Extending a Microservice Architecture

Nathan R. Hall, Benjamin Kurzyna, Ji Hyuk Shon

Ji

- About Me
  - Computer Science Student at Pitt; Graduating April 2020
- Why I chose this project
  - Docker
  - Testing
- Favorite part of the project
  - The entire process of enhancing the project.

Ben

- About Me
  - Computer Science Student at Pitt; Graduating April 2020
- Why I chose this project
  - Testing
  - Developer Experience
  - Containerization Technology
- Favorite part of the project
  - Learning about Continuous Integration

Nathan

- About Me
  - Computer Science Student at Pitt; Graduating August 2020
- Why I chose this project
  - Microservice Architecture
  - RESTful APIs
  - Containerization Technology
- Favorite part of the project
  - Learning how to host and use RESTful APIs in Docker

Outline

- Midterm Review
- Tech Stack
- Experience
- Communication
- Challenges
- Goals
- Final Implementations
- Demo
- Questions

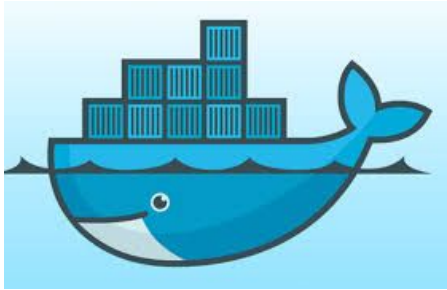
Midterm Review

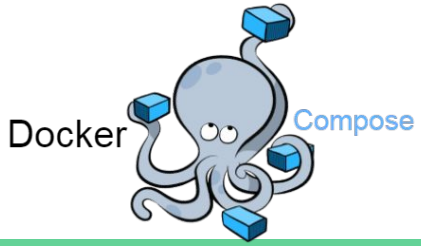
- Initial Research
  - Database
  - Testing Framework
  - Continuous Integration
- Created a Test Microservice
- Implemented a Database per Microservice Pattern

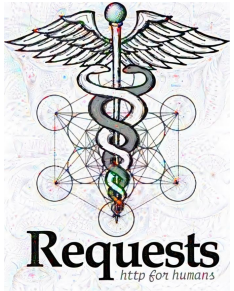
Technologies & Frameworks

GIVEN

NEW







Experience

- Remote
  - New experience for everyone involved
  - Adapting to the situation for better results
- Incorporating enhancements to the project
- Focusing on the software quality assurance aspect
  - 8 basic CRUD tests to now 20 tests

Communication

- Business as usual communication even in a remote environment
  - Slack Workspace
  - Weekly Zoom Meetings
- AGILE Development Methodology
  - GitHub Kanban Board
  - Sprint Planning and Sprint Review
  - Pull Requests and Code Review

Challenges

1. Adapting to a fully remote experience
2. Creatively making edge case and input validation tests
    a. Fixing the bugs that came up when some tests failed
3. Making the test microservice scalable for future microservices
4. Implementing the continuous integration aspect

Goals

1. Modify the Test Container for efficiency, organization, and scalability.
2. Create additional test cases to improve code quality and find defects.
3. Develop GitHub plug-ins to support continuous integration.

Test Per Microservice Pattern

- Wanted to adapt our test from one single test.py to individual tests for each microservice
- 2 Main Approaches to Solving This
  - Have test.py go to each container and run their tests at spin up time
  - Appending each test into the main test.py file

Appending Each Test

1. Each microservice will create a test directory and put the test file in that directory
2. When Build.py is ran it reads each microservice and if there is a test file it appends each individual test into the main test microservice
3. Test.py is now dynamically made and runs all tests at spin up time

Appending Each Test

- In Build.py
- Python OS Library
- write_into_test(testfile)

```python
def make_master_test_file():
    # Go into microservices
    os.chdir('microservices')

    # loop through each service in microservices
    for service in os.listdir():
        # Go into the microservice directory
        os.chdir(service)

        # loop through files in the directory
        for d in os.listdir():
            # check for test directory
            if d == 'test':
                os.chdir('test')

                # List the test directory and write the test files
                for test_file in os.listdir():
                    cwd = os.getcwd()
                    test_ = cwd + '/' + test_file
                    write_into_test(test_)
                os.chdir('..')
        os.chdir('..')

    # Leave Microservice Directory and return function.
    os.chdir('..')
```

Test Methodology

1. Try to create a test case for every method in our code.

2. Test for edge cases and invalid inputs.

3. Review and modify code after seeing automated tests finish.

Edge Case Testing

1. Making sure that functionality of REST APIs are intact.

2. Testing negative integers for specified, unique ID’s.

3. Testing empty input for content and header.

Total 10 Bugs Found

1. User Post null content gives 201
2. User Patch null headers gives 201
3. User Get on deleted id returns 200
4. User Get on negative id returns 200
5. Pet Delete on negative id returns 200
6. Pet Post null content gives 201
7. Pet Patch null header gives 201
8. Pet Get on deleted id returns 200
9. Pet Get on negative id returns 200
10. Pet Delete on negative id returns 200

Continuous Integration

- Initially interested in Github plugin for continuous integration
- New plugins looked at:
  - Buddy
  - Azure Pipelines
  - Semaphore
- Issues with plugins
  - Incompatible with Docker
  - Free options limit parallel jobs that would be required

Continuous Integration: Github Action

- Github Action
  - Create .github/workflows
  - Make actions to spin up Docker containers and run test suite
  - Run on pull request
  - Display results in github ui
- Advantages of our implementation:
  - Fully customizable
  - Run as many parallel jobs as we want
  - Easily maintainable

Demo

CS1980-Capstone-Spring-2020 / api_gateway_microservice / microservices / test

test_gen

Project

- CS1980-Capstone-Spring-2020 ~/Desktop/Spring2
  - api_gateway_microservice
    - config
    - database
    - microservices
      - api_gateway
      - pet
      - test
        - .pytest_cache
        - Dockerfile
        - requirements.txt
        - test.py
      - ui
      - user
    - api_gateway.yaml
    - build.py
    - docker-compose.yaml
    - master_yaml.py
    - pystache_engine.py
    - README.md

test.py | test_pet.py | test_user.py | docker-compose.yaml

```python
# Makes sure that we can't get deleted information for USER
def test_get_deleted_user():
    url = "http://172.16.238.8:8080/user/0"
    # Send a GET request
    response = requests.get(url)
    print(response.json())
    assert response.status_code == 404

# Tests to see if a negative integer will throw an error. In user.py user_len(id) must be > 0.
def test_negative_id_user():
    url = "http://172.16.238.8:8080/user/-1"
    response = requests.get(url)
    print(response.json())
    assert response.status_code == 404

# added edge case test (USER)
# Tests to see if a negative integer will delete or modify anything. In user.py user_len(id) must be > 0.
def test_negative_id_delete_user():
    url = "http://172.16.238.8:8080/user/-1"
    # Send a GET request
    response = requests.delete(url)
```

Terminal: Local | Local (2)

```
  Downloading https://files.pythonhosted.org/packages/f6/d5/1ecdac957e3ea12c1b319fcdee8b6917ffaff8b4644d673c4d72d2f20b49/wcwidth-0.1.9-py2.py3-none-any.whl
Collecting pluggy<1.0,>=0.12
  Downloading https://files.pythonhosted.org/packages/a0/28/85c7aa31b80d150b772fbe4a229487bc6644da9ccb7e427dd8cc60cb8a62/pluggy-0.13.1-py2.py3-none-any.whl
Collecting packaging
  Downloading https://files.pythonhosted.org/packages/62/0a/34641d2bf5c917c96db0ded85ae4da25b6cd922d6b794648d4e7e07c88e5/packaging-20.3-py2.py3-none-any.whl
Collecting urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
  Downloading https://files.pythonhosted.org/packages/e8/74/6e4f91745020f967d09332bb2b8b9b10090957334692eb88ea4afe91b77f/urllib3-1.25.8-py2.py3-none-any.whl (125kB)
Collecting idna<3,>=2.5
  Downloading https://files.pythonhosted.org/packages/89/e3/afebe61c546d18fb1709a61bee788254b40e736cff7271c7de5de2dc4128/idna-2.9-py2.py3-none-any.whl (58kB)
Collecting chardet<4,>=3.0.2
  Downloading https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8/chardet-3.0.4-py2.py3-none-any.whl (133kB)
Collecting certifi>=2017.4.17
  Downloading https://files.pythonhosted.org/packages/57/2b/26e37a4b034800c960a00c4e1b3d9ca5d7014e983e6e729e33ea2f36426c/certifi-2020.4.5.1-py2.py3-none-any.whl (157kB)
```

6: TODO | 9: Version Control | Terminal | Python Console | Event Log

Dockerfile detection: You may setup Docker deployment run configuration for the following file(s): api_gateway_microservice/microservices... (6 minutes ago) 16:13 LF UTF-8 4 spaces Git: Bug_Fixing Python 3.8 (Venv)

Suggestions for the Future

1. Incorporating integration testing
2. Extending the microservice architecture further and adding more microservices.
3. Develop out the Front end Site
4. Add better communication from CI actions by adding compatibility with Slack or other platform using github plugins

Thank You Note

- Thank you to the following developers and engineers who played a crucial role in this project:
  - Twesha Mitra
  - Anuradha Kulkarni
  - Tim Banyas
- Your support and guidance during this project has helped us tremendously.

Questions/Comments?