`

EXPERIMENT NO 9

Aim- Containerizing App with Docker

Theory-

What is Containerization?

Containerization is the process of packaging an application along with all its dependencies, libraries, and configuration files in to a single unit called a container.

This ensures the app behaves the same way in development, testing, and production environments.

- Benefits of Containerization:
    - Consistent runtime environment:The app runs the same on every computer because Docker packs everything it needs inside one container.
    - Faster deployment and scaling:Containers start in just a few seconds, and we can easily add more containers when more users come.
    - Simplified management of application dependencies:All the app's software and files are already inside the container, so we don't have to install anything extra.
    - Easy rollback and version control:If a new version has a problem, we can quickly go back to the old working version with just one command.
    - If docker image are myapp:v1
    - myapp:v2
    - docker run -d --name myapp_old -p 3000:3000 myapp:v1

What is Docker?

Docker is an open-source platform that allows developers to automate the deployment of applications inside lightweight, portable containers.

It ensures that the application runs consistently on any system regardless of differences in software or configuration.

- Key features of Docker:

`

- **Lightweight and fast**: Docker containers start quickly and use fewer resources because they share the system's operating system instead of running a full one.
- **Platform-independent**:Apps made with Docker can run on any computer — Windows, Mac, or Linux — without changing the code.

- **Easy deployment and scaling:**You can easily launch, stop, or run multiple containers, making it simple to deploy updates or handle more users.

- **Uses less memory than virtual machines**:Since containers share the same OS, they take up much less space and memory compared to heavy virtual machines.

DockerFile:
A Dockerfile is a simple text file that contains a script of instructions for building a Docker image. Think of it as a recipe for creating your application's environment. The Docker engine reads this file and executes the commands in order, layer by layer, to assemble a final, runnable image.

What is an Nginx image?

An **Nginx image** is a **ready-made Docker image** that contains:

- The **Nginx web server** (software)
- A small **Linux system** (usually Alpine Linux)
- All the tools needed to run Nginx immediately

Steps to Containerize the ExploreEase App:

1. Install Docker
    a. Download and install Docker Desktop from https://www.docker.com/.
    Verify installation:
    docker —version
2. Create a Dockerfile.
    Inside your ExploreEase project folder, create a file named Dockerfile.

```
# Use official Node.js LTS image

FROM node:18
```

`

```
# Set working directory

WORKDIR /usr/src/app


# Copy package.json and package-lock.json

COPY package*.json ./


# Install dependencies

RUN npm install


# Copy the rest of the application code

COPY . .


# Expose the port the app runs on

EXPOSE 3000


# Set environment variables (override in production as needed)

ENV NODE_ENV=production


# Start the app

CMD ["node", "app.js"]
```

Our Project DockerFile:

```
# Build frontend
```

```
FROM node:18-alpine AS frontend-build

WORKDIR /app

COPY package.json package-lock.json ./
RUN npm install

COPY . .
RUN npm run build

# Production image
FROM node:18-alpine

# Install nginx
RUN apk add --no-cache nginx

# Create nginx directories
RUN mkdir -p /run/nginx /var/log/nginx /var/cache/nginx

# Copy nginx config
RUN echo 'server {\
    listen 80;\
    server_name localhost;\
```

```
    root /usr/share/nginx/html;\
    index index.html;\
    location / {\
        try_files $uri $uri/ /index.html;\
    }\
    location /api {\
        proxy_pass http://localhost:5000;\
        proxy_set_header Host $host;\
        proxy_set_header X-Real-IP $remote_addr;\
    }\
}' > /etc/nginx/http.d/default.conf

# Copy built frontend
COPY --from=frontend-build /app/dist /usr/share/nginx/html

# Copy backend
WORKDIR /app/backend
COPY backend/package.json ./
RUN npm install --production

COPY backend/ .

# Expose ports
```

`

```
EXPOSE 80 5000



# Start both services

CMD nginx && npm start
```

3. Build the Docker Image
   Run the following command in the terminal:
   Docker build -t my-website
4. Run the Container
   Start your container with:
   docker run -d -p 4040:80 my-website-container3
5. Access the App
   a. Open your browser and go to:
      👉 http://localhost:8080
   b. You'll see your WanderHub Travel App running inside a Docker container!

30% extra -

What is MongoDB?

MongoDB is a NoSQL (non-relational) database that stores data in JSON-like documents instead of tables.

It is widely used in modern web apps like ExploreEase Travel App because it provides:

- High scalability and flexibility
- Easy handling of unstructured data
- Faster performance for read/write operations
- Support for geolocation, which is useful for travel-based apps

Why Use MongoDB with Docker?

By containerizing MongoDB, we can run the database and application in separate, isolated containers, ensuring:

`

- Easy setup without manual database installation
- Consistent environment across systems
- Simplified networking between backend and database
- Portability and quick restoration using images and volumes

How WanderHub Uses MongoDB:

In the ExploreEase app, MongoDB can store:

- User profiles and login details
- Trip packages and destinations
- Bookings, reviews, and ratings
- Travel history and recommendations

This makes the app dynamic and data-driven.

How User Data is Managed

In **WanderHub**, users can sign up, log in, and explore travel packages.

Their details — such as name, email, phone number, and travel interests — are securely stored in the MongoDB database.


```
{
  "name": "Khushi Panjwani",
  "email": "khushi@example.com",
  "phone": "+91 9876543210",
  "preferred_destination": "Nainital",
  "bookings": ["Trip_001", "Trip_004"]
}
```


This allows the app to:

`

- Personalize travel suggestions
- Track booking history
- Provide a smooth login experience

When a user signs up:

1. The frontend sends their data to the backend API.
2. The backend (in a Docker container) connects to MongoDB (in another container).
3. Data is stored in a collection named users.

Conclusion:

The ExploreEase Travel App was successfully containerized using Docker.

This process made the application portable, easily deployable, and ensured environment consistency across systems.

Containerization helps developers avoid the “it works on my machine” problem and supports modern DevOps practices like Continuous Integration and Deployment (CI/CD).