

Cloud Computing - Mini Project Report

Breaking Down Monoliths

April 2023

Submitted By:

Avanish Bhat - PES1UG20CS092

Manas Chebrolu - PES1UG20CS111

Atharv Tiwari - PES1UG20CS087

Aryan Karn - PES1UG20CS080

VI Semester Section B

PES University

Short Description and Scope of the Project

- The first task of the project involves adding the required modules to the requirements file, and then writing a Dockerfile in order to deploy a Docker container which runs the Flask application.
- The next task is to debug the following issues in a web based calculator application
 - Python treating the inputs as strings, hence the arithmetic operations not working as expected
 - Since by default there are no values in each field, a None type exception is raised
- The next task is “breaking the monolith”; that is splitting each of the calculator’s functions into separate flask applications, so that each function may work as required when the main landing service goes down.
- Then we added 7 more services, namely:
 - LCM
 - GCD
 - Modulus
 - Less Than
 - Greater Than
 - Exponent
 - Equals
- The project is written in a very modular fashion, hence can be extended to add more functionality with minimal effort.

Methodology

- We split each microservice into a separate flask application, that ran on different port numbers
- In order to achieve modularity, mapping of ports and functions is done by simply adding an entry to a python dictionary

```
operation_mapping = {}  
    "add": {  
        "operation": "addition",  
        "port": 5051,  
        "method": "add",  
    },  
    "subtract": {  
        "operation": "subtraction",  
        "port": 5052,  
        "method": "sub"  
    },  
    "multiply": {  
        "operation": "multiplication",  
        "port": 5053,  
        "method": "mul"  
    },  
    "divide": {  
        "operation": "division",  
        "port": 5054,  
        "method": "div"  
    },  
    "gcd": {  
        "operation": "gcd",  
        "port": 5055,  
        "method": "gcd"
```

- The application takes the numbers and function as input from the landing page, and then sends a POST request to the appropriate URL.
- The response, which contains the result of applying the function on the inputs, is received by the landing page and displayed.
- Any errors raised, for example by inappropriate input, would be caught and displayed.

Testing

LCM operation

localhost:5050

localhost:5050

Arithmetic Microservices

Enter the First number
1200

Enter the Second number
51

Enter Operation
lcm

Submit

The result of operation lcm on 1200.0 and 51.0 is 20400.0

Multiplication operation

localhost:5050

localhost:5050

Arithmetic Microservices

Enter the First number
2

Enter the Second number
3

Enter Operation
multiply

Submit

The result of operation multiply on 2.0 and 3.0 is 6.0

Exponent operation

The screenshot shows a web browser window with the address bar displaying 'localhost:5050'. The page title is 'Arithmetic Microservices'. The interface is a green box on an orange background. It contains the following elements:

- Input field 'Enter the First number' with the value '5'.
- Input field 'Enter the Second number' with the value '8'.
- Dropdown menu 'Enter Operation' with the selected value 'exponent'.
- A red 'Submit' button.
- A dashed box containing the text: 'The result of operation exp on 5.0 and 8.0 is 390625.0'.

Default value of 0 taken when no input provided

The screenshot shows the same web browser window as above, but with different input values. The address bar still shows 'localhost:5050'. The page title is 'Arithmetic Microservices'. The interface is a green box on an orange background. It contains the following elements:

- Input field 'Enter the First number' with the value '0'.
- Input field 'Enter the Second number' with the value '0'.
- Dropdown menu 'Enter Operation' with the selected value 'addition'.
- A red 'Submit' button.
- A dashed box containing the text: 'The result of operation add on 0.0 and 0.0 is 0.0'.

Error raised when string provided as input

localhost:5050

Arithmetic Microservices

Enter the First number
1.5

Enter the Second number
1.5

Enter Operation
addition

Submit

could not convert string to float: 'sample'

Zero Division error

localhost:5050

Arithmetic Microservices

Enter the First number
5

Enter the Second number
0

Enter Operation
division

Submit

float division by zero

Results and Conclusions

- The monolithic application has been broken down into simpler microservices enabling us to extend the application to multiple services.
- The initial services (addition, subtraction, multiplication and division) have been split into their own services that will process the data it receives from the landing service and return the result.
- Some simple error handling has been done to ensure that the error message is displayed when an error occurs.
- The addition, subtraction, multiplication and division microservices work as expected.
- Additional services such as modulus, gcd, lcm etc. were added to extend the functionality of the application.
- The extra services added have also been modularized and work as expected.

Hence we conclude that breaking down a complex app into separate microservices helps increase readability, ability to debug, extendability, high flexibility etc.