These findings are the result of the research done for https://cyber-solutions.atlassian.net/browse/PA-58. We needed to find a way of running a queue consumer (SQS in this case) in the same docker container with our application (a python web app api).

The solution we settled for is uwsgi spooler: https://uwsgi-docs.readthedocs.io/en/latest/Spooler.html

We tried to find a solution that will not only match the current project (Amazon Pricing Feed) but all existing and future projects that will use a queue consumer.

Since uwsgi is used in all our existing application using this feature will be easy to implement. What uwsgi spooler does is to run in a different process(es) than the uwsgi workers that will serve the application and will import a python module. Here are the settings needed in the uwsgi configuration (.ini) file:

```
spooler = spools
import = pricing.workers.consumer
spooler-frequency = 1
spooler-processes = 1
```

where spools is an existing folder, `pricing.workers.consumer` is the module that contains the code that has to run (in our case the sqs consumer).

There are two decorators that can be used from the **uwsgidecorators** library: @timer and @spool

For our specific case the best solution is the **timer** decorator that will take as a parameter the number of seconds at which will run the decorated function. For example, if in the imported module there is the following code:

```
from uwsgidecorators import timer

@timer(3)
def sqs_consumer(t):
    pass
```

the sqs_consumer function will be executed every 3 seconds.

The other decorator **spool** is more useful for one-time tasks, like calling **auto-ordering api** from **integration-service**, or saving a **PSE Feed** to S3, that will shorten the response time for those services.

The way that feature works is: in the imported module there is a function decorated with **spool**:

```
from uwsgidecorators import timer

@spool
def one_time_task(args: Dict):
    queue_url = args.get('queue_url')
    pass
```

and in one of the application files:

```
from pricing.workers import consumer

consumer.one_time_task.spool(queue_url=config.SQS_QUEUE_URL)
```

this line will create a task file in the `spools` folder that was specified in the configuration file, and the spool decorated function will be executed.

There are also another spoole decorators available in the **uwsgidecorators** library:

spoolforever and **spoolraw** that also create/trigger decorated functions, but as the name suggests **spoolforever** will run continuously the decorated function by always return uwsgi.SPOOL_RETRY, while **spoolraw** passes the responsibility of returning either uwsgi.SPOOL_OK or uwsgi.SPOOL_RETRY to the decorated function.

Example of a sqs consumer running in a uwsgi spooler is implemented here:

https://gitlab.com/cst-pse/amazon-pricing-feed/blob/PO-58/src/pricing/workers/consumer.py