Department of Computer Science, Electrical and Space Engineering Luleå University of Technology

Cloud Services

Lab 4 Report: Programming Cloud Services - RESTful APIs

Author:
Hanna Ogbazghi
Elyas Khorasani
Yacine Rabehi

Supervisors:
Karan Mitra

10 December 2021

Contents

1 Objective

The objective of this lab is to:
- Understand methodologies for developing cloud application and services.
- Program a simple RESTful API.

2 Question

2.1 Question 1: What are microservices?

Microservices are a common software design paradigm that separates monolithic systems into smaller units that are self-contained services[1]. Each application is developed as an independent component. These applications run as independent services and communicate with each other via protocols such as HTTP and TCP. There are two types of microservices, namely:
- Serverful microservices
- Serverless microservices

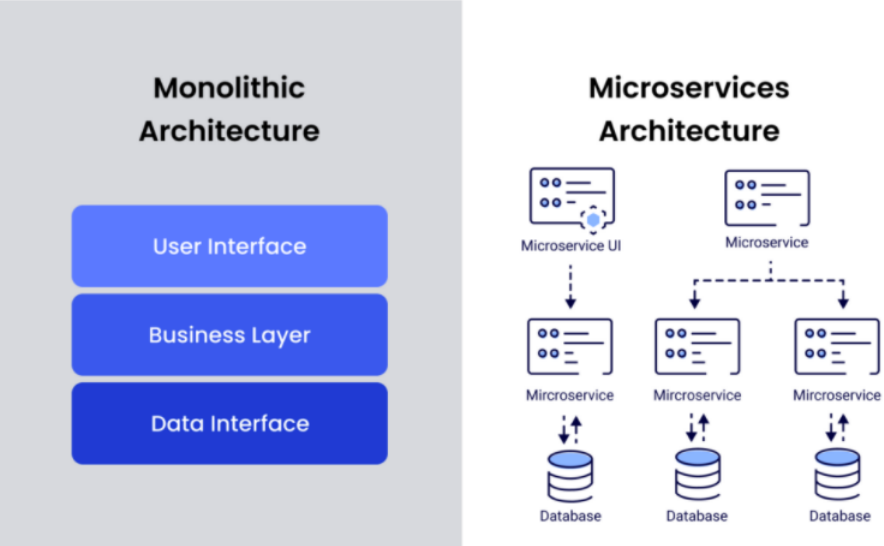

Figure 1: Microservices Architecture[2]

2.2 Question 2: Describe in detail the pros and cons of microservices architecture by giving examples

Microservices	
Pros	**Cons**
Can be developed and deployed on different platforms.	More services will require more storage
Use APIs and communication protocols to interact with each other, but they don't rely on each other otherwise	Complex communication between services
Each microservice and be developed, deployed and maintained independently.	Testing and debugging globally is difficult
Easier to debug individual microservice	Challenges in deployment
Better scalability	Is not practical for small applications
Better failure tolerance	

Figure 2: Pros and Cons Of Microservices Architecture

3 Exercise

3.1 Question 1: a: In this exercise, you will learn how to develop a simple RESTful API for the application of your choice.

We have decided to build a Student record-keeping application. The application contained Names, the class number they are enrolled in and nicknames if applicable. The application is developed in Django REST Framework.

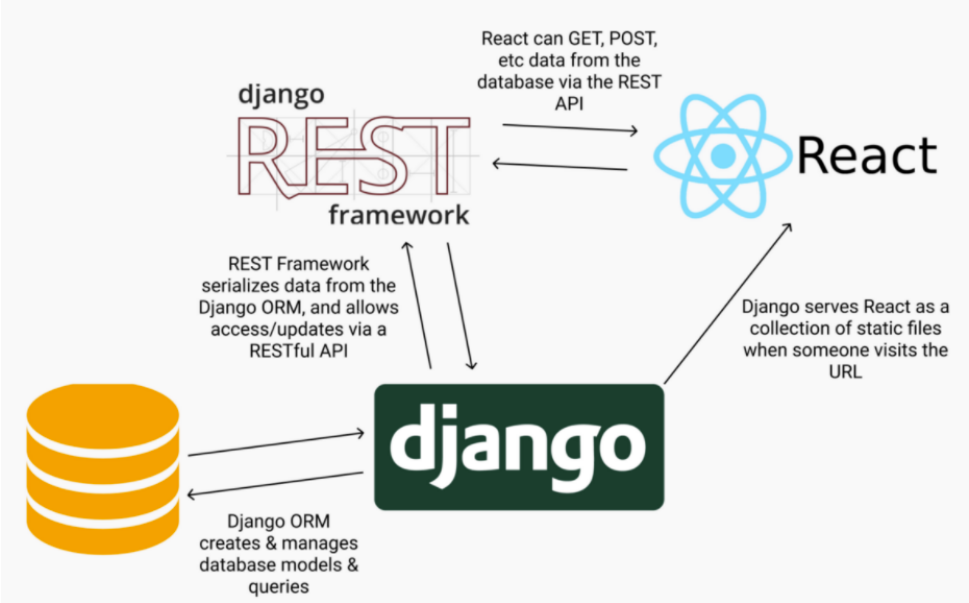

Figure 3: A typical Django application that uses React as a front end. It needs an API to allow React to consume data from the database.[4]

GET ,PUT, DELETE and REST methods have been implemented for different requests.

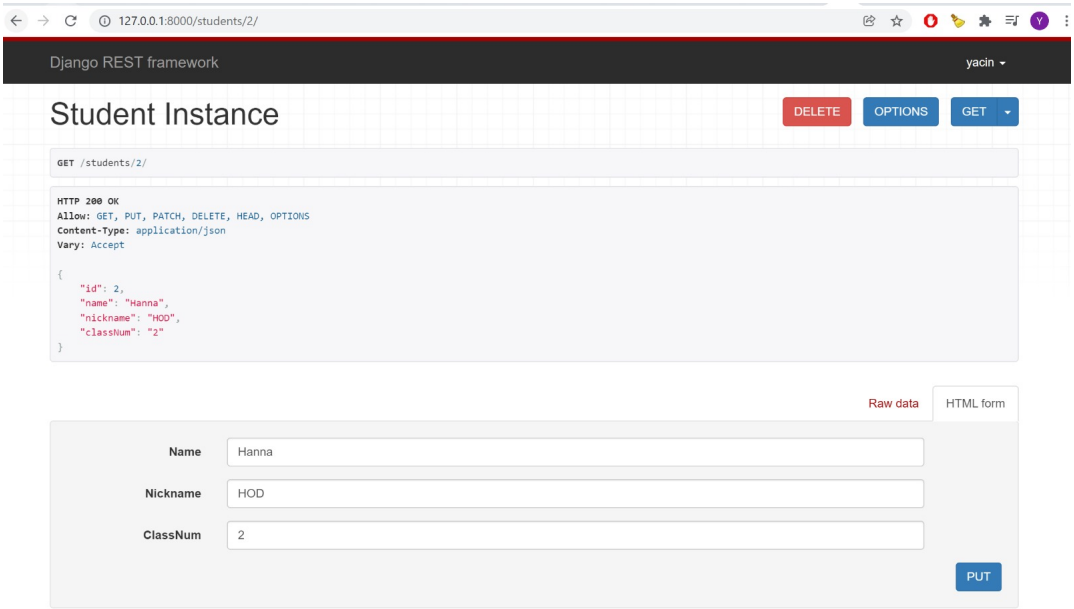

Figure 4: Student Record Tracking Application

3.1.1 Run the application locally in a docker Container

Docker container is used to deploy the service. The image is built using a docker file. The following steps were taken to install the docker container and build an image in a docker container.

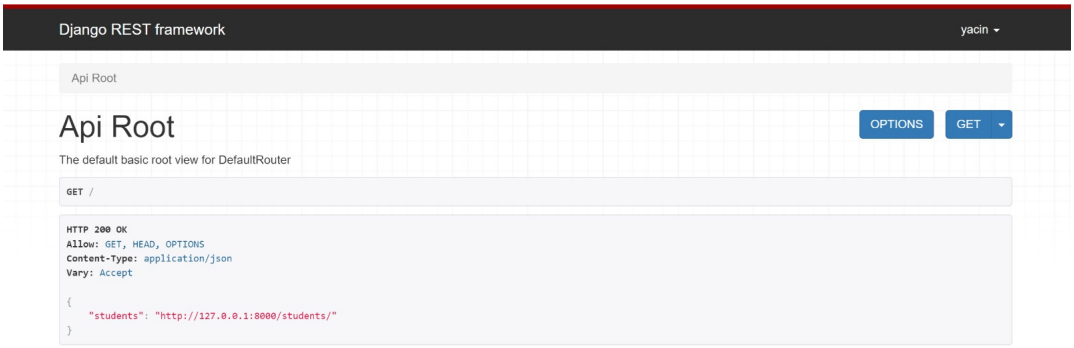

Figure 5: REST API

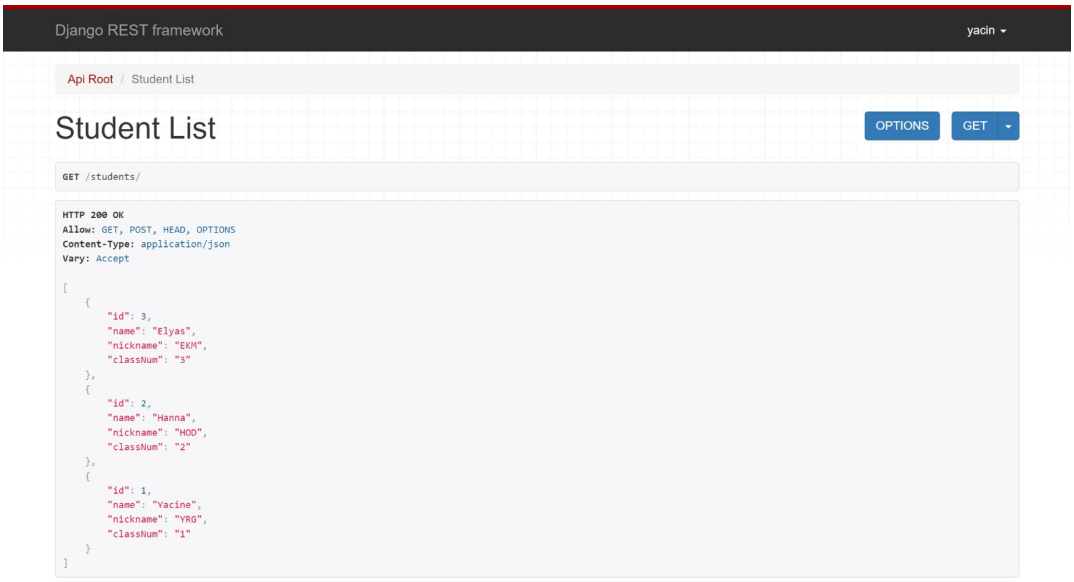

Figure 6: An example of API [GET]

1. Install docker container
2. Add files to docker
3. Declare necessary install files in requirements.txt in docker. This will install all the declared install files such as Django, djangorestframework, gunicorn
4. Create docker image.

N.B All of the above steps excluding step one, can be done using Visual code studio. Nonetheless, a docker extension must be installed for the above functionalities to work.

Step 1: Install docker container

The following link has been used to install docker in windows OS. (https://docs.docker.com/desktop/windows/install/)

Step 2:Add files to docker

After creating the application and API using Django and REST, we added the manage.py file to docker workspace, and docker file, requirement.txt, and docker ignore were created.

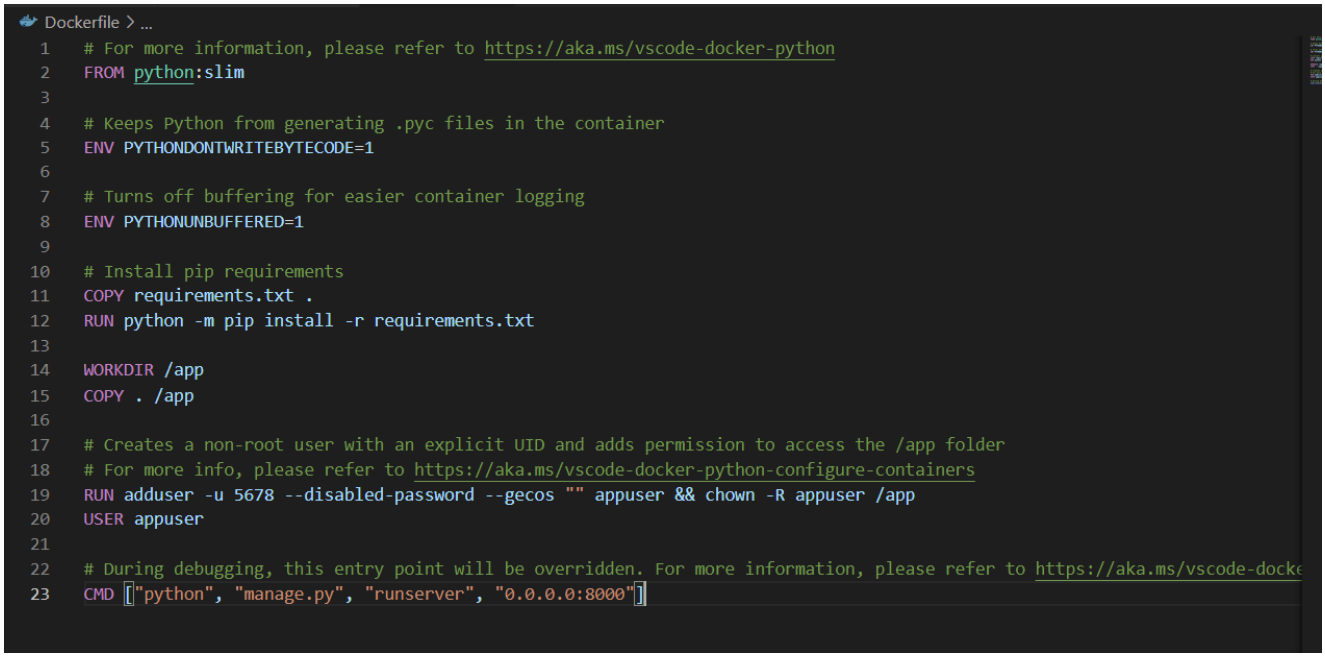

Figure 7: Docker File

Step 3: Declare necessary install files in requirements.txt

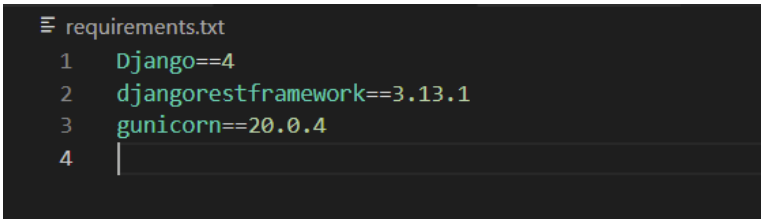

Figure 8: Requirement.txt File

Step 4: Create a docker image. There are two ways one can create an image. One can either use the following command
docker build -t students
Alternatively, right-click on the docker file in visual code studio and select the build an image option.

The created image was then pushed to a docker hub to be accessed anywhere.

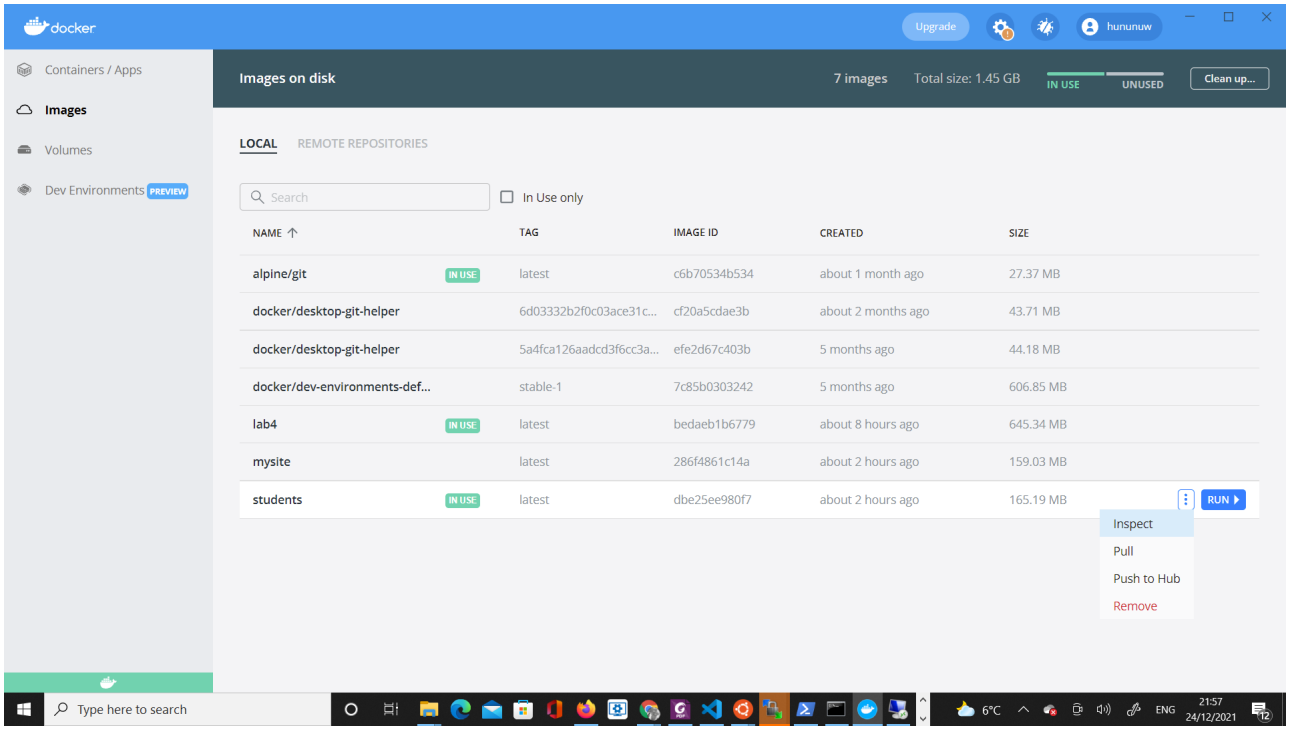

Figure 9: Push Image to Hub

3.1.2 Run the application in an EC2 Instance docker Container

To deploy the application as a service, we decided to launch an EC2 instance to deploy our image in AWS and run it as a service. The instance description is as follows.

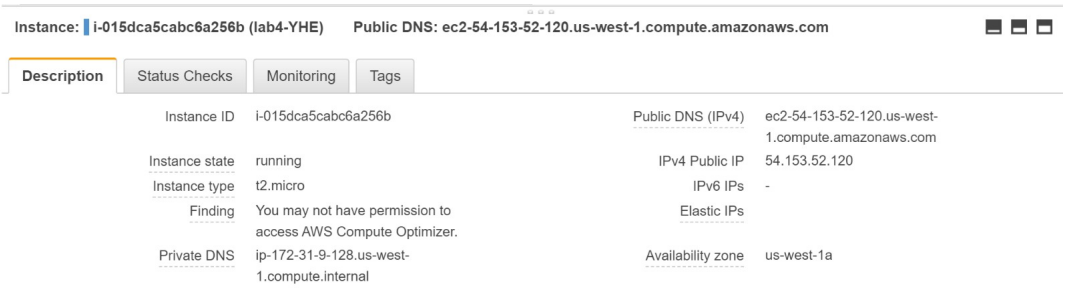

Figure 10: EC2 Instance Description

The following steps were taken to complete the deployment of the created student record tracking application.
Step 1: Install docker in the instance.
Use the following command to install docker in an Ubuntu AMI.
sudo yum update -y
sudo amazon-linux-extras install docker
sudo yum install docker
sudo service docker start
sudo usermod -a -G docker ec2-user

docker info

Step 2: Pull the image from the docker hub.

```
ubuntu@ip-172-31-9-128:~$ sudo docker images
REPOSITORY           TAG       IMAGE ID       CREATED       SIZE
yacine554/students   latest    bbed00873929   2 hours ago   165MB
```

Figure 11: Docker Image in EC2

After these steps are completed, we use the public address of the instance and port 8000. The application deployed as a microservices on the instance and can be accessed through this address http://54.153.52.120:8000/students/.

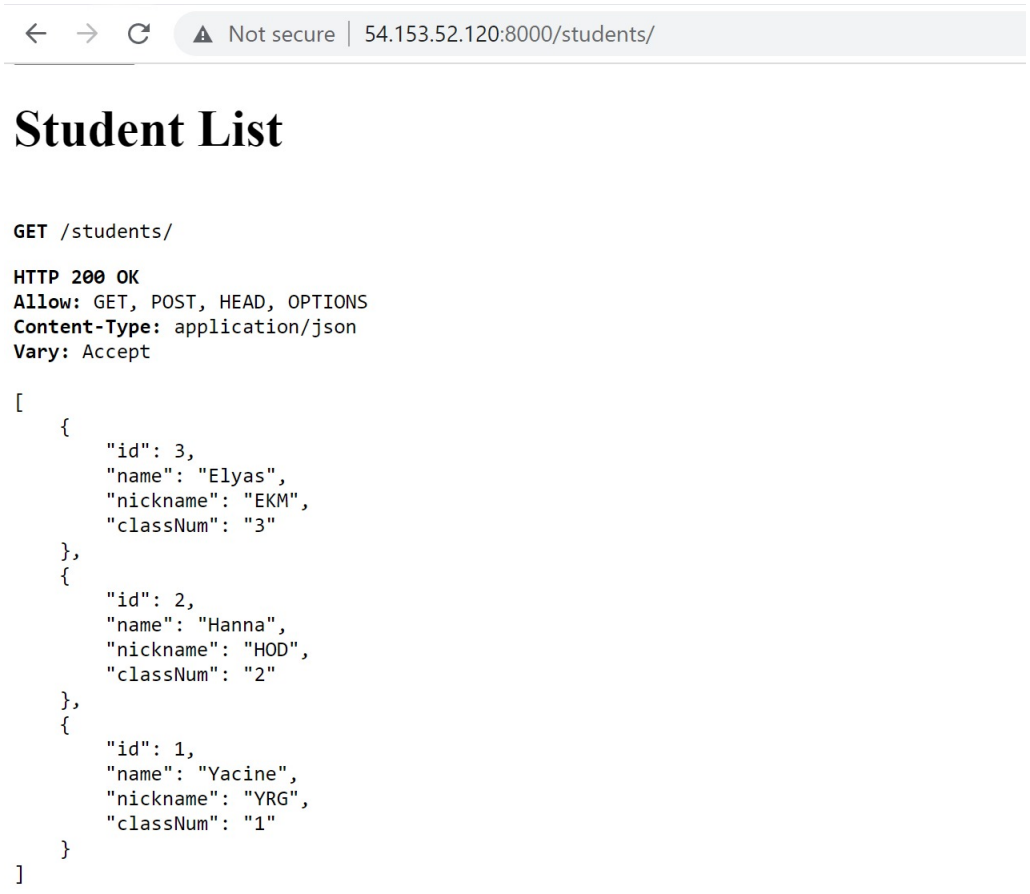

Figure 12: Student tracking app as Microservice

4 Libcloud for S3

Apache libcloud is a python library that allows the management of different cloud resources through an easy use of API. It hides the differences between different cloud provider API[3].

4.1 Question 1: Create a bucket

```python
# Create a bucket using cloud lib

def create_bucket(container_name):

    client = get_driver(Provider.S3_us_west_1)
    s3 = client(aws_id, aws_secret)
    container = s3.create_container(container_name=container_name)

    print(container.name + "has been created" )
```

Figure 13: Creating a bucket

4.2 Question 2: List the buckets in the region

```python
#List buckets that are found in us_west_3
def list_bucket():
    s3Driver = get_s3_driver(s3_provider.S3_us_west_1)
    driver = s3Driver(aws_id, aws_secret)
    sizes = driver.list_containers()
    for size in sizes:
        print(size)
```

Figure 14: Listing buckets in a region

4.3 Question 3: Upload and Download objects

```python
#Upload a pdf file in a bucket

def upload_file( file, container_name):

    client = get_driver(Provider.S3_eu_north_1)
    s3 = client(aws_id, aws_secret)
    container = s3.get_container(container_name=container_name)



    with open(file, 'rb') as iterator:
        obj = s3.upload_object_via_stream(iterator=iterator,
                                          container=container,
                                          object_name='file.pdf',
                                          )
```

Figure 15: Uploading a file to an bucket

```python
#Download a file from a bucket

def Download_file(container_name):

    client = get_driver(Provider.S3_Aeu_north_1)
    s3 = client(aws_id, aws_secret)
    container = s3.get_container(container_name=container_name)


    con_list = s3.iterate_container_objects(container=container_name)
    con_list2 = s3.iterate_containers()
    return container.list_objects()
```

Figure 16: Download a file from a bucket

4.4 Question 4: Delete an object

```python
#Delete a bucket

def delete_bucket(container_name):

    client = get_driver(Provider.S3_EU_NORTH1)
    s3 = client(aws_id, aws_secret)
    container = s3.get_container(container_name=container_name)
    delete_item = s3.delete_container(container=container)

    print("The bucket has been deleted")
```

Figure 17: Deleting a bucket

5 Bibliography

1.Raygun Blog. (n.d.).What are microservices? The pros, cons, and how they work. [online] Available at: https://raygun.com/blog/what-are-microservices/.

2.Amazon Web Services, Inc. (2019). What are Microservices? | AWS. [online] Available at: https://aws.amazon.com/microservices/.

3.https://pypi.org/project/apache-libcloud/

4.Garner, B. (2020). Build your first REST API with Django REST Framework. [online] Medium. Available at: https://medium.com/swlh/build-your-first-rest-api-with-django-rest-framework-e394e39a482c.