BACTERIA SEARCH ENGINE

Cloud computing final project

SAPIENZA
UNIVERSITÀ DI ROMA

Matteo Migliarini 1886186
Giuseppe Di Poce 2072371
Enrico Grimaldi 1884443

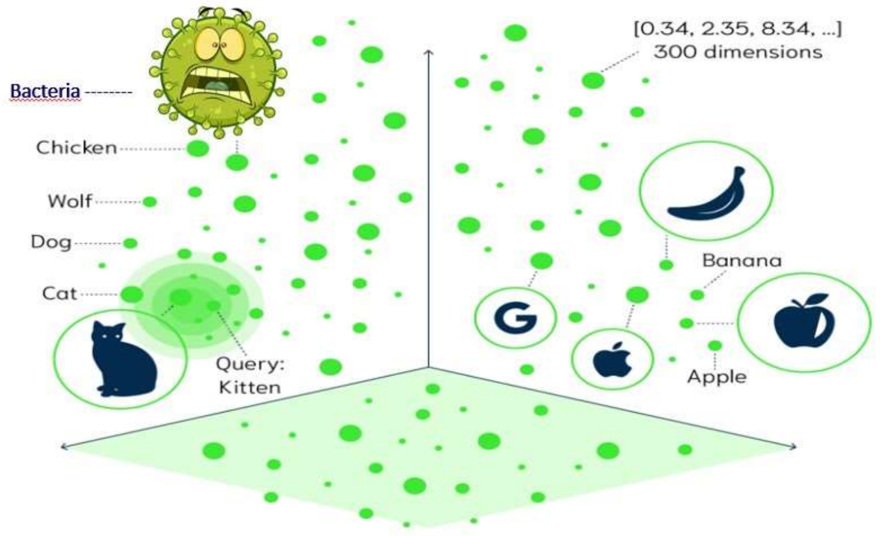
Problem Description

Weaviate vectorial database to implement an image search engine;

- The goal of the project is to develop an application that uses Weaviate to create a search engine for images of bacteria. ***Weaviate*** is known for its speed and flexibility, allowing for efficient search operations on large datasets.

- The process involves using a pre-trained neural network to extract features from the images and generate vector ***embeddings.*** These embeddings are then stored in the Weaviate database, enabling quick searches for similar images based on a given query image.

- When a user *uploads an image*, Weaviate computes the vector embedding for that image and compares it with the embeddings of reference images in the index to find the most similar matches.

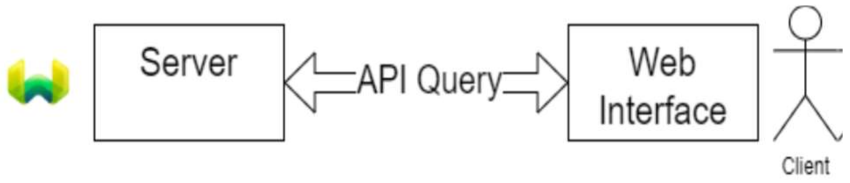


Solution implementation

we want to implement a scalable web application, capable of rapid elasticity.

The application is divided into *microservices*:

- ***Web Interface***: The client-side is a fat client, where a static HTML/CSS/JS webpage is served from an edge node. The webpage performs API calls to the server to query and retrieve results.The client performs computations to update the webpage, but it shouldn't be computationally intensive.

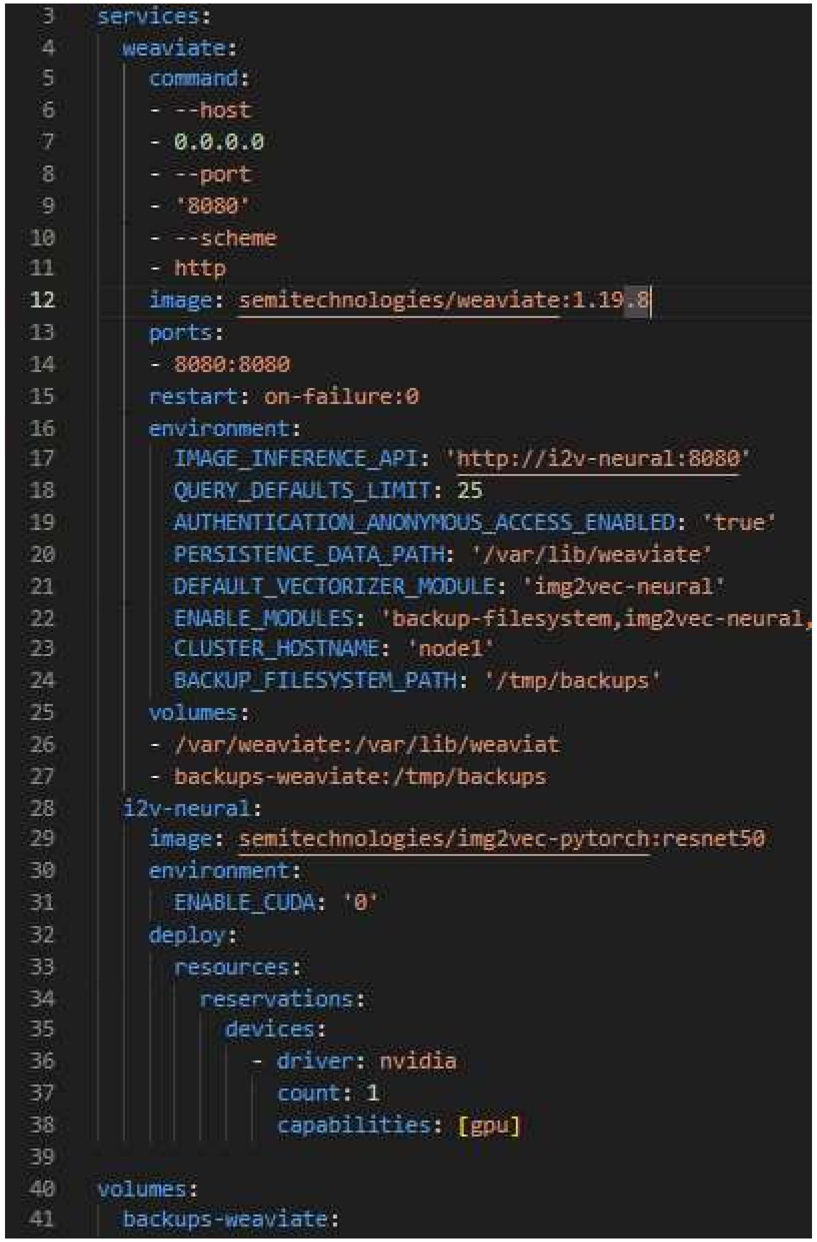


- ***API Call***: The server-side implements a REST API, particularly using POST requests to handle the query image in the request payload and return a list of images in the response.

- ***Vectorial Database***: Weaviate is chosen as the vector database. It uses vector representations of items and cosine similarity to find similar items to a query. The vector database consists of a pretrained neural network that transforms items into vectors and the database software itself, which handles queries, schema, and vector-item mappings.

Docker Containers

- To Develop our web app we need to ensure consistent and reliable deployment across different environments.
- The application utilizes ***Docker containers*** for Weaviate and the pre-trained neural network ***images***.
- Docker provides a lightweight and portable solution, encapsulating the entire application stack along with dependencies and configurations.
- In our implementation we will run on top of an AWS service two different Docker containers that will communicate each other (Weaviate and NN).
- Each container could be allocated on specific computing resources, <u>for example the NN needs GPU resources while the Weaviate container needs a fast memory.</u>
  In this way we ensure optimal usage.
- The system can scale by deploying multiple containers in parallel, achieving high availability and load balancing.



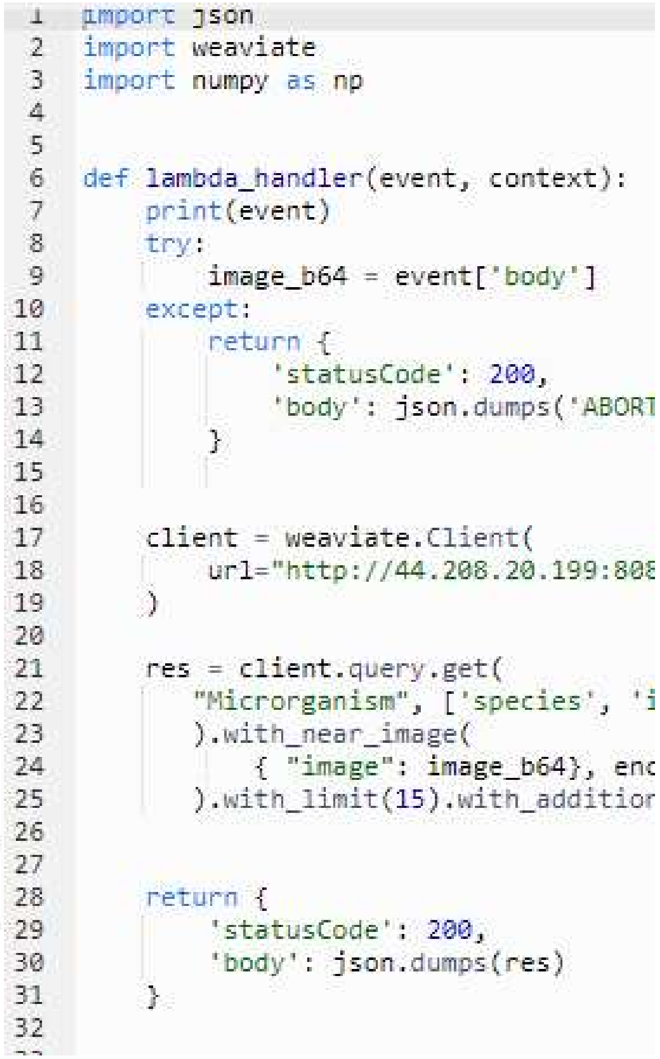
```yaml
services:
  weaviate:
    command:
    - --host
    - 0.0.0.0
    - --port
    - '8080'
    - --scheme
    - http
    image: semitechnologies/weaviate:1.19.8
    ports:
    - 8080:8080
    restart: on-failure:0
    environment:
      IMAGE_INFERENCE_API: 'http://i2v-neural:8080'
      QUERY_DEFAULTS_LIMIT: 25
      AUTHENTICATION_ANONYMOUS_ACCESS_ENABLED: 'true'
      PERSISTENCE_DATA_PATH: '/var/lib/weaviate'
      DEFAULT_VECTORIZER_MODULE: 'img2vec-neural'
      ENABLE_MODULES: 'backup-filesystem,img2vec-neural,
      CLUSTER_HOSTNAME: 'node1'
      BACKUP_FILESYSTEM_PATH: '/tmp/backups'
    volumes:
    - /var/weaviate:/var/lib/weaviat
    - backups-weaviate:/tmp/backups
  i2v-neural:
    image: semitechnologies/img2vec-pytorch:resnet50
    environment:
      ENABLE_CUDA: '0'
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: 1
              capabilities: [gpu]

volumes:
  backups-weaviate:
```

Lambda Function

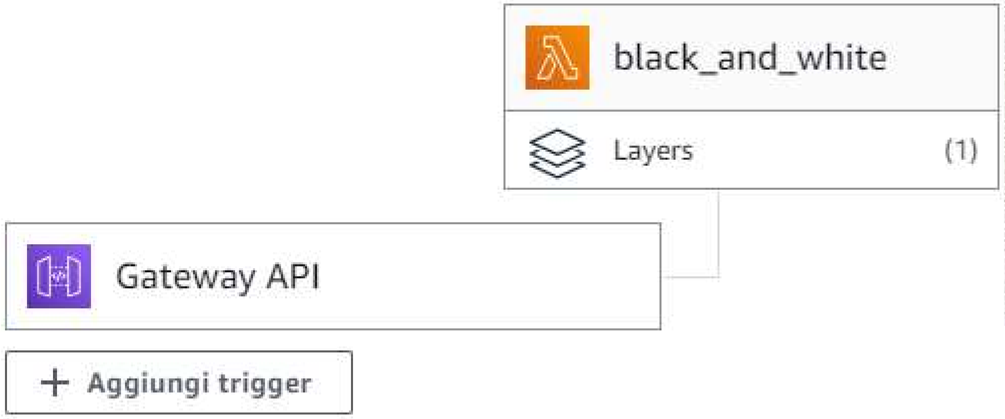
- The lambda function get triggered only when an image is uploaded to the web app by the user, *through a REST request (POST).*

- The lambda will communicate by an API call with the web app and connect with Weaviate by the IPv4 address of the Application Load Balancer service.

- This decoupling allows for greater scalability and responsiveness. The Lambda function can be invoked as needed, providing real-time responses without continuous resource allocation, since it's managed by AWS.



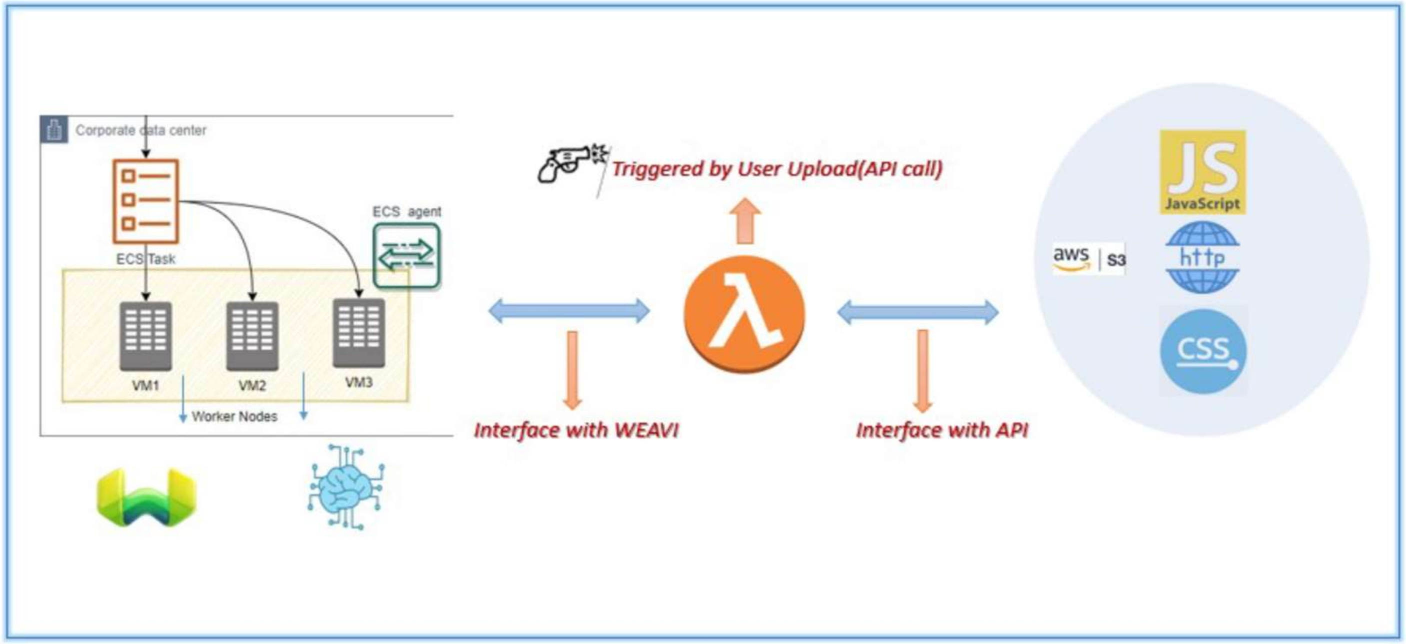
```
import json
import weaviate
import numpy as np


def lambda_handler(event, context):
    print(event)
    try:
        image_b64 = event['body']
    except:
        return {
            'statusCode': 200,
            'body': json.dumps('ABORT
        }


    client = weaviate.Client(
        url="http://44.208.20.199:808
    )

    res = client.query.get(
       "Microrganism", ['species', 'i
       ).with_near_image(
          { "image": image_b64}, enc
       ).with_limit(15).with_additior


    return {
        'statusCode': 200,
        'body': json.dumps(res)
    }

```



Why to scale?

ECS *modus operandi*

This solution leverages Amazon ECS. Docker images are pushed to Amazon ECR, and ECS task definitions and services are created. Both service runs on the same EC2 instance, and load balancing is achieved through an Application Load Balancer (ALB).

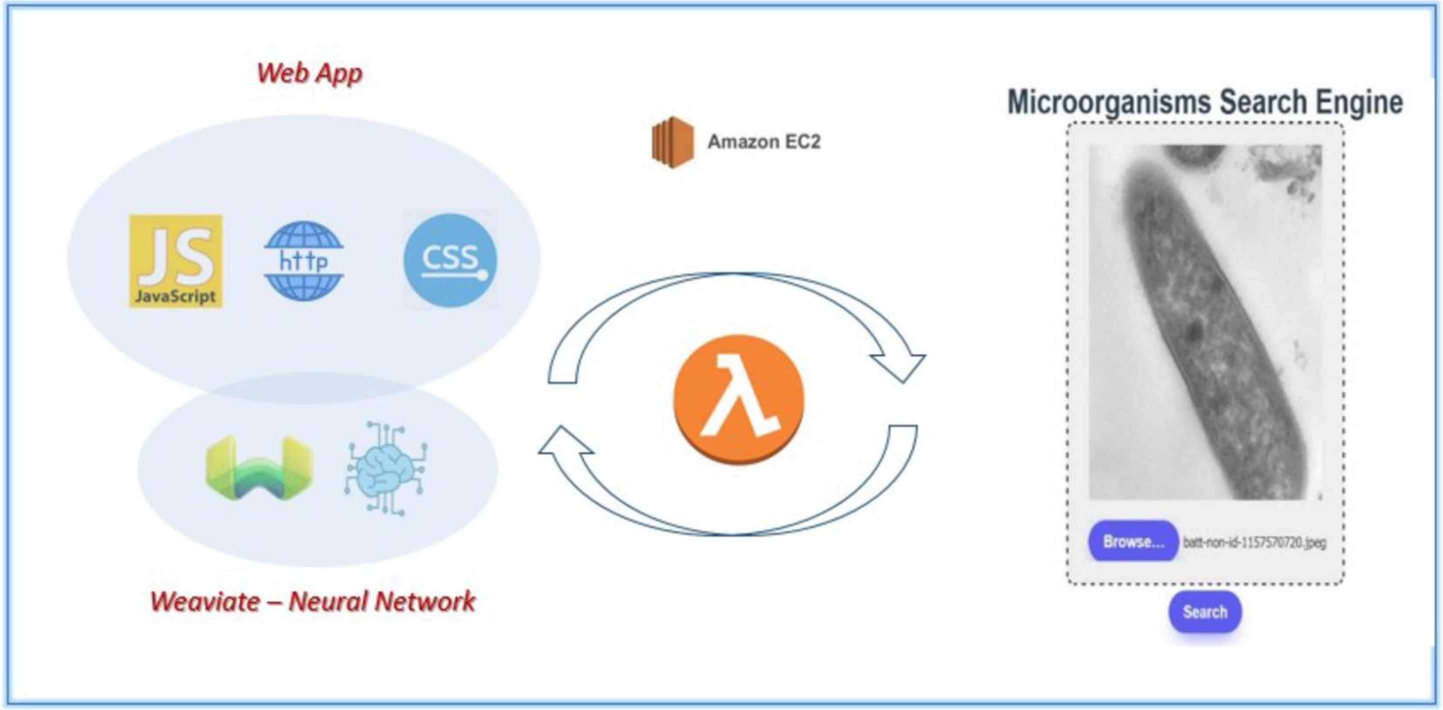


Scaling, monitoring, and adjustment of resources are performed for each ECS service based on specific requirements. This approach provides better performance optimization by allowing separate scaling policies for different components.

Why to scale?

EC2 *modus operandi*

Scaling the application enhances performance by distributing the workload across multiple EC2 instances, reducing response times, and improving query processing. It also enhances availability by using multiple instances and an elastic load balancer to redirect traffic in case of instance failure;


Web App
JS JavaScript
http
CSS
Weaviate – Neural Network
Amazon EC2
Microorganisms Search Engine
Browse...
batt-non-id-1157570720.jpeg
Search

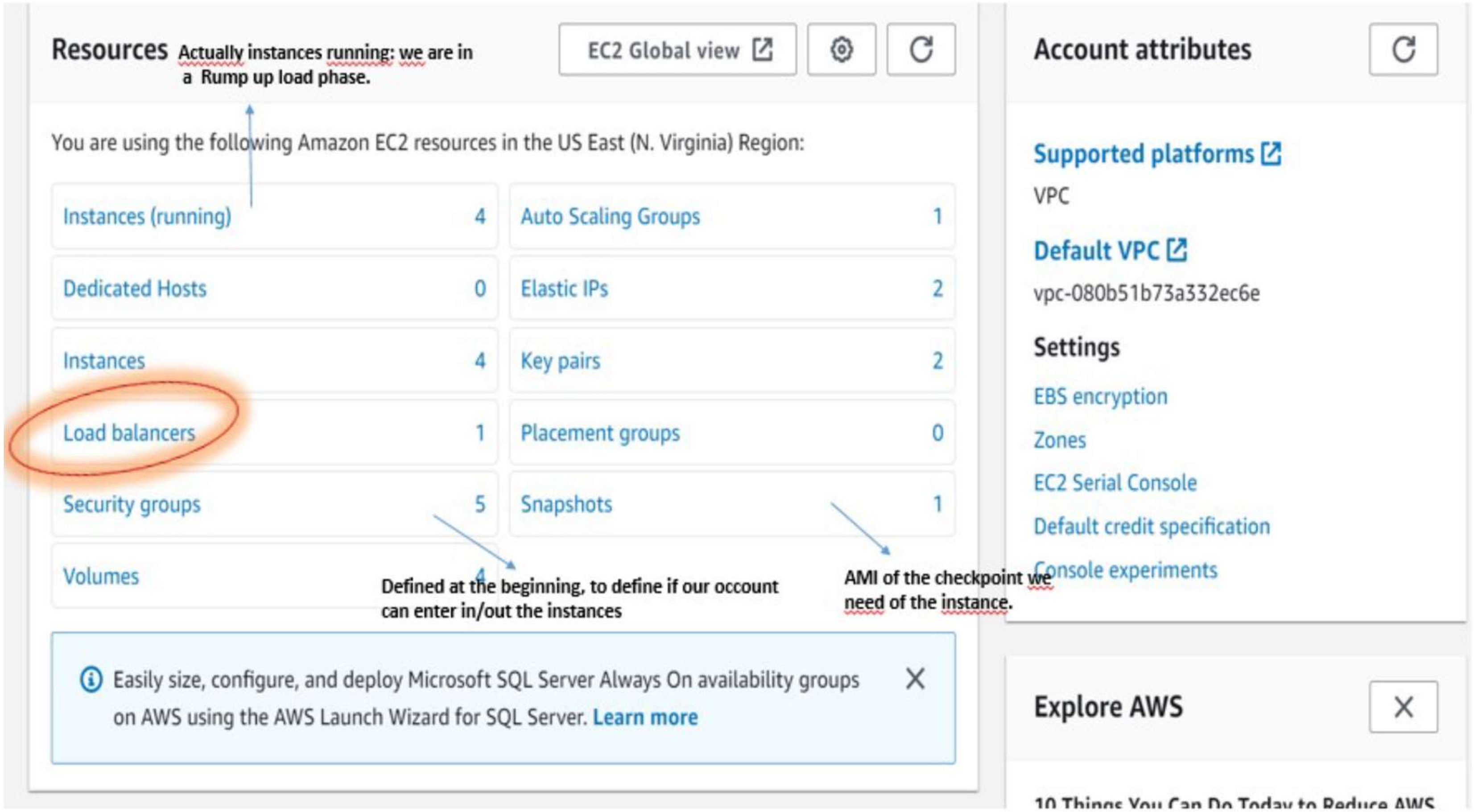
Due to *restrictions issues* on the Learner Lab account we couldn't leverage ECS properly, so we resorted to ***EC2 service***, running on multiple instances our web app tools.

Scalable Architecture

To deploy the application and create a scalable architecture, the following steps are involved:

- Create Virtual Private Cloud (VPC) (create security group and configure inbound rules). Notice that we allow only the lambda function to access the ELB (and instances of ec2);
- Create Web Group & Define EC2 Rules;
- Define Launch template of EC2 Instance, also creating a custom AMI;
- Scale and Load Balance the Architecture:
- Configure the Auto Scaling Group with a minimum and maximum number of instances, such as a *minimum of 2 and a maximum of 6 EC2 instances* for our purposes.
- The ALB makes intelligent routing decisions based on the content of the HTTP/HTTPS client requests, while also ensuring high availability and scalability.

Scalable Architecture



Resources Actually instances running: we are in a Rump up load phase.

EC2 Global view

You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Instances (running)	4	Auto Scaling Groups	1
Dedicated Hosts	0	Elastic IPs	2
Instances	4	Key pairs	2
Load balancers	1	Placement groups	0
Security groups	5	Snapshots	1
Volumes	4		

Defined at the beginning, to define if our occount can enter in/out the instances

AMI of the checkpoint we need of the instance.

Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. Learn more

Account attributes

Supported platforms
VPC

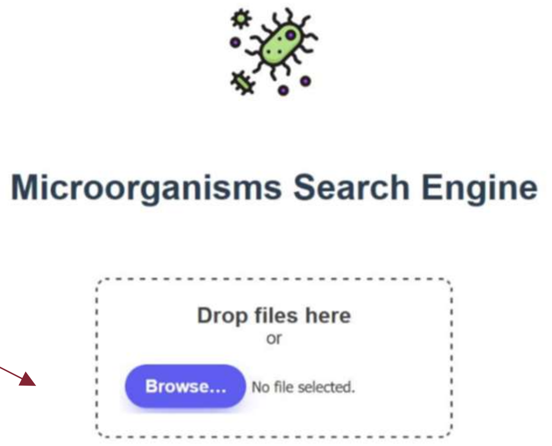
Default VPC
vpc-080b51b73a332ec6e

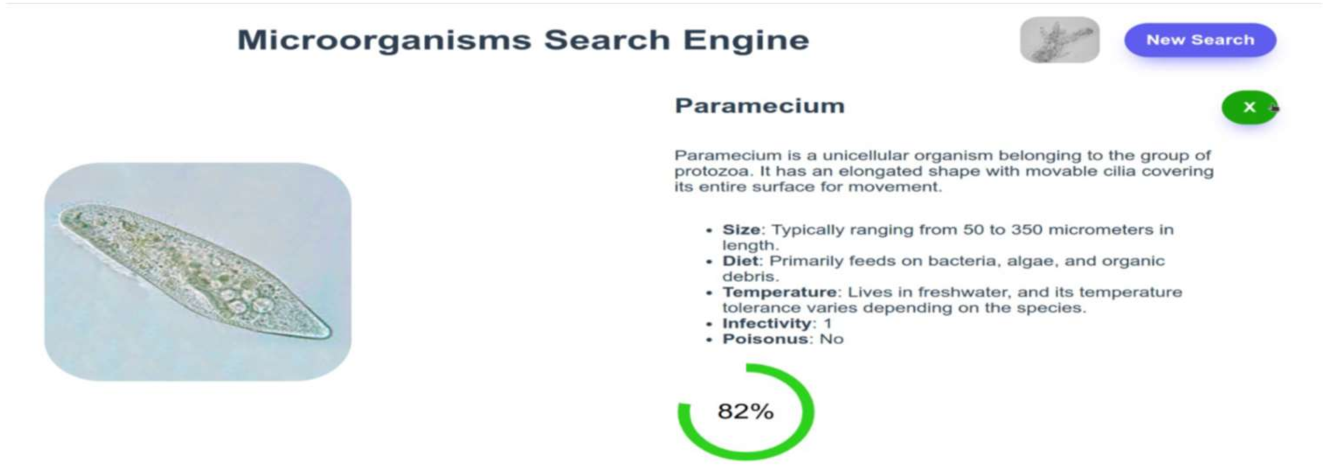
Settings
EBS encryption
Zones
EC2 Serial Console
Default credit specification
Console experiments

Explore AWS

Web App results

- The client will upload here the query image of bacteria this event will act as a trigger for our lambda function.
- Remember that the lambda interface with the API with the Web app and with the EC2 instance to communicate With Weaviate.





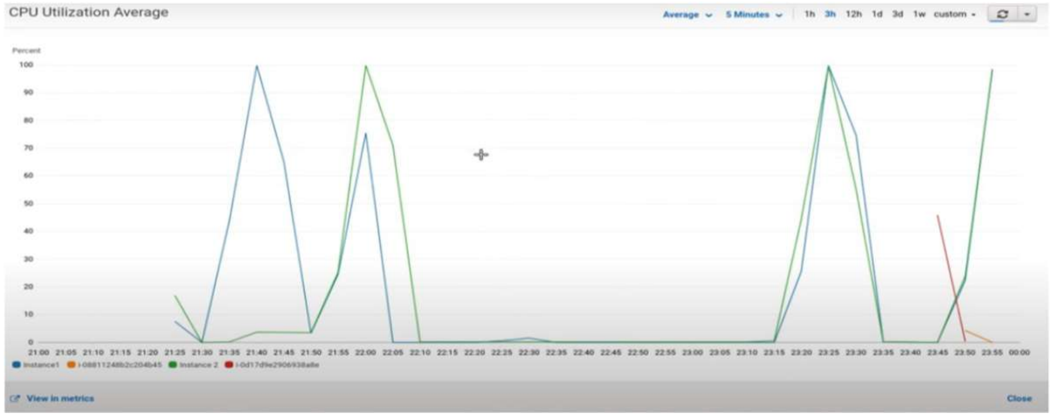
Paramecium

Paramecium is a unicellular organism belonging to the group of protozoa. It has an elongated shape with movable cilia covering its entire surface for movement.

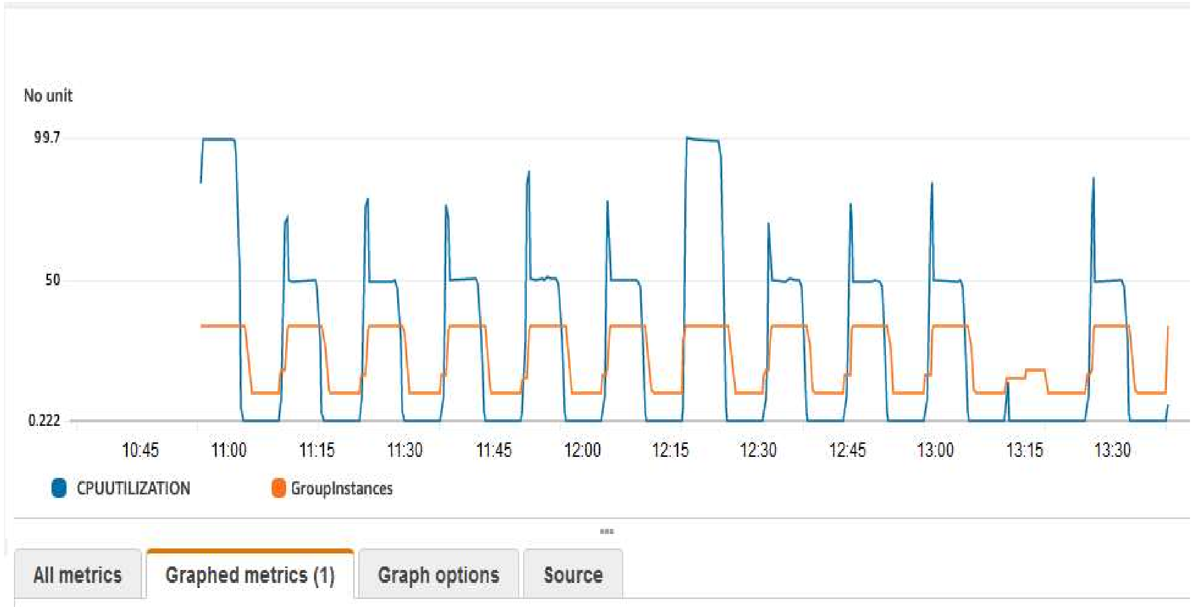
- **Size**: Typically ranging from 50 to 350 micrometers in length.
- **Diet**: Primarily feeds on bacteria, algae, and organic debris.
- **Temperature**: Lives in freshwater, and its temperature tolerance varies depending on the species.
- **Infectivity**: 1
- **Poisonus**: No

- Each image will have a description of each type of bacteria.
- Notice that the web site is developed in a static way and it’s just a bunch of files onto a S3 bucket.
- This ensures that the website is highly available since the distribution of the web site is menaged by AWS.

Scalability results



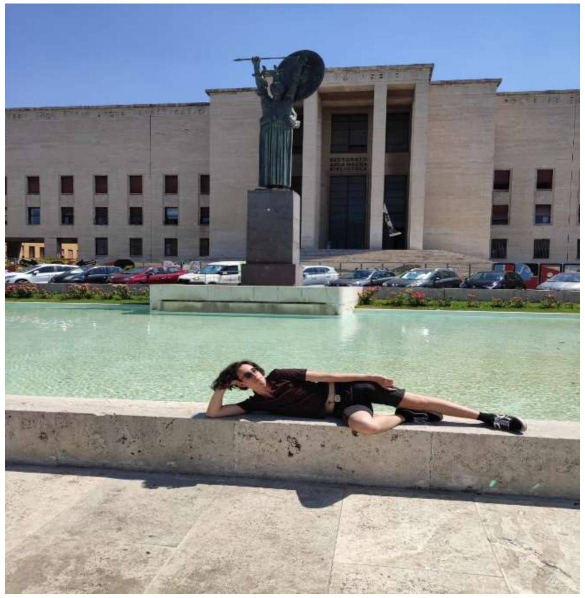
- Peaks of the CPU utilization of the two baseline instance when stressed.



- CPU usage versus number of instances running in Autoscaling Group.

Thanks for your attention

ENRICO GRIMALDI



MATTEO MIGLIARINI

GIUSEPPE DI POCE