2"
231.07 KiB / 53.00 MiB [>_____________________________________________________] 0.43% ? p/s ?455.
```

:-Configure Docker to run containers with the least privilege.

Use Non-Root Users in Docker files**: Modify your Docker file to create and use a non-root user.**

:-Use tools like Docker Bench for Security to audit your Docker environment.

Install Docker Bench for Security:

Clone the Docker Bench for Security repository:

git clone https://github.com/docker/docker-bench-security.git
cd docker-bench-security

Run the Docker Bench script:
sudo sh docker-bench-security.sh

```
anil@IN-PG02P670:~/ld/java-project/student/docker-bench-security$ sudo sh docker-bench-security.sh
# ------------------------------------------------------------------------------------------
# Docker Bench for Security v1.6.0
#
# Docker, Inc. (c) 2015-2024
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Based on the CIS Docker Benchmark 1.6.0.
# ------------------------------------------------------------------------------------------

Initializing 2024-09-12T07:26:37+05:30


Section A - Check results

[INFO] 1 - Host Configuration
[INFO] 1.1 - Linux Hosts Specific Configuration
WARNING: No blkio throttle.read_bps_device support
WARNING: No blkio throttle.write_bps_device support
WARNING: No blkio throttle.read_iops_device support
WARNING: No blkio throttle.write_iops_device support
[PASS] 1.1.1 - Ensure a separate partition for containers has been created (Automated)
[INFO] 1.1.2 - Ensure only trusted users are allowed to control Docker daemon (Automated)
[INFO]       * Users: anil
```