Docker To-Do List Application Documentation

Introduction

This document provides a comprehensive overview of the Docker-based To-Do List application developed following the Docker workshop. The project involves containerizing an application, updating it, sharing it, persisting its database, managing multi-container applications, and using Docker Compose.

Containerizing the Application

Steps Followed

1. Cloned the repository and created a simple Node.js app

2. Added a **Dockerfile**:

```
Dockerfile U ×   .dockerignore
getting-started-app > Dockerfile > ...
1  # syntax=docker/dockerfile:1
2
3  FROM node:lts-alpine
4  WORKDIR /app
5  COPY . .
6  RUN yarn install --production
7  CMD ["node", "src/index.js"]
8  EXPOSE 3000
```

3. Built and ran the container:

```
PS E:\docker\getting-started-app> docker build -t getting-started .
[+] Building 63.0s (13/13) FINISHED docker:desktop-linux=> transferring context: 6.42MB          1.7s
 => [internal] load build definition from Doc  0.0s
 => => transferring dockerfile: 189B           0.0s
 => resolve image config for docker-image://d  3.5s
 => [auth] docker/dockerfile:pull token for r  0.0s
 => docker-image://docker.io/docker/dockerfil  4.8s
 => => resolve docker.io/docker/dockerfile:1@  0.0s
```

```
Containers / clever_kalam

clever_kalam
48769147c885   getting-started:latest
3000:3000

STATUS
Running (58 seconds ago)

Logs   Inspect   Bind mounts   Exec   Files   Stats

2025-03-26 16:27:03 Using sqlite database at /etc/todos/todo.db
2025-03-26 16:27:03 Listening on port 3000
```

Updating the Application

Steps Followed

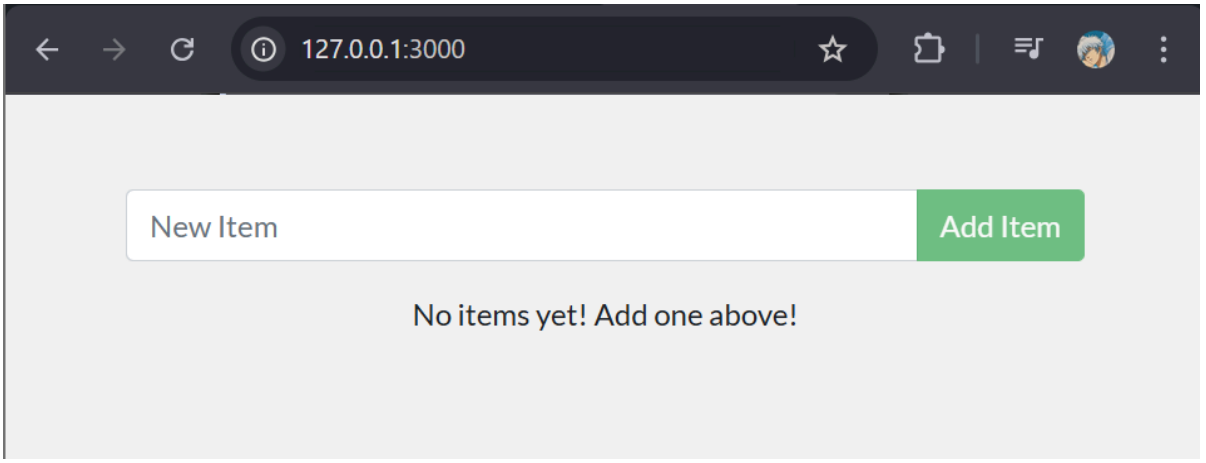
1. Modified the app
2. Rebuilt the image and restarted the container:

```
55            {items.length === 0 && (
56                <p className="text-center">You have no todo items yet! Add one above!</p>
57            )}
58            {items.map(item => (
59                <ItemDisplay
60                    item={item}
61                    key={item.id}
62                    onItemUpdate={onItemUpdate}
63                    onItemRemoval={onItemRemoval}
64                />
65            ))}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell - getting-started-app

```
 => => naming to docker.io/library/getting-started:latest              0.0s
 => => unpacking to docker.io/library/getting-started:latest           2.0s
PS E:\docker\getting-started-app> docker run -dp 127.0.0.1:3000:3000 getting-started
863b3f8ff1b64e44ed56a3fa77a0bdd47616ebb93bdffa2887d8c5ff0116a235
docker: Error response from daemon: failed to set up container networking: driver failed programming external connectivity on endpoint jolly_kare (76f5285765aa00a2
e78937ae1ea31e5244e19f2babb3fd4502843b41737cd4c9): Bind for 127.0.0.1:3000 failed: port is already allocated

Run 'docker run --help' for more information
PS E:\docker\getting-started-app>
```



Sharing the Application

Steps Followed

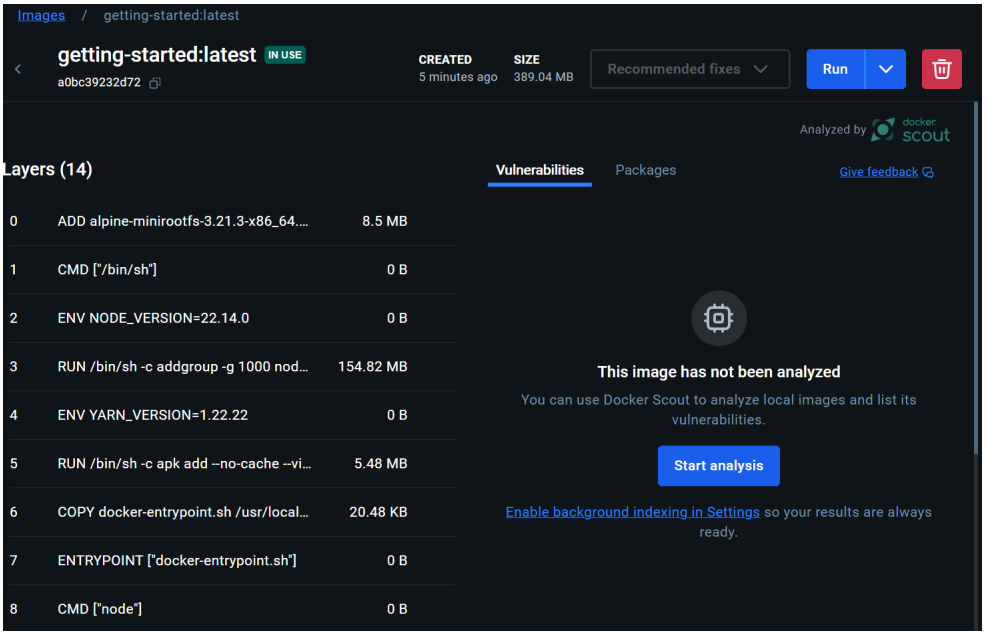
1. Tagged the image and pushed it to **Docker Hub**:

   docker tag todo-app vvndza/todo-app
   docker push vvndza/todo-app

2. Pulled the image on another system:

```
PS E:\docker\getting-started-app> docker run -dp 127.0.0.1:3000:3000 getting-started
863b3f8ff1b64e44ed56a3fa77a0bdd47616ebb93bdffa2887d8c5ff0116a235

Run 'docker run --help' for more information
PS E:\docker\getting-started-app> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                     NAMES
48769147c885   a0bc39232d72   "docker-entrypoint.s…"   22 minutes ago   Up 22 minutes   127.0.0.1:3000->3000/tcp   clever_kalam
PS E:\docker\getting-started-app> docker stop 48769147c885
48769147c885
PS E:\docker\getting-started-app> docker rm 48769147c885
48769147c885
```



Persisting the Database

Steps Followed

1. Used **Docker volumes** to persist MySQL data:

   docker volume create todo-db

```
PS E:\docker\getting-started-app> docker volume inspect todo-db
[
    {
        "CreatedAt": "2025-03-26T11:44:17Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/todo-db/_data",
        "Name": "todo-db",
        "Options": null,
        "Scope": "local"
    }
]
```

2. Ran MySQL container:

```
PS E:\docker\getting-started-app> docker run -d `
>>     --network todo-app --network-alias mysql `
>>     -v todo-mysql-data:/var/lib/mysql `
>>     -e MYSQL_ROOT_PASSWORD=secret `
>>     -e MYSQL_DATABASE=todos `
>>     mysql:8.0
```

3. Connected the app to MySQL.

Multi-Container App with Docker Network

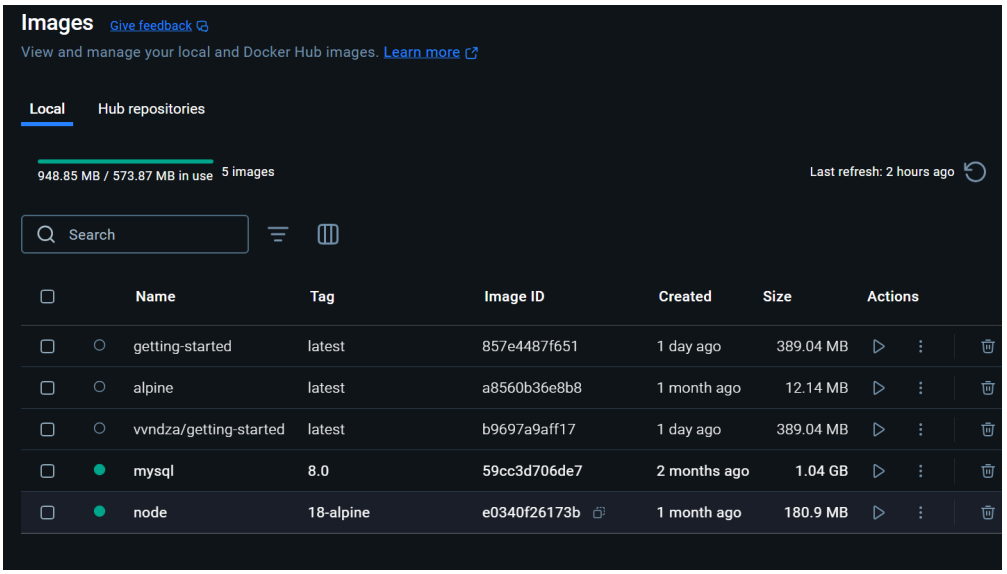
Steps Followed

1. Created a **Docker network**:

```
PS E:\docker\getting-started-app> docker network create todo-app
5cfcef744d8361857c186a4e00f97225af2866a4c4246ffaa789089d4c425f77
```

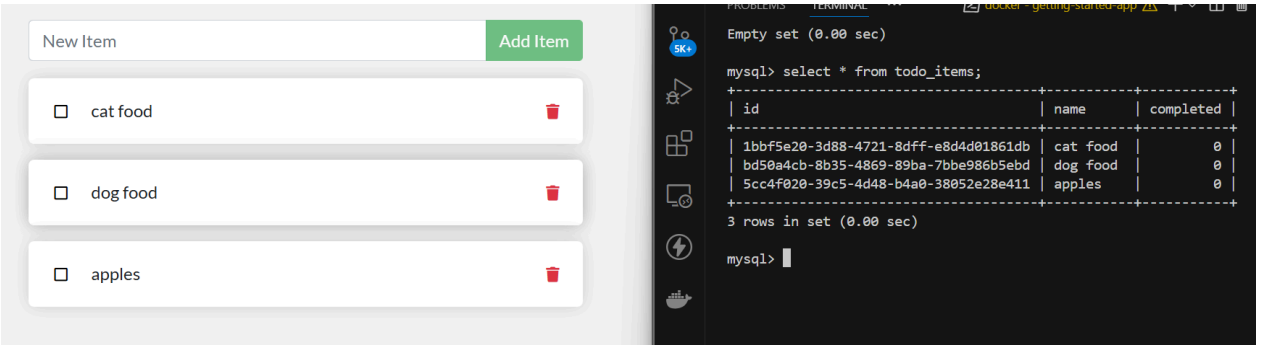
2. Connected both containers to the network:

```
PS E:\docker\getting-started-app> docker run -d `
>>     --network todo-app --network-alias mysql `
>>     -v todo-mysql-data:/var/lib/mysql `
>>     -e MYSQL_ROOT_PASSWORD=secret `
>>     -e MYSQL_DATABASE=todos `
>>     mysql:8.0
```



3. Started the app container:

```
2e6311f703c324dd4a6648053a45bd8664a2e7c2690e92607266531ecc5849ee
PS E:\docker\getting-started-app> docker exec -it 2e6311f703c324dd4a6648053a45bd8664a2e7c2690e92607266531ecc5849ee mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
```



Using Docker Compose

Steps Followed

1. Created a `compose.yml` file:

```
services:
  app:
    image: node:18-alpine
    command: sh -c "yarn install && yarn run dev"
    ports:
      - 127.0.0.1:3000:3000
    working_dir: /app
    volumes:
      - ./:/app
    environment:
      MYSQL_HOST: mysql
      MYSQL_USER: root
      MYSQL_PASSWORD: secret
      MYSQL_DB: todos
```

```
mysql:
  image: mysql:8.0
  volumes:
    - todo-mysql-data:/var/lib/mysql
  environment:
    MYSQL_ROOT_PASSWORD: secret
    MYSQL_DATABASE: todos

volumes:
  Todo-mysql-data:
```

Dockerfile U | app.js M | compose.yaml U

getting-started-app > compose.yaml

```
1   services:
16    mysql:
17      image: mysql:8.0
18      volumes:
19        - todo-mysql-data:/var/lib/mysql
20      environment:
21        MYSQL_ROOT_PASSWORD: secret
22        MYSQL_DATABASE: todos
23
24  volumes:
25    todo-mysql-data:
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
vdsou@DESKTOP-55IA22L MINGW64 /e/docker/getting-started-app (main)
$ docker run -d \
  --network todo-app --network-alias mysql \
  -v todo-mysql-data:/var/lib/mysql \
  -e MYSQL_ROOT_PASSWORD=secret \
  -e MYSQL_DATABASE=todos \
  mysql:8.0
 - Container getting-started-app-app-1  Starting0.7s
```

Containers Give feedback

View all your running containers and applications. Learn more

Container CPU usage: 63.32% / 1200% (12 CPUs available)

Container memory usage: 515.2MB / 7.43GB

Show charts

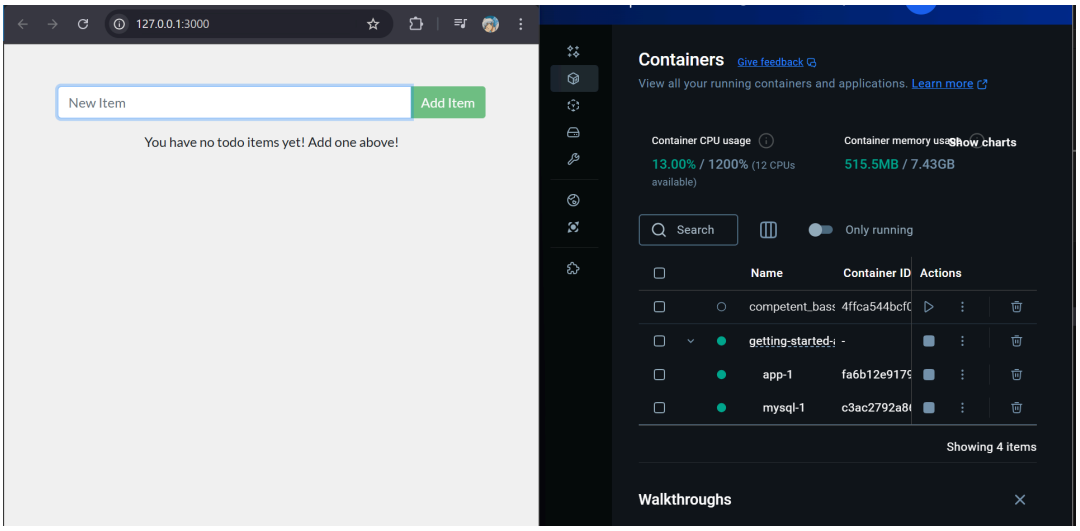
Search | Only show running containers

Name	Container ID	Image	Port(s)	CPU (%)	Last start	Actions
competent_bas	4ffca544bcf0	mysql:8.0		0%	6 minutes	
getting-started-	-	-	-	63.32%	2 minutes	
app-1	fa6b12e91793	node:18-al	3000:3000	61.69%	2 minutes	
mysql-1	c3ac2792a869	mysql:8.0		1.63%	2 minutes	

Showing 4 items

2. Ran the entire setup:

```
vdsou@DESKTOP-55IA22L MINGW64 /e/docker/getting-started-app (main)
$ docker compose up -d
[+] Running 3/3
 ✓ Volume "getting-started-app_todo-mysql-data"  Created
 ✓ Container getting-started-app-mysql-1         Started
 ✓ Container getting-started-app-app-1           Started
```



Conclusion

This project demonstrated how to containerize, update, share, and scale an application using Docker.