

Lousanje Ramírez Navarro

## Eagle Eye Networks Skill Test

This document contains a quick rundown of the process followed to get to the solution for the skill test

### Assumptions

These are some assumptions I made when coding the solution:

- The worker greenlets are coded as the minimum expression of them, that is, they are a single function that calls an asynchronous request to the API
- Concurrency was managed with a Pool to maximize the throughput (at first I thought of making batches of 5 greenlets or `gevent.spawns` and running those batches sequentially, but a pool seemed like a better option)
- The solution is just really 2 files, since that seemed like an adequate MVP
- `monkey.patch_all()` was used to make 'requests' play nice with `gevent`
- The `een.py` file is in great part not my code, I found it on github after a google search and decided to use it as it was a public repo

# Output

Here are some samples of the script running:

First some proof that it is working asynchronously, all 20 images in 1 shot, we can see all of them start at once and finish in different order:

```
(image-download) → ImageDownload git:(master) x /home/loucho/anaconda3/envs/image-download/bin/python /home/loucho/www/ImageDownload/src/imageload/main.py
Task 140573139343560 started
Task 140573139344648 started
Task 140573139344920 started
Task 140573126381640 started
Task 140573126381912 started
Task 140573126382184 started
Task 140573126382456 started
Task 140573126382728 started
Task 140573126383000 started
Task 140573126383272 started
Task 140573126383544 started
Task 140573126383816 started
Task 140573126384088 started
Task 140573126384360 started
Task 140573126384632 started
Task 140573126384904 started
Task 140573126385176 started
Task 140573126389832 started
Task 140573126390104 started
Task 140573126390376 started
Task 140573139343560 done
Task 140573126390104 done
Task 140573139344648 done
Task 140573126382184 done
Task 140573126381640 done
Task 140573139344920 done
Task 140573126381912 done
Task 140573126384088 done
Task 140573126384904 done
Task 140573126384360 done
Task 140573126385176 done
Task 140573126384632 done
Task 140573126382728 done
Task 140573126390376 done
Task 140573126383816 done
Task 140573126382456 done
Task 140573126383272 done
Task 140573126383544 done
Task 140573126383000 done
Task 140573126389832 done
(image-download) → ImageDownload git:(master) x
```

Now running in a pool of 5, we can see new ones start as soon as a worker is done:

```
(image-download) → ImageDownload git:(master) x /home/loucho/anaconda3/envs/image-download/bin/python /home/loucho/www/ImageDownload/src/imageload/main.py
Task 140088601513224 started
Task 140088601513496 started
Task 140088588542024 started
Task 140088588542296 started
Task 140088588542568 started
Task 140088601513224 done
Task 140088588542840 started
Task 140088588542296 done
Task 140088588543112 started
Task 140088601513496 done
Task 140088588542568 done
Task 140088588542024 done
Task 140088588543384 started
Task 140088588543656 started
Task 140088588543928 started
Task 140088588543656 done
Task 140088588544200 started
Task 140088588542840 done
Task 140088588544472 started
Task 140088588543112 done
Task 140088588544744 started
Task 140088588544200 done
Task 140088588545016 started
Task 140088588544744 done
Task 140088588545288 started
Task 140088588544472 done
Task 140088588545560 started
Task 140088588543928 done
Task 140088571465800 started
Task 140088588543384 done
Task 140088571466072 started
Task 140088588545016 done
Task 140088571466344 started
Task 140088571466072 done
Task 140088571466616 started
Task 140088588545560 done
Task 140088588545288 done
Task 140088571465800 done
Task 140088571466344 done
Task 140088571466616 done
(image-download) → ImageDownload git:(master) x
```

## The Requirements

Here are the relevant segments of code to cover the requirements:

Login to the API, grab a random camera:

```
context = een.login("demo@een.com", "bettercloud")
camera_list = context.camera_list()
img_list = []
#seems sometimes there are no images on the camera, let's ensure we get one with at least 20
while len(img_list) < 20:
    #choose a random camera to pull images from (index 1 contains the camera id)
    camera = random.choice(camera_list)[1]
```

Get 20 images (we pass end\_timestamp as now and get the latest 20):

```
img_list = context.image_list(esn=camera, asset_type="preview", time=een.timestamp("now"), count=-20)
```

Download them with 5 greenlet workers using gevent (in this case gevent.pool as Pool):

```
#fetch an image and save it the "downloads" folder, include camera name to identify them
def fetch_image(image):
    print('Task %s started' % id(getcurrent())) #get the thread id so we can see the concurrence at work
    img = context.fetch_image(esn=camera, time=image['s'], asset_type="preview")
    fh = open("downloads/" + camera + "-" + image['s'] + ".jpeg", "wb")
    fh.write(img)
    fh.close()
    print('Task %s done' % id(getcurrent()))

#fetch them all!
#threads = [gevent.spawn(fetch_image, val, index) for index, val in enumerate(img_list)]
#threads = [gevent.spawn(fetch_image, image) for image in img_list]
#gevent.joinall(threads)

#fetch them using a pool of 5 greenlets
pool = Pool(5)
pool.map(fetch_image, img_list)
```

## The Repository

The github repo is this: <https://github.com/loucho/image-download>

## Things that I tried or would like to have tried

- Give this a nice UI, even as a console app that could be invoked with some parameters (number of images to download, number of concurrent workers, user and password)
- Encapsulate some of the API responses and some of the behavior in classes (especially since the list of cameras is returning arrays of arrays, so accessing the data is done by the index of the array instead of a named property), this would probably be fun to do, but a bit overkill for the current requirements
- Maybe making it an API endpoint that a webapp can consume and display the results, or something
- Add unit tests and probably research on best practices