PYCON ODESSA

Go && Python
Event driven ML

Lalafo case

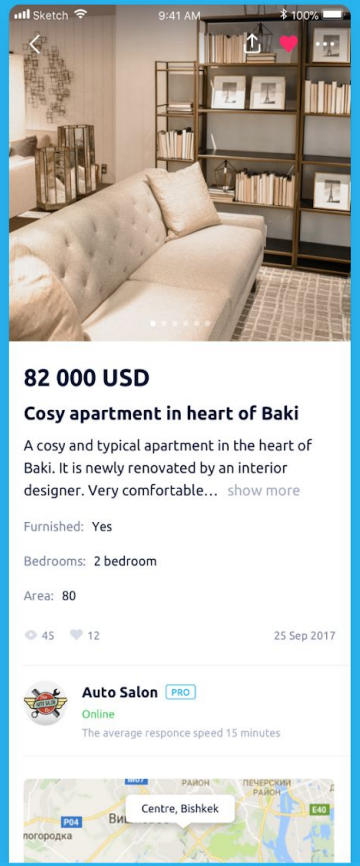
Новые функции

Искусственный интеллект помогает продавать

lalafo использует собственную технологию компьютерного зрения и распознает товары по фотографии.

Bicycle

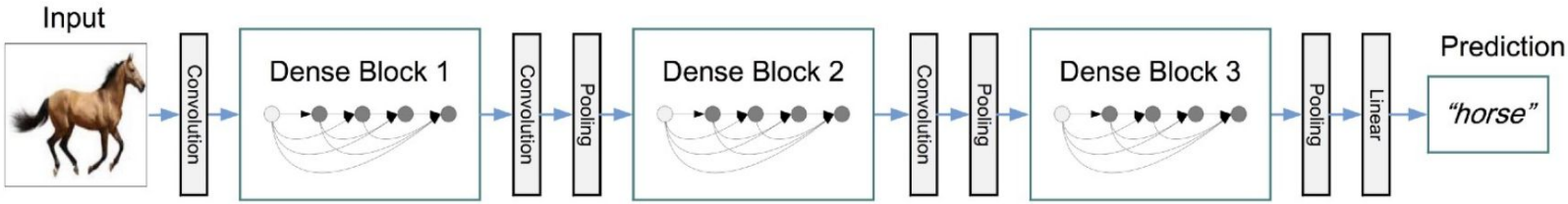
Loudspeakers



lalafo

How it all started

Part 1. The Phantom Menace



PYTORCH

python™

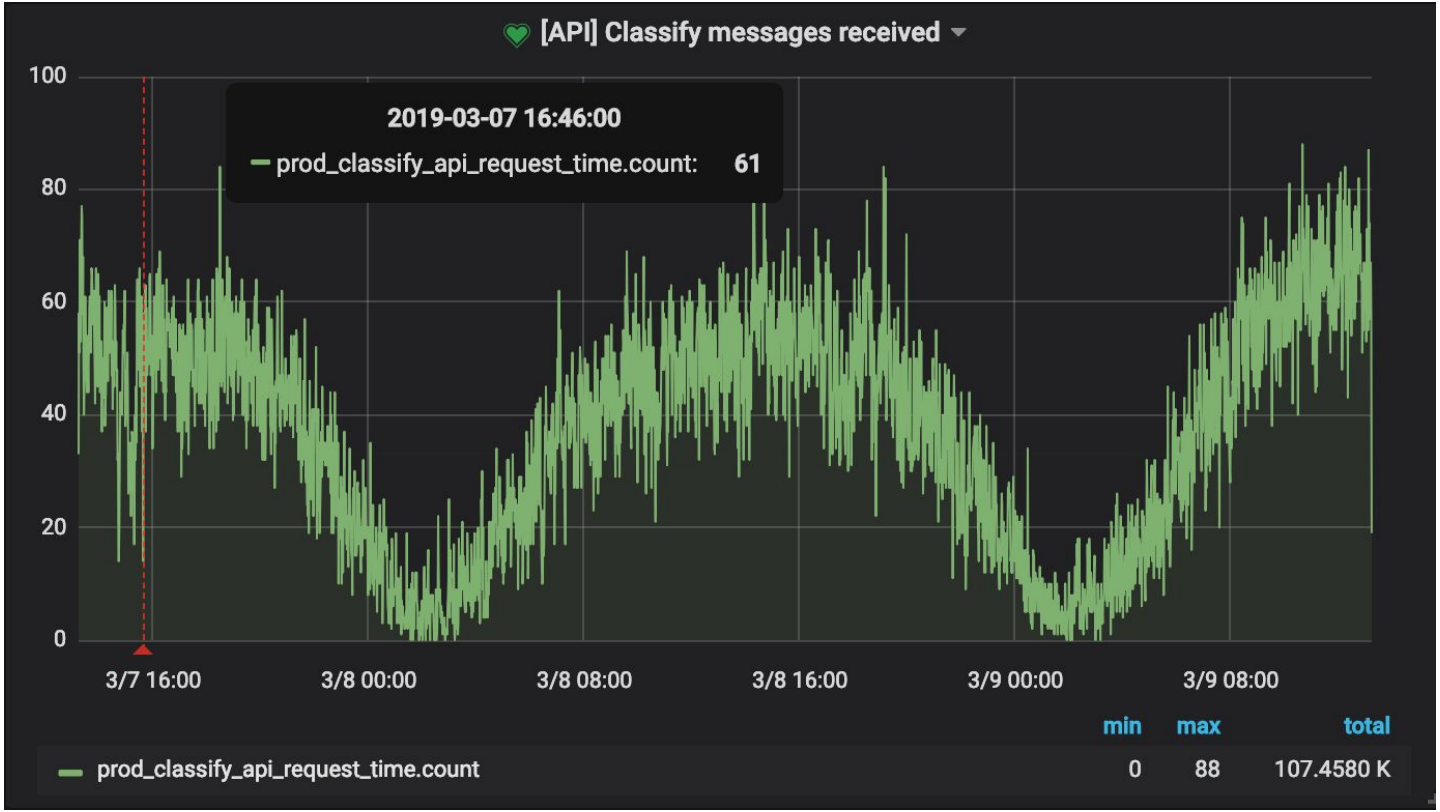
Tornado

amazon web services™

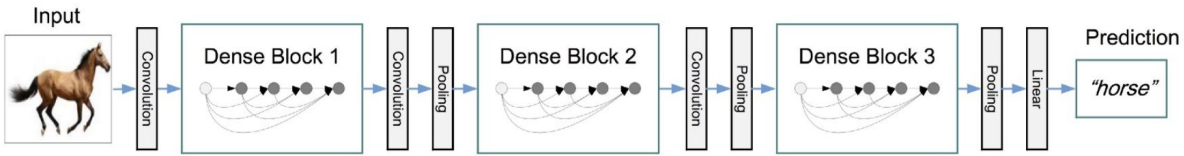
```python
class ClassifyHandler(tornado.web.RequestHandler):
    executor = ThreadPoolExecutor(2)

    @run_on_executor
    def predict(self, data):
        results = self.classifier.predict(data)
        self.storage: RedisStorage = self.get_storage()
        self.storage.save(results)

    async def post(self):
        # get data
        request_data = self.get_request_data()
        await self.predict(request_data)

class GetClassifyHandler(tornado.web.RequestHandler):
    async def get(self):
        ad_id = self.get_request_data()['ad_id']
        self.storage: RedisStorage = self.get_storage()
        results = self.storage.get(ad_id)
        self.finish({"status": "Success", "results": results})

application = tornado.web.Application([
    (r"/classify", ClassifyHandler),
    (r"/classify/results", GetClassifyHandler),
])
```

StackOverflow: Ready to rescue

200 RPM


[API] Classify messages received
100
80
60
40
20
0
2019-03-07 16:46:00
prod_classify_api_request_time.count: 61
3/7 16:00
3/8 00:00
3/8 08:00
3/8 16:00
3/9 00:00
3/9 08:00
min
max
total
prod_classify_api_request_time.count
0
88
107.4580 K

lalafo



PYTORCH

python™

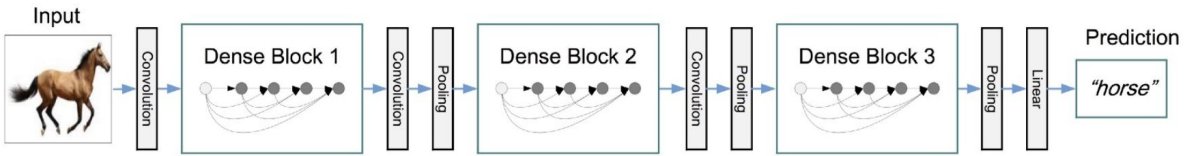
Tornado

$

amazon web services™

mongoDB.

Feedback



PYTORCH

python™

Processes Monitoring

Tornado

$

amazon web services™

RAM - expiration?

Relational DB?

mongoDB.

Feedback

lalafo

Any issues there?

1. Request monitoring (?)
2. Hard to reason GPU usage (varies 3 to 7GB)
3. ~5Gb RAM out of the box (scale?)
4. Redis is in-memory storage (temporary)
5. Threading != Parallel

lalafo

PRESS IN DIRE SITUATIONS

If you're reading this on and Android, we've released the Noooooooooooooo button as a Free App

http://www.noooooooooooooooo.com/

lalafo

What if we split API and ML

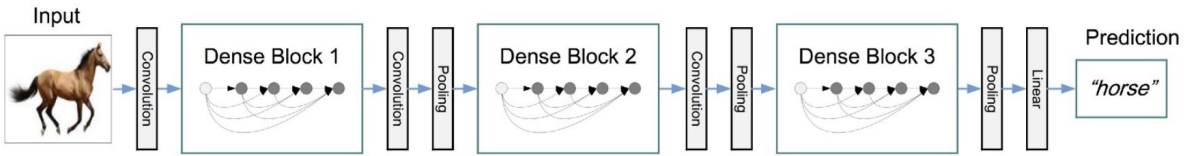
Part 2. A new hope

lalafo

Requirements

1. Single image/request processing time: < 2 seconds
2. Visibility
3. Persistence
4. Scalability
5. Make it extendable for new features
    a. Price prediction
    b. Similarity search
    c. Segmentation
6. SDK friendly (well documented, tested etc)

Lalafo case



PYTORCH

python™

Processes Monitoring

Tornado

$

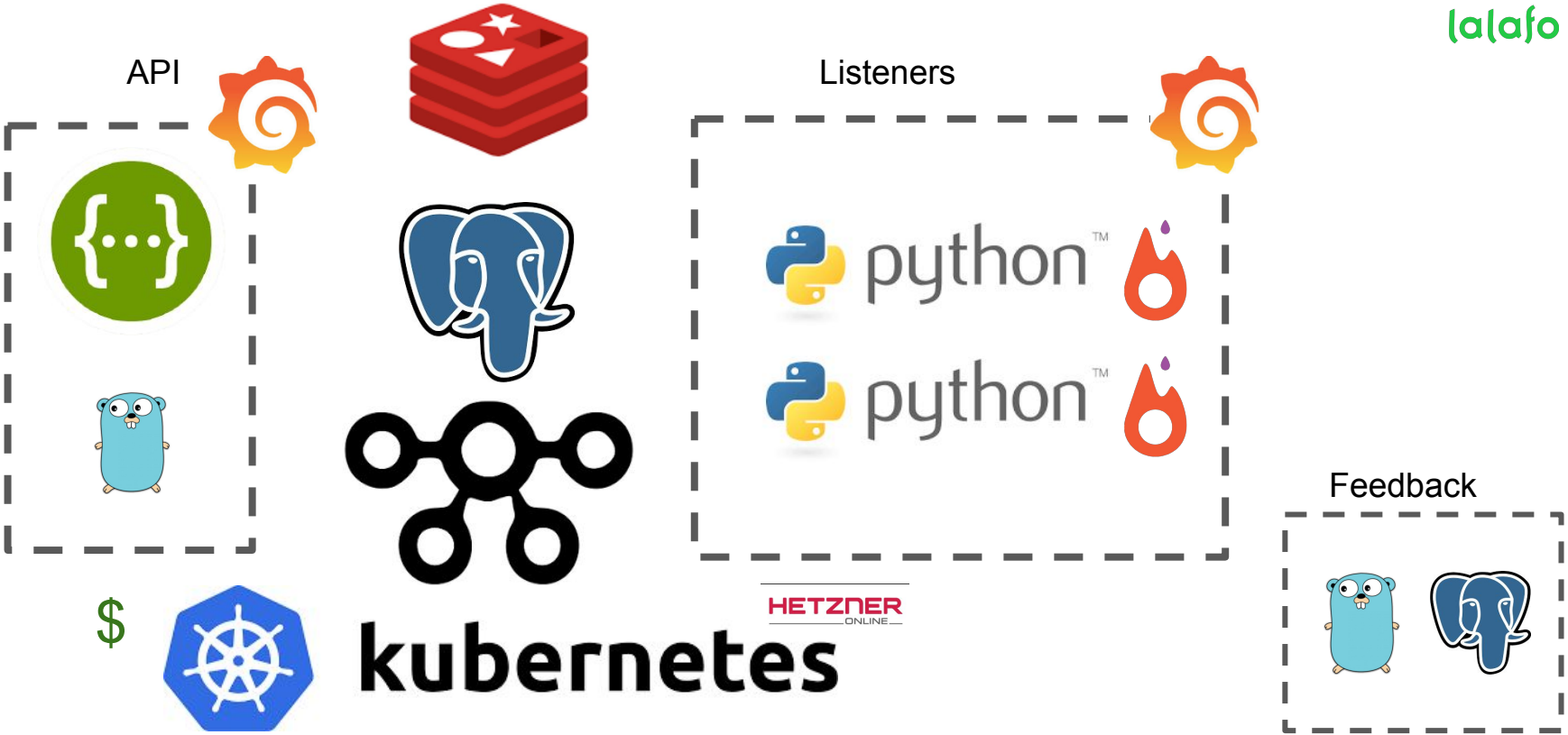
amazon web services™

RAM - expiration?

Relational DB?

mongoDB.

Feedback



After 2 months

lalafo

Requirements

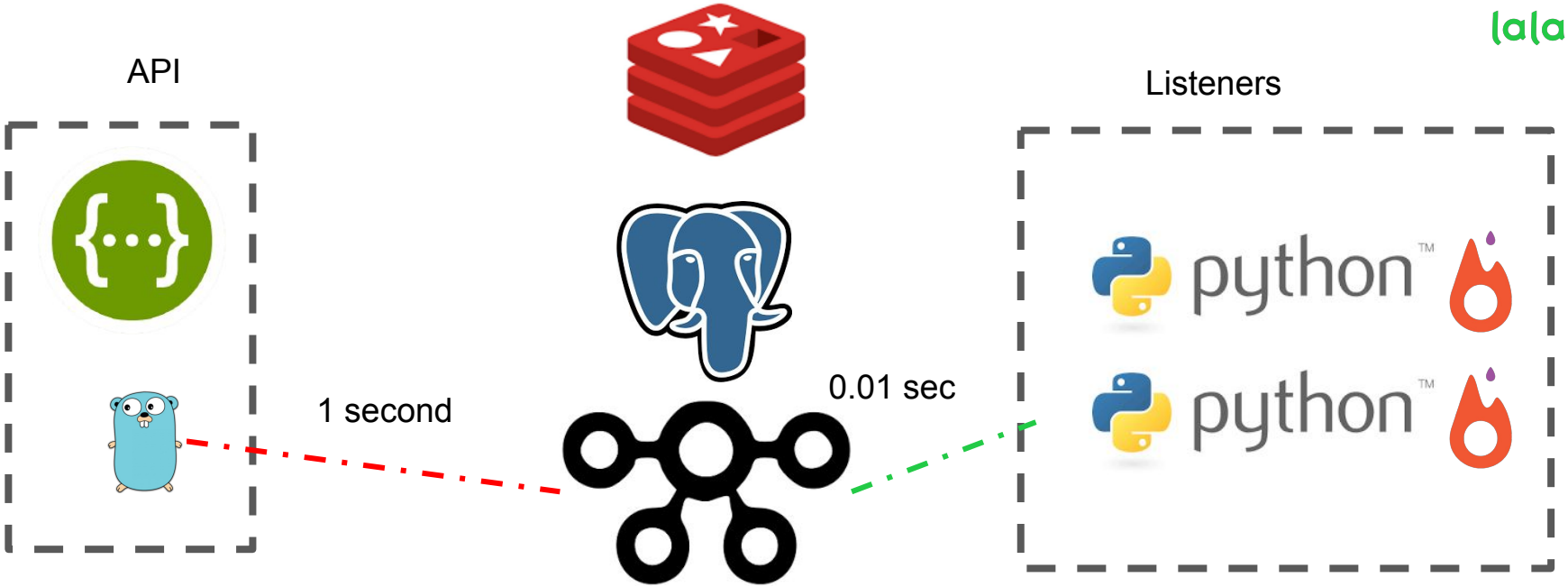
1. Single image/request processing time: < 2 seconds
2. Visibility - Decouple request and prediction
3. Persistence
4. Scalability kubernetes
5. Make it extendable for new features
   a. Price prediction
   b. Similarity search
   c. Segmentation
6. SDK friendly (well documented, tested etc)

Lalafo case

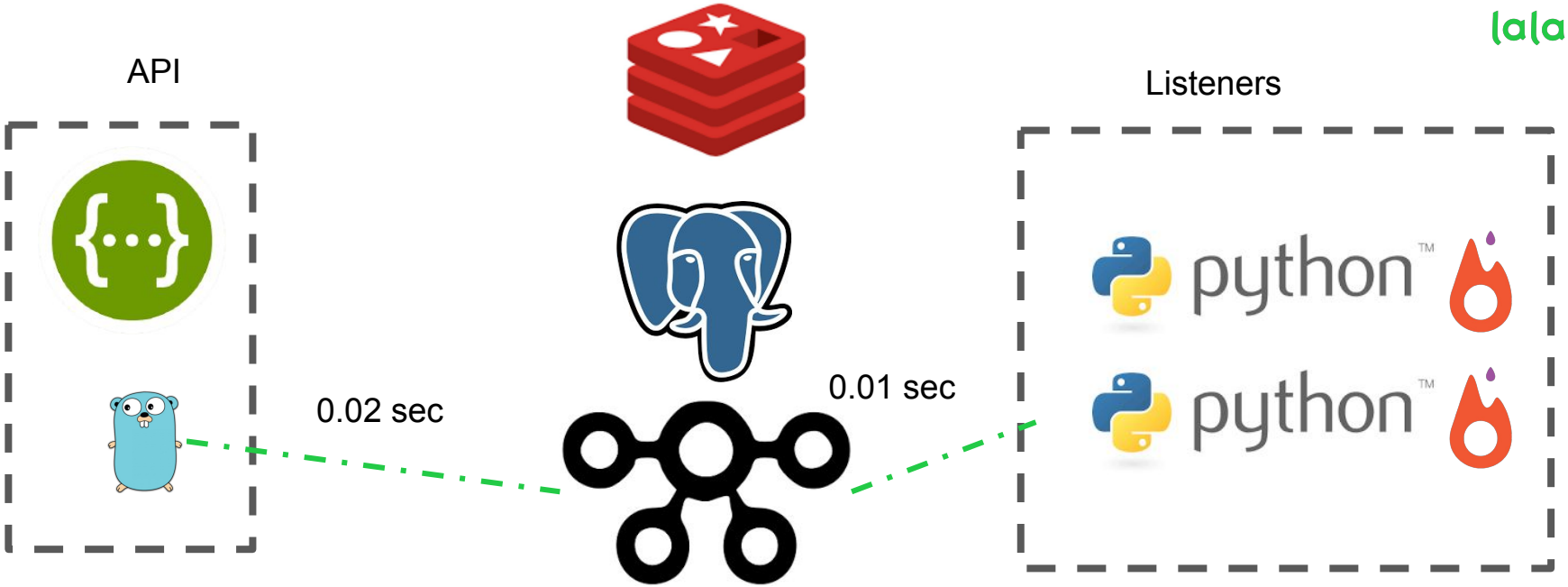
lalafo

2 seconds per request

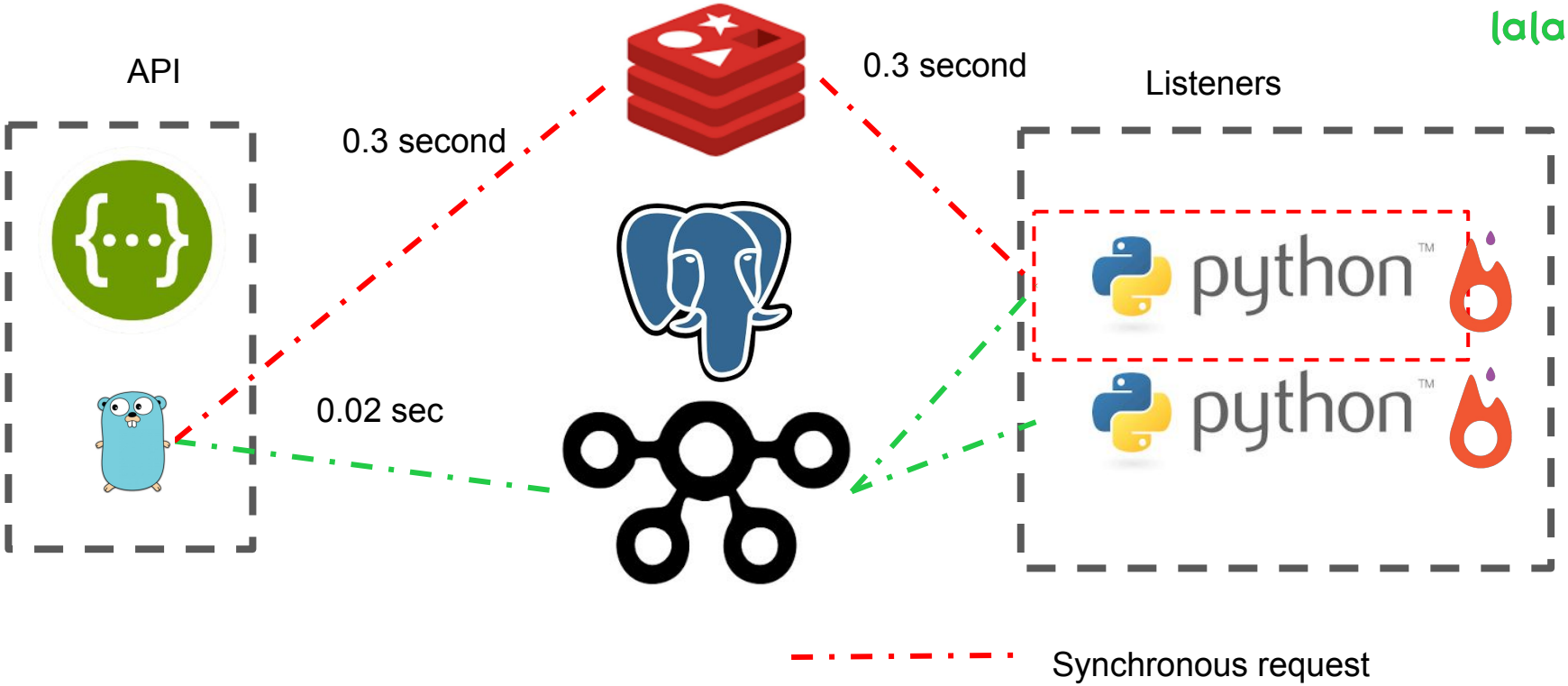
Part 3. The Empire Strikes Back



4 seconds per request



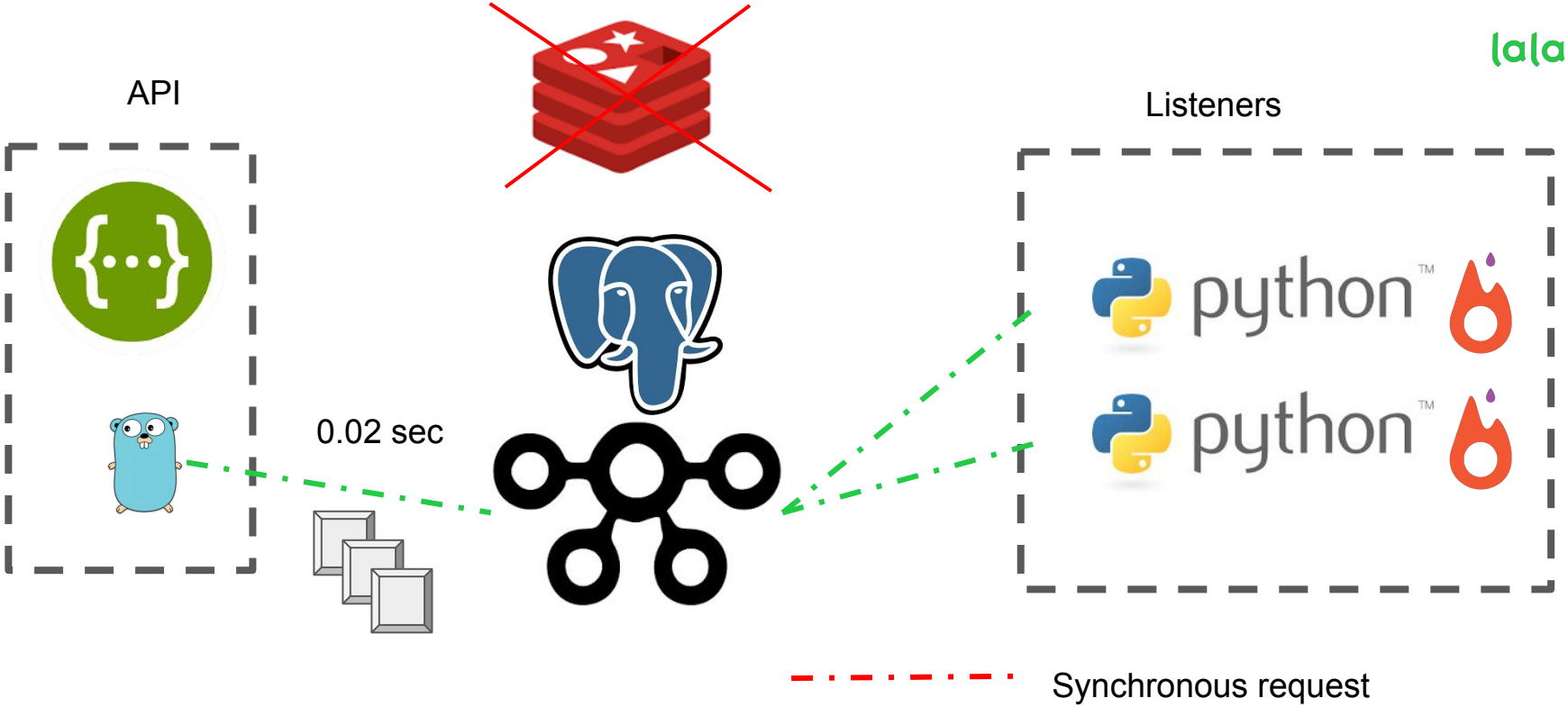
segmentio/kafka-go -> shopify/sarama



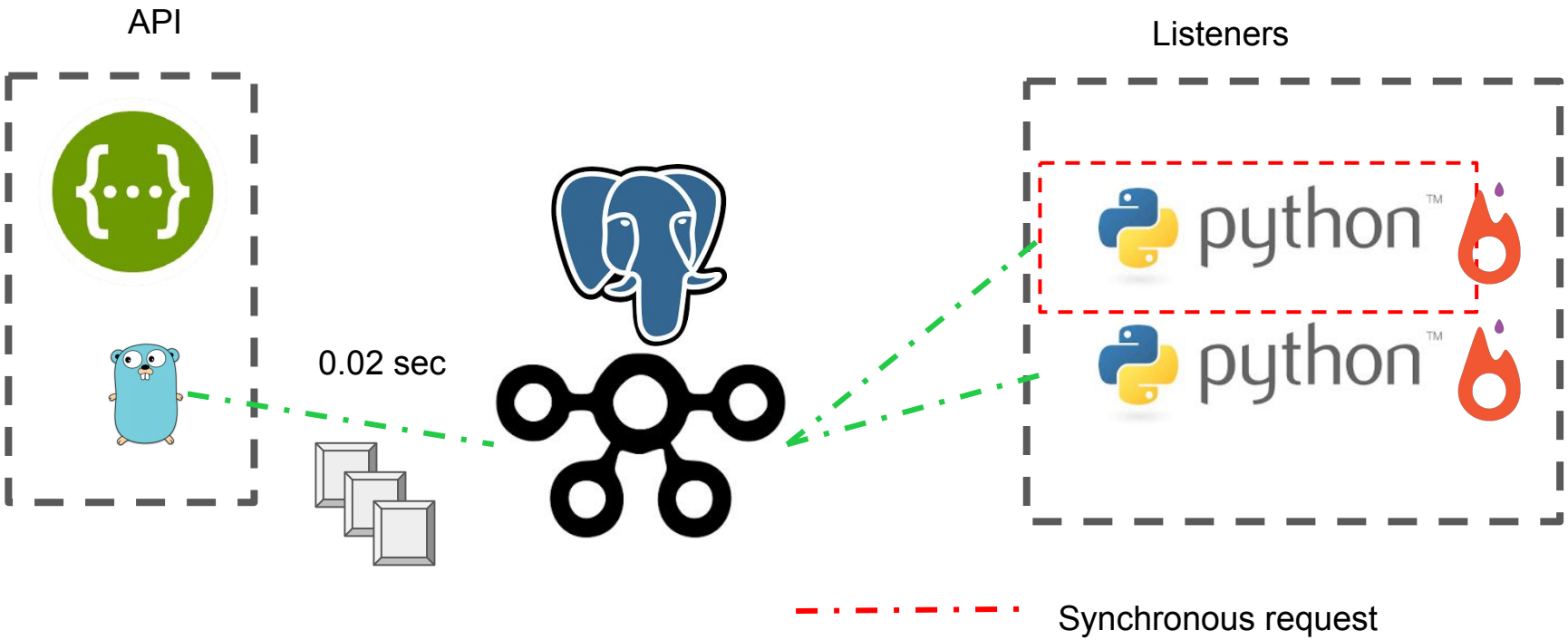
3 seconds per request

lalafo

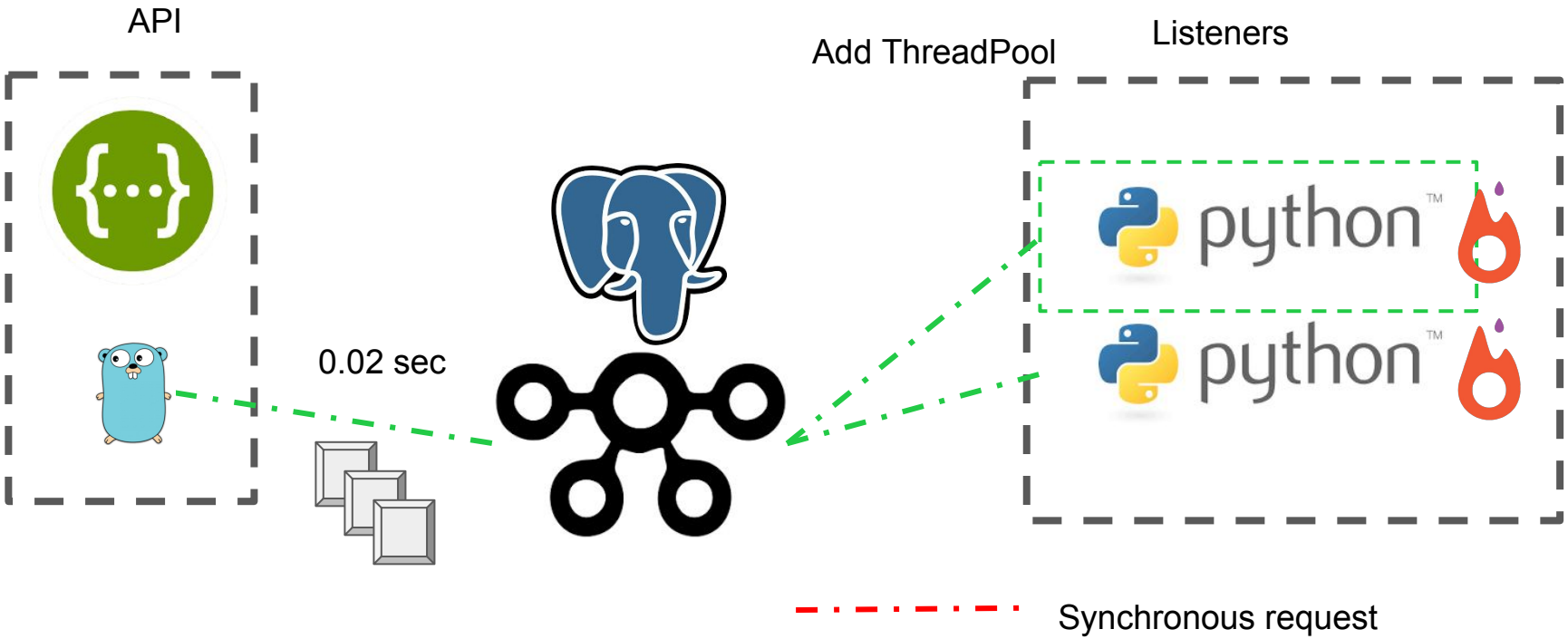
How large is every image



2.4 seconds per request



2.4 seconds per request

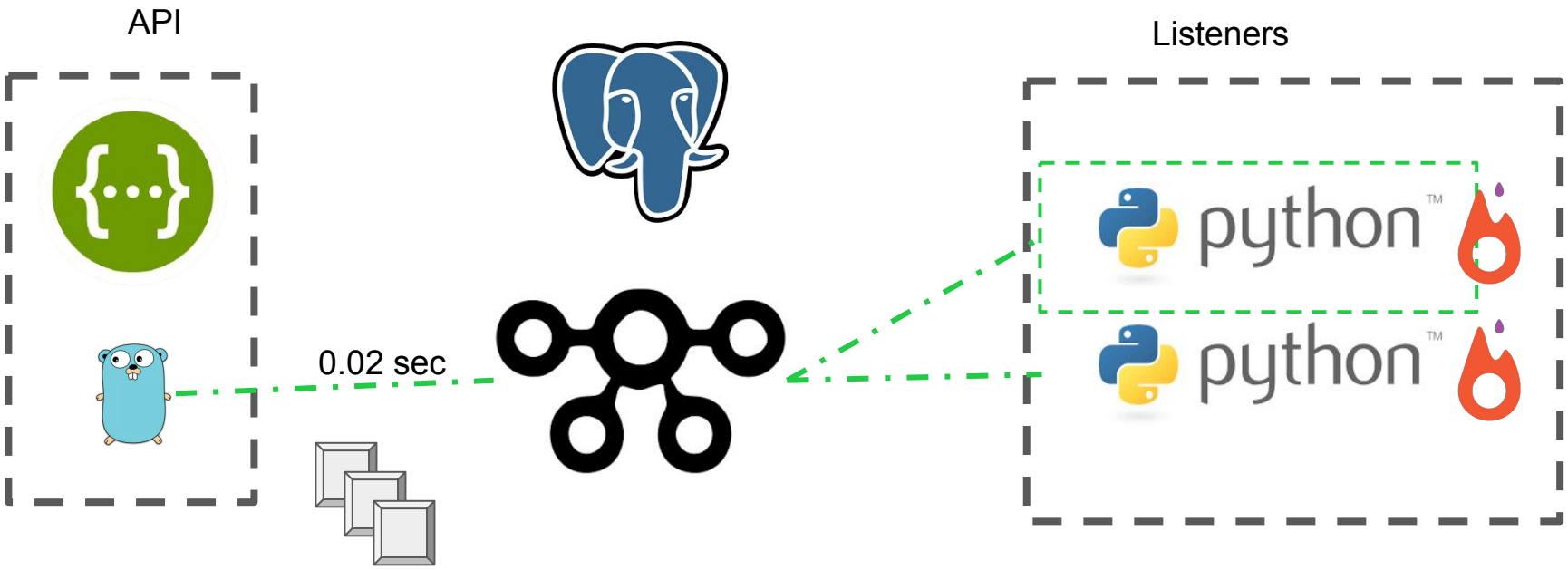


2 seconds per request

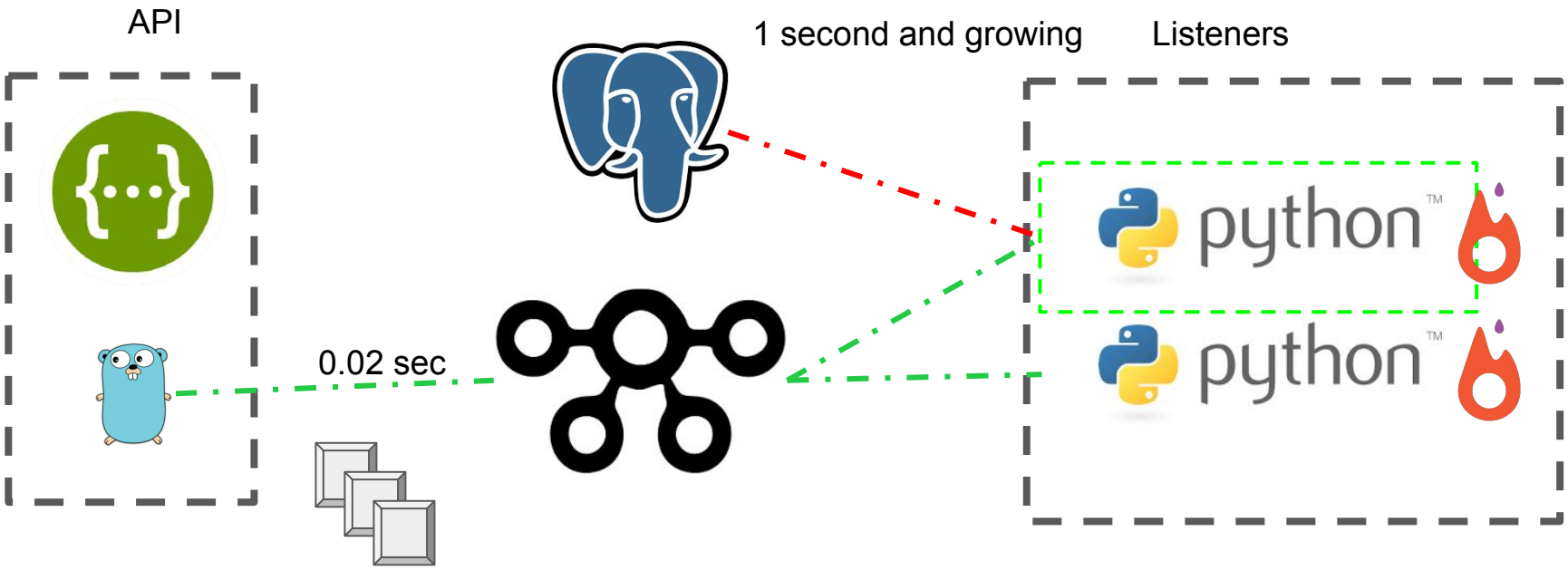
lalafo

Success?
Not yet

2 seconds per request

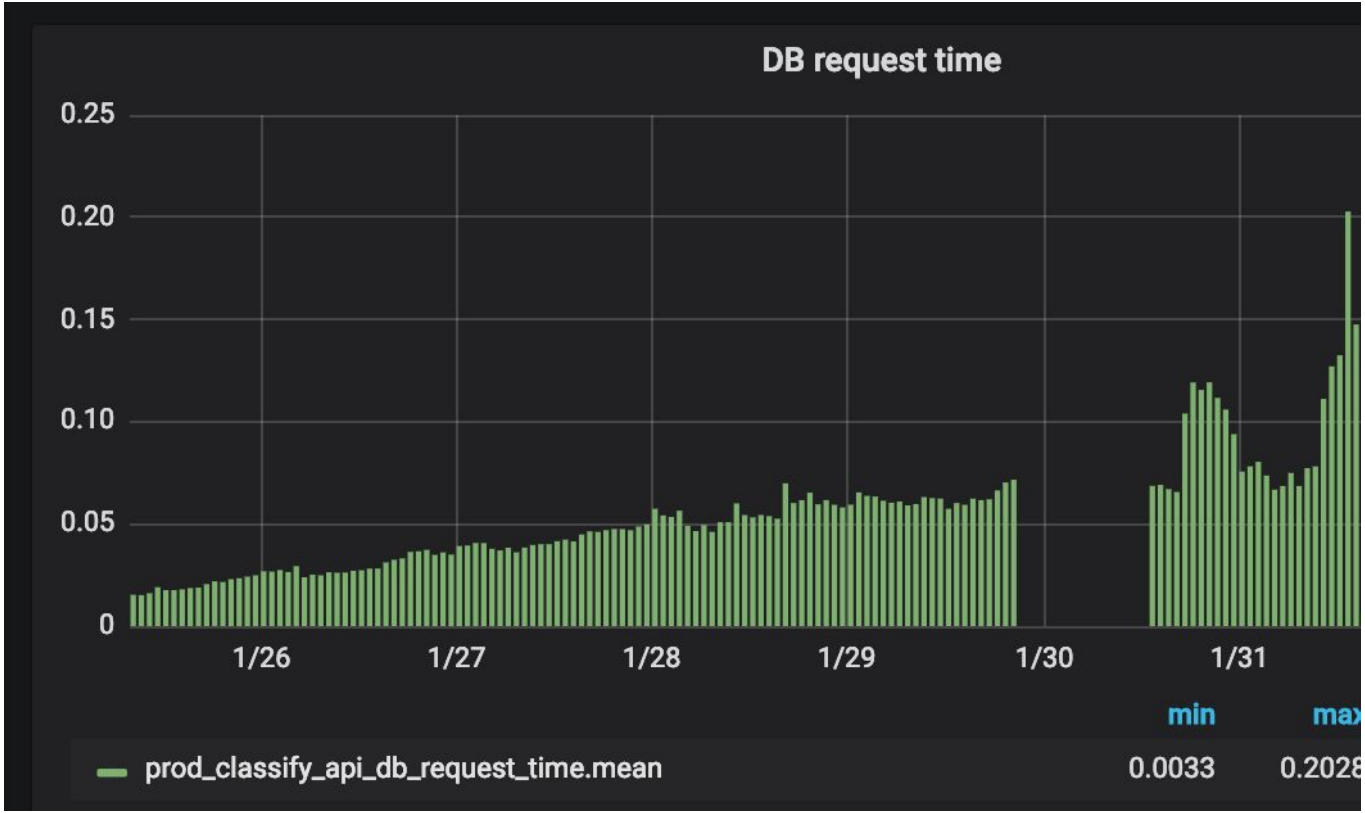


3 seconds per request



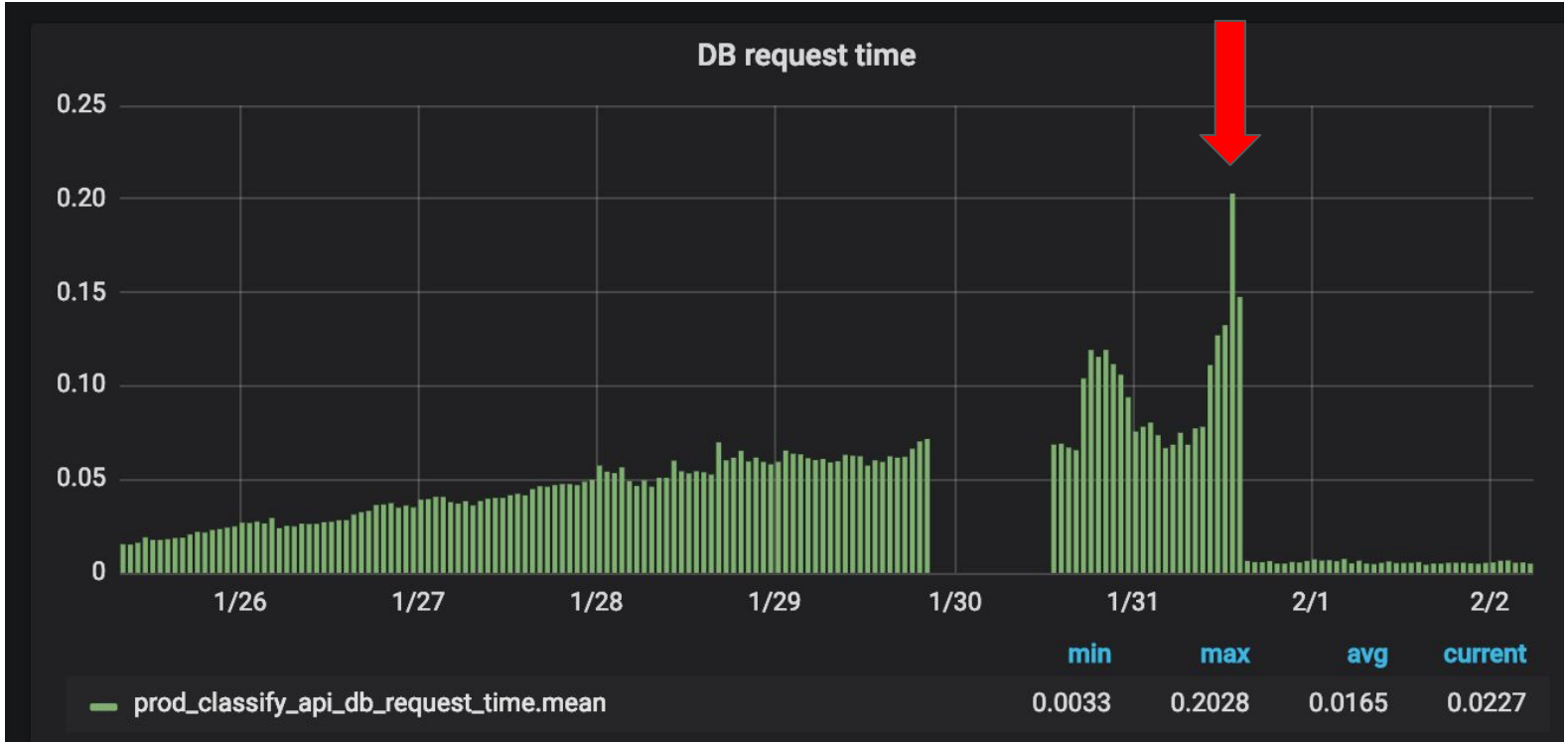
3 seconds per request

lalafo



3 seconds per request

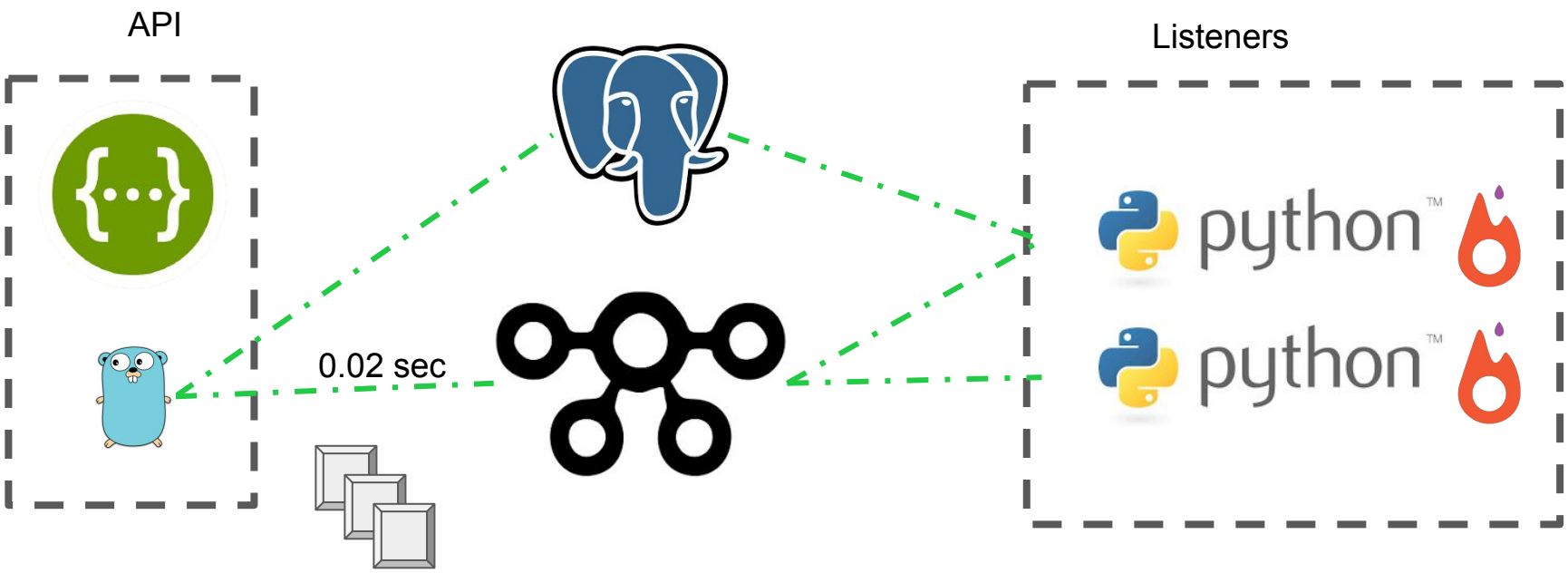
lalafo



2 seconds per request

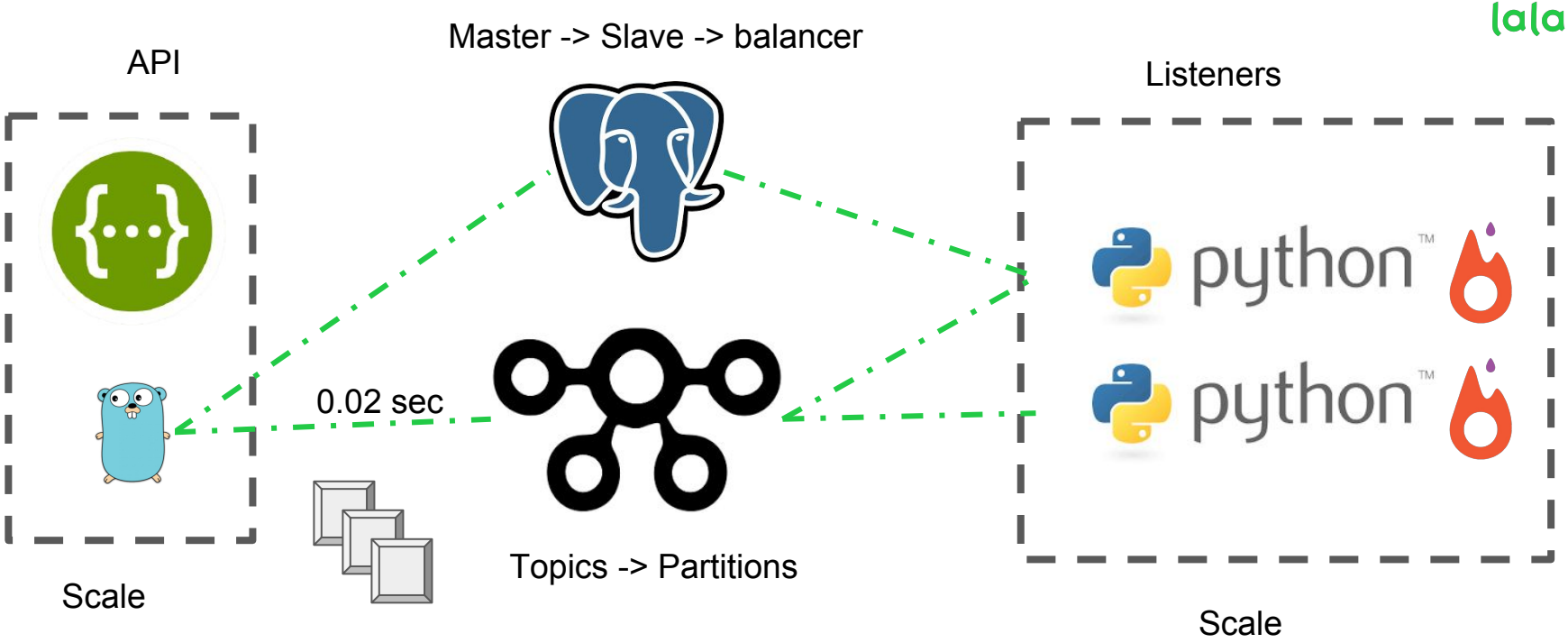
lalafo

¯_(ツ)_/¯



2 seconds per request

200 RPM -> 1200 RPM



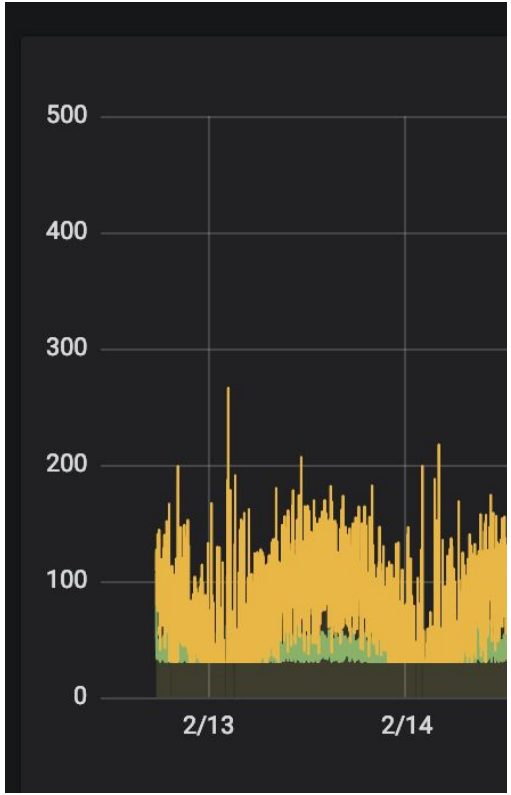
2 seconds per request

lalafo

Success?
Not yet

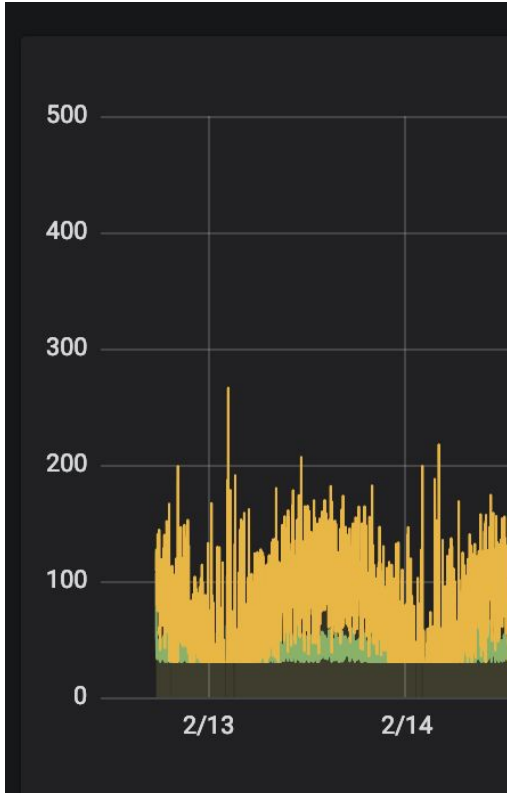
< 1 second per request

lalafo




```
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0     46897      C   /bin/python                             7652MiB    |
```

lalafo

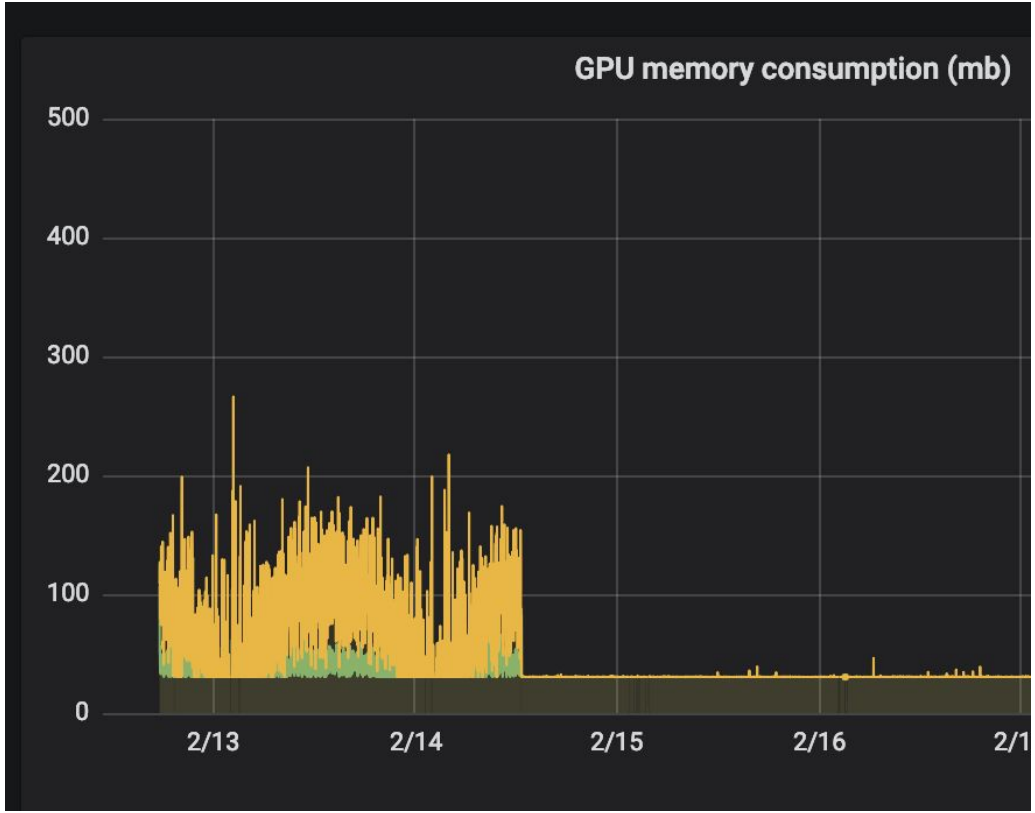


python™

Out of 8Gb

```
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0     46897      C   /bin/python                             7652MiB    |
```

lalafo



How to PyTorch in production

lafo

```
Sat Mar  9 11:43:08 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.39       Driver Version: 418.39       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 1080    Off  | 00000000:01:00.0 Off |                  N/A |
| 46%   57C    P2    40W / 180W |   1294MiB /  8119MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0     15795      C   ...it- classify -p 6066 /app/classify-data   689MiB |
|    0   1546563      C   python3                                      595MiB |
+-----------------------------------------------------------------------------+
```

7.6 Gb -> 600 Mb

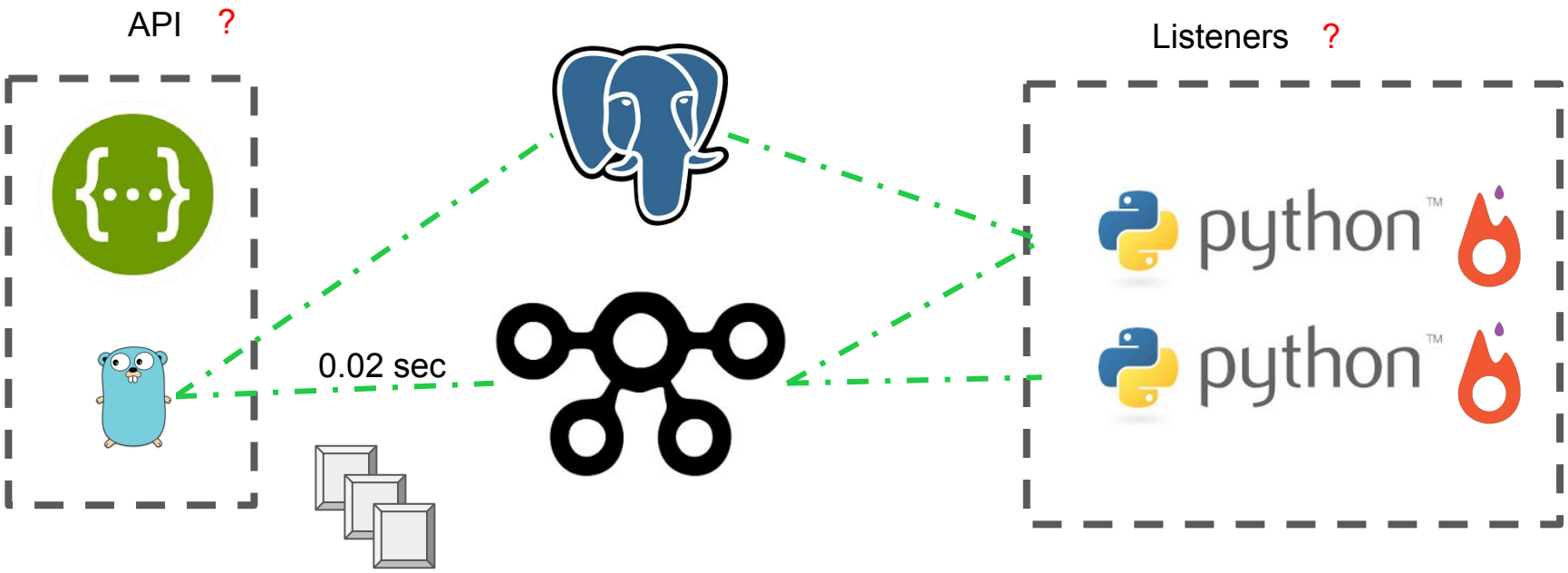
lalafo

Success?
Not yet

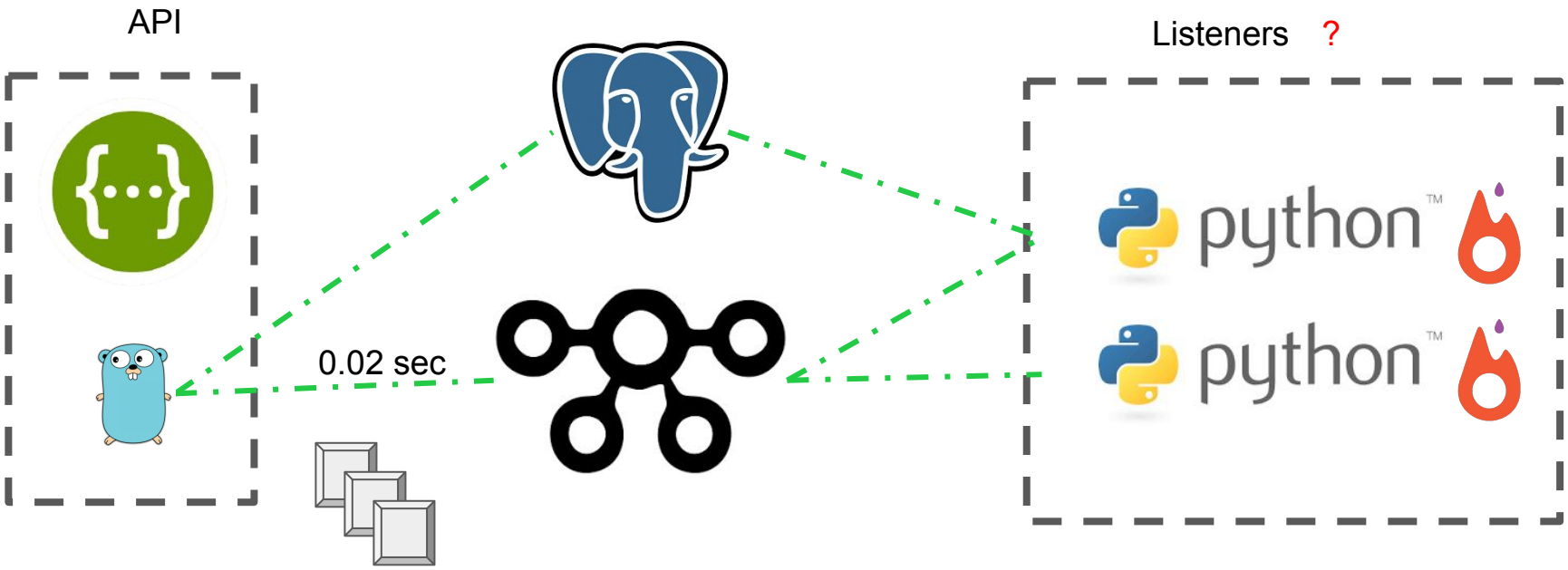
1 second per request

lalafo

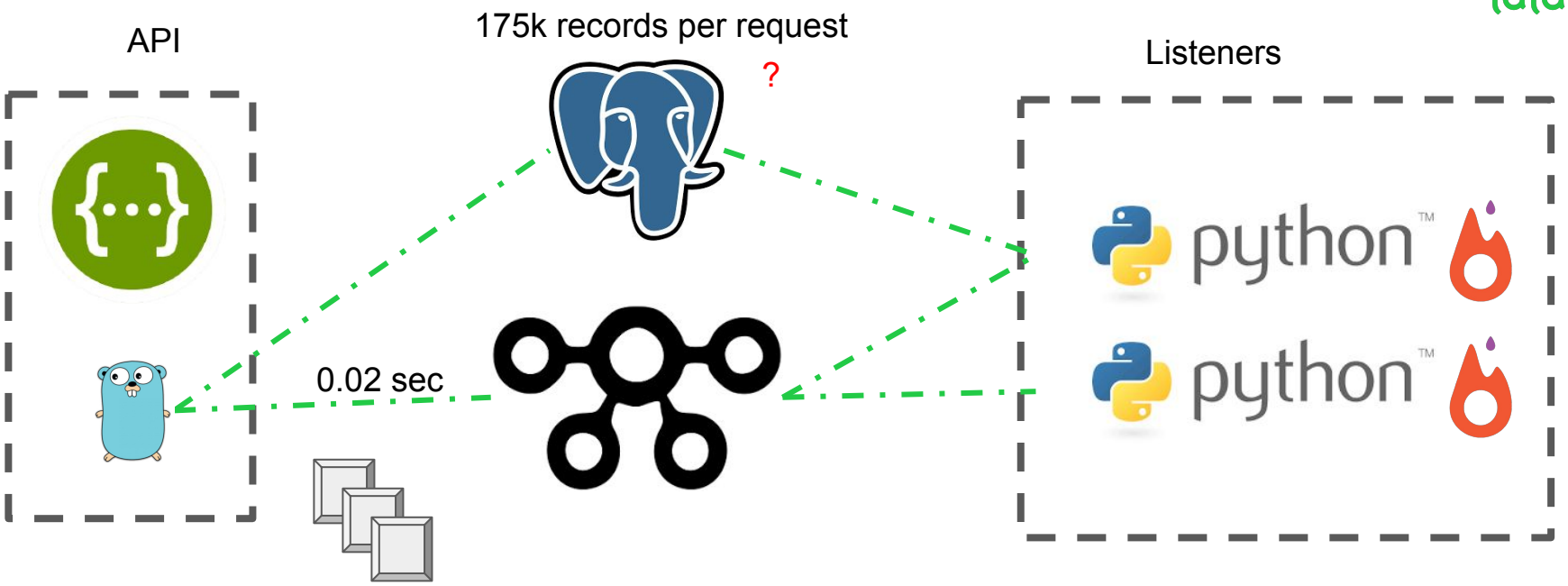
Back to 2, 5, 10 seconds per request



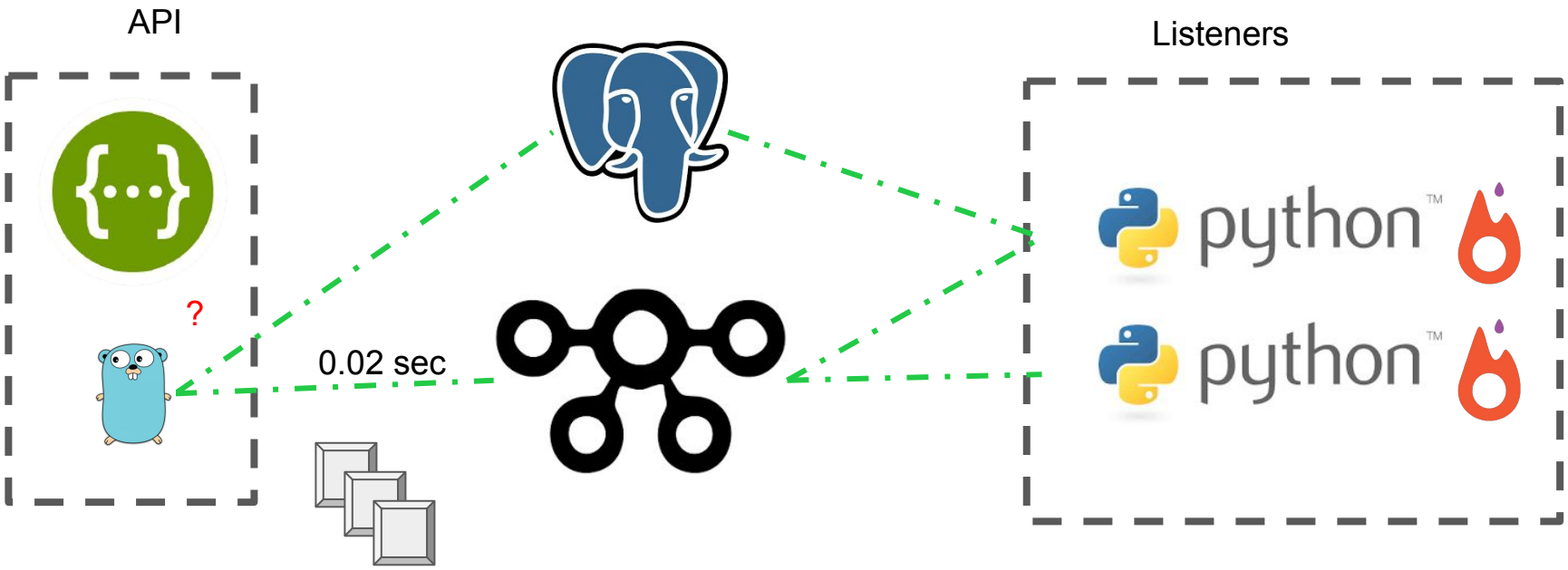
Troubleshooting



Troubleshooting



Troubleshooting



Troubleshooting

API

lalafo

```
-       err := db.FirstOrCreate(&ad, Ad{CatalogAdID: ad.CatalogAdID, CatalogName: ad.CatalogName}).Error
+       ad = Ad{CatalogAdID: ad.CatalogAdID, CatalogName: ad.CatalogName}
+       err := db.Create(&ad).Error
        influxTags := map[string]string{"action": "/v4/classify/add_images", "catalog": ad.CatalogName}
        iw.WritePoint(dbRequestTime, influxTags, time.Since(funcStart).Seconds())

        if err != nil {
-               log.Errorf("couldn't save ad to the DB: %v\n  Ad: %v\n", err, ad)
+               if pqErr, _ := err.(*pq.Error); pqErr.Code == "23505" { // Retry once for integrity errors
+                       err = db.First(&ad).Error
+               }
+               log.Errorf("couldn't save ad to the DB: %v\n Ad: %v \n", err, ad)
                c.JSON(http.StatusInternalServerError, LegacyApiError{Status: "Error", Message: "Could not
```

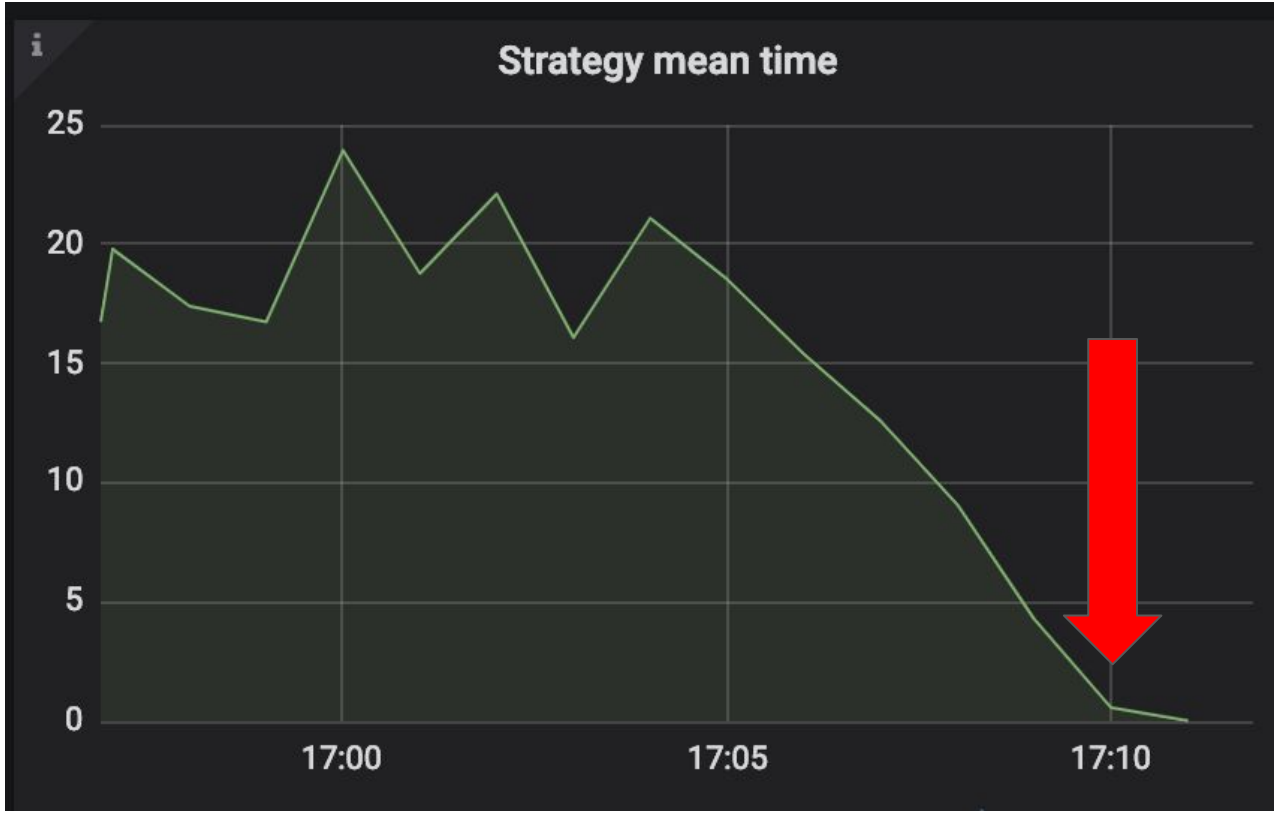
Troubleshooting

lalafo

```
-       err := db.FirstOrCreate(&ad, Ad{CatalogAdID: ad.CatalogAdID, CatalogName: ad.CatalogName}).Error
+       ad = Ad{CatalogAdID: ad.CatalogAdID, CatalogName: ad.CatalogName}
+       err := db.Create(&ad).Error
        influxTags := map[string]string{"action": "/v4/classify/add_images", "catalog": ad.CatalogName}
        iw.WritePoint(dbRequestTime, influxTags, time.Since(funcStart).Seconds())

        if err != nil {
-               log.Errorf("couldn't save ad to the DB: %v\n  Ad: %v\n", err, ad)
+               if pqErr, _ := err.(*pq.Error); pqErr.Code == "23505" { // Retry once for integrity errors
+                       err = db.First(&ad).Error
+               }
+               log.Errorf("couldn't save ad to the DB: %v\n Ad: %v \n", err, ad)
                c.JSON(http.StatusInternalServerError, LegacyApiError{Status: "Error", Message: "Could not
```

```
\"uix_catalog_name_ad_id\"" {
-                       err = db.First(&ad).Error
+                       err = db.First(&ad, &Ad{CatalogName: ad.CatalogName, CatalogAdID:
ad.CatalogAdID}).Error
```

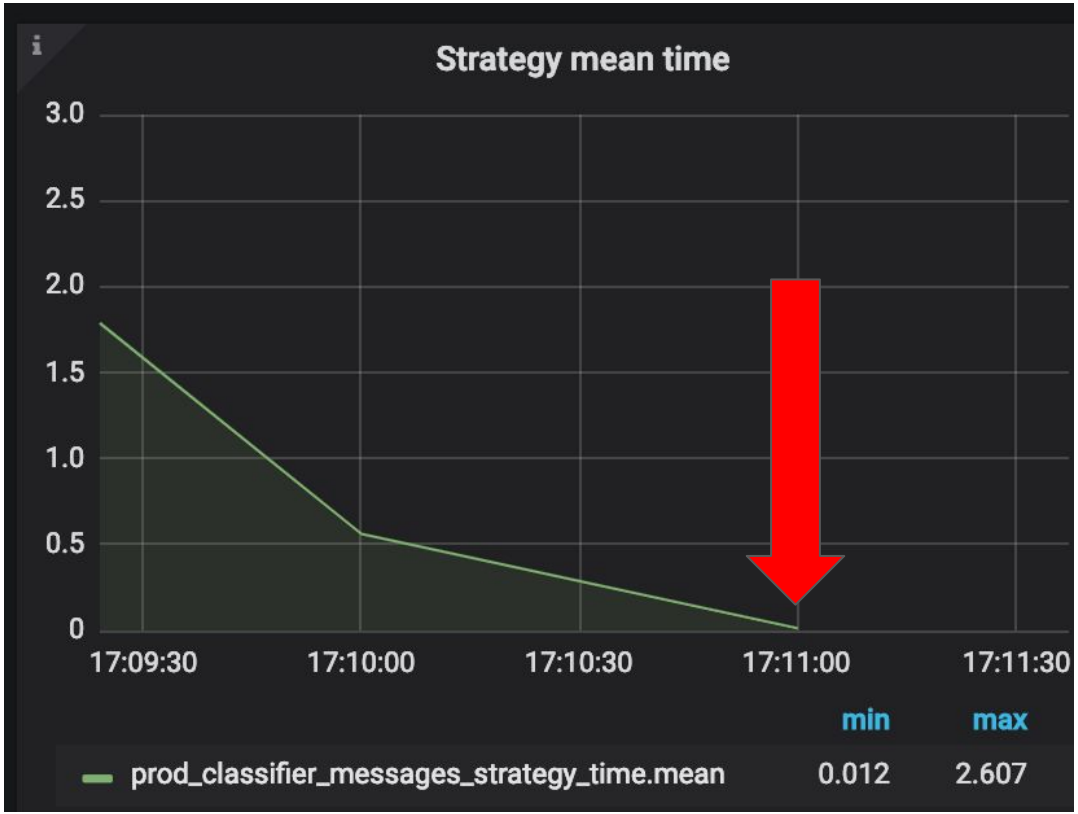
SELECT * FROM ads LIMIT 1



lalafo

25 seconds -> 0.02 seconds

lalafo



0.02 seconds -> 0.002 second

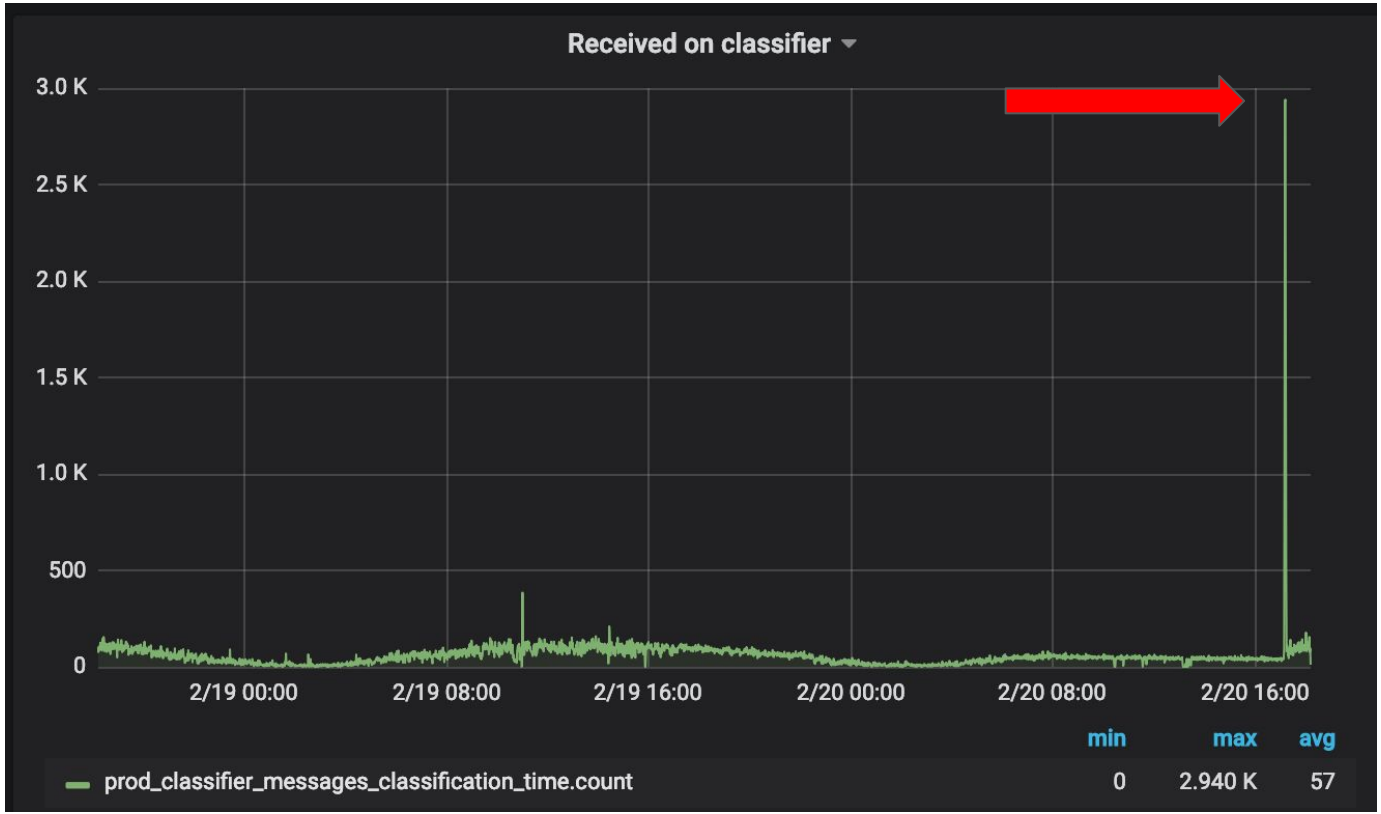
lalafo

Success?
Not yet

lalafo

https://github.com/robinhood/faust

Faust 1.4.6: No latest offset



3K images per minute 0_o

lalafo

Success?
Not yet

lalafo

Issues to solve

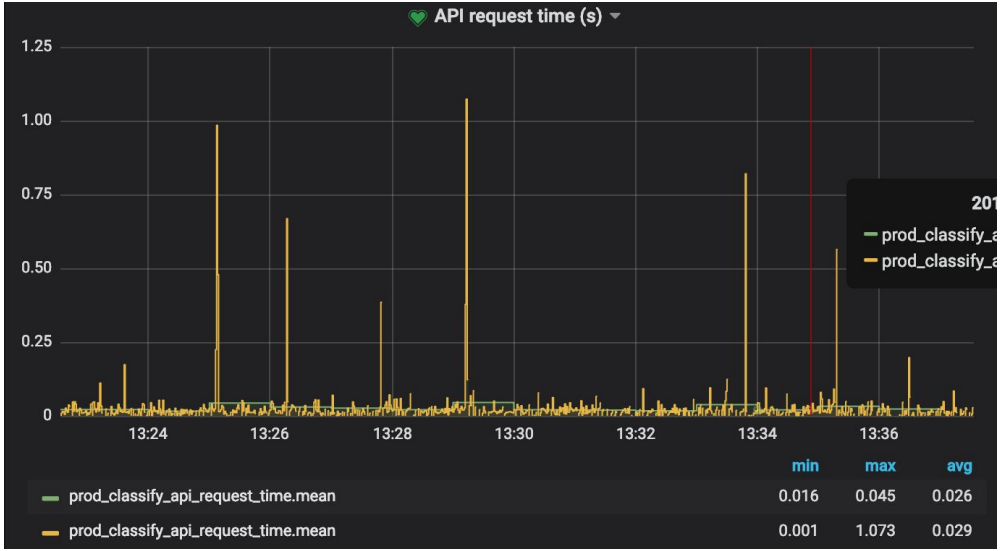
1. Occasional spikes in performance (GC, network latency)
2. Message broker (Kafka rebalancing, offset etc)
3. How to handle DB migrations
4. Something we are not aware of yet

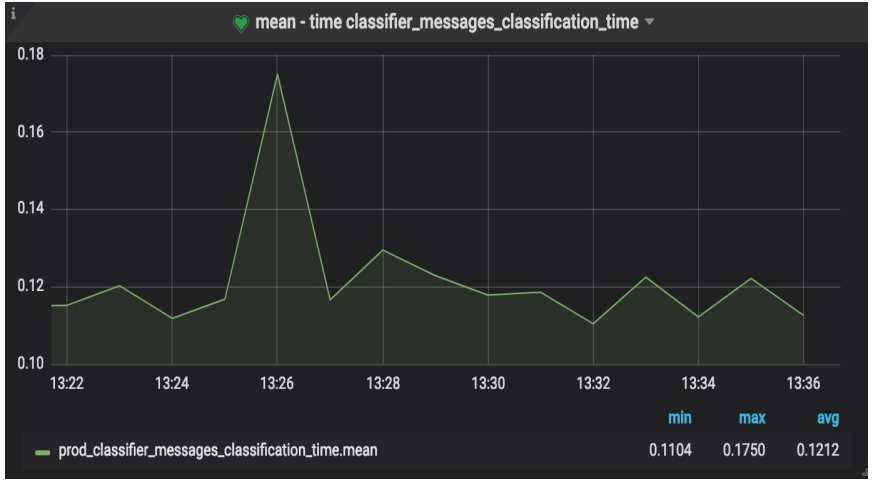
lalafo

Lessons learnt

1. CPU bound tasks != IO bound (¯_(ツ)_/¯)
2. High coupling - low cohesion
3. You need to know how to cook MongoDB
4. Go is not that obvious and library reach as Python
5. Simple != Easier
6. Concurrency != Parallelism (obviously)

lalafo

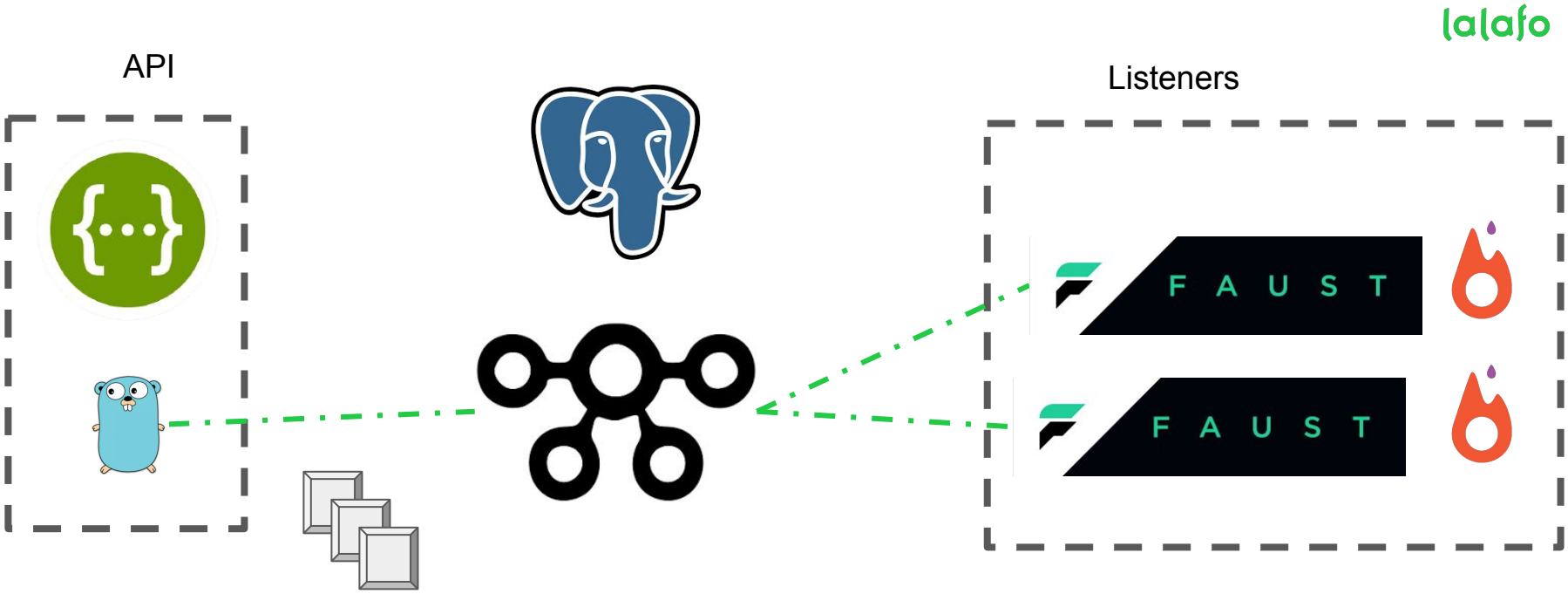




min	max	avg
0.015	0.045	0.027
0.001	1.073	0.029

min	max	avg
0.1025	0.1750	0.1208

< 1 second per request



+ Live statistics from PostgreSQL

lalafo

Success?

Thank you everyone