Bharatiya Vidya Bhavan's

SARDAR PATEL INSTITUTE OF TECHNOLOGY

An Autonomous Institute Affiliated To University Of Mumbai

Munshi Nagar,Andheri (W) Mumbai 400 058

C.I.T.L. EXPERIMENT 10

Topic : Inventory Management System

Submitted By:

Akash Panicker	2021300089
Mahesh Patil	2021300095
Rohit Phalke	2021300100
Adwait Purao	2021300101

Submitted To:

Prof. Sunil Ghane

Aim:
Use docker-compose to run a multi-container web application

Problem Statement:
Develop an inventory management system for a retail store that efficiently tracks and manages the inventory of products. The system should provide real-time updates on stock levels, generate alerts for low stock items, enable easy addition and removal of products, and offer insights into sales trends to optimize restocking decisions.

Theory:

- **Containerization and Docker**
  Containerization is a lightweight virtualization technology that allows developers to package software applications and their dependencies into self-contained units called containers. Containers are isolated from each other, making them highly portable and scalable. Docker is the most popular containerization platform, providing a comprehensive set of tools for building, running, and managing containers.

- **Multi-Container Web Applications**
  Web applications often consist of multiple components, such as a web server, a database, and a cache. Traditionally, these components would be deployed on separate servers, which can be complex to manage and scale. Containerization offers a more efficient and flexible solution for deploying multi-container web applications.

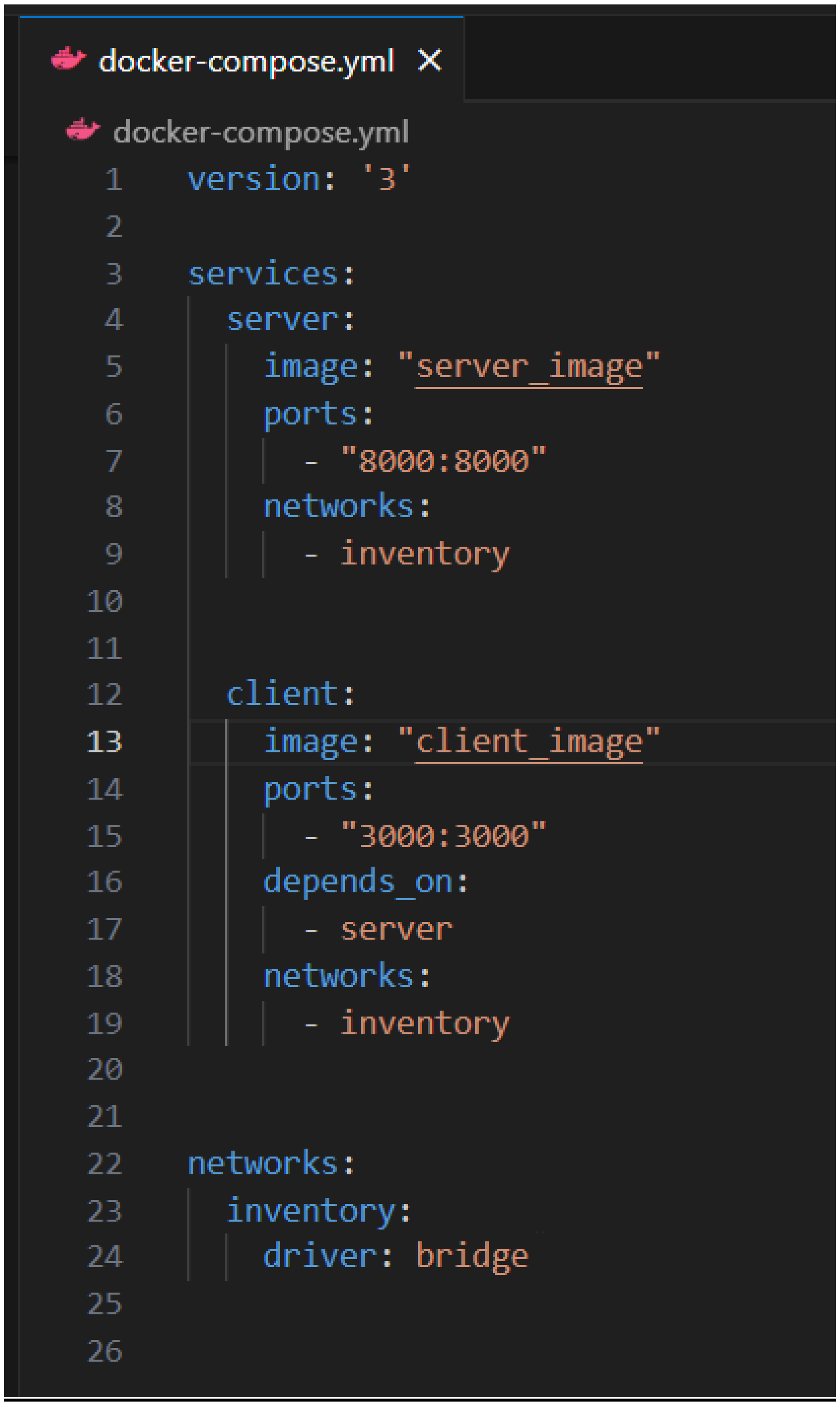
- **Docker Compose**
  Docker Compose is a tool for defining and running multi-container Docker applications. It simplifies the process of managing multiple containers, allowing developers to start, stop, and update all containers in a stack with a single command. Docker Compose also handles networking and dependency management between containers.

- **Running a Multi-Container Web Application with Docker Compose**
  To run a multi-container web application with Docker Compose, you typically follow these steps:
  - **Create a docker-compose.yml file:** This file defines the services that make up your application, specifying their images, ports, volumes, and other configuration options.
  - **Build the application images:** If you have Dockerfiles for your application components, you need to build them before running Docker Compose.
  - **Start the application:** Run the docker-compose up command to start all the containers defined in the docker-compose.yml file.

Docker Compose will handle the communication and dependency management between the containers, allowing your web application to run seamlessly.

- **Benefits of Using Docker Compose for Multi-Container Web Applications**
    - Docker Compose offers several benefits for running multi-container web applications:
    - Simplified deployment and management: With Docker Compose, you can start, stop, and update all containers in your application with a single command.
    - Portable and scalable: Docker Compose applications can be easily deployed across different environments, from local development machines to production servers.
    - Isolated and secure: Containers are isolated from each other, making them more secure and preventing conflicts between applications.
    - Resource-efficient: Containers share the host machine's kernel, making them more resource-efficient than traditional virtualization technologies.

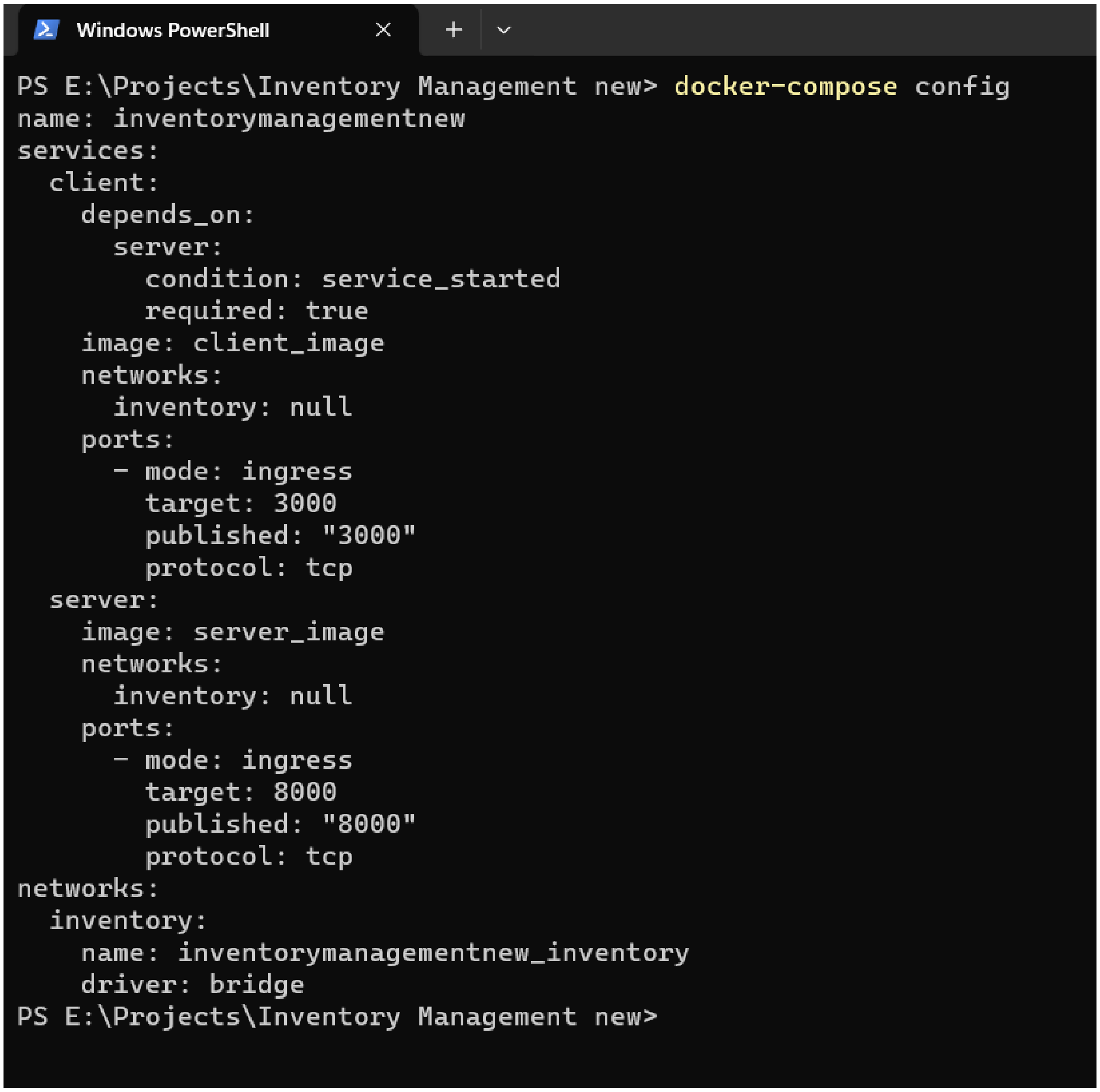
<u>Screenshots:</u>
docker-compose.yml file


docker-compose.yml ×

docker-compose.yml

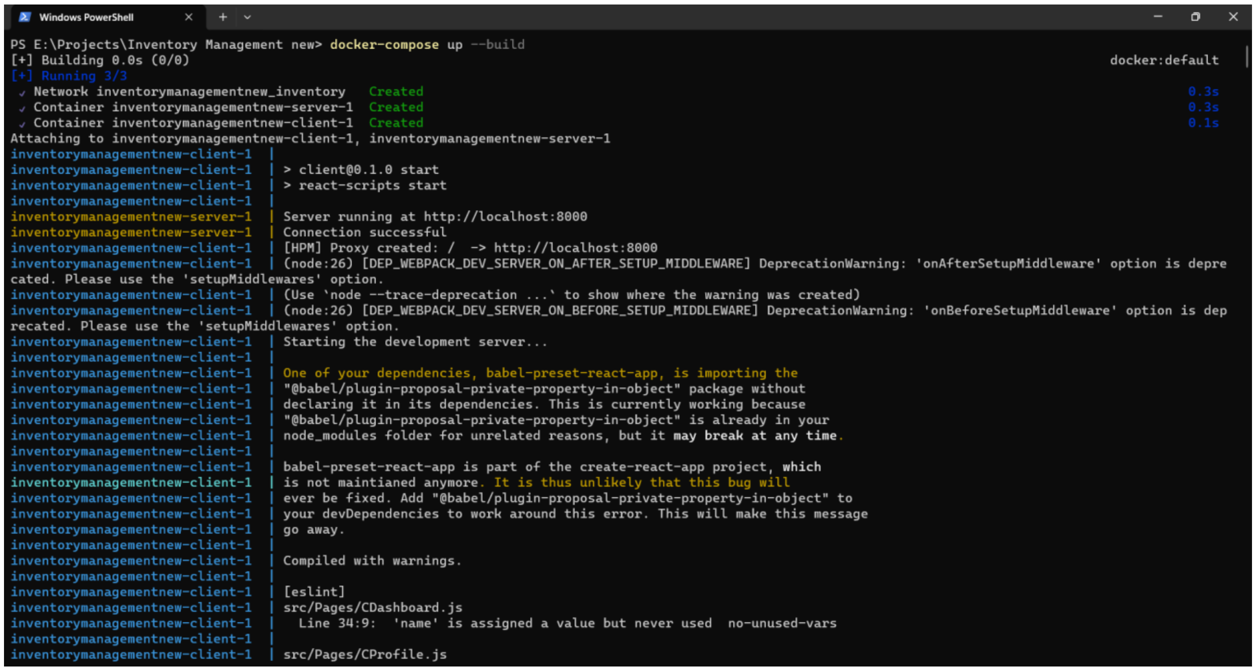
```yaml
version: '3'

services:
  server:
    image: "server_image"
    ports:
      - "8000:8000"
    networks:
      - inventory


  client:
    image: "client_image"
    ports:
      - "3000:3000"
    depends_on:
      - server
    networks:
      - inventory


networks:
  inventory:
    driver: bridge


```

Docker compose configuration



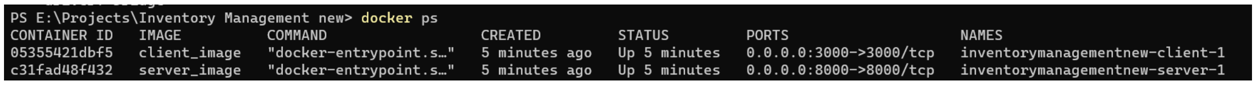
```
PS E:\Projects\Inventory Management new> docker-compose config
name: inventorymanagementnew
services:
  client:
    depends_on:
      server:
        condition: service_started
        required: true
    image: client_image
    networks:
      inventory: null
    ports:
      - mode: ingress
        target: 3000
        published: "3000"
        protocol: tcp
  server:
    image: server_image
    networks:
      inventory: null
    ports:
      - mode: ingress
        target: 8000
        published: "8000"
        protocol: tcp
networks:
  inventory:
    name: inventorymanagementnew_inventory
    driver: bridge
PS E:\Projects\Inventory Management new>
```

Running docker-compose.yml file thus running a multiple container application



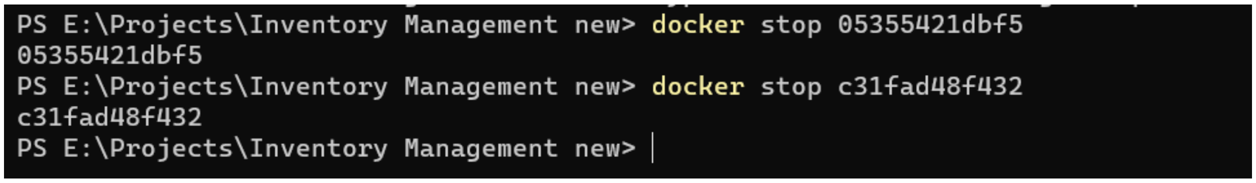
```
PS E:\Projects\Inventory Management new> docker-compose up --build
[+] Building 0.0s (0/0)                                                                              docker:default
[+] Running 3/3
 ✓ Network inventorymanagementnew_inventory   Created                                                          0.3s
 ✓ Container inventorymanagementnew-server-1  Created                                                          0.3s
 ✓ Container inventorymanagementnew-client-1  Created                                                          0.1s
Attaching to inventorymanagementnew-client-1, inventorymanagementnew-server-1
inventorymanagementnew-client-1  |
inventorymanagementnew-client-1  | > client@0.1.0 start
inventorymanagementnew-client-1  | > react-scripts start
inventorymanagementnew-client-1  |
inventorymanagementnew-server-1  | Server running at http://localhost:8000
inventorymanagementnew-server-1  | Connection successful
inventorymanagementnew-client-1  | [HPM] Proxy created: /  -> http://localhost:8000
inventorymanagementnew-client-1  | (node:26) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is depre
cated. Please use the 'setupMiddlewares' option.
inventorymanagementnew-client-1  | (Use `node --trace-deprecation ...` to show where the warning was created)
inventorymanagementnew-client-1  | (node:26) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is dep
recated. Please use the 'setupMiddlewares' option.
inventorymanagementnew-client-1  | Starting the development server...
inventorymanagementnew-client-1  |
inventorymanagementnew-client-1  | One of your dependencies, babel-preset-react-app, is importing the
inventorymanagementnew-client-1  | "@babel/plugin-proposal-private-property-in-object" package without
inventorymanagementnew-client-1  | declaring it in its dependencies. This is currently working because
inventorymanagementnew-client-1  | "@babel/plugin-proposal-private-property-in-object" is already in your
inventorymanagementnew-client-1  | node_modules folder for unrelated reasons, but it may break at any time.
inventorymanagementnew-client-1  |
inventorymanagementnew-client-1  | babel-preset-react-app is part of the create-react-app project, which
inventorymanagementnew-client-1  | is not maintianed anymore. It is thus unlikely that this bug will
inventorymanagementnew-client-1  | ever be fixed. Add "@babel/plugin-proposal-private-property-in-object" to
inventorymanagementnew-client-1  | your devDependencies to work around this error. This will make this message
inventorymanagementnew-client-1  | go away.
inventorymanagementnew-client-1  |
inventorymanagementnew-client-1  | Compiled with warnings.
inventorymanagementnew-client-1  |
inventorymanagementnew-client-1  | [eslint]
inventorymanagementnew-client-1  | src/Pages/CDashboard.js
inventorymanagementnew-client-1  |   Line 34:9:  'name' is assigned a value but never used  no-unused-vars
inventorymanagementnew-client-1  |
inventorymanagementnew-client-1  | src/Pages/CProfile.js
```

Multiple containers running



```
PS E:\Projects\Inventory Management new> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                    NAMES
05355421dbf5   client_image   "docker-entrypoint.s…"   5 minutes ago   Up 5 minutes   0.0.0.0:3000->3000/tcp   inventorymanagementnew-client-1
c31fad48f432   server_image   "docker-entrypoint.s…"   5 minutes ago   Up 5 minutes   0.0.0.0:8000->8000/tcp   inventorymanagementnew-server-1
```

Containers stopped



```
PS E:\Projects\Inventory Management new> docker stop 05355421dbf5
05355421dbf5
PS E:\Projects\Inventory Management new> docker stop c31fad48f432
c31fad48f432
PS E:\Projects\Inventory Management new>
```

Conclusion:

Docker Compose is a valuable tool for simplifying the development and management of multi-container web applications. It provides a straightforward way to define, run, and scale complex application stacks, offering numerous advantages over traditional deployment methods. By leveraging Docker Compose, developers can streamline their workflow, enhance portability, improve resource utilization, and strengthen application security.