

# GCP FaaS

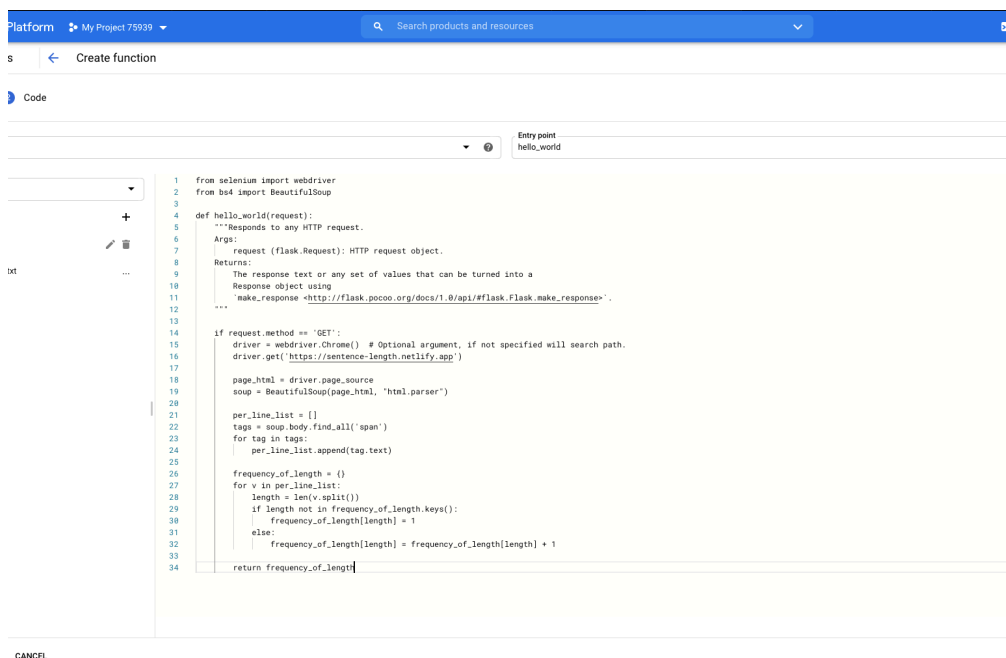
FaaS link: [https://funky1-xoya5vvlsq-uc.a.run.app/input?link=\\*anyurl\\*](https://funky1-xoya5vvlsq-uc.a.run.app/input?link=*anyurl*)

Example: <https://funky1-xoya5vvlsq-uc.a.run.app/input?link=https://sentence-length.netlify.app>

**Output format:** *JSON where key is the the number of words in a line and the value is the count of such lines with the same length.*  
*To note: I implement my solution in Python.*

## INITIAL SOLUTION (THAT DIDN'T WORK!):

Initially, I built a simple cloud function in my project to parse html, and calculate the distribution in the output format mentioned above. I used python to do this, with urllib3 and beautifulsoup4 as my main dependencies. The dependencies were added in **requirements.txt** while the main code was written in **main.py**.



```
1 from selenium import webdriver
2 from bs4 import BeautifulSoup
3
4 def hello_world(request):
5     """Responds to any HTTP request.
6
7     Args:
8         request (flask.Request): HTTP request object.
9
10    Returns:
11        The response text or any set of values that can be turned into a
12        Response object using
13        'make_response' ->http://flask.pocoo.org/docs/1.0/api/#flask.Flask.make_response' .
14    """
15
16    if request.method == 'GET':
17        driver = webdriver.Chrome() # Optional argument, if not specified will search path.
18        driver.get('https://sentence-length.netlify.app')
19        page_html = driver.page_source
20        soup = BeautifulSoup(page_html, 'html.parser')
21
22        per_line_list = []
23        tags = soup.body.find_all('span')
24        for tag in tags:
25            per_line_list.append(tag.text)
26
27        frequency_of_length = {}
28        for v in per_line_list:
29            length = len(v.split())
30            if length not in frequency_of_length.keys():
31                frequency_of_length[length] = 1
32            else:
33                frequency_of_length[length] = frequency_of_length[length] + 1
34
35    return frequency_of_length
```

## FaaS with GCP Cloud functions

However, the above cloud function did not work because even though it succeeded in calculating frequency distribution in static webpages, most webpages are dynamic generated at runtime (like with javascript). Typical HTML parsers and web-scrappers don't work with that because they're not able to fetch the actual content of the webpage that we see in the browser. For example, in the link shared, we don't see any content of the Christmas Carol with our static crawler.

## SOLUTION THAT WORKS:

To capture the content of javascript generated webpages, I use Selenium's Chrome driver to crawl and parse dynamic webpages.

**However, using selenium and chrome driver requires a chrome binary to be installed with an executable path in the system.**

**That's a problem if we want to run serverless cloud functions!**

Headless Chrome with Selenium in general doesn't work with GCP Cloud functions. Out of all the languages supported for GCP Cloud functions, the only compatible solution is using *Node.js* with *puppeteer*.

Alternatively, I used Google Cloud Run to pre-build an image with Google Chrome Driver and all the python dependencies with Docker and push the image with my code (main.py).

The Dockerfile is attached in the submission and looks like:

```

all the missing libraries

$ sudo dpkg --get-selections | grep install
$ sudo apt-get install \
  gconf-service libasound2 libatk1.0-0 libcairo2 libcups2 libfontconfig1 libgdk-pixbuf2.0-0 libgtk-3-0 libnspr4 libnss3 \
  libpango1.0-0 libpangocairo-1.0-0 libpangox-1.0-0 libx11-xcb1 libxcomposite1 libxdamage1 libxext6 libxfixes3 libxi6 libxrandr2 \
  libxrender1 libxss1 libxtst6 xdg-utils

$ curl -O https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
$ dpkg -i google-chrome-stable_current_amd64.deb; apt-get -f install

$ curl -O https://dl.google.com/linux/direct/chromedriver_linux64.zip
$ unzip chromedriver_linux64.zip

$ sudo mv /usr/bin/chromedriver
$ sudo chown root:root /usr/bin/chromedriver
$ sudo chmod 755 /usr/bin/chromedriver

$ pip install -r requirements.txt
$ python requirements.txt

$ docker build -t myapp .
$ docker run --rm myapp

# To run the container image.

$ docker run --rm -p 4444:4444 --name myapp myapp
$ docker ps
$ docker logs myapp

# To run the container image with multiple CPU cores.
$ docker run --rm -p 4444:4444 --name myapp --cpus 4 myapp
$ docker ps
$ docker logs myapp

# To run the container image with multiple CPU cores and 8 threads.
$ docker run --rm -p 4444:4444 --name myapp --cpus 4 --threads 8 myapp
$ docker ps
$ docker logs myapp

# To run the container image with multiple CPU cores, increase the number of workers
$ docker run --rm -p 4444:4444 --name myapp --cpus 4 --workers 4 --threads 8 myapp
$ docker ps
$ docker logs myapp

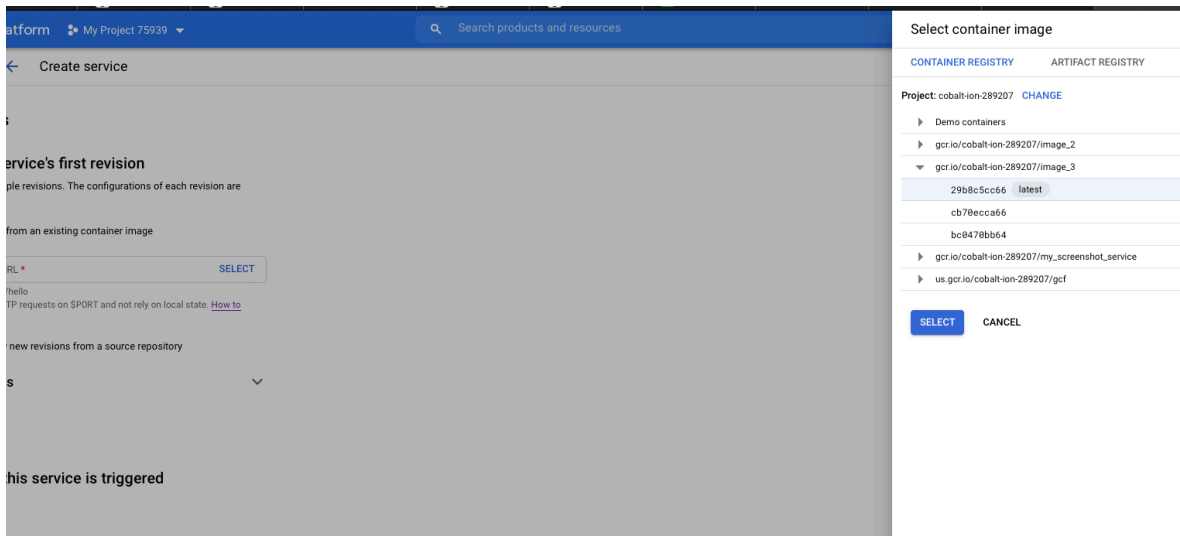
# To run the container image with multiple CPU cores and 8 threads.
$ docker run --rm -p 4444:4444 --name myapp --cpus 4 --threads 8 --workers 4 myapp
$ docker ps
$ docker logs myapp

```

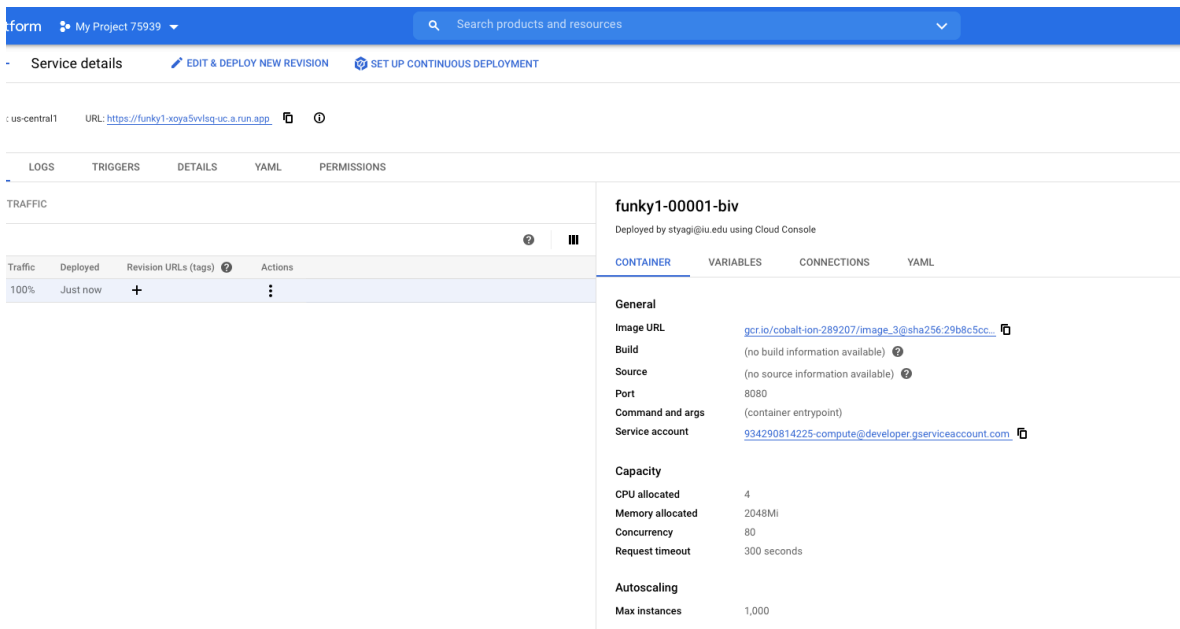
## Dockerfile

I build and push my image as named ***image\_3*** under project ***cobalt-ion-289207*** with:

```
gcloud builds submit --tag gcr.io/cobalt-ion-289207/image_3
```



## Creating our Function with custom image



## FaaS running successfully