BHARATIYA VIDYA BHAVAN'S

SARDAR PATEL INSTITUTE OF TECHNOLOGY

(Empowered Autonomous Institute Affiliated to University of Mumbai)

[Knowledge is Nectar]

Department of Computer Engineering

Course - DC

UID	2023800056
Name	Pratham Masurkar
Class and Batch	TE Computer Science and Engineering - Batch - A4
Lab #	10
Aim	To implement and demonstrate static and dynamic load balancing in distributed systems.
Objective	● To understand the concept of load balancing in distributed systems. ● To differentiate between static and dynamic load balancing techniques. ● To implement algorithms that distribute workload among multiple nodes or servers. ● To analyze system performance, resource utilization, and response time under different balancing strategies. ● To demonstrate how load balancing improves the efficiency and fault tolerance of distributed systems. ● To observe how dynamic load balancing adapts to runtime conditions compared to static methods. ● To simulate real-world scenarios such as web request distribution or task scheduling
Theory	**Load Balancing in Distributed Systems** Load balancing is a method used to distribute workloads evenly across multiple computing resources—such as servers, processors, or virtual machines—to maximize efficiency, prevent any single node from becoming overloaded, and improve the overall performance of a distributed system. It plays a crucial role in ensuring **high availability, scalability, and fault tolerance**, especially in environments with variable or unpredictable workloads. --- **Types of Load Balancing Techniques** **1. Static Load Balancing** Static load balancing assigns tasks to servers based on **predefined rules or fixed algorithms**, without considering the system's real-time conditions. These algorithms are

generally simpler and best suited for systems with consistent workloads.

Common static load balancing strategies include:

- **Round Robin:** Tasks or requests are sequentially distributed among all servers in a cyclical manner, ensuring even allocation.
- **Least Connections:** New requests are sent to the server handling the fewest active connections at that moment.

While static methods are easy to implement and perform well in stable environments, they **lack adaptability** in dynamic or unpredictable systems since they don't account for real-time variations in server load [1].

---

2. Dynamic Load Balancing

Dynamic load balancing continuously monitors the current state of the system—such as CPU usage, response time, and network latency—and distributes tasks accordingly. This allows it to **respond adaptively to workload fluctuations**, preventing performance bottlenecks.

Popular dynamic load balancing techniques include:

- **Weighted Round Robin (WRR):** Each server is assigned a weight according to its capacity or performance level, and tasks are distributed proportionally.
- **Least Response Time:** Requests are routed to the server that currently offers the fastest response time, improving user experience.
- **Randomized Algorithms:** These methods use probabilistic or feedback-based approaches to adjust load distribution dynamically.

Dynamic approaches are especially effective in **highly variable systems**, ensuring balanced workloads and avoiding server overloads [2].

Advantages and Challenges of Load Balancing

Advantages:
Load balancing enhances **system reliability, scalability, and resource utilization**, resulting in **lower response times** and improved user satisfaction. It also contributes to

	fault tolerance by redirecting traffic if a server fails. **Challenges:** Although dynamic load balancing provides flexibility and adaptability, it introduces **overhead from continuous monitoring and decision-making**. This increases the **complexity of implementation and maintenance**, particularly in large-scale distributed environments.
Code	Folder Structure (see below)

Folder Structure

```
load_balancer_demo/
├── app/
│   ├── app.py
│   ├── Dockerfile
│   └── requirements.txt
├── nginx/
│   └── nginx.conf
├── docker-compose.yml
└── load_test.py
```

app.py

```python
from flask import Flask
import socket

app = Flask(__name__)
@app.route("/")
def index():
    return f"Hello from {socket.gethostname()}"
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

Dockerfile

```dockerfile
FROM python:3.10-slim
WORKDIR /app

# Copy requirements and install
```

```
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy app code
COPY . .

# Expose port (documentation only)
EXPOSE 5000

# Start app
CMD ["python", "app.py"]
```

nginx.conf

```
events { }

http {
    upstream backend {
        # service names defined in docker-compose
        server app1:5000;
        server app2:5000;
        server app3:5000;
    }

    server {
        listen 80;
        location / {
            proxy_pass http://backend;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
        }
    }
}
```

docker-compose.yml

```yaml
version: '3'
services:
 app1:
   build: ./app
   container_name: app1
   expose:
     - "5000"


 app2:
   build: ./app
   container_name: app2
   expose:
     - "5000"


 app3:
   build: ./app
   container_name: app3
   expose:
     - "5000"


 app4:
   build: ./app
   container_name: app4
   expose:
     - "5000"


 app5:
   build: ./app
   container_name: app5
   expose:
     - "5000"
```

```
nginx:
  image: nginx:latest
  container_name: nginx
  ports:
    - "8080:80"
  volumes:
    - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
  depends_on:
    - app1
    - app2
    - app3
    - app4
    - app5
```

load_test.py

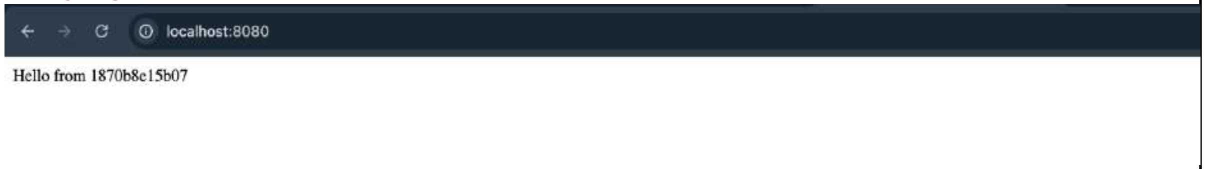
```
import requests
import time
from collections import defaultdict

url = "http://localhost:8080"
counter = defaultdict(int)
errors = 0

print("Sending 1000 requests to load balancer...\n")

for i in range(1000):
    try:
        response = requests.get(url, timeout=2)
        # response body is: "Hello from <hostname>"
        hostname = response.text.strip().split()[-1]
        counter[hostname] += 1
        print(f"[{i+1}] Handled by: {hostname}")
    except Exception as e:
        errors += 1
        print(f"[{i+1}] Error: {e}")
```

<table>
<tr><td></td><td>

```
print("\n--- Load Distribution Summary ---")
for name, count in counter.items():
    print(f"{name}: {count} requests")
print(f"\nFailed requests: {errors}")
```

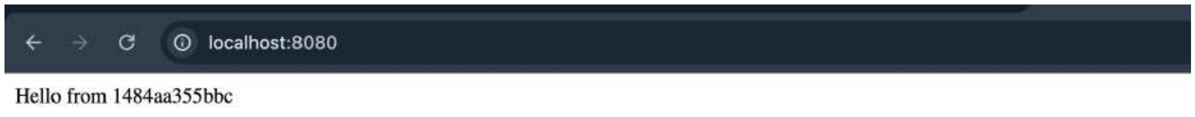
</td></tr>
<tr><td>**Output**</td><td>

Verifying setup :



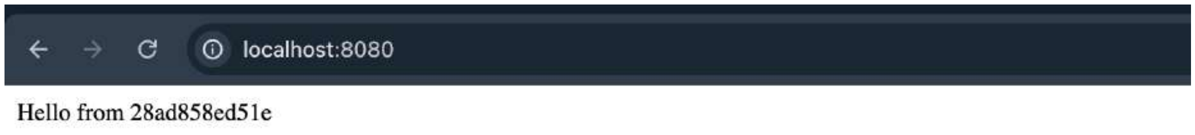
localhost:8080

Hello from 1870b8e15b07



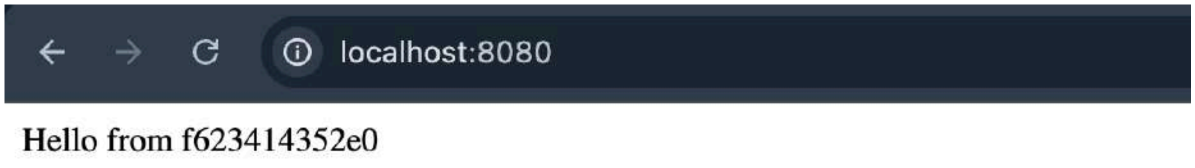
localhost:8080

Hello from 1484aa355bbc



localhost:8080

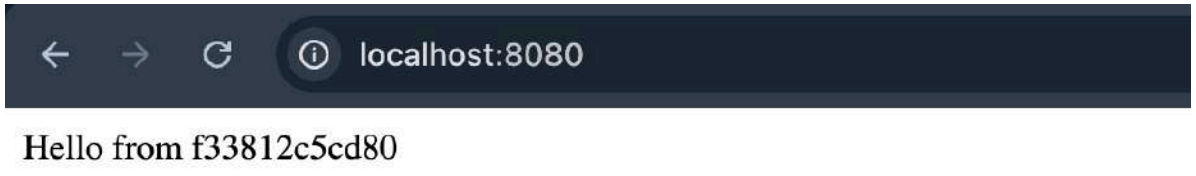
Hello from 28ad858ed51e

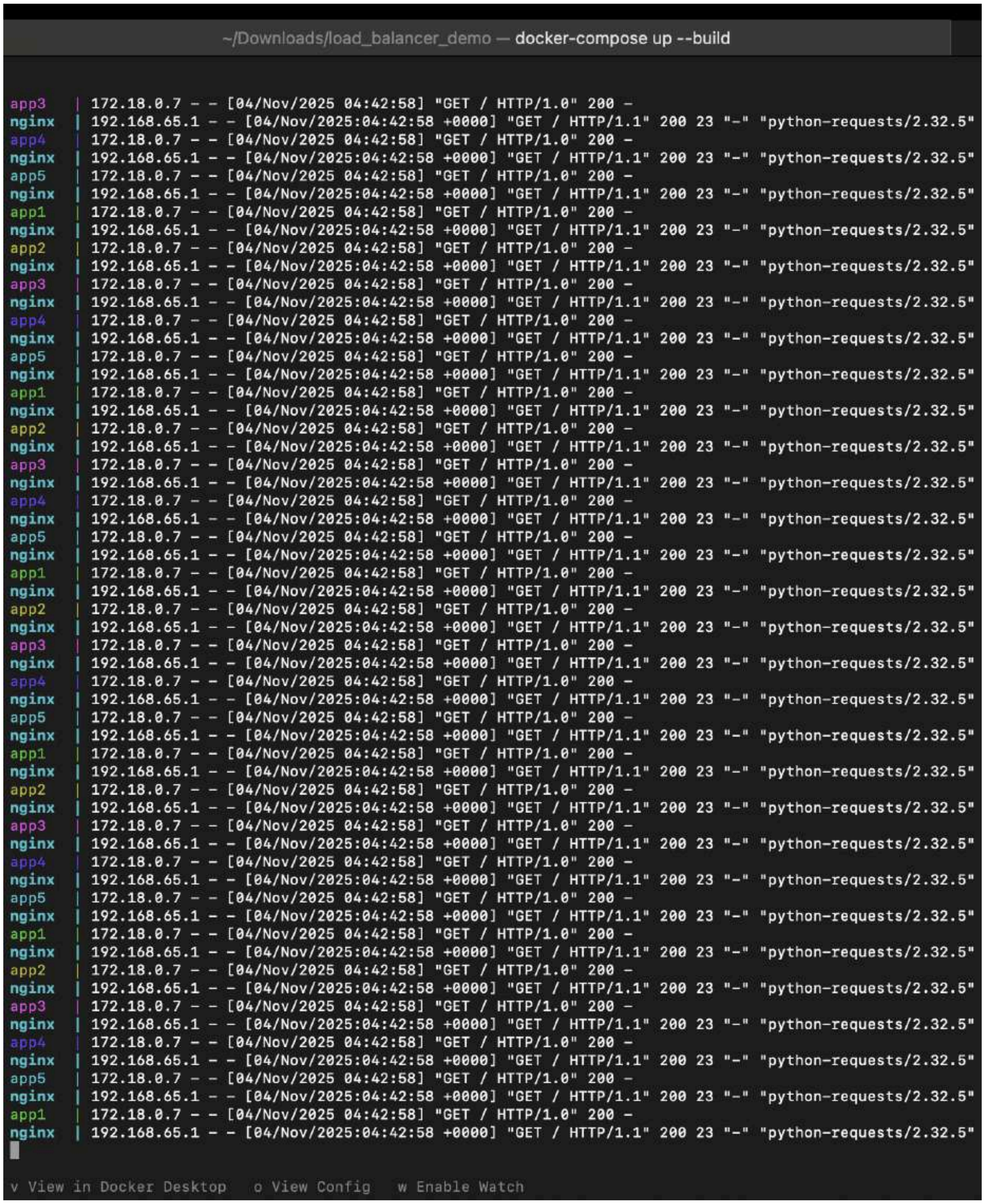


localhost:8080

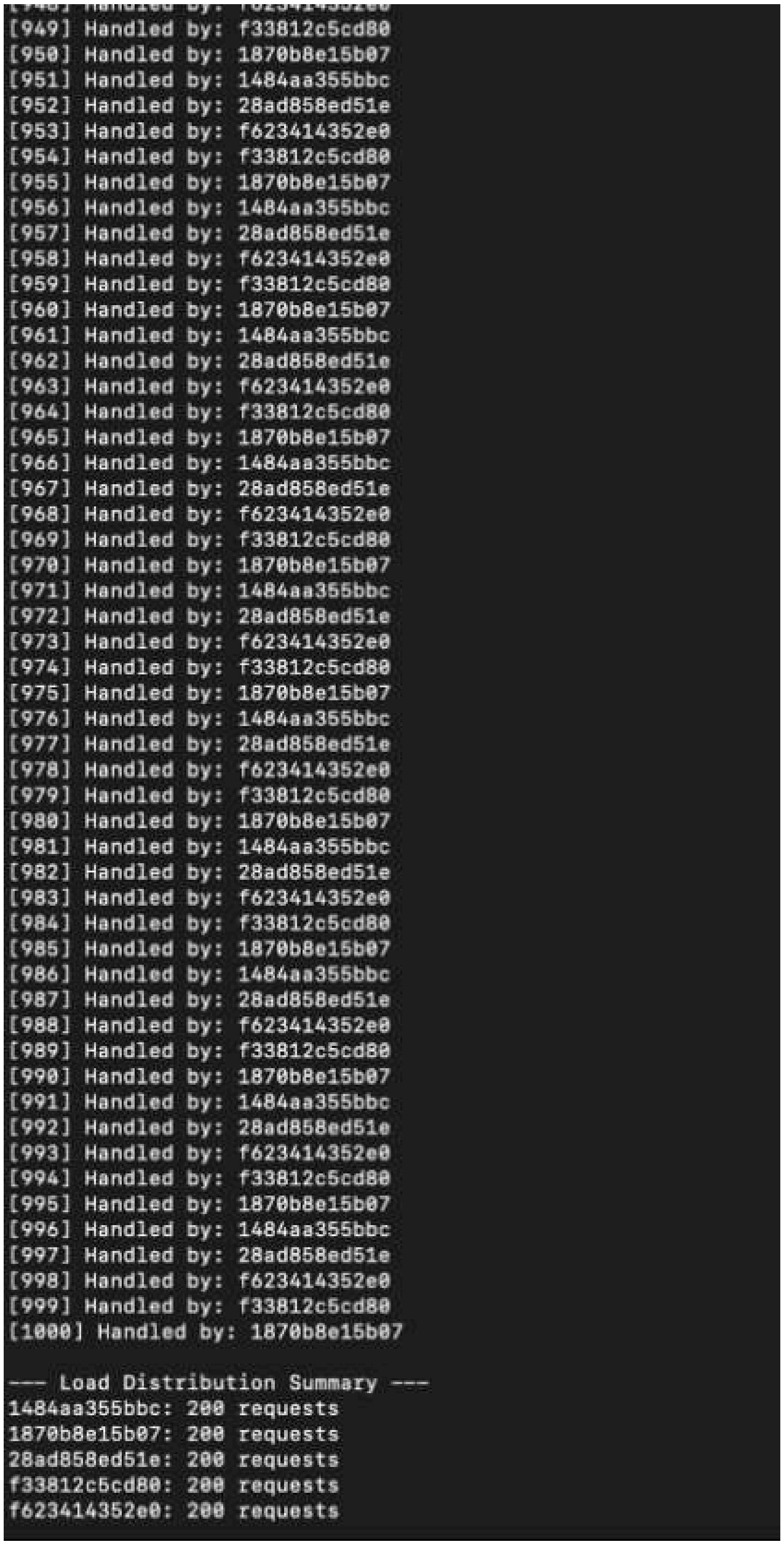
Hello from f623414352e0

</td></tr>
</table>

	 localhost:8080 Hello from f33812c5cd80 **Part 1(Static load balancing) :**

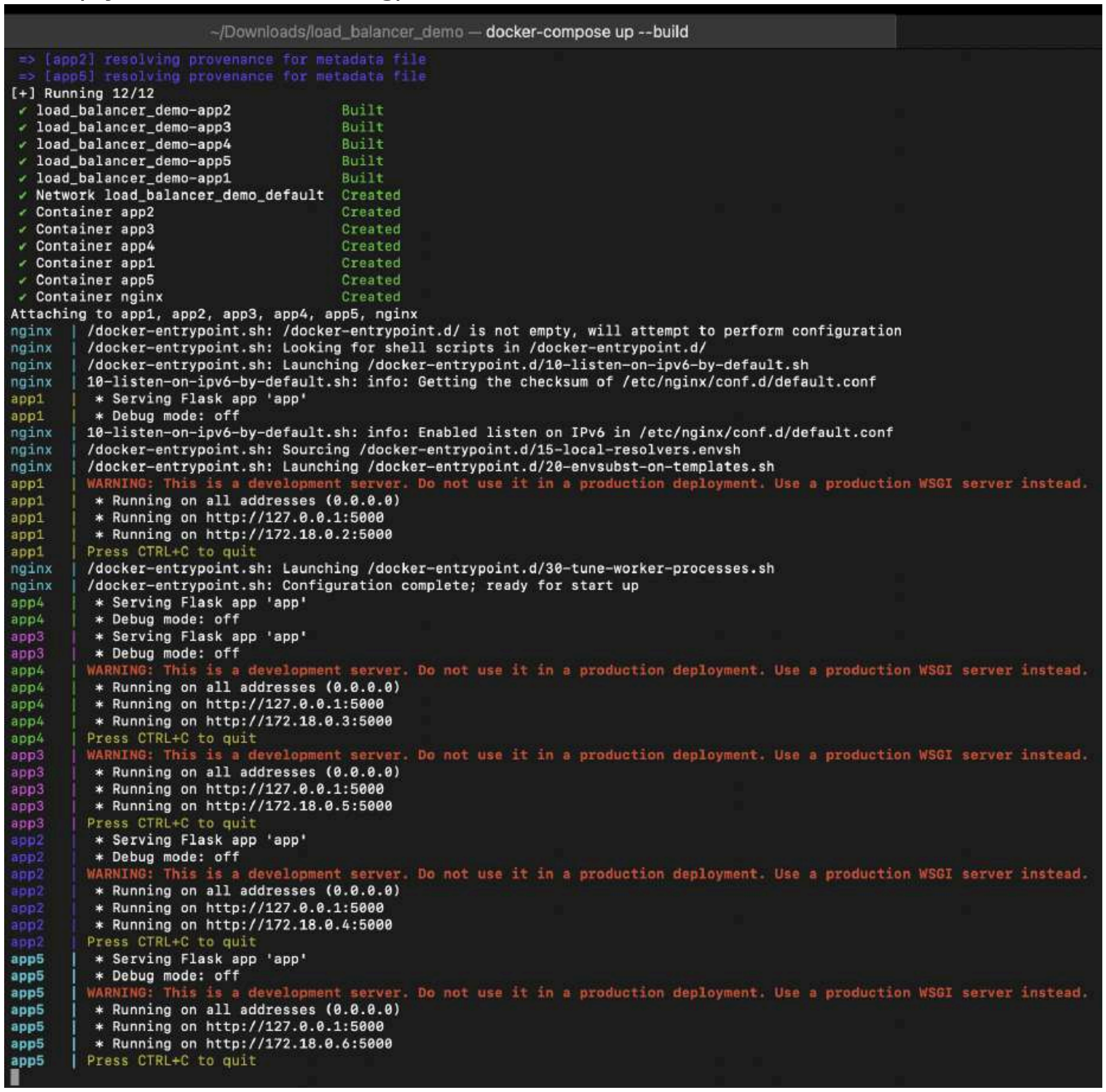


```
~/Downloads/load_balancer_demo — docker-compose up --build

app3   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app4   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app5   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app1   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app2   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app3   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app4   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app5   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app1   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app2   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app3   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app4   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app5   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app1   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app2   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app3   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app4   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app5   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app1   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app2   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app3   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app4   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app5   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app1   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app2   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app3   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app4   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app5   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"
app1   | 172.18.0.7 - - [04/Nov/2025 04:42:58] "GET / HTTP/1.0" 200 -
nginx  | 192.168.65.1 - - [04/Nov/2025:04:42:58 +0000] "GET / HTTP/1.1" 200 23 "-" "python-requests/2.32.5"

v View in Docker Desktop   o View Config   w Enable Watch
```

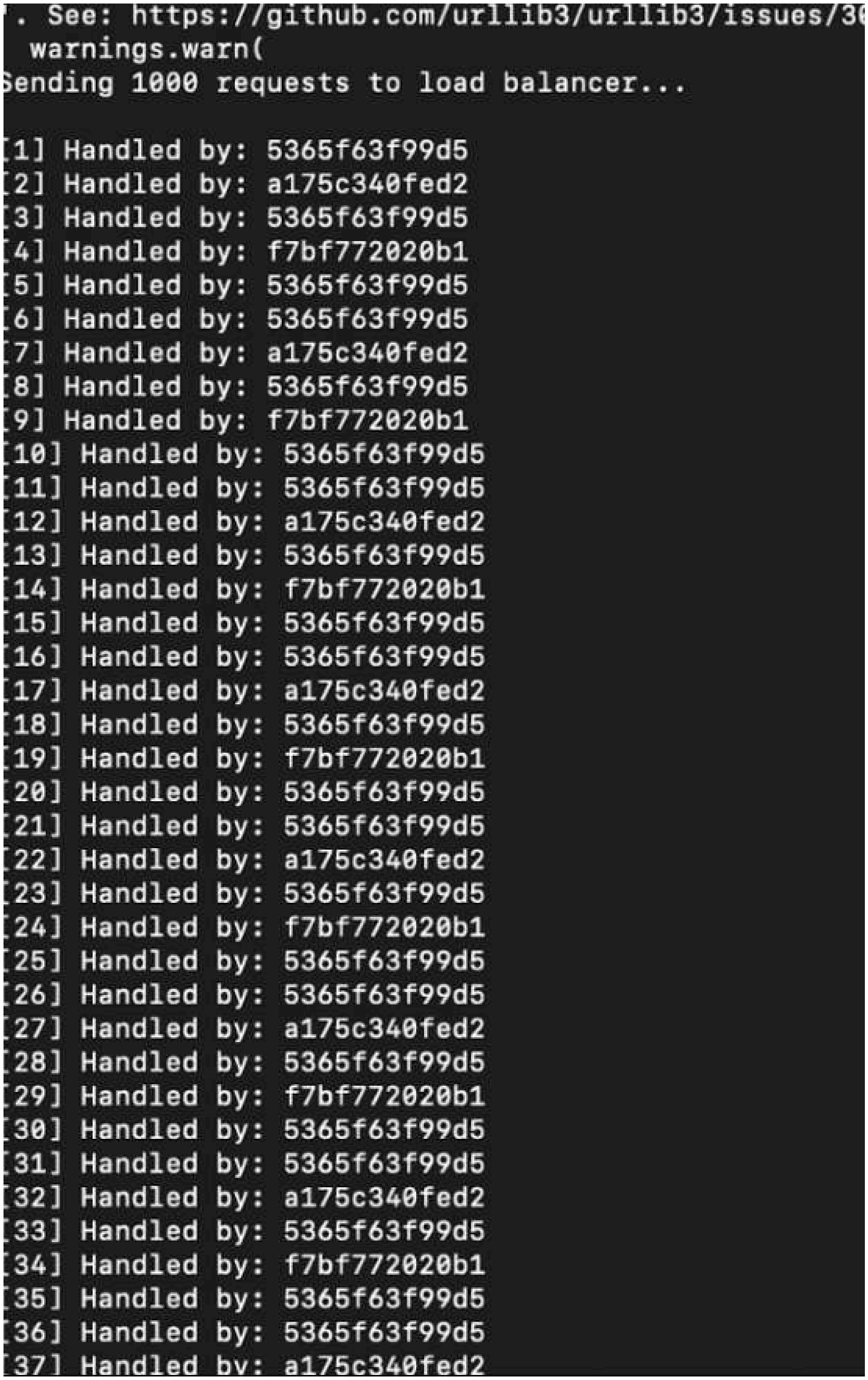


```
[949] Handled by: f33812c5cd80
[950] Handled by: 1870b8e15b07
[951] Handled by: 1484aa355bbc
[952] Handled by: 28ad858ed51e
[953] Handled by: f623414352e0
[954] Handled by: f33812c5cd80
[955] Handled by: 1870b8e15b07
[956] Handled by: 1484aa355bbc
[957] Handled by: 28ad858ed51e
[958] Handled by: f623414352e0
[959] Handled by: f33812c5cd80
[960] Handled by: 1870b8e15b07
[961] Handled by: 1484aa355bbc
[962] Handled by: 28ad858ed51e
[963] Handled by: f623414352e0
[964] Handled by: f33812c5cd80
[965] Handled by: 1870b8e15b07
[966] Handled by: 1484aa355bbc
[967] Handled by: 28ad858ed51e
[968] Handled by: f623414352e0
[969] Handled by: f33812c5cd80
[970] Handled by: 1870b8e15b07
[971] Handled by: 1484aa355bbc
[972] Handled by: 28ad858ed51e
[973] Handled by: f623414352e0
[974] Handled by: f33812c5cd80
[975] Handled by: 1870b8e15b07
[976] Handled by: 1484aa355bbc
[977] Handled by: 28ad858ed51e
[978] Handled by: f623414352e0
[979] Handled by: f33812c5cd80
[980] Handled by: 1870b8e15b07
[981] Handled by: 1484aa355bbc
[982] Handled by: 28ad858ed51e
[983] Handled by: f623414352e0
[984] Handled by: f33812c5cd80
[985] Handled by: 1870b8e15b07
[986] Handled by: 1484aa355bbc
[987] Handled by: 28ad858ed51e
[988] Handled by: f623414352e0
[989] Handled by: f33812c5cd80
[990] Handled by: 1870b8e15b07
[991] Handled by: 1484aa355bbc
[992] Handled by: 28ad858ed51e
[993] Handled by: f623414352e0
[994] Handled by: f33812c5cd80
[995] Handled by: 1870b8e15b07
[996] Handled by: 1484aa355bbc
[997] Handled by: 28ad858ed51e
[998] Handled by: f623414352e0
[999] Handled by: f33812c5cd80
[1000] Handled by: 1870b8e15b07

--- Load Distribution Summary ---
1484aa355bbc: 200 requests
1870b8e15b07: 200 requests
28ad858ed51e: 200 requests
f33812c5cd80: 200 requests
f623414352e0: 200 requests
```

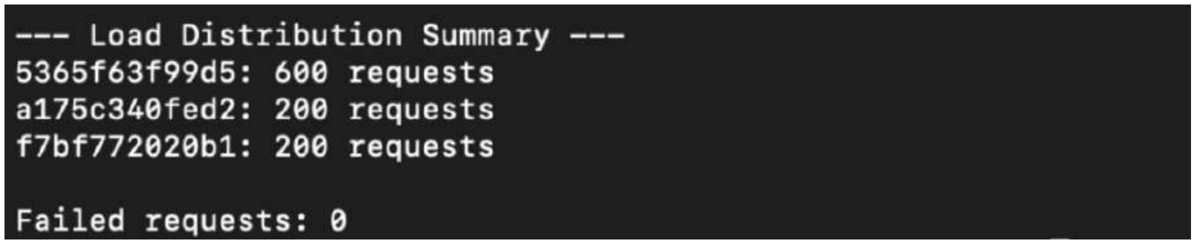
Part 2(Dynamic load balancing) :



```
~/Downloads/load_balancer_demo — docker-compose up --build
 => [app2] resolving provenance for metadata file
 => [app5] resolving provenance for metadata file
[+] Running 12/12
 ✔ load_balancer_demo-app2            Built
 ✔ load_balancer_demo-app3            Built
 ✔ load_balancer_demo-app4            Built
 ✔ load_balancer_demo-app5            Built
 ✔ load_balancer_demo-app1            Built
 ✔ Network load_balancer_demo_default  Created
 ✔ Container app2                     Created
 ✔ Container app3                     Created
 ✔ Container app4                     Created
 ✔ Container app1                     Created
 ✔ Container app5                     Created
 ✔ Container nginx                    Created
Attaching to app1, app2, app3, app4, app5, nginx
nginx  | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
nginx  | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
nginx  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
nginx  | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
app1   |  * Serving Flask app 'app'
app1   |  * Debug mode: off
nginx  | 10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
nginx  | /docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
nginx  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
app1   | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
app1   |  * Running on all addresses (0.0.0.0)
app1   |  * Running on http://127.0.0.1:5000
app1   |  * Running on http://172.18.0.2:5000
app1   | Press CTRL+C to quit
nginx  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
nginx  | /docker-entrypoint.sh: Configuration complete; ready for start up
app4   |  * Serving Flask app 'app'
app4   |  * Debug mode: off
app3   |  * Serving Flask app 'app'
app3   |  * Debug mode: off
app4   | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
app4   |  * Running on all addresses (0.0.0.0)
app4   |  * Running on http://127.0.0.1:5000
app4   |  * Running on http://172.18.0.3:5000
app4   | Press CTRL+C to quit
app3   | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
app3   |  * Running on all addresses (0.0.0.0)
app3   |  * Running on http://127.0.0.1:5000
app3   |  * Running on http://172.18.0.5:5000
app3   | Press CTRL+C to quit
app2   |  * Serving Flask app 'app'
app2   |  * Debug mode: off
app2   | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
app2   |  * Running on all addresses (0.0.0.0)
app2   |  * Running on http://127.0.0.1:5000
app2   |  * Running on http://172.18.0.4:5000
app2   | Press CTRL+C to quit
app5   |  * Serving Flask app 'app'
app5   |  * Debug mode: off
app5   | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
app5   |  * Running on all addresses (0.0.0.0)
app5   |  * Running on http://127.0.0.1:5000
app5   |  * Running on http://172.18.0.6:5000
app5   | Press CTRL+C to quit
```



```
. See: https://github.com/urllib3/urllib3/issues/3
  warnings.warn(
Sending 1000 requests to load balancer...

[1] Handled by: 5365f63f99d5
[2] Handled by: a175c340fed2
[3] Handled by: 5365f63f99d5
[4] Handled by: f7bf772020b1
[5] Handled by: 5365f63f99d5
[6] Handled by: 5365f63f99d5
[7] Handled by: a175c340fed2
[8] Handled by: 5365f63f99d5
[9] Handled by: f7bf772020b1
[10] Handled by: 5365f63f99d5
[11] Handled by: 5365f63f99d5
[12] Handled by: a175c340fed2
[13] Handled by: 5365f63f99d5
[14] Handled by: f7bf772020b1
[15] Handled by: 5365f63f99d5
[16] Handled by: 5365f63f99d5
[17] Handled by: a175c340fed2
[18] Handled by: 5365f63f99d5
[19] Handled by: f7bf772020b1
[20] Handled by: 5365f63f99d5
[21] Handled by: 5365f63f99d5
[22] Handled by: a175c340fed2
[23] Handled by: 5365f63f99d5
[24] Handled by: f7bf772020b1
[25] Handled by: 5365f63f99d5
[26] Handled by: 5365f63f99d5
[27] Handled by: a175c340fed2
[28] Handled by: 5365f63f99d5
[29] Handled by: f7bf772020b1
[30] Handled by: 5365f63f99d5
[31] Handled by: 5365f63f99d5
[32] Handled by: a175c340fed2
[33] Handled by: 5365f63f99d5
[34] Handled by: f7bf772020b1
[35] Handled by: 5365f63f99d5
[36] Handled by: 5365f63f99d5
[37] Handled by: a175c340fed2
```

	 `--- Load Distribution Summary ---`
	`5365f63f99d5: 600 requests`
	`a175c340fed2: 200 requests`
	`f7bf772020b1: 200 requests`
	
	`Failed requests: 0`

Conclusion	In this experiment, I explored both static and dynamic load balancing techniques to evaluate their performance within a distributed system. Static load balancing assigns tasks based on predefined rules to ensure an even distribution, whereas dynamic load balancing responds to real-time system conditions, enhancing fault tolerance and overall efficiency. Using a practical setup involving Flask, NGINX, and Docker, I simulated a load balancing environment where client requests were routed to multiple backend servers. The results showed that dynamic load balancing is more effective at adapting to workload fluctuations, preventing server overloads, and optimizing resource utilization. Testing with live requests also revealed that each approach offers distinct benefits depending on the specific use case and system requirements.