Tasks:

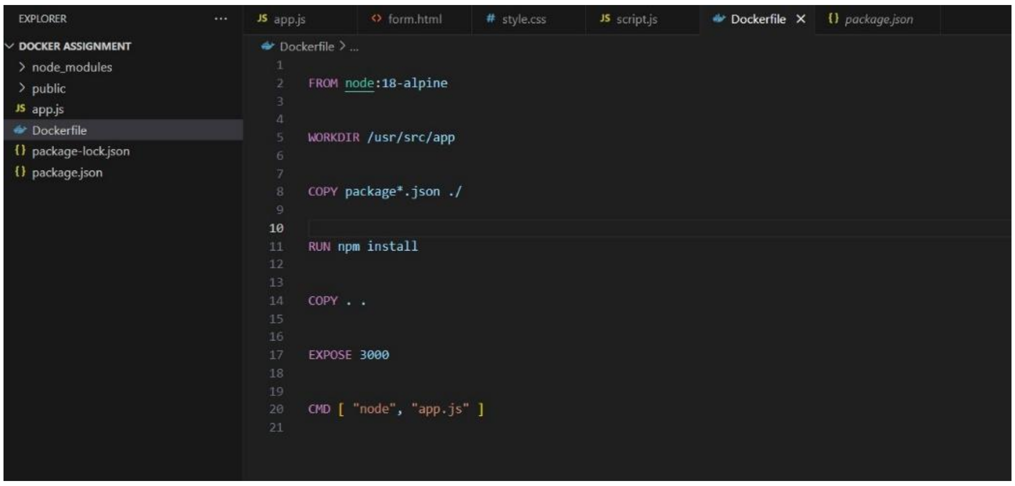
Create a simple Docker container for a Node.js web application.

1. **Create a Node.js web application or use an existing one.**
2. **Write a Dockerfile to containerize the Node.js application.**
3. **Build the Docker image.**
4. **Push it to Docker hub.**
5. **Run the Docker container, exposing the application on a specific port.**
6. **Access the web application in a web browser.**
7. **Application should be up and running, also create documentation for the whole process.**

1. Prerequisites:

- Ensure Docker is installed on your system.
- Have a Node.js web application ready or create a simple one.

2. Dockerfile Explanation:



```
FROM node:18-alpine

WORKDIR /usr/src/app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 3000

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

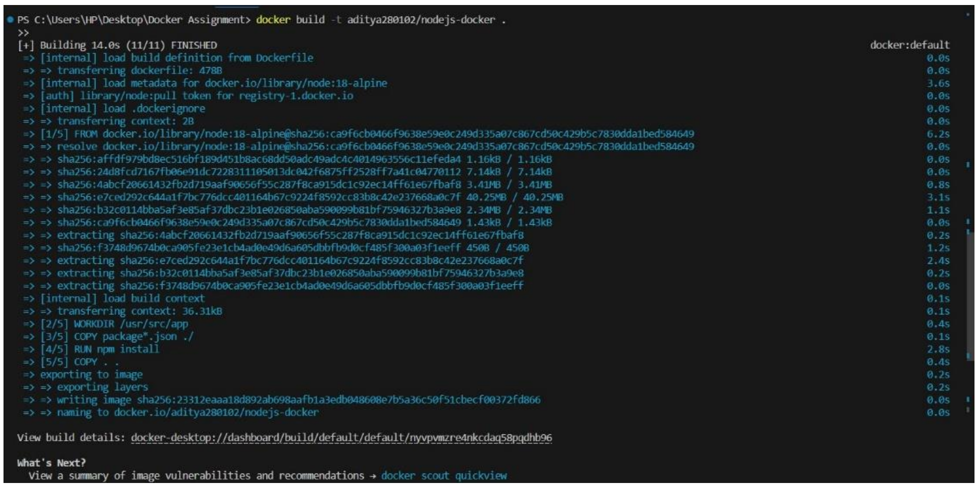
- **FROM node:18-alpine :** Use the official Node.js 18 Alpine image as the base image.
- **WORKDIR /usr/src/app :** Set the working directory within the container to /usr/src/app.
- **COPY package*.json ./ :** Copy package.json and package-lock.json to the working directory.
- **RUN npm install :** Install Node.js dependencies.

- **COPY . . :** Copy the rest of the application files to the working directory.
- **EXPOSE 3000 :** Expose port 3000, which is the default port for Node.js applications.
- **CMD ["node", "app.js"] :** Set the default command to start the application.

3. Build and Push Docker Image:

- **Build the Docker image:**

  docker build -t aditya280102/nodejs-docker .

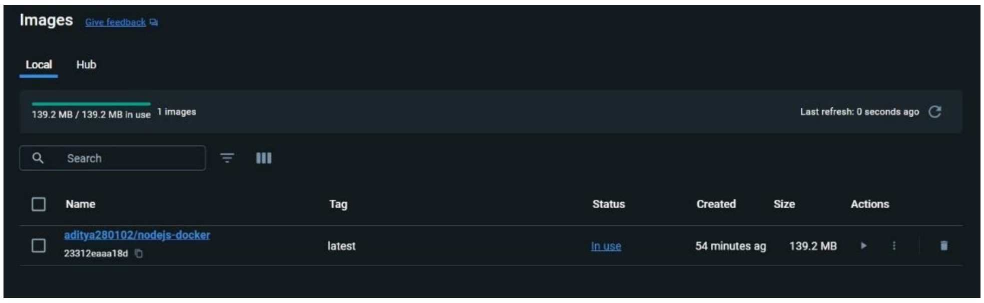


```
PS C:\Users\HP\Desktop\Docker Assignment> docker build -t aditya280102/nodejs-docker .
>>
[+] Building 14.0s (11/11) FINISHED                                                                          docker:default
 => [internal] load build definition from Dockerfile                                                                   0.0s
 => => transferring dockerfile: 478B                                                                                   0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                                                      3.6s
 => [auth] library/node:pull token for registry-1.docker.io                                                            0.0s
 => [internal] load .dockerignore                                                                                      0.0s
 => => transferring context: 2B                                                                                        0.0s
 => [1/5] FROM docker.io/library/node:18-alpine@sha256:ca9f6cb0466f9638e59e0c249d335a07c867cd50c429b5c7830dda1bed584649  6.2s
 => => resolve docker.io/library/node:18-alpine@sha256:ca9f6cb0466f9638e59e0c249d335a07c867cd50c429b5c7830dda1bed584649  0.0s
 => => sha256:affdf979bd8ec516bf189d451b8ac68dd50adc49adc4c4014963556c11efeda4 1.16kB / 1.16kB                         0.0s
 => => sha256:24d8fcd7167fb06e91dc7228311105013dc042f6875ff2528ff7a41c04770112 7.14kB / 7.14kB                         0.0s
 => => sha256:4abcf20661432fb2d719aaf90656f55c287f8ca915dc1c92ec14ff61e67fbaf8 3.41MB / 3.41MB                         0.8s
 => => sha256:e7ced292c644a1f7bc776dcc401164b67c9224f8592cc83b8c42e237668a0c7f 40.25MB / 40.25MB                       3.1s
 => => sha256:b32c0114bba5af3e85af37dbc23b1e026850aba590099b81bf75946327b3a9e8 2.34MB / 2.34MB                         1.1s
 => => sha256:ca9f6cb0466f9638e59e0c249d335a07c867cd50c429b5c7830dda1bed584649 1.43kB / 1.43kB                         0.0s
 => => extracting sha256:4abcf20661432fb2d719aaf90656f55c287f8ca915dc1c92ec14ff61e67fbaf8                              0.2s
 => => sha256:f3748d9674b0ca905fe23e1cb4ad0e49d6a605dbbfb9d0cf485f300a03f1eeff 450B / 450B                             1.2s
 => => extracting sha256:e7ced292c644a1f7bc776dcc401164b67c9224f8592cc83b8c42e237668a0c7f                              2.4s
 => => extracting sha256:b32c0114bba5af3e85af37dbc23b1e026850aba590099b81bf75946327b3a9e8                              0.2s
 => => extracting sha256:f3748d9674b0ca905fe23e1cb4ad0e49d6a605dbbfb9d0cf485f300a03f1eeff                              0.0s
 => [internal] load build context                                                                                      0.1s
 => => transferring context: 36.31kB                                                                                   0.1s
 => [2/5] WORKDIR /usr/src/app                                                                                         0.4s
 => [3/5] COPY package*.json ./                                                                                        0.1s
 => [4/5] RUN npm install                                                                                              2.8s
 => [5/5] COPY . .                                                                                                     0.4s
 => exporting to image                                                                                                 0.2s
 => => exporting layers                                                                                                0.2s
 => => writing image sha256:23312eaaa18d892ab698aafb1a3edb048608e7b5a36c50f51cbecf00372fd866                           0.0s
 => => naming to docker.io/aditya280102/nodejs-docker                                                                  0.0s

View build details: docker-desktop://dashboard/build/default/default/nyvpvmzre4nkcdaq58pqdhb96

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
```

Result:

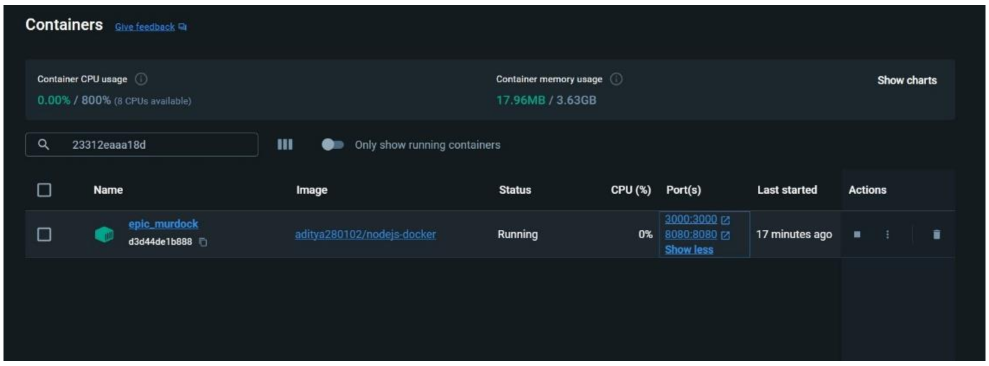


Images

139.2 MB / 139.2 MB in use 1 images

Last refresh: 0 seconds ago

Name	Tag	Status	Created	Size	Actions
aditya280102/nodejs-docker 23312eaaa18d	latest	In use	54 minutes ag	139.2 MB	



Containers

Container CPU usage
0.00% / 800% (8 CPUs available)

Container memory usage
17.96MB / 3.63GB

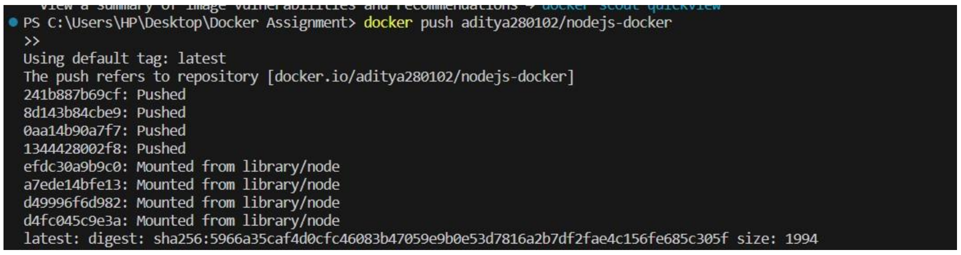
23312eaaa18d

Only show running containers

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
epic_murdock d3d44de1b888	aditya280102/nodejs-docker	Running	0%	3000:3000 8080:8080 Show less	17 minutes ago	

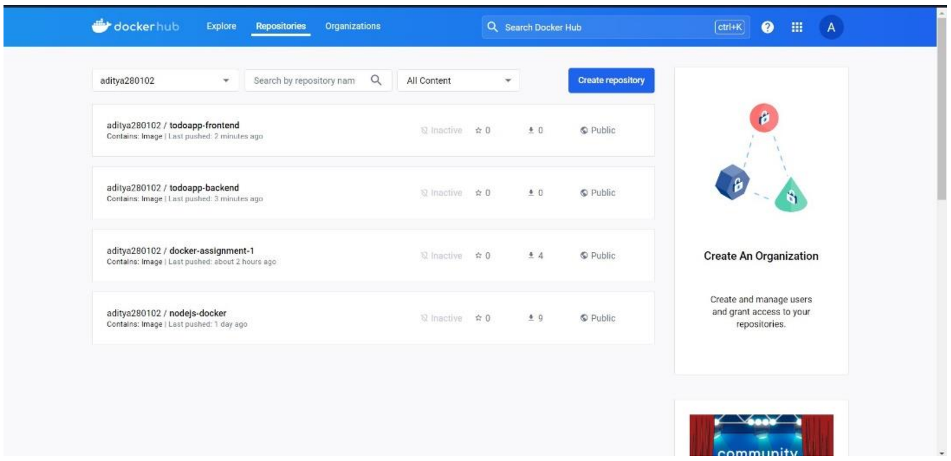
- **Push the Docker image to Docker Hub: (Before this login to your docker)**

  docker push aditya280102/nodejs-docker



```
  View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\HP\Desktop\Docker Assignment> docker push aditya280102/nodejs-docker
>>
Using default tag: latest
The push refers to repository [docker.io/aditya280102/nodejs-docker]
241b887b69cf: Pushed
8d143b84cbe9: Pushed
0aa14b90a7f7: Pushed
1344428002f8: Pushed
efdc30a9b9c0: Mounted from library/node
a7ede14bfe13: Mounted from library/node
d49996f6d982: Mounted from library/node
d4fc045c9e3a: Mounted from library/node
latest: digest: sha256:5966a35caf4d0cfc46083b47059e9b0e53d7816a2b7df2fae4c156fe685c305f size: 1994
```

Result:



aditya280102 / todoapp-frontend
Contains: Image | Last pushed: 2 minutes ago — Inactive ☆ 0 ⭳ 0 Public

aditya280102 / todoapp-backend
Contains: Image | Last pushed: 3 minutes ago — Inactive ☆ 0 ⭳ 0 Public

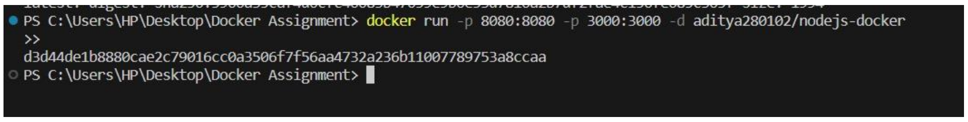
aditya280102 / docker-assignment-1
Contains: Image | Last pushed: about 2 hours ago — Inactive ☆ 0 ⭳ 4 Public

aditya280102 / nodejs-docker
Contains: Image | Last pushed: 1 day ago — Inactive ☆ 0 ⭳ 9 Public

4. Run Docker Container:

- **Run the Docker container:**

  docker run -p 8080:8080 -p 3000:3000 -d aditya280102/nodejs-docker

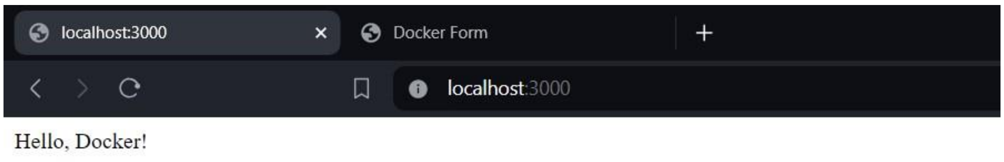


```
PS C:\Users\HP\Desktop\Docker Assignment> docker run -p 8080:8080 -p 3000:3000 -d aditya280102/nodejs-docker
>>
d3d44de1b8880cae2c79016cc0a3506f7f56aa4732a236b11007789753a8ccaa
PS C:\Users\HP\Desktop\Docker Assignment> 
```

- **Explanation:**
  - **-p 8080:8080**: Map port 8080 on the host to port 8080 in the container.
  - **-p 3000:3000**: Map port 3000 on the host to port 3000 in the container.
  - **-d**: Run the container in detached mode.

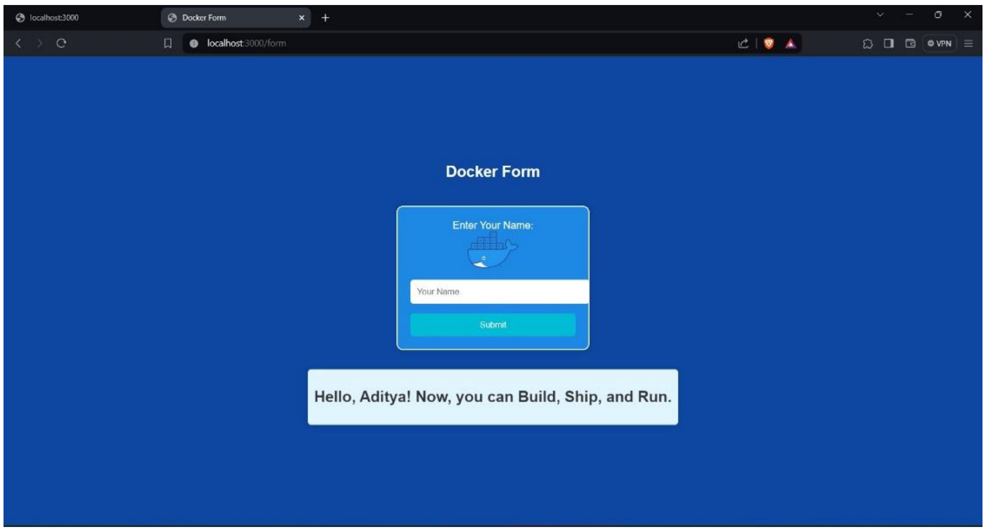
5. Access the Web Application:

- Open a web browser and go to **http://localhost:3000** to access the web application.



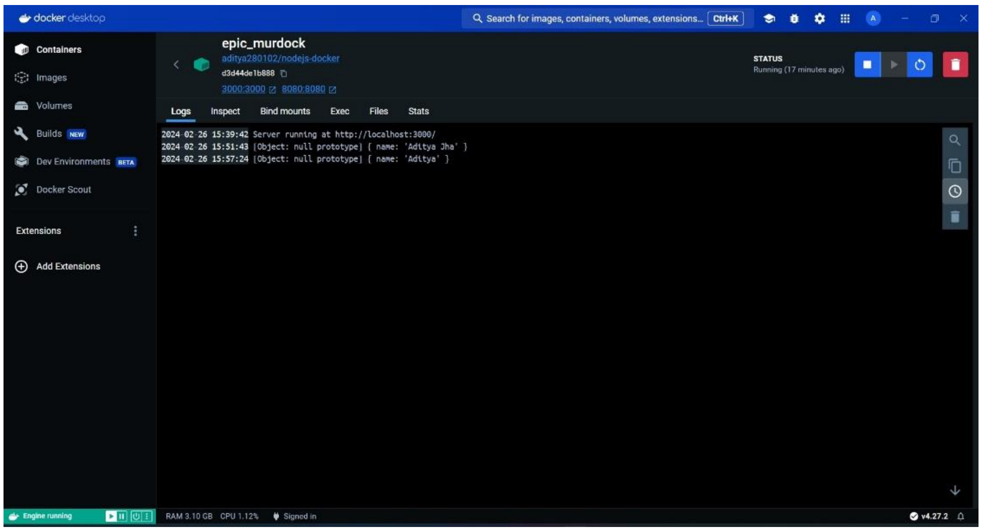
Hello, Docker!

Also open http://localhost:3000/form to access the form



Hello, Aditya! Now, you can Build, Ship, and Run.

You can see the status in the docker desktop.



epic_murdock
aditya280102/nodejs-docker
d3d44de1b888
3000:3000 8080:8080

STATUS
Running (17 minutes ago)

```
2024-02-26 15:39:42 Server running at http://localhost:3000/
2024-02-26 15:51:43 [Object: null prototype] { name: 'Aditya Jha' }
2024-02-26 15:57:24 [Object: null prototype] { name: 'Aditya' }
```

6. Cleanup: (When done testing or deploying)

- Stop the running container:

```
docker stop <container_id>
```

- Remove the container:

```
docker rm <container_id>
```

- Remove the Docker image (optional):

```
docker rmi aditya280102/nodejs-docker
```

7. Conclusion:

- The Node.js web application is now containerized using Docker and can be easily deployed in different environments.
- The Docker image is available on Docker Hub for distribution.
- Follow the cleanup steps when done testing or deploying the application.