# Three-body Problem
# By Runge-Kutta Solver Infrastructure
(Earth, Sun and Jupiter)
(Project 1)

Aleksandar Mitic (2035177)
Dominik Wirsig (2020067)
Ravishankar Selvaraj (2036915)

8th February 2022

Virtualization 2 | Project Phase

BERGISCHE UNIVERSITÄT WUPPERTAL

# Table of contents



Virtualization 2 | Project Phase

BERGISCHE UNIVERSITÄT WUPPERTAL

# 1. Introduction

- In this project, we develop a simulation of Three Body problem which represents the movement of Earth, Sun and Jupiter. The solution is being done numerically via Runge-Kutta method.
- Results are then represented graphically via Frontend website, which are given by plots and animation.
- Project contains docker compose file, Dockerfile, traefik.yml file, HTML file and requirements text file.
- TLS security is included but without real certificates. Containers are running on every machine. All of our work is spread and done in Backend and Frontend.
- Our codes can be found on GitHub website.

# 2. Backend

## 2.1. Dockerfile

- Docker can build images automatically by reading the instructions from a Dockerfile.
- A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.
- We are using 5 main commands in order to use the image, defining working directory, copying code, install libraries and run the script solver.

```
1  FROM python:latest
2
3  WORKDIR /usr/src/app/
4
5  COPY RKmethod.py ./
6  COPY requirements.txt ./
7
8  RUN pip install -r requirements.txt
9  RUN apt update &&apt -y install ffmpeg
10
11 CMD ["python" ,"RKmethod.py"]
```

Virtualization 2 | Project Phase

BERGISCHE UNIVERSITÄT WUPPERTAL

# 2. Backend

## 2.2. Main script

- Using Runge-Kutta method of 4th order, we are solving the 3-Body problem in order to simulate the movement of Sun, Earth and Jupiter.
- All the inputs are preset, which are taken from the NASA website.
- Outputs are plots and animation.

```python
# Differential equation solver
# ================================================================

def RK4Solver(t,r,v,h,planet,ro,vo):
    k11 = dr_dt(t,r,v,planet,ro,vo)
    k21 = dv_dt(t,r,v,planet,ro,vo)

    k12 = dr_dt(t + 0.5*h,r + 0.5*h*k11,v + 0.5*h*k21,planet,ro,vo)
    k22 = dv_dt(t + 0.5*h,r + 0.5*h*k11,v + 0.5*h*k21,planet,ro,vo)

    k13 = dr_dt(t + 0.5*h,r + 0.5*h*k12,v + 0.5*h*k22,planet,ro,vo)
    k23 = dv_dt(t + 0.5*h,r + 0.5*h*k12,v + 0.5*h*k22,planet,ro,vo)

    k14 = dr_dt(t + h,r + h*k13,v + h*k23,planet,ro,vo)
    k24 = dv_dt(t + h,r + h*k13,v + h*k23,planet,ro,vo)

    y0 = r + h * (k11 + 2.*k12 + 2.*k13 + k14) / 6.
    y1 = v + h * (k21 + 2.*k22 + 2.*k23 + k24) / 6.

    z = np.zeros([2,2])
    z = [y0, y1]
    return z
```

```python
Me = 6e24                   # Mass of Earth in kg
Ms = 2e30                   # Mass of Sun in kg
Mj = 1.9e27                 # Mass of Jupiter

G = 6.673e-11               # Gravitational Constant

RR = 1.496e11               # Normalizing distance in km (= 1 AU)
MM = 6e24                   # Normalizing mass
TT = 365*24*60*60.0         # Normalizing time (1 year)
```

Virtualization 2 | Project Phase

BERGISCHE UNIVERSITÄT WUPPERTAL

# 2. Backend

## 2.3. Requirements

- To keep the main Dockerfile simple, a separate file for the libraries is created (requirements.txt).
- In order to be able to run our container in any environment, libraries are mentioned as a prerequisite.
- If any problem occurs while running docker Container, first check on the libraries.

```
numpy
pylab-sdk
requests
ffmpeg-python
ipython
matplotlib
moviepy
```

# 3. Frontend

## 3.1 Web Server

**Why we need a Web server?** A web server is computer software and underlying hardware that accepts requests via HTTP or its secure variant HTTPS.

A user agent, commonly a web browser, initiates communication by making a request for a web page or other resource using HTTP, and the server responds with the content of that resource or an error message.

Web servers are used to present the results graphically on a website.

e.g. Apache and nginx, etc.,

**Why we are using nginx image as webserver?**

- Lightweight (low memory usage)
- High performance

```
webserver:
  image: nginx
  restart: always
  volumes:
    - ./Container:/usr/share/nginx/html
  depends_on:
    - rk_method
    - traefik
  labels:
    - "traefik.enable=true"
    - "traefik.http.routers.nginxrouter.entrypoints=web,websecure"
    - "traefik.http.routers.nginxrouter.rule=Host(`localhost`) || Host(`compass27.physik.uni-wuppertal.de`)"
    - "traefik.http.services.nginxservice.loadbalancer.server.port=80"
    - "traefik.http.routers.nginxrouter.tls=true"
```

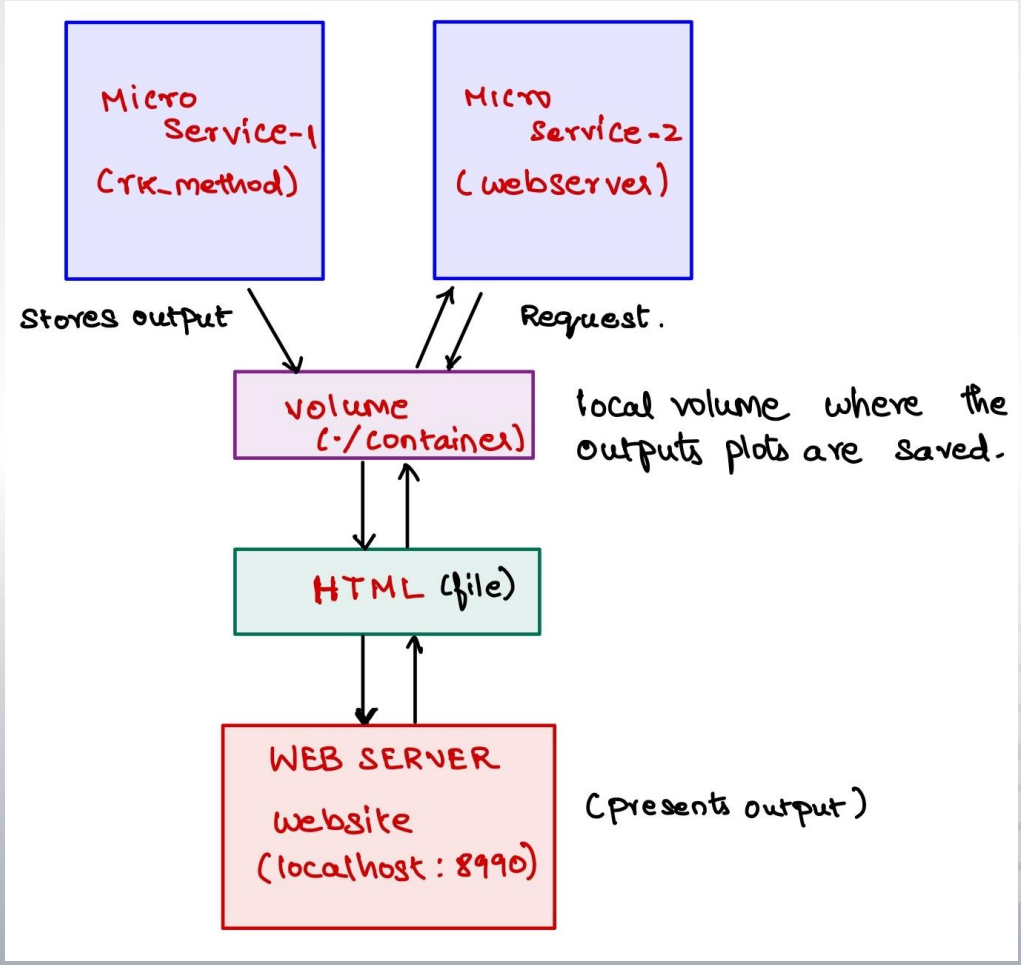
Virtualization 2 | Project Phase

BERGISCHE UNIVERSITÄT WUPPERTAL

# 3. Frontend

## 3.2 HTML

HTML (Hypertext Markup Language) is the code that is used to structure a web page and its content.

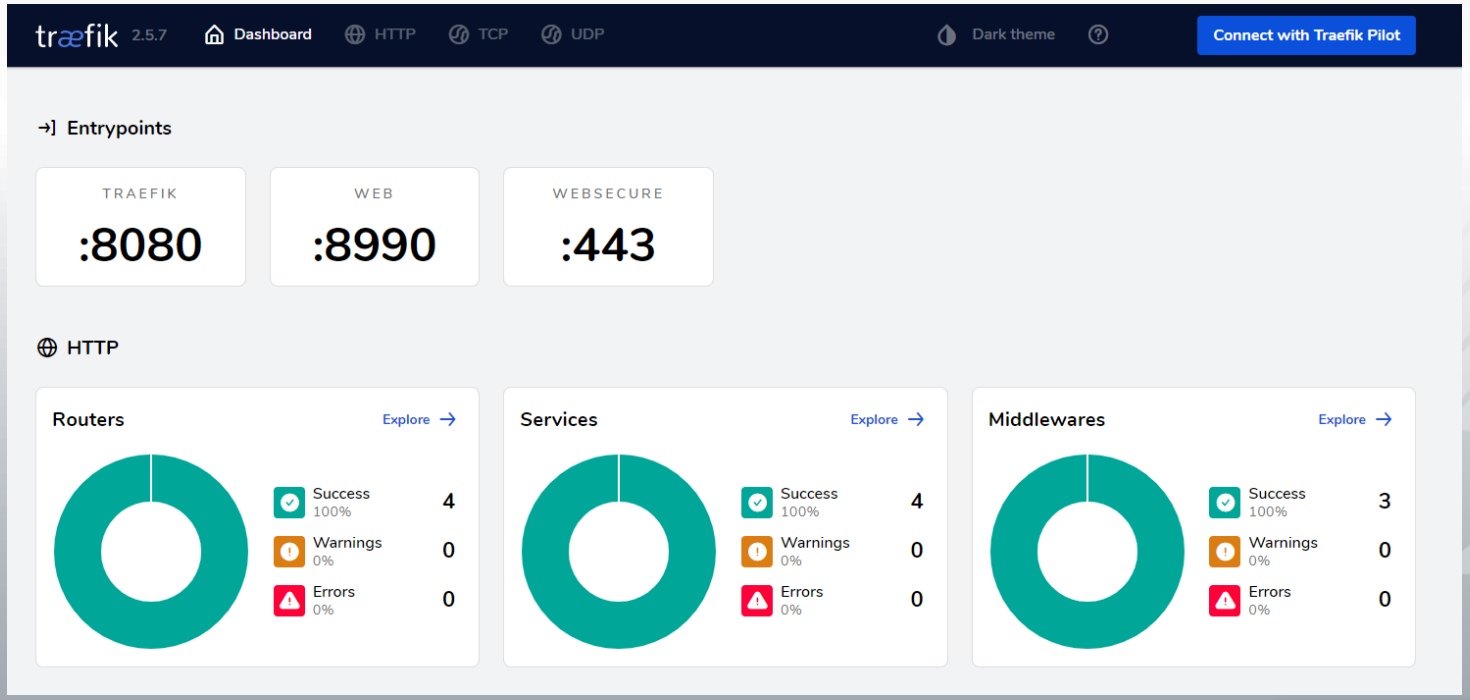


# 4. Traefik

**What traefik is used for?**

Traefik is used for:

- Routing
- Load Balancing
- Proxy and Reverse proxy, etc.




Virtualization 2 | Project Phase

BERGISCHE UNIVERSITÄT WUPPERTAL

# 4. Traefik

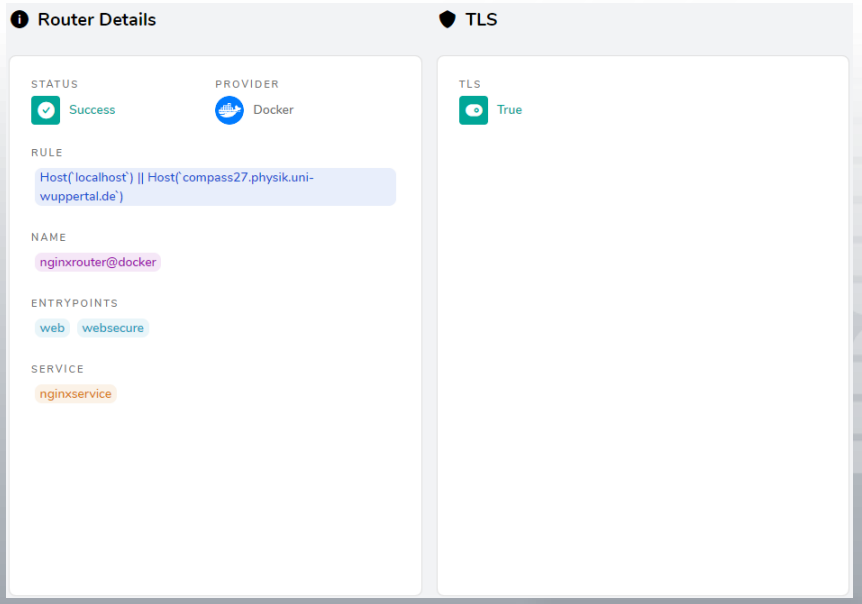
## 4.1 Entrypoints

```yaml
traefik:
  image: "traefik:v2.5"
  container_name: "traefik"
  ports:
    - 8990:8990
    - 443:443
    - 8080:8080
  volumes:
    - /etc/localtime:/etc/localtime:ro
    - /var/run/docker.sock:/var/run/docker.sock:ro
    - /home/jarvis/Virtualization/Three-Body-problem/traefik/data/traefik.yml:/traefik.yml:ro
```

- We used Entrypoint 443 for websecure
- Entrypoint 8080 for dashboard
- Entrypoint 8990 (http) for Frontend webserver, which will redirect to secured website (https-Entrypoint:443)

```yaml
entryPoints:
  web:
    address: :8990
    # (Optional) Redirect to HTTPS
    # ---
    http:
      redirections:
        entryPoint:
          to: websecure
          scheme: https

  websecure:
    address: :443
```




Virtualization 2 | Project Phase

BERGISCHE UNIVERSITÄT WUPPERTAL

# 5. Docker-Compose

## 5.1 Docker-compose

**Docker-Compose** is a tool that was developed to help define and share multi-container applications. With Compose, we can create a YAML file to define the services and with a single command, can spin everything up or tear it all down.

Command: docker-compose up –d
docker-compose down

For traefik: docker logs traefik

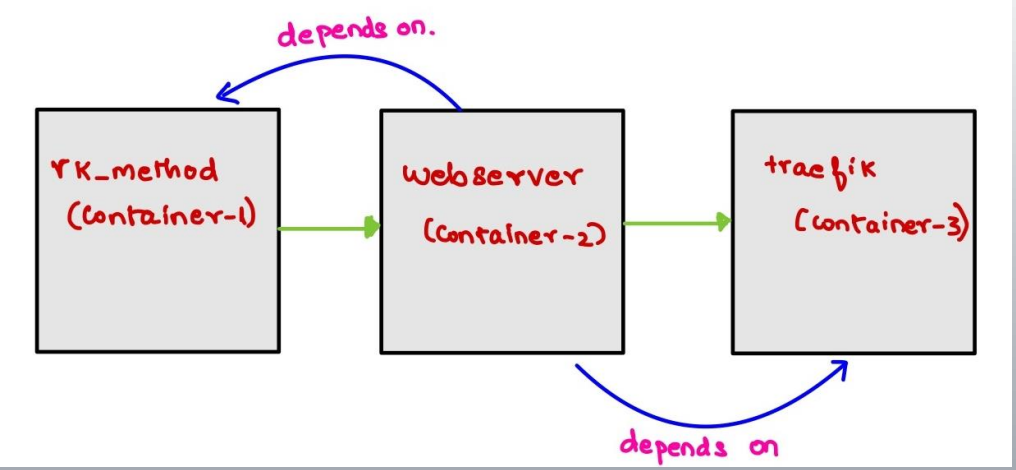
We have 3 microservices running inside docker-compose

- rk_method
- webserver (which connects to frontend website)
- traefik (which used for routing)

# 5. Docker-Compose

**What happens when you run our docker-compose?**

- Initially, the first container (rk_method) is executed, which contains a Docker file and a solver. And it stores the outputs in a local volume
- Then the second container (web server) is started, which requests the output to the first container and projects the results to the website via Traefik routing
- Finally, it runs the Third container (traefik) on which webserver (container -2 ) depends.

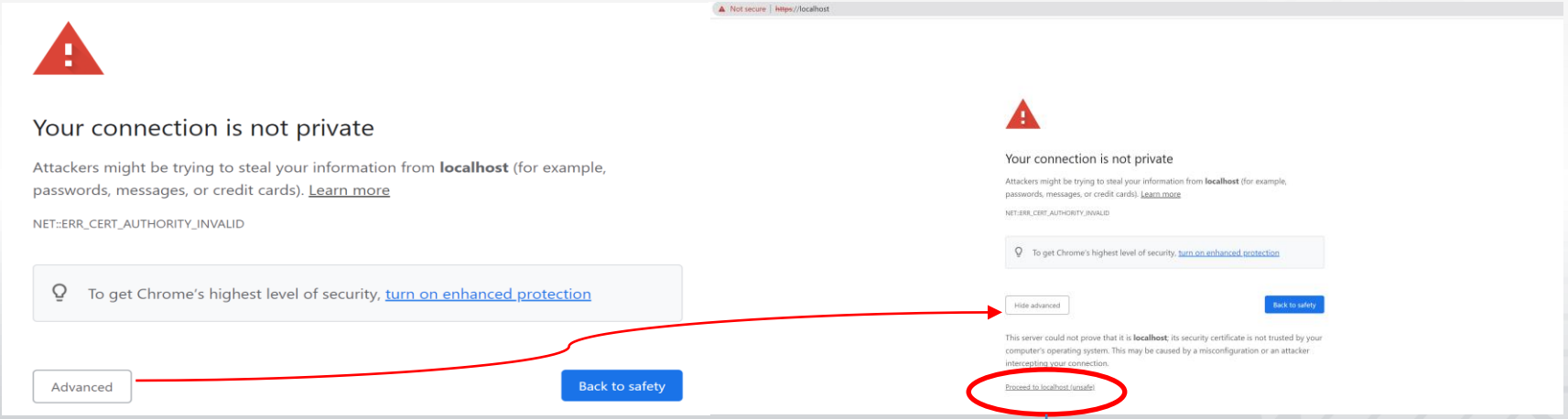



Virtualization 2 | Project Phase

BERGISCHE UNIVERSITÄT WUPPERTAL

# 6. Results

Entering localhost:8990 in browser it redirects to secured frontend website with Entrypoint localhost:443

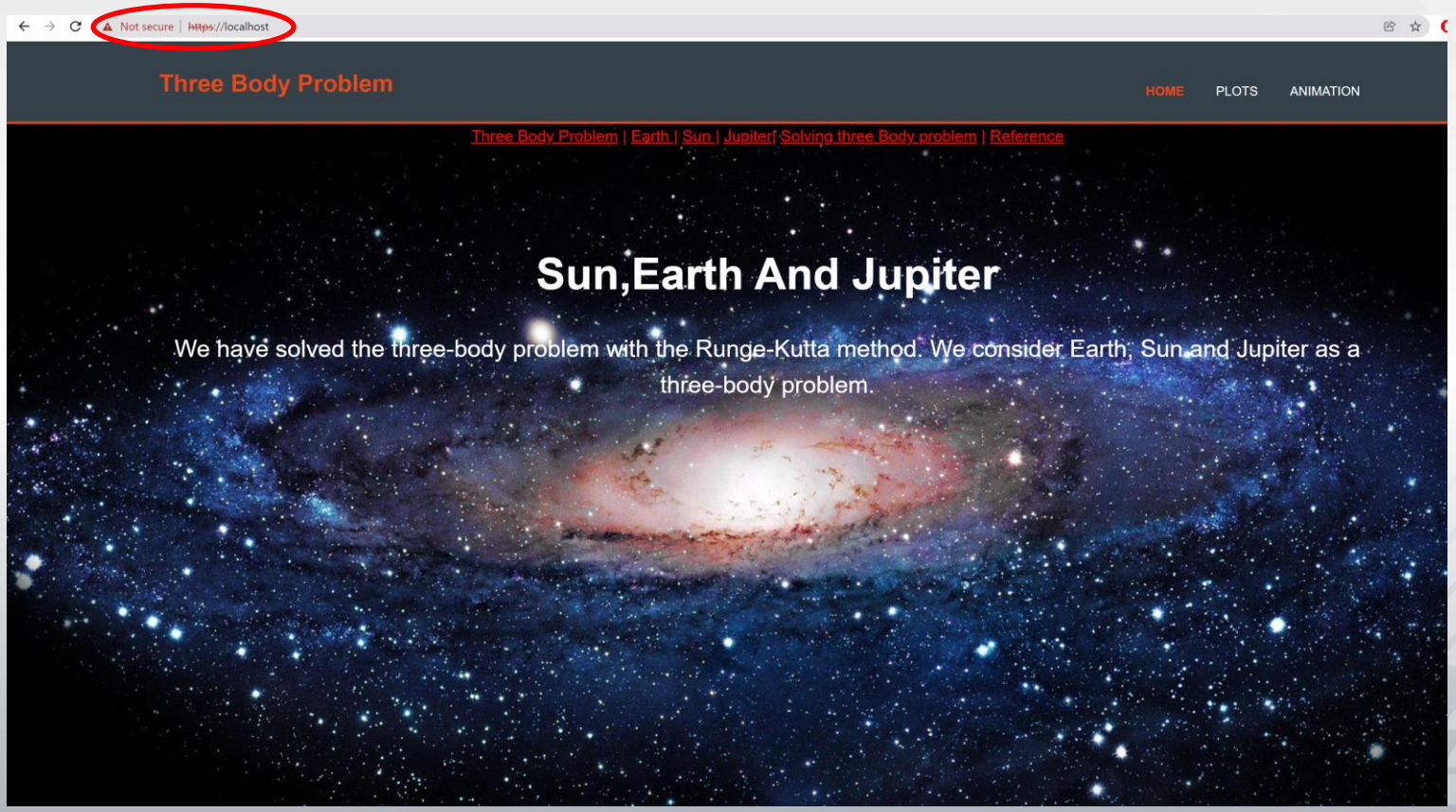



Proceed to localhost(unsafe)

Virtualization 2 | Project Phase

BERGISCHE UNIVERSITÄT WUPPERTAL

# 6. Results




Virtualization 2 | Project Phase

BERGISCHE UNIVERSITÄT WUPPERTAL

# Problems we face during project

- **Saving animation -** Library that allows us to save the animation in local volume
- **Allocating Volume** - Same volume for two Containers (to address the graphical results)
- **HTML -** User interface, workflow and design
- **Web server image -** nginx
- **Database -** storing the results
- **Traefik routing -** Entrypoints
- **Labels -** rule=host

# References


1. Three-body-problem by Runge-kutta method

   https://www.phas.ubc.ca/~berciu/TEACHING/PHYS349/karla.pdf

2. Earth fact sheet: https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html

3. Jupiter fact sheet: https://nssdc.gsfc.nasa.gov/planetary/factsheet/jupiterfact.html

4. Sun fact sheet: https://nssdc.gsfc.nasa.gov/planetary/factsheet/sunfact.html

5. RK method: https://github.com/zaman13/Three-Body-Problem-Gravitational-System/tree/master/Python%20script

# THANK YOU

Virtualization 2 | Project Phase

BERGISCHE UNIVERSITÄT WUPPERTAL