Microservices Assignment 1

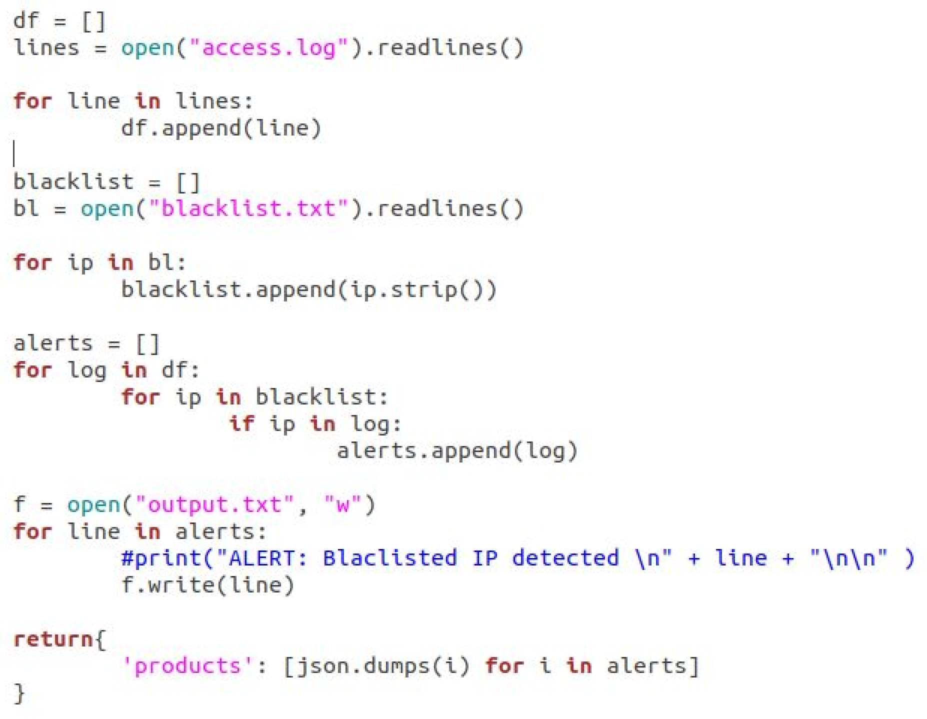
Cian Roddis - R00140685

Approach

I will go through all these topics in depth but first an overview of my approach. My first thought about this assignment was that REST would probably the best means of communication for the webservice over Messaging or GRPC. This did turn out to be the case as REST gave me no problems creating an api that fed JSON to the webpage as log alerts were fed in. I have a webservice updating when a alert occurs. The page will update as it reads in the new alerts. I have a database service that saves the alerts to an output file. I didn't use redis as my main datastore but I do have it as a docker service anyway just to show how it works. I have an email service that I attempted to send real emails but failed to do so. I just emulated it by outputting to a text file the alerts. All of these features are their own service. They are all python docker containers bar the webservice one, which is a php container. I have all of these services then orchestrated via a docker-compose YML file.

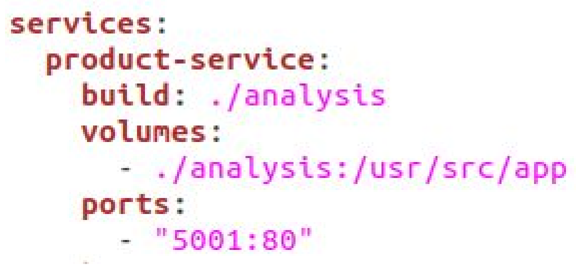
Analysis

My analysis container is a python container that uses Flask and Flask_restful to emulate an API to a server. It reads in the log file line by line and appends them to list. I read in the list of blacklisted IPs and add them to a list. I then check if any of the logs have any of the IPs contained within them. If they do that log is added to a list of alerts which is then fed as raw JSON to a webserver.



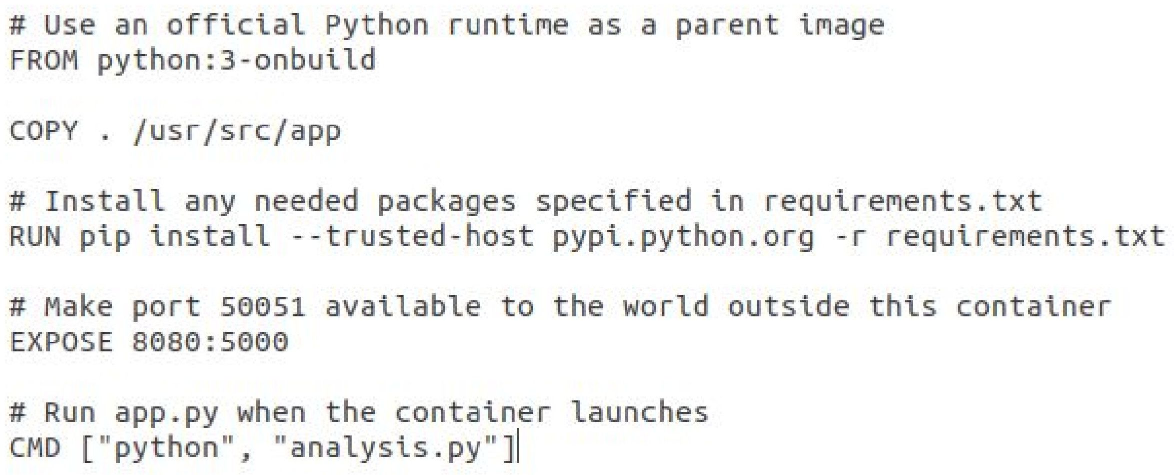
```
df = []
lines = open("access.log").readlines()

for line in lines:
        df.append(line)

blacklist = []
bl = open("blacklist.txt").readlines()

for ip in bl:
        blacklist.append(ip.strip())

alerts = []
for log in df:
        for ip in blacklist:
                if ip in log:
                        alerts.append(log)

f = open("output.txt", "w")
for line in alerts:
        #print("ALERT: Blaclisted IP detected \n" + line + "\n\n" )
        f.write(line)

return{
        'products': [json.dumps(i) for i in alerts]
}
```

I then write a Dockerfile along with a requirements file for dependencies like Flask and flask restful. I then add that service to my docker-compose yml file.



```yaml
services:
  product-service:
    build: ./analysis
    volumes:
      - ./analysis:/usr/src/app
    ports:
      - "5001:80"
```

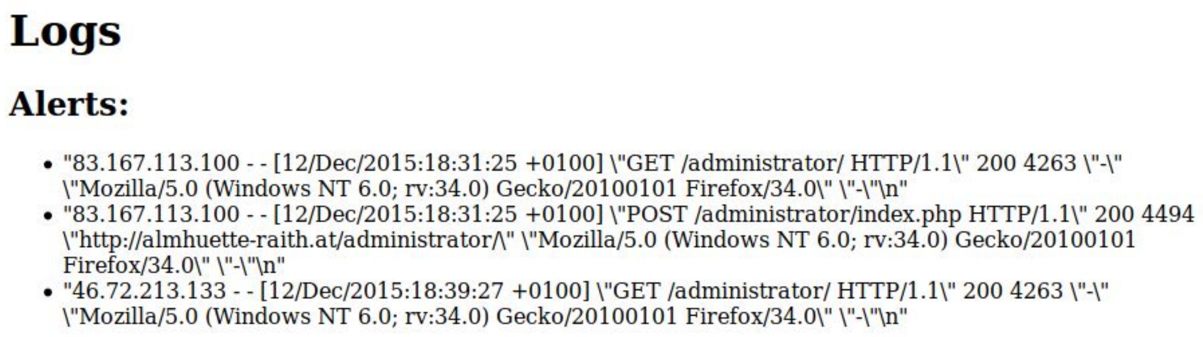
My Dockerfiles vary very little for all my python containers but here's what the analysis one looks like:



```dockerfile
# Use an official Python runtime as a parent image
FROM python:3-onbuild

COPY . /usr/src/app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 50051 available to the world outside this container
EXPOSE 8080:5000

# Run app.py when the container launches
CMD ["python", "analysis.py"]
```

Webservice

My webservice is a PHP image that gets fed JSON from analysis. It is given alerts and then out puts them to the screen in bullet point format.

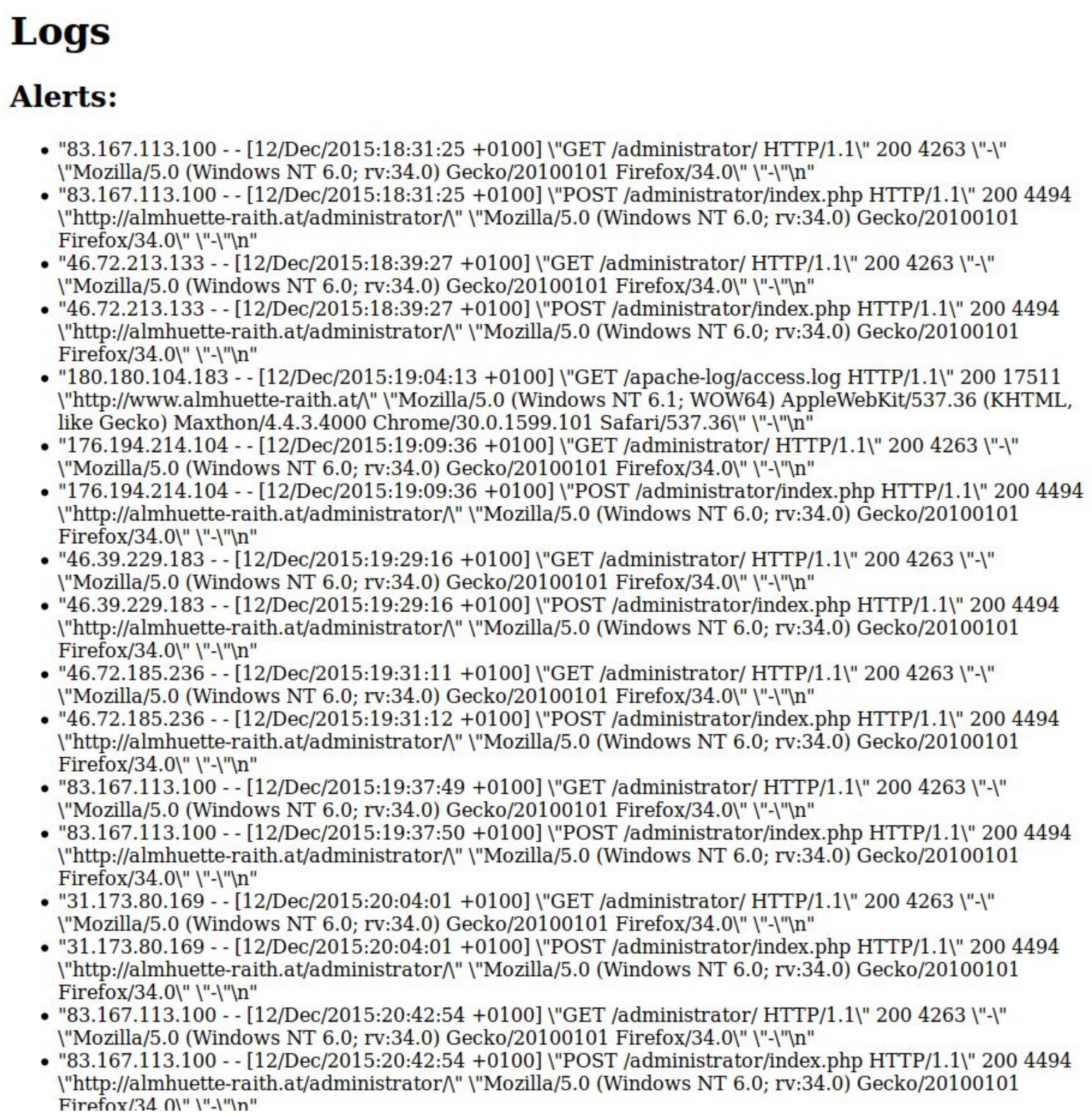


Logs

Alerts:

```
• "83.167.113.100 - - [12/Dec/2015:18:31:25 +0100] \"GET /administrator/ HTTP/1.1\" 200 4263 \"-\"
  \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
• "83.167.113.100 - - [12/Dec/2015:18:31:25 +0100] \"POST /administrator/index.php HTTP/1.1\" 200 4494
  \"http://almhuette-raith.at/administrator/\" \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101
  Firefox/34.0\" \"-\"\n"
• "46.72.213.133 - - [12/Dec/2015:18:39:27 +0100] \"GET /administrator/ HTTP/1.1\" 200 4263 \"-\"
  \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
```

After a while the page will fill up with alerts as they are read in:

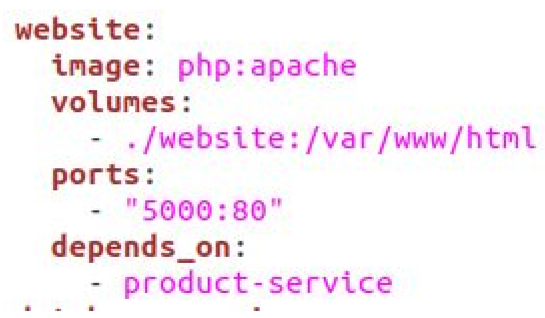


Logs

Alerts:

```
• "83.167.113.100 - - [12/Dec/2015:18:31:25 +0100] \"GET /administrator/ HTTP/1.1\" 200 4263 \"-\" \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
• "83.167.113.100 - - [12/Dec/2015:18:31:25 +0100] \"POST /administrator/index.php HTTP/1.1\" 200 4494 \"http://almhuette-raith.at/administrator/\" \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
• "46.72.213.133 - - [12/Dec/2015:18:39:27 +0100] \"GET /administrator/ HTTP/1.1\" 200 4263 \"-\" \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
• "46.72.213.133 - - [12/Dec/2015:18:39:27 +0100] \"POST /administrator/index.php HTTP/1.1\" 200 4494 \"http://almhuette-raith.at/administrator/\" \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
• "180.180.104.183 - - [12/Dec/2015:19:04:13 +0100] \"GET /apache-log/access.log HTTP/1.1\" 200 17511 \"http://www.almhuette-raith.at/\" \"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Maxthon/4.4.3.4000 Chrome/30.0.1599.101 Safari/537.36\" \"-\"\n"
• "176.194.214.104 - - [12/Dec/2015:19:09:36 +0100] \"GET /administrator/ HTTP/1.1\" 200 4263 \"-\" \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
• "176.194.214.104 - - [12/Dec/2015:19:09:36 +0100] \"POST /administrator/index.php HTTP/1.1\" 200 4494 \"http://almhuette-raith.at/administrator/\" \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
• "46.39.229.183 - - [12/Dec/2015:19:29:16 +0100] \"GET /administrator/ HTTP/1.1\" 200 4263 \"-\" \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
• "46.39.229.183 - - [12/Dec/2015:19:29:16 +0100] \"POST /administrator/index.php HTTP/1.1\" 200 4494 \"http://almhuette-raith.at/administrator/\" \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
• "46.72.185.236 - - [12/Dec/2015:19:31:11 +0100] \"GET /administrator/ HTTP/1.1\" 200 4263 \"-\" \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
• "46.72.185.236 - - [12/Dec/2015:19:31:12 +0100] \"POST /administrator/index.php HTTP/1.1\" 200 4494 \"http://almhuette-raith.at/administrator/\" \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
• "83.167.113.100 - - [12/Dec/2015:19:37:49 +0100] \"GET /administrator/ HTTP/1.1\" 200 4263 \"-\" \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
• "83.167.113.100 - - [12/Dec/2015:19:37:50 +0100] \"POST /administrator/index.php HTTP/1.1\" 200 4494 \"http://almhuette-raith.at/administrator/\" \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
• "31.173.80.169 - - [12/Dec/2015:20:04:01 +0100] \"GET /administrator/ HTTP/1.1\" 200 4263 \"-\" \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
• "31.173.80.169 - - [12/Dec/2015:20:04:01 +0100] \"POST /administrator/index.php HTTP/1.1\" 200 4494 \"http://almhuette-raith.at/administrator/\" \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
• "83.167.113.100 - - [12/Dec/2015:20:42:54 +0100] \"GET /administrator/ HTTP/1.1\" 200 4263 \"-\" \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
• "83.167.113.100 - - [12/Dec/2015:20:42:54 +0100] \"POST /administrator/index.php HTTP/1.1\" 200 4494 \"http://almhuette-raith.at/administrator/\" \"Mozilla/5.0 (Windows NT 6.0; rv:34.0) Gecko/20100101 Firefox/34.0\" \"-\"\n"
```

There is no need for a dockerfile or requirements for a php image because I just pulled the container in straight from docker-compose:

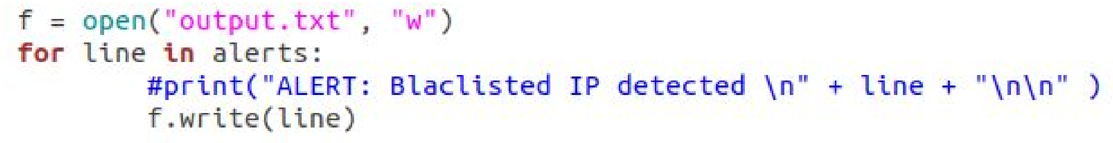


```yaml
website:
  image: php:apache
  volumes:
    - ./website:/var/www/html
  ports:
    - "5000:80"
  depends_on:
    - product-service
```

It’s mounted on volumes so it updates without refreshing, however it does depend on product-service, a.k.a my analysis container.

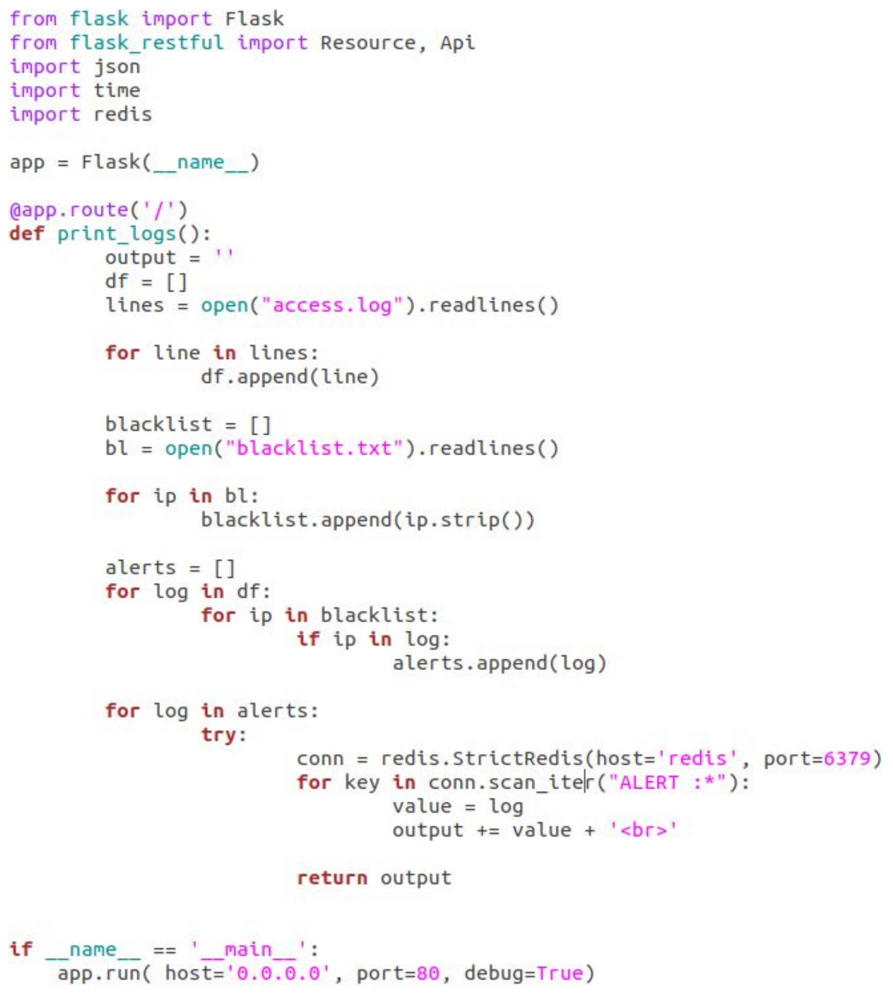
Database

I tried to implement redis into my database container but I struggled and ended up just coupling db creation in the last container. If you noticed:



```
f = open("output.txt", "w")
for line in alerts:
        #print("ALERT: Blaclisted IP detected \n" + line + "\n\n" )
        f.write(line)
```

This just outputs to a dummy db, a text file. My database container code looks like this:

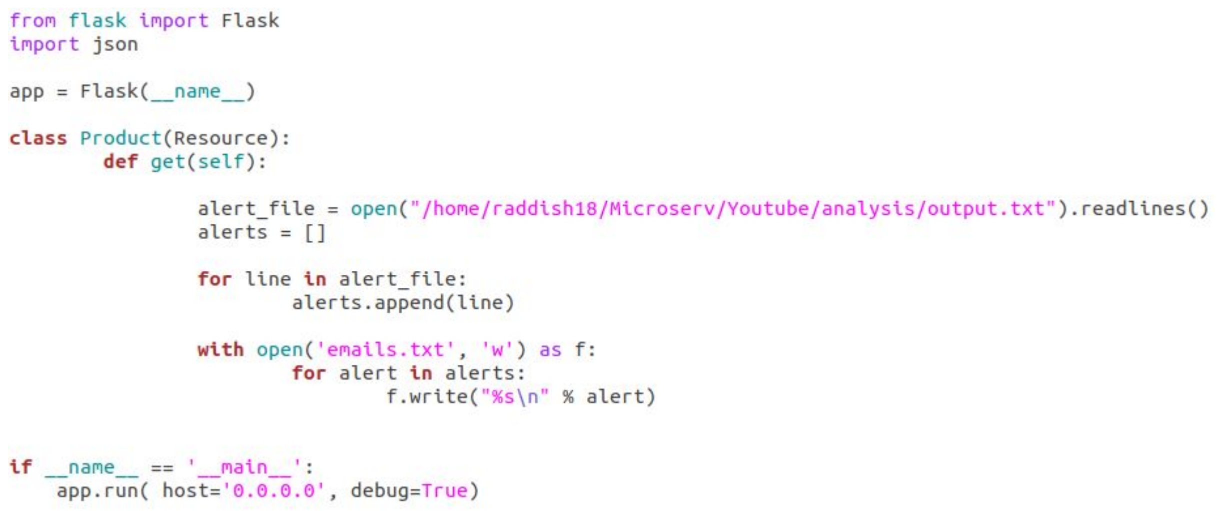


```
from flask import Flask
from flask_restful import Resource, Api
import json
import time
import redis

app = Flask(__name__)

@app.route('/')
def print_logs():
        output = ''
        df = []
        lines = open("access.log").readlines()

        for line in lines:
                df.append(line)

        blacklist = []
        bl = open("blacklist.txt").readlines()

        for ip in bl:
                blacklist.append(ip.strip())

        alerts = []
        for log in df:
                for ip in blacklist:
                        if ip in log:
                                alerts.append(log)

        for log in alerts:
                try:
                        conn = redis.StrictRedis(host='redis', port=6379)
                        for key in conn.scan_iter("ALERT :*"):
                                value = log
                                output += value + '<br>'

                        return output


if __name__ == '__main__':
    app.run( host='0.0.0.0', port=80, debug=True)
```

The dockerfile and requirements dont vary from the last container.

Email

I really did try to get a full email service up and running but it never came to fruition however I just tried to output to a text file instead. It reads the alerts that were created by the analysis service.



```
from flask import Flask
import json

app = Flask(__name__)

class Product(Resource):
        def get(self):

                alert_file = open("/home/raddish18/Microserv/Youtube/analysis/output.txt").readlines()
                alerts = []

                for line in alert_file:
                        alerts.append(line)

                with open('emails.txt', 'w') as f:
                        for alert in alerts:
                                f.write("%s\n" % alert)


if __name__ == '__main__':
    app.run( host='0.0.0.0', debug=True)
```

Here’s my attempt at the email service and sorting by 24 hour periods:

```python
app = Flask(__name__)

@app.route("/")

def email():
        port = 465  # For SSL
        smtp_server = "smtp.gmail.com"
        sender_email = "gibbytestman@gmail.com"  # Enter your address
        receiver_email = "gibbytestman@gmail.com"  # Enter receiver address
        password = "P@ssw0rd42"

        alert_file = open("/home/raddish18/Microserv/Youtube/analysis/output.txt", "r")
        blacklist = []
        bl = open("blacklist.txt").readlines()
        for ip in bl:
                blacklist.append(ip.strip())

        df = []
        for line in alert_file:
                #print line
                df.append(map(''.join, re.findall(r'\"(.*?)\"|\[(.*?)\]|(\S+)', line)))
                #print re.match(regex, line).groups()

        df = filter(None, df)

        regex = '([(\d\.)]+) - - \[(.*?)\] "(.*?)" (\d+) - "(.*?)" "(.*?)"'

        # Create a secure SSL context
        context = ssl.create_default_context()
        last_entry_time = []
        fmt = "%d-%m-%Y %H:%M:%S +%d%d%d%d"

        with smtplib.SMTP_SSL(smtp_server, port, context=context) as server:
                server.login("gibbytestman@gmail.com", password)

                for line in df:
                        for ip in bl:
                                if (ip not in last_entry_time):
                                        date_only = datetime.datetime.strptime(line[3], fmt)
                                        last_entry_time.append(ip, date_only)
                                        message = "ALERT BLACKLISTED IP TRIGGERED:\n" + ip + "\t" +
line[3]
                                        server.sendmail(sender_email, receiver_email, message)

                                elif(line[0] in last_entry_time):
                                        if(datetime.datetime.strptime(line[3], fmt) -
last_entry_time[1] >= timedelta(hours=24)):
                                                #print( ip + "\t" + line[3])
                                                message = "ALERT BLACKLISTED IP TRIGGERED:\n" + ip +
"\t" + line[3]
                                                server.sendmail(sender_email, receiver_email, message)


if __name__ == '__main__':
    app.run( host='0.0.0.0', port=80, debug=True)
```