Report – Challenge 4

Scaling a Node.js Application with Docker Compose

Introduction

Scaling an application effectively is crucial for handling increased traffic and ensuring high availability. This tutorial will demonstrate how to scale a Node.js service from one to three instances using Docker Compose.

Prerequisites

- **Docker**: Docker must be installed on your machine. To install Docker, visit Docker's official website and choose the appropriate version for your operating system.
- **Basic understanding of command-line tools**: Familiarity with using terminal or command prompt.
- **Node.js and Docker familiarity**: Basic understanding of Node.js and Docker concepts will be beneficial.

Project Structure Overview

Directory Layout: The project directory consists of directories for each component (api, db, nginx) and essential files such as docker-compose.yml, .env, and configuration files (nginx.conf, server.js, init.sql).

Key Files Explained:

- **docker-compose.yml:** Manages the multi-container setup.
- **Dockerfile:** Contains instructions for building Docker images for each component.
- **.env:** Contains environment variables required for configuration.
- **nginx.conf:** Configuration file for the Nginx server.
- **server.js:** Script for the Node.js application.
- **init.sql:** SQL script to initialize the database.

Detailed Setup Steps

1: Setting Up Your Environment

- **Install Docker**: Follow the instructions on Docker’s website to download and install Docker Desktop.

- **Verify Installation**: Open your terminal or command prompt and type **docker --version** to ensure Docker was installed correctly.

2: Understanding the Docker Compose File

Before scaling, it's important to understand the components of your Docker Compose file. Here is a brief overview of the service definitions:

- **nginx**: Serves as the reverse proxy to your Node.js application.
- **node-service**: The Node.js application you wish to scale.
- **db**: A MariaDB instance serving as the database for your application.

3: Modifying the Docker Compose File

To enable scaling of the Node service, modify the **docker-compose.yml** file:

1. Open your Docker Compose file (**docker-compose.yml**) in a text editor.
2. Find the **node-service** section and add a **deploy** key with **replicas: 3** under it as shown below:

```
node-service:
 build: ./docker/api
environment:
DB_HOST: db
DB_USERNAME: ${DB_USERNAME}
DB_PASSWORD: ${DB_PASSWORD}
DB_DATABASE: ${DB_DATABASE}
PORT: 3000
depends_on: - db
deploy: replicas: 3
```

Step 4: Running and Scaling the Application

- **Launch Docker Compose**:
  - Open a terminal and navigate to the directory containing your **docker-compose.yml**.
  - Run the following command to start and scale your services:

```
docker-compose up --scale node-service=3 -d
```

- **Verify Scaling**:
  - Make multiple requests to **http://localhost:8080/api/stats** and observe if the **hostname** changes, which indicates that different instances are serving the requests.

Step 5: Documenting the Output

- Run **docker-compose ps** to see the list of running containers. You should see three instances of the **node-service**.
- Record this output along with the varied **hostname** responses as evidence of successful scaling.

Conclusion

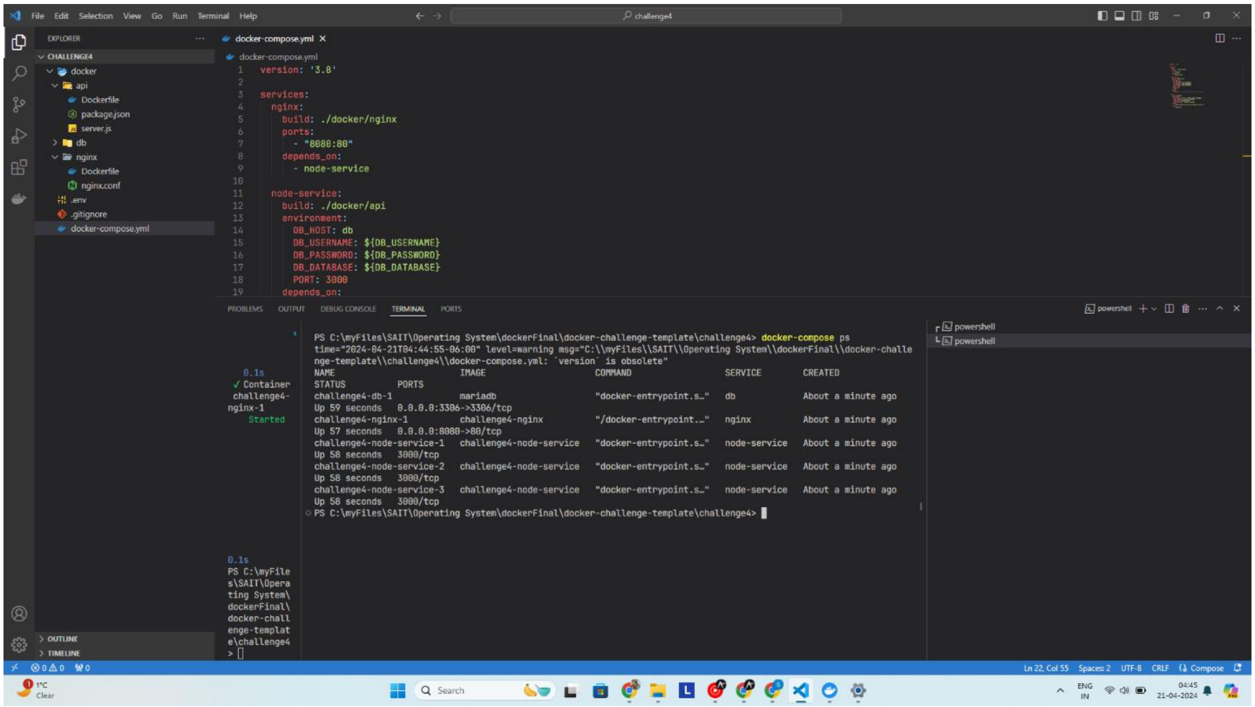
You have successfully scaled a Node.js application using Docker Compose. This setup demonstrates basic load balancing across multiple instances, improving the application's ability to handle traffic and providing redundancy.

References:

Docker, "Docker Documentation," Docker. [Online]. Available: https://docs.docker.com. [Accessed: Apr. 20, 2024].

Node.js, "Node.js Docker Best Practices," Node.js. [Online]. Available: https://nodejs.org/en/docs/guides/nodejs-docker-webapp/. [Accessed: Apr. 20, 2024].

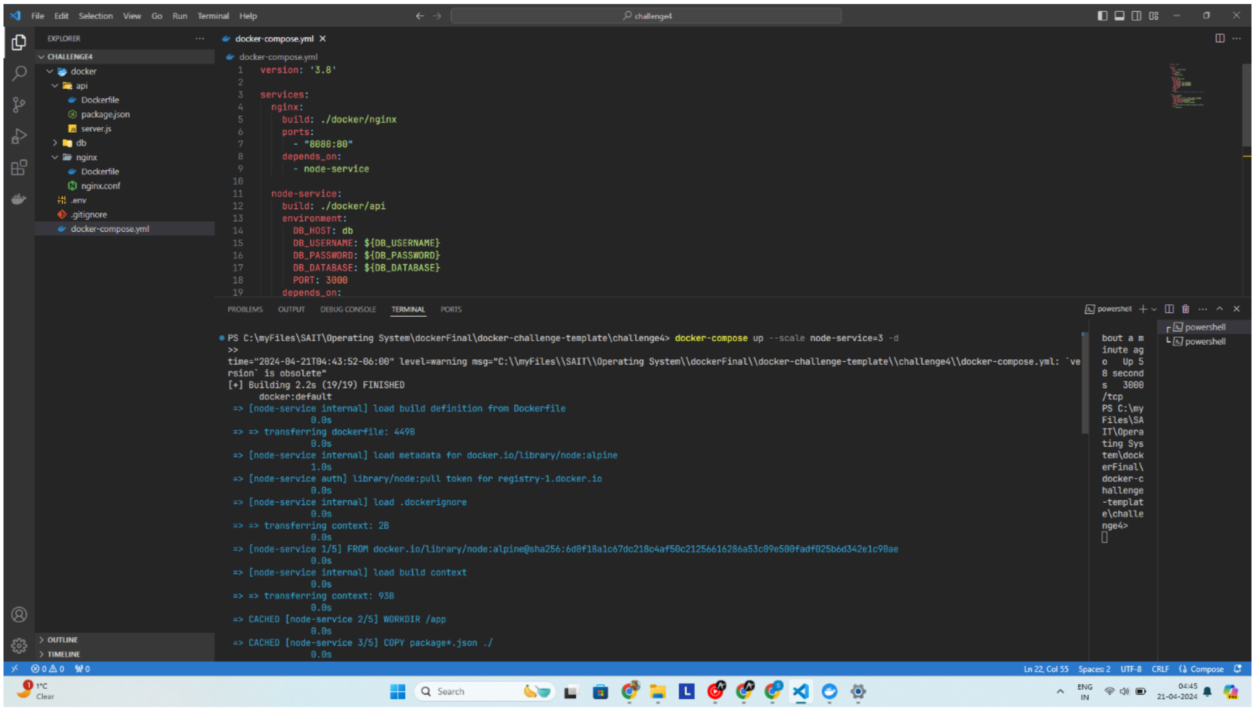
MariaDB, "MariaDB Docker Image," Docker Hub. [Online]. Available: https://hub.docker.com/_/mariadb. [Accessed: Apr. 20, 2024].



```
version: '3.8'

services:
  nginx:
    build: ./docker/nginx
    ports:
      - "8080:80"
    depends_on:
      - node-service

  node-service:
    build: ./docker/api
    environment:
      DB_HOST: db
      DB_USERNAME: ${DB_USERNAME}
      DB_PASSWORD: ${DB_PASSWORD}
      DB_DATABASE: ${DB_DATABASE}
      PORT: 3000
    depends_on:
```

```
  0.1s
 ✓ Container
 challenge4-
nginx-1
    Started

0.1s
PS C:\myFile
s\SAIT\Opera
ting System\
dockerFinal\
docker-chall
enge-templat
e\challenge4
> 
```

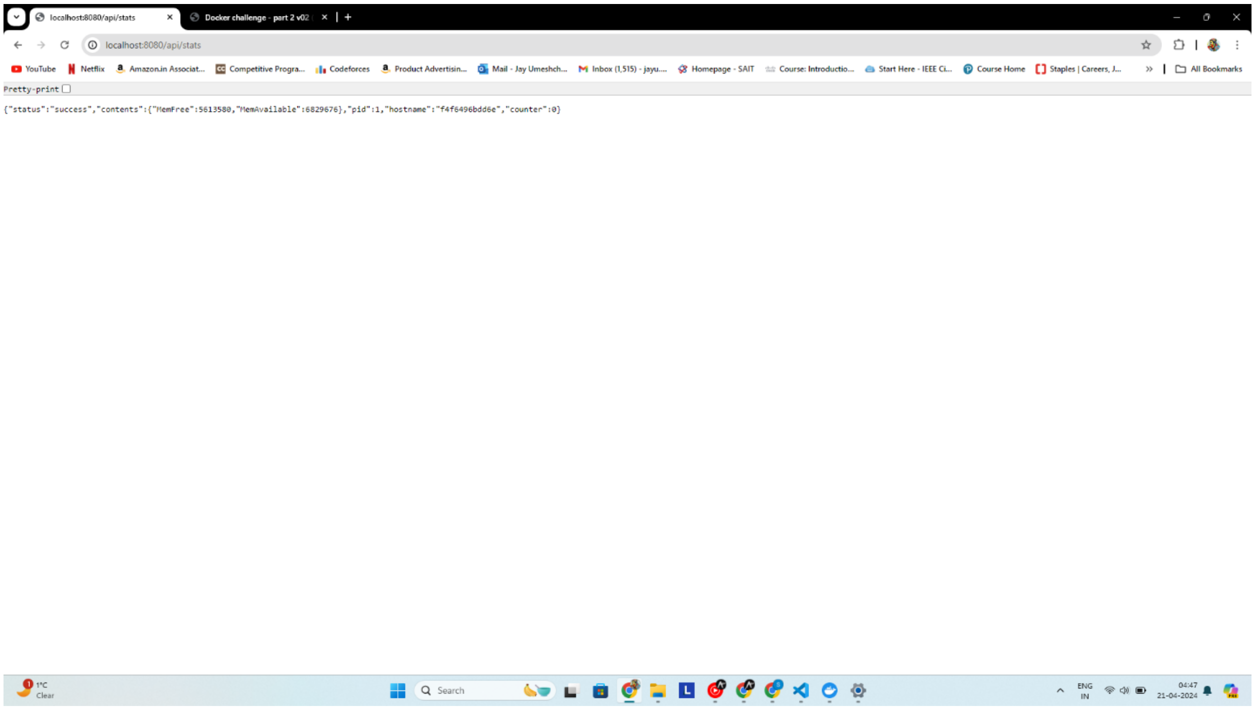
```
PS C:\myFiles\SAIT\Operating System\dockerFinal\docker-challenge-template\challenge4> docker-compose ps
time="2024-04-21T04:44:55-06:00" level=warning msg="C:\\myFiles\\SAIT\\Operating System\\dockerFinal\\docker-challe
nge-template\\challenge4\\docker-compose.yml: `version` is obsolete"
NAME                         IMAGE                    COMMAND                  SERVICE        CREATED
STATUS          PORTS
challenge4-db-1              mariadb                  "docker-entrypoint.s…"   db             About a minute ago
Up 59 seconds   0.0.0.0:3306->3306/tcp
challenge4-nginx-1           challenge4-nginx         "/docker-entrypoint.…"   nginx          About a minute ago
Up 57 seconds   0.0.0.0:8080->80/tcp
challenge4-node-service-1    challenge4-node-service  "docker-entrypoint.s…"   node-service   About a minute ago
Up 58 seconds   3000/tcp
challenge4-node-service-2    challenge4-node-service  "docker-entrypoint.s…"   node-service   About a minute ago
Up 58 seconds   3000/tcp
challenge4-node-service-3    challenge4-node-service  "docker-entrypoint.s…"   node-service   About a minute ago
Up 58 seconds   3000/tcp
PS C:\myFiles\SAIT\Operating System\dockerFinal\docker-challenge-template\challenge4> 
```



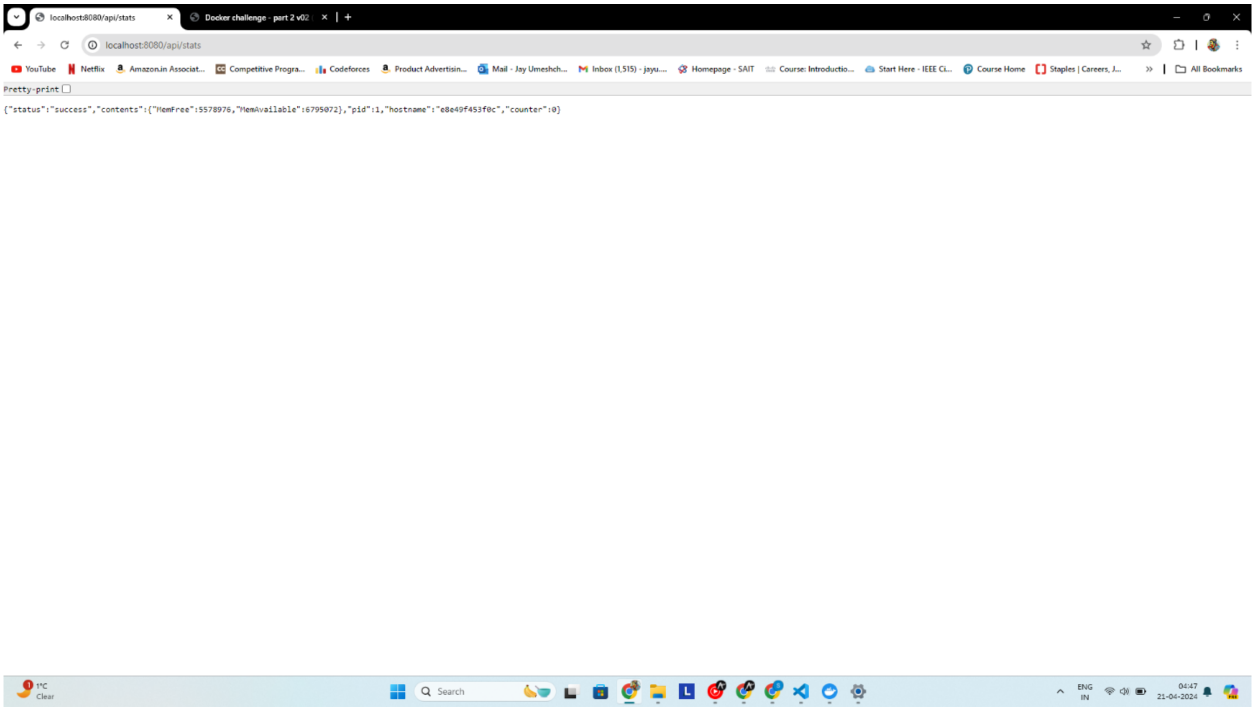
```
version: '3.8'

services:
  nginx:
    build: ./docker/nginx
    ports:
      - "8080:80"
    depends_on:
      - node-service

  node-service:
    build: ./docker/api
    environment:
      DB_HOST: db
      DB_USERNAME: ${DB_USERNAME}
      DB_PASSWORD: ${DB_PASSWORD}
      DB_DATABASE: ${DB_DATABASE}
      PORT: 3000
    depends_on:
```

```
PS C:\myFiles\SAIT\Operating System\dockerFinal\docker-challenge-template\challenge4> docker-compose up --scale node-service=3 -d
>>
time="2024-04-21T04:43:52-06:00" level=warning msg="C:\\myFiles\\SAIT\\Operating System\\dockerFinal\\docker-challenge-template\\challenge4\\docker-compose.yml: `ve
rsion` is obsolete"
[+] Building 2.2s (19/19) FINISHED
     docker:default
 => [node-service internal] load build definition from Dockerfile
           0.0s
 => => transferring dockerfile: 449B
           0.0s
 => [node-service internal] load metadata for docker.io/library/node:alpine
           1.0s
 => [node-service auth] library/node:pull token for registry-1.docker.io
           0.0s
 => [node-service internal] load .dockerignore
           0.0s
 => => transferring context: 2B
           0.0s
 => [node-service 1/5] FROM docker.io/library/node:alpine@sha256:6d0f18a1c67dc218c4af50c21256616286a53c09e500fadf025b6d342e1c90ae
           0.0s
 => [node-service internal] load build context
           0.0s
 => => transferring context: 93B
           0.0s
 => CACHED [node-service 2/5] WORKDIR /app
           0.0s
 => CACHED [node-service 3/5] COPY package*.json ./
           0.0s
```

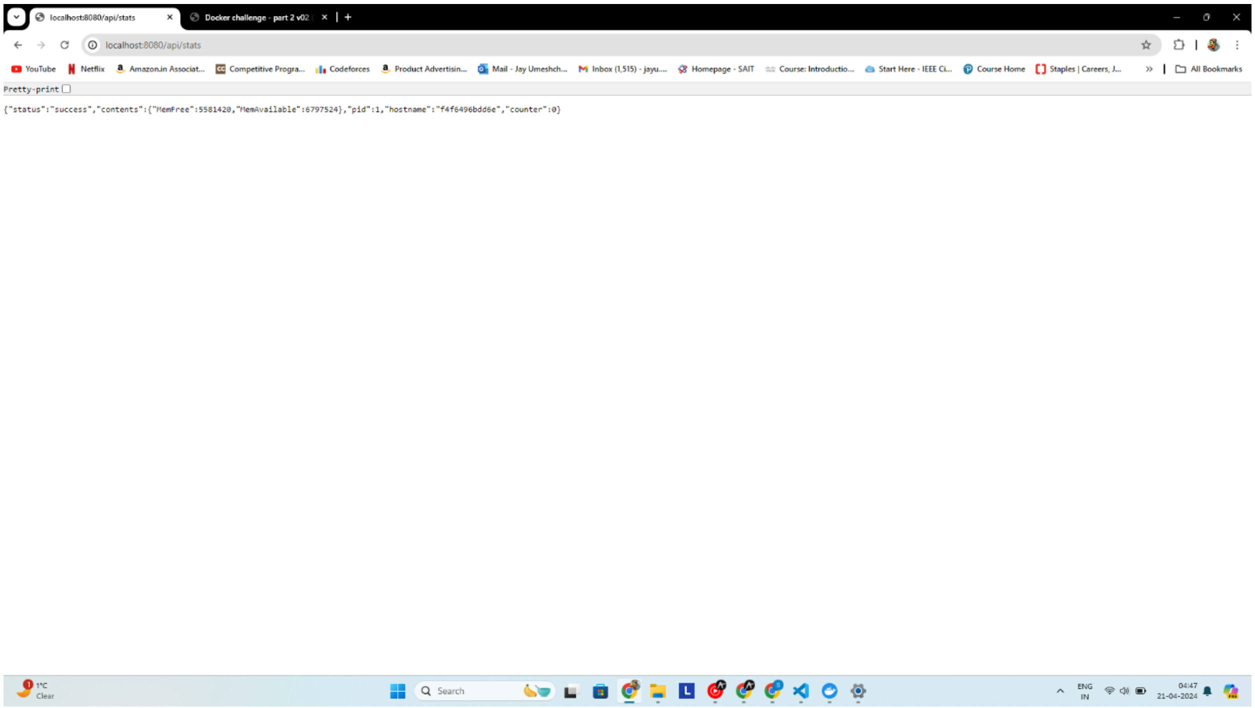
```
bout a m
inute ag
o   Up 5
8 second
s   3000
/tcp
PS C:\my
Files\SA
IT\Opera
ting Sys
tem\dock
erFinal\
docker-c
hallenge
-templat
e\challe
nge4>
```



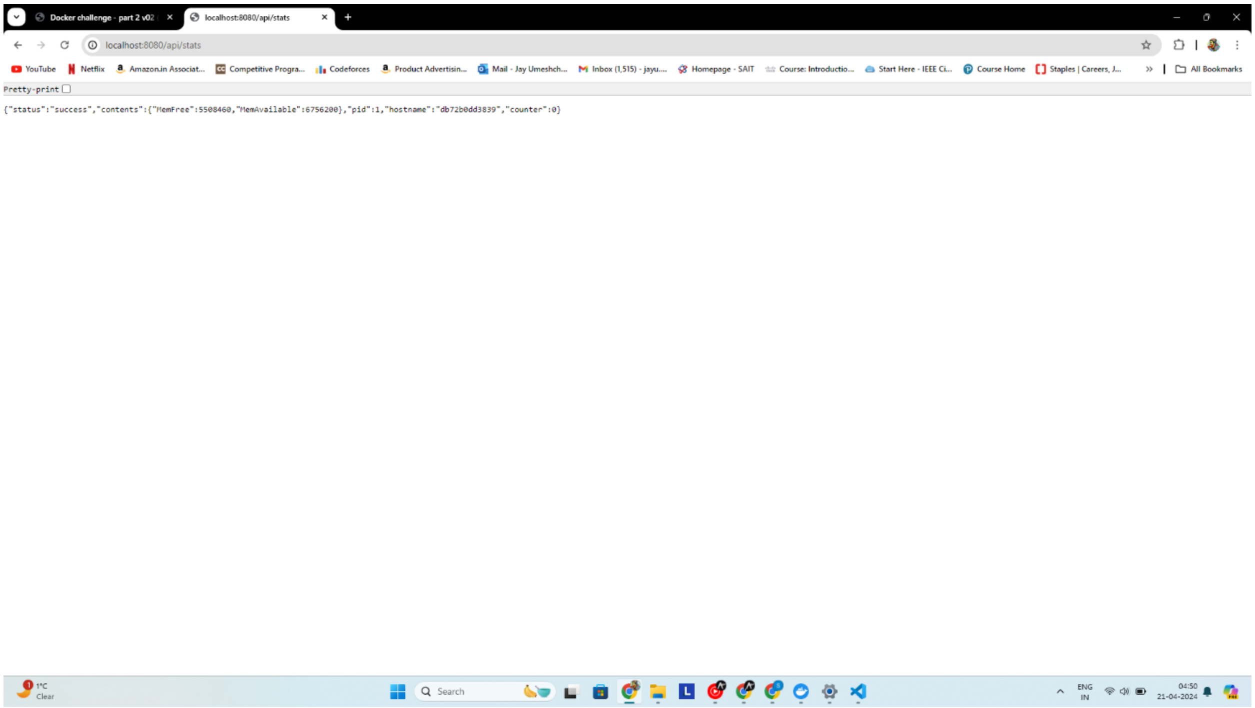
```
{"status":"success","contents":{"MemFree":5613580,"MemAvailable":6829676},"pid":1,"hostname":"f4f6496bdd6e","counter":0}
```



```
{"status":"success","contents":{"MemFree":5578976,"MemAvailable":6795072},"pid":1,"hostname":"e8e49f453f0c","counter":0}
```



```
{"status":"success","contents":{"MemFree":5581420,"MemAvailable":6797524},"pid":1,"hostname":"f4f6496bdd6e","counter":0}
```



```
{"status":"success","contents":{"MemFree":5508460,"MemAvailable":6756200},"pid":1,"hostname":"db72b0dd3839","counter":0}
```