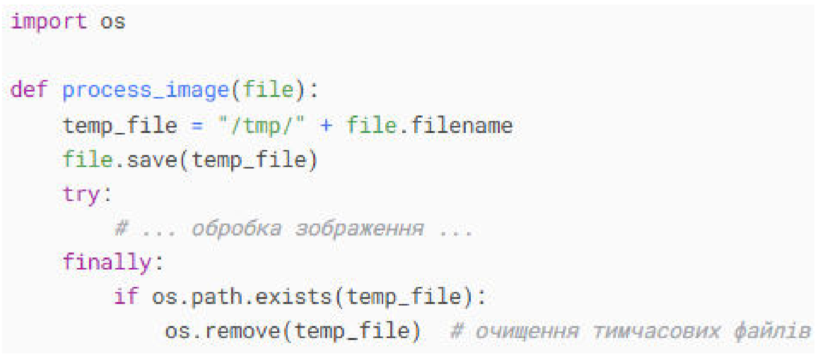
PREVENTING INCREASED MEMORY USAGE USING DOCKER CONTAINER STATISTICS

Serhii Popovniak, Nataliia Khatsko, Yevhenii Shebanov, Kyrylo Khatsko
National Technical University «Kharkiv Polytechnic Institute», Kharkiv

Docker statistics provide metrics on the current state of a container, reflecting CPU, memory, network traffic, and disk I/O usage [1]. With this data, you can monitor resource consumption, detect anomalies, optimize performance, and plan for scaling, such as adding replicas under stable load conditions. In practice, the following situation may occur: an image processing service written in Python periodically crashes with an Out of Memory-error. After restarting, it works, but after 2–3 hours, the issue recurs. The reason is that during operation, the service does not delete temporary files after processing. Specifically, the process_image() function saves files in the /TMP directory without subsequent cleanup. To identify the problem, use the Docker command "docker stats my-image-processor" and analyze the gradual increase in memory usage. During the experiment, the initial metrics were 450MiB out of 2GiB (22%), and after 2 hours, it reached 1.95GiB out of 2GiB (97%). Docker logs confirmed the accumulation of temporary files. To solve the problem, a software code is proposed that will prevent memory leaks during the application's operation (Fig. 1).



```python
import os

def process_image(file):
    temp_file = "/tmp/" + file.filename
    file.save(temp_file)
    try:
        # ... обробка зображення ...
    finally:
        if os.path.exists(temp_file):
            os.remove(temp_file)  # очищення тимчасових файлів
```

Fig. 1. – Code to delete files in the temporary directory

The code performs cleaning of temporary files, which prevents the increase in the size of the temporary storage directory /TMP. Additionally, an extra tool can be used – the software systems Grafana or Prometheus – which will allow creating conditions for notifying the developer to timely respond to increased memory usage. Thus, it is proposed to use the following approach in the application code – check for the presence of a file in /tmp and delete it after processing. This approach will help prevent memory leaks during the operation of the image processing service. Implementation of this software approach to the algorithm in work [2] will also help conserve resources in a Distributed IoT System in a Cloud Service.

References:

1. View container statistics https://docs.docker.com/engine/reference/commandline/stats/
2. Zamkovyi, M., Gavrylenko, S., Khatsko, K. and Khatsko, N. Algorithmic Support for Building a Distributed IoT System in a Cloud Service. 2023 IEEE 4th KhPI Week on Advanced Technology, Kharkiv, Ukraine, 2023, pp. 1-6, doi: 10.1109/KhPIWeek61412.2023.10312994.