Report – Challenge 4: Scaling a Node.js Application Using Docker Compose

Introduction

This guide focuses on effectively scaling a Node.js application using Docker Compose, moving from a single instance to three to enhance the application's ability to manage increased traffic and maintain availability.

Prerequisites

- Docker Installation: Docker must be installed on your system. Installation instructions are available on Docker's official website.
- Command-Line Proficiency: Basic familiarity with terminal or command prompt commands is necessary.
- Technical Understanding: Foundational knowledge of Node.js and Docker concepts is beneficial for this tutorial.

Setup Instructions

1)Environment Preparation

- Install Docker by following the instructions provided on Docker’s official website.
- Verify the installation by checking Docker's version in your command-line interface.

2)Docker Compose File Overview

- nginx: Acts as the reverse proxy for the Node.js application.
- node-service: The Node.js application you will scale.
- db: The database used by the application.

3)Modifying Docker Compose for Scaling

- Open your Docker Compose file and modify the node-service section to enable scaling.
- This involves specifying multiple instances (replicas) for the Node.js service.

4)Launching and Scaling the Application

- Start and scale your services by executing a command in your command-line interface that instructs Docker Compose to scale the Node.js service to three instances.
- Verify the scaling by observing if the responses come from different instances when accessing the application’s statistics page.

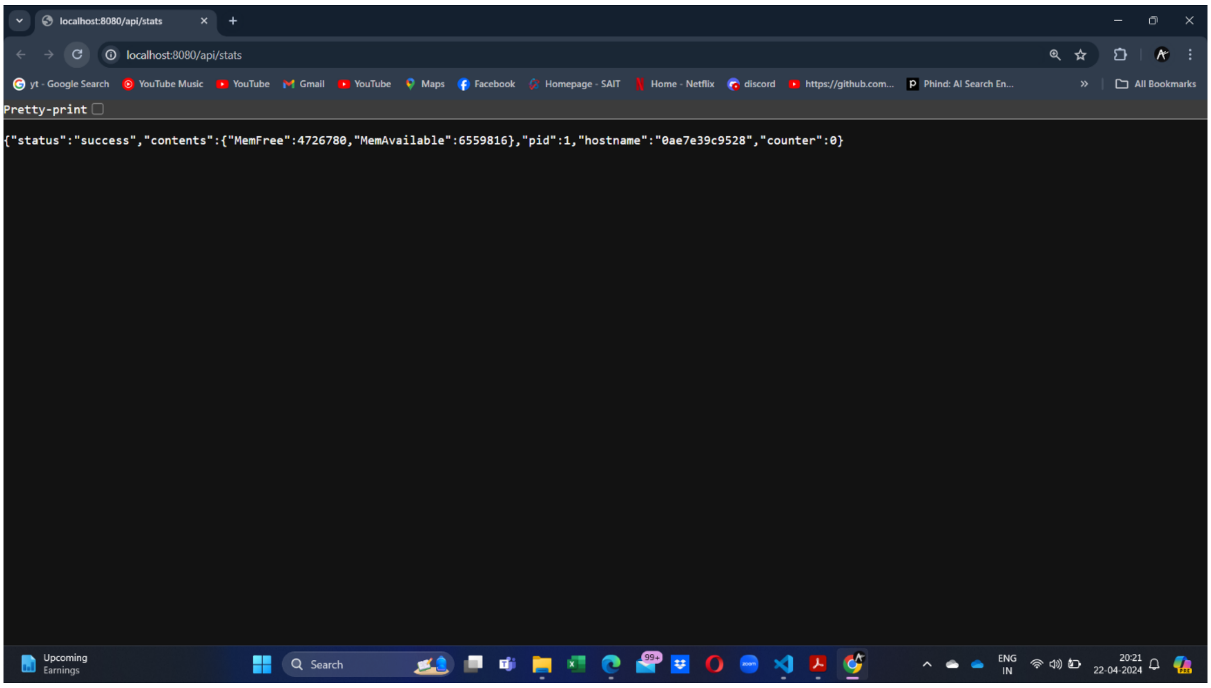
5)Output Documentation

- View the list of running containers to ensure that three instances of the Node.js service are active.
- Record the output and responses from different instances as evidence of successful scaling.

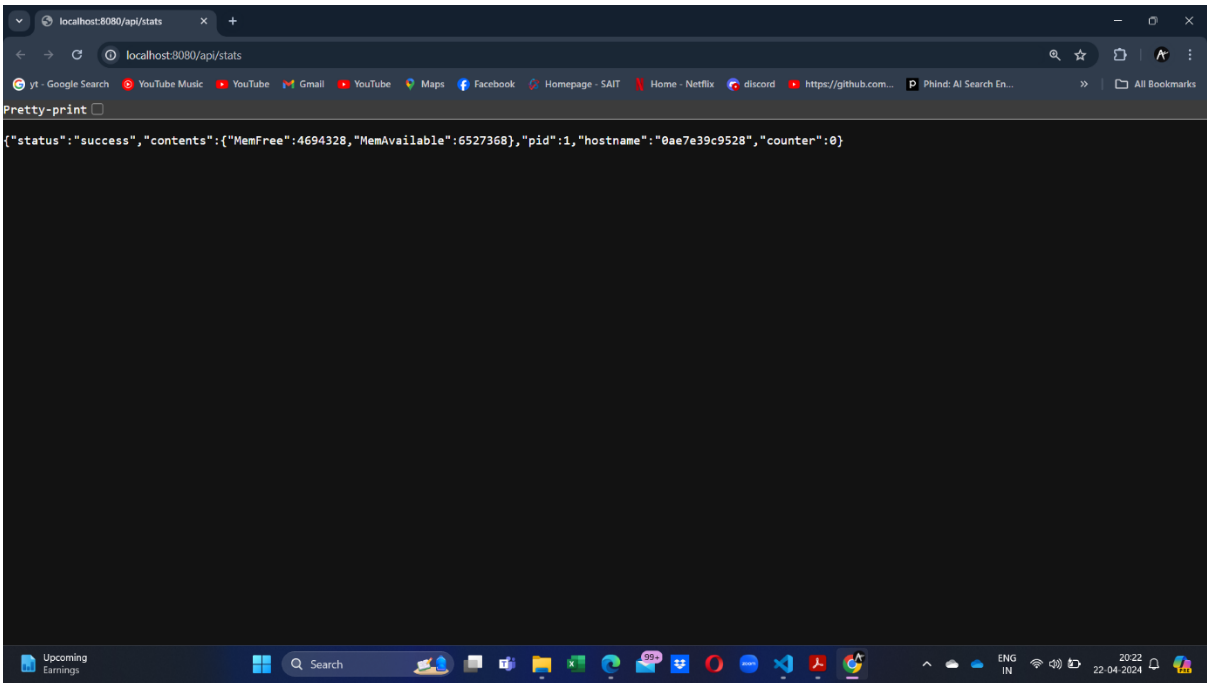
References

Docker, "Docker Documentation" [Online] Available: https://docs.docker.com [Accessed: Apr 22,2024]

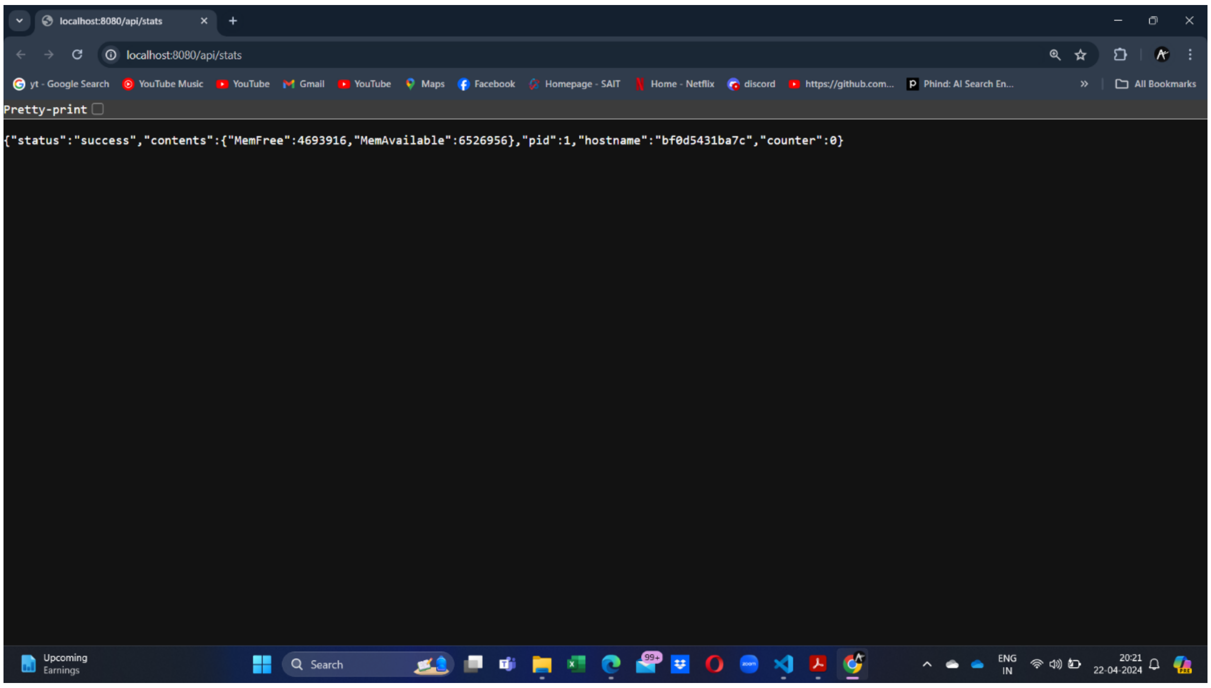
Screenshots



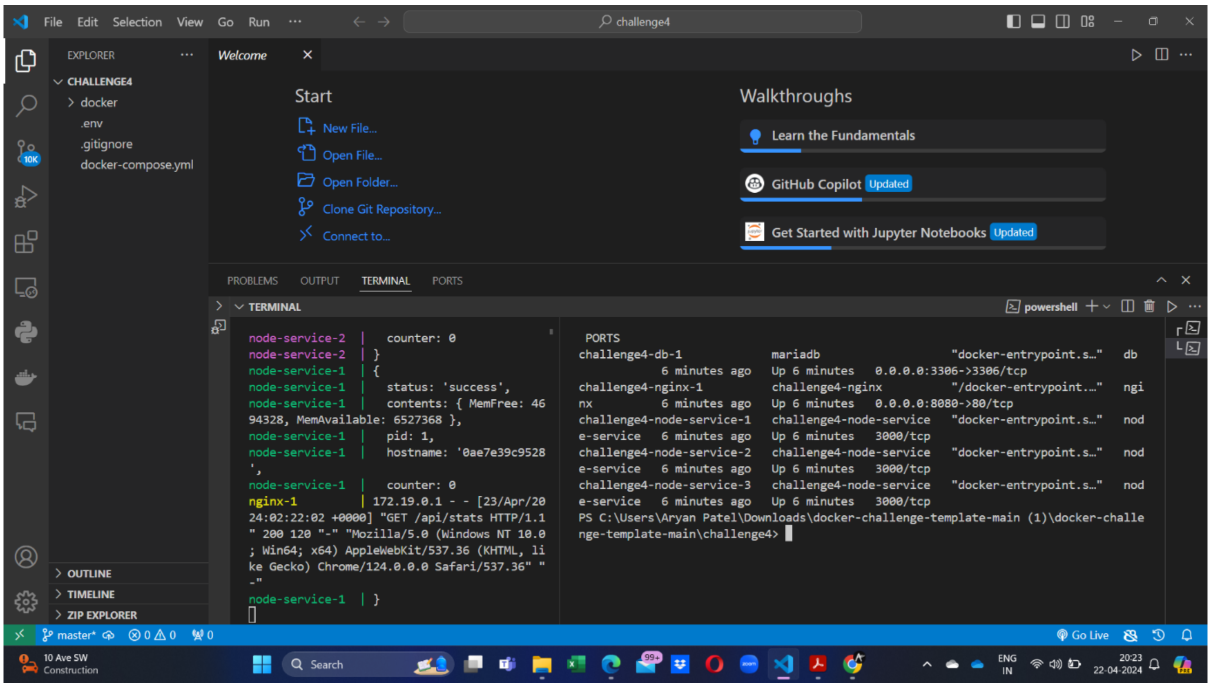
```
{"status":"success","contents":{"MemFree":4726780,"MemAvailable":6559816},"pid":1,"hostname":"0ae7e39c9528","counter":0}
```



```
{"status":"success","contents":{"MemFree":4694328,"MemAvailable":6527368},"pid":1,"hostname":"0ae7e39c9528","counter":0}
```

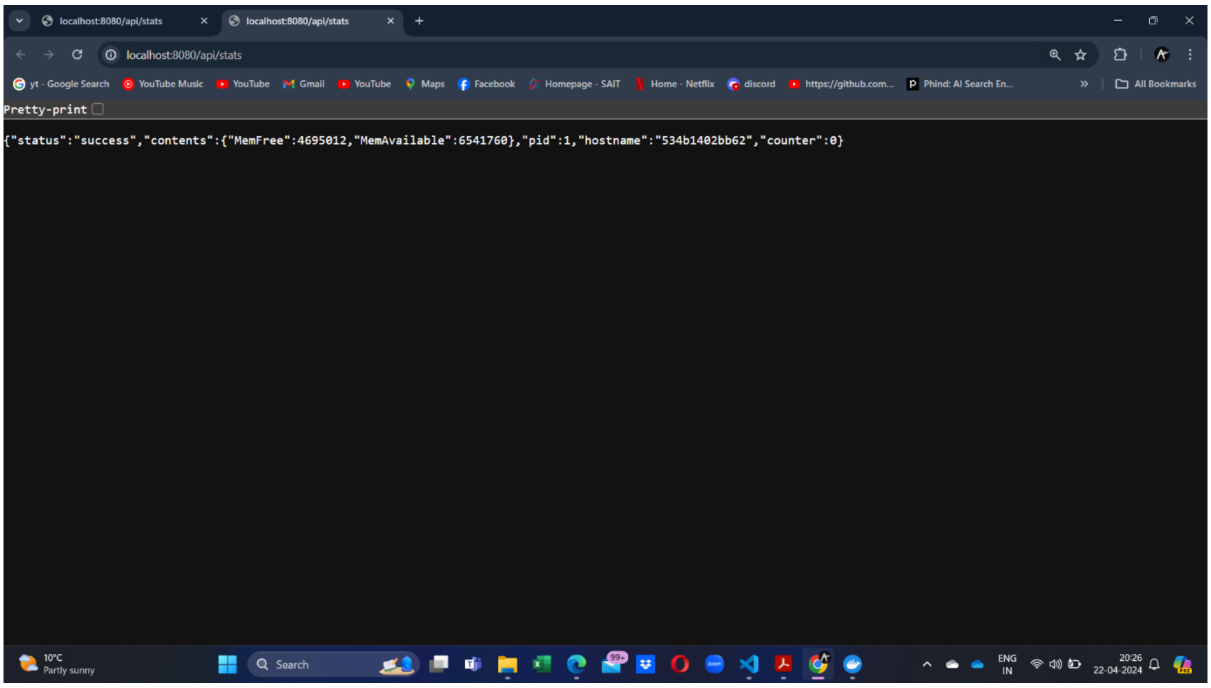


```
{"status":"success","contents":{"MemFree":4693916,"MemAvailable":6526956},"pid":1,"hostname":"bf0d5431ba7c","counter":0}
```



```
node-service-2  |   counter: 0
node-service-2  | }
node-service-1  | {
node-service-1  |   status: 'success',
node-service-1  |   contents: { MemFree: 46
94328, MemAvailable: 6527368 },
node-service-1  |   pid: 1,
node-service-1  |   hostname: '0ae7e39c9528
',
node-service-1  |   counter: 0
nginx-1         | 172.19.0.1 - - [23/Apr/20
24:02:22:02 +0000] "GET /api/stats HTTP/1.1
" 200 120 "-" "Mozilla/5.0 (Windows NT 10.0
; Win64; x64) AppleWebKit/537.36 (KHTML, li
ke Gecko) Chrome/124.0.0.0 Safari/537.36" "
-"
node-service-1  | }
```

```
 PORTS
challenge4-db-1             mariadb                  "docker-entrypoint.s…"   db
               6 minutes ago   Up 6 minutes   0.0.0.0:3306->3306/tcp
challenge4-nginx-1          challenge4-nginx         "/docker-entrypoint.…"   ngi
nx             6 minutes ago   Up 6 minutes   0.0.0.0:8080->80/tcp
challenge4-node-service-1   challenge4-node-service   "docker-entrypoint.s…"   nod
e-service      6 minutes ago   Up 6 minutes   3000/tcp
challenge4-node-service-2   challenge4-node-service   "docker-entrypoint.s…"   nod
e-service      6 minutes ago   Up 6 minutes   3000/tcp
challenge4-node-service-3   challenge4-node-service   "docker-entrypoint.s…"   nod
e-service      6 minutes ago   Up 6 minutes   3000/tcp
PS C:\Users\Aryan Patel\Downloads\docker-challenge-template-main (1)\docker-challe
nge-template-main\challenge4>
```



```
{"status":"success","contents":{"MemFree":4695012,"MemAvailable":6541760},"pid":1,"hostname":"534b1402bb62","counter":0}
```