HTTPq: DDoS-as-a-Service

How MessageBird delivers millions of webhooks daily

Disclaimer

MessageBird **condemns** any malicious attack (DDoS or not) and works together both with customers and engineers everywhere to make sure security and operations are taken **seriously**. Any situations presented here are anecdotal scenarios gathered across multiple years operating a system designed to perform as fast as it can for customers who need as many webhooks as they’d need.

About me

Brazilian bird 🇧🇷

(slightly obsessed) cat person 🐈‍⬛

On the (tech) road for 17 years 🦉

Tech geek since forever 👾

Business Infra Team Lead @ MessageBird 🐦

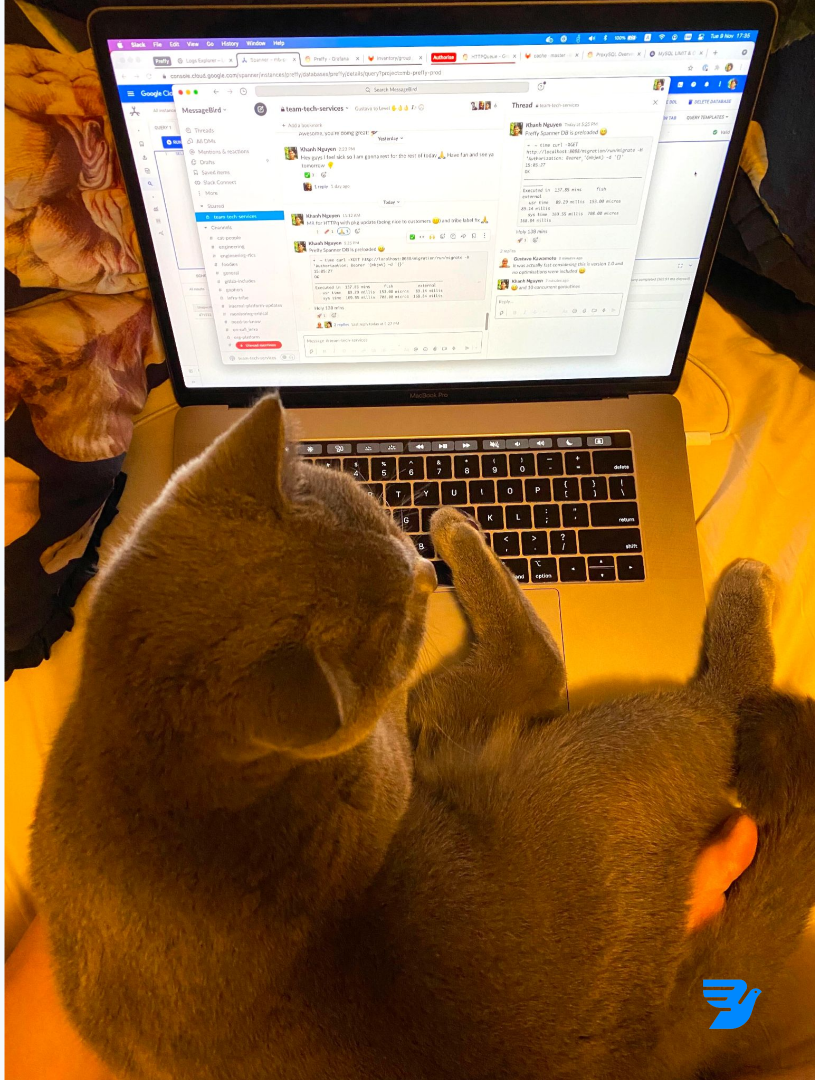
Business Infra

Manages product infrastructure services

Authentication, queues, preferences

Flamingos and cats everywhere

🇧🇷🇳🇱🇻🇳+🇫🇷



About MessageBird

Communications Platform as a Service (CPaaS)

Global scale with big customers

A lot of challenges scaling and innovating

Fun place to work (we’re hiring!)

Let’s talk about HTTPq

```go
// RunWorker is responsible for fetching a job from beanstalk and passing that
// job to handleJob().
func RunWorker() {
    for {
        handleJob(<-consumerPool.C)
    }
}

// handleJob extracts the Request ID from the job's JSON data, retrieves the
// Request from the database and tries to perform an HTTP(S) request.
func handleJob(job *beanstalk.Job) {
    var (
        request    *httpqueue.Request
        requestJob beanstalkRequest
    )

    // Extract the ID from the job's JSON data and fetch the Request object.
    err := json.Unmarshal(job.Body, &requestJob)
    if err == nil {
        request, err = httpqueue.RequestWithID(requestJob.ID)
    }
    if err != nil {
        log.Printf("Unable to fetch request for beanstalk job %d: %s", job.ID, err)
        if err := job.Bury(); err != nil {
            log.Printf("Unable to bury beanstalk job %d: %s", job.ID, err)
        }

        return
    }

    Debug.Printf("%d: Worker parsing request", request.ID)

    // If the request is ready, perform the actual HTTP request.
    if request.IsReady() {
        Debug.Printf("%d: Worker performing request", request.ID)

        if err := request.Do(); err != nil && request.Status == httpqueue.StatusFailed {
            log.Printf("Request %d failed after %d attempts: %s", request.ID, request.Attempts, err)
        }

        if err := request.Save(); err != nil {
            log.Printf("Unable to save request %d: %s", request.ID, err)
        }
    }

    // If the current state requires a callback, do it here.
    if request.ShouldCallback() {
        Debug.Printf("%d: Worker performing callback for status '%s'", request.ID, request.Status)

        if err := request.DoCallback(producerPool); err != nil {
            if request.IsFinished() {
                log.Printf("Request %d was unable to perform final callback: %s", request.ID, err)
                if err := job.Bury(); err != nil {
                    log.Printf("Unable to bury beanstalk job %d: %s", job.ID, err)
                }

                return
            }
        }

        if err := request.Save(); err != nil {
            log.Printf("Unable to save request %d: %s", request.ID, err)
        }
    }

    // If there is no more work for this request, delete the job.
    if request.IsFinished() && !request.ShouldCallback() {
        Debug.Printf("%d: Worker deleting beanstalk job", request.ID)
```

What is HTTPq?

Developed in 2016 by our dinosaurs

Queue for webhooks written in Go

Connects MessageBird services & customers

Deliver consistently across multiple clients

MessageBird delivers a lot of webhooks daily

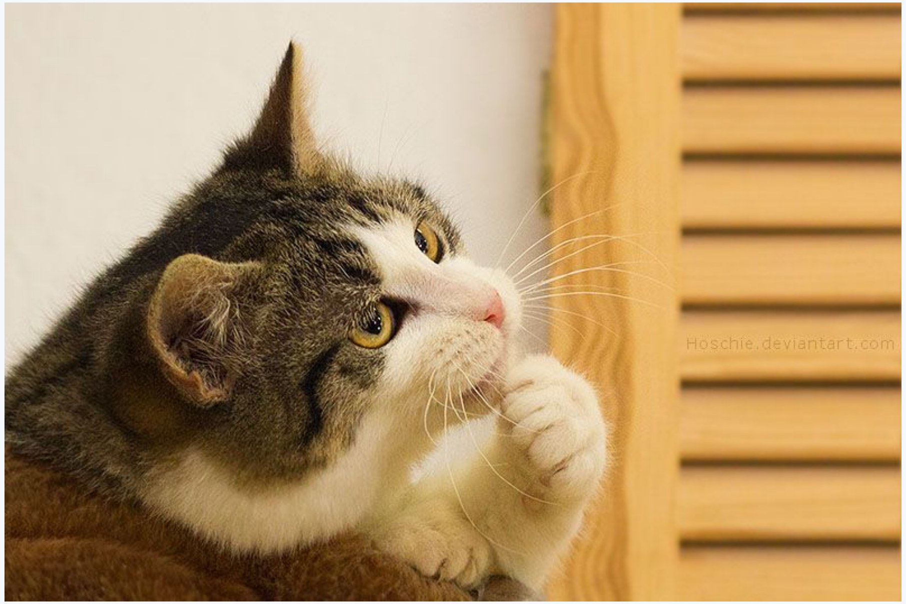
Question: how many webhooks a day on average?

a) Tens of thousands

b) Hundreds of thousands

c) Tens of millions

d) Hundreds of millions



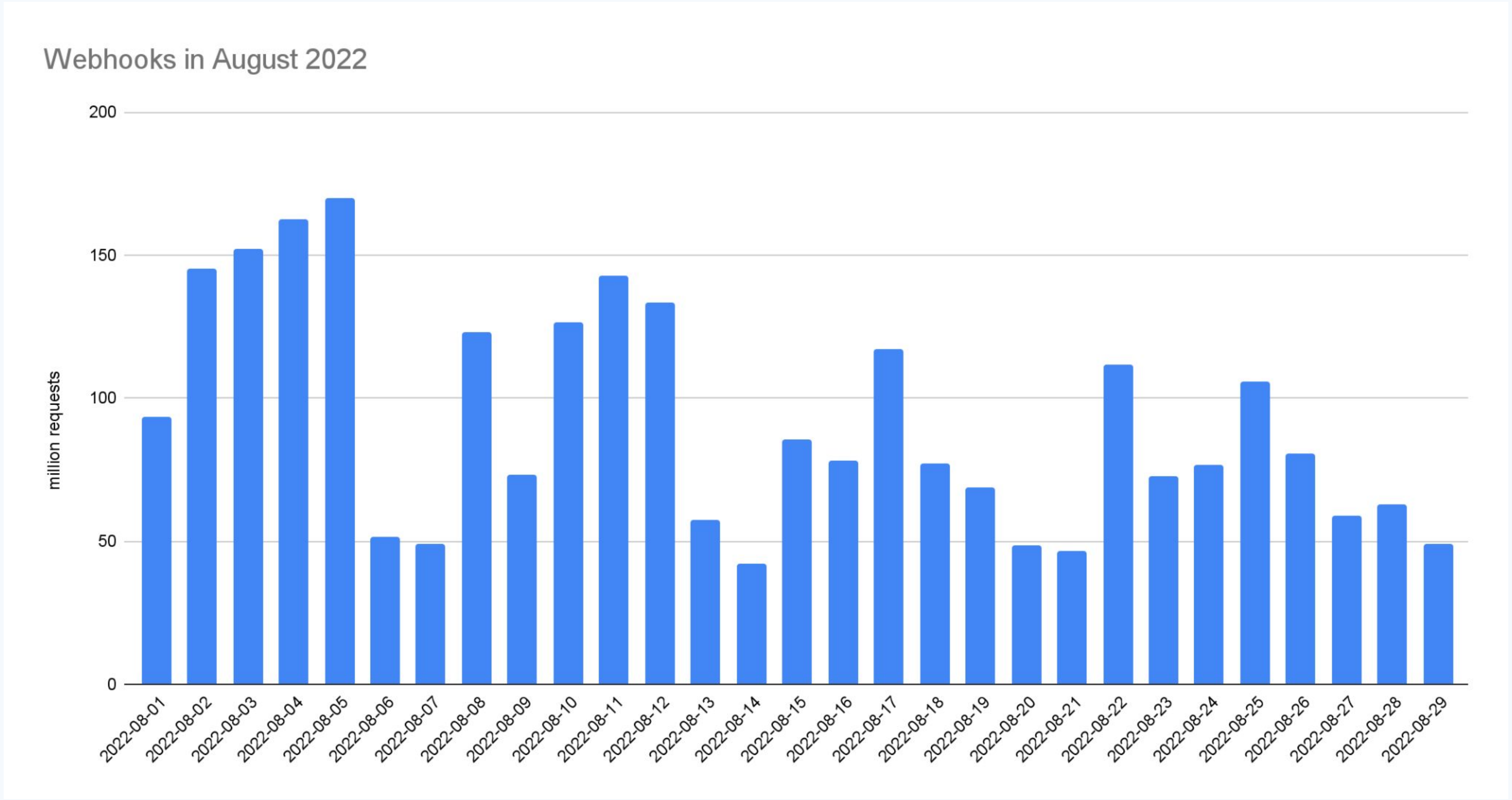
Question: how many webhooks a day on average?

a) Tens of thousands

b) Hundreds of thousands

c) Tens of millions

d) Hundreds of millions ✅


Webhooks in August 2022
200
150
100
50
0
million requests
2022-08-01
2022-08-02
2022-08-03
2022-08-04
2022-08-05
2022-08-06
2022-08-07
2022-08-08
2022-08-09
2022-08-10
2022-08-11
2022-08-12
2022-08-13
2022-08-14
2022-08-15
2022-08-16
2022-08-17
2022-08-18
2022-08-19
2022-08-20
2022-08-21
2022-08-22
2022-08-23
2022-08-24
2022-08-25
2022-08-26
2022-08-27
2022-08-28
2022-08-29

Breaking down HTTPq

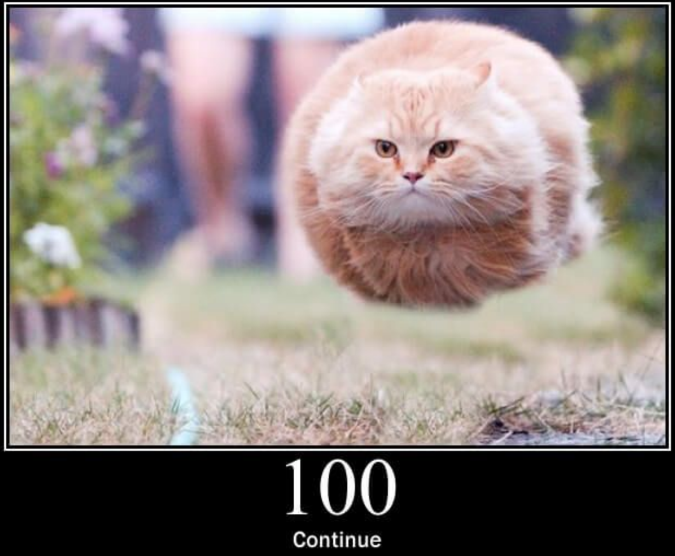
Breaking down HTTPq

Self-contained single binary

KISS principle

Secure by default

Three main components in the service



Breaking down HTTPq: API

HTTP and GRPC APIs available

Single endpoint which users POST requests

Runs on Go’s standard HTTP mux

Globally available (thanks to Google GLB)

Horizontal Pod Autoscaling to adapt to traffic

99.95% submissions under 250ms internal SLO

Breaking down HTTPq: Beanstalk

Open source queue manager

Simple protocol

Delaying and priority commands

Fast, single-threaded process

Nice open source Go client written by one of our birds

https://github.com/beanstalkd/beanstalkd
https://github.com/prep/beanstalk

Breaking down HTTPq: Worker

Configurable number of workers

Handles the heavy lifting for webhooks

Main source of our issues in the past

Socket ports hungry

HPA to adapt to traffic

99.95% deliveries under 500ms internal SLO

(measured by the moment the webhook request comes in)

Lifecycle of a webhook

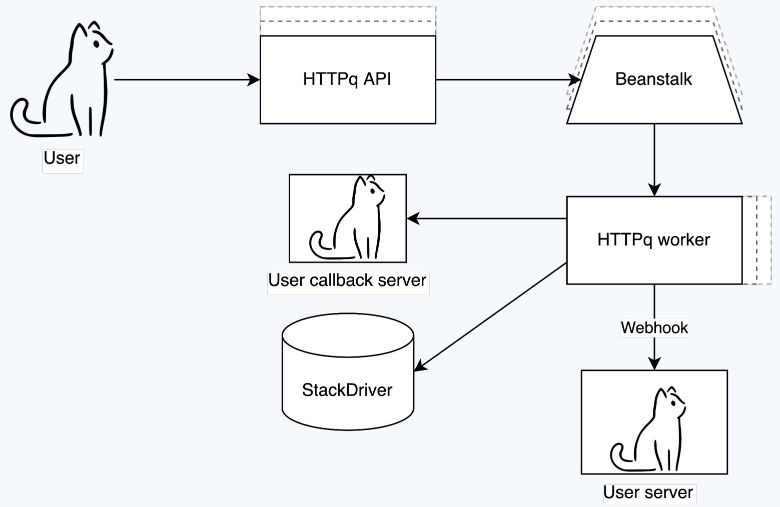
Schedule webhook deliveries using HTTPq's API

Webhooks are queued on beanstalk

Workers pick up pending webhook requests from beanstalk

Request arrives to target server

(Optional) Callback request is queued



Operating & maintaining HTTPq

Scaling HTTPq

Horizontal pod autoscaling

Multi-region installation

Global load balancing

Dedicated nodes

Finding the resource quotas sweet spot

HTTP2 client issues

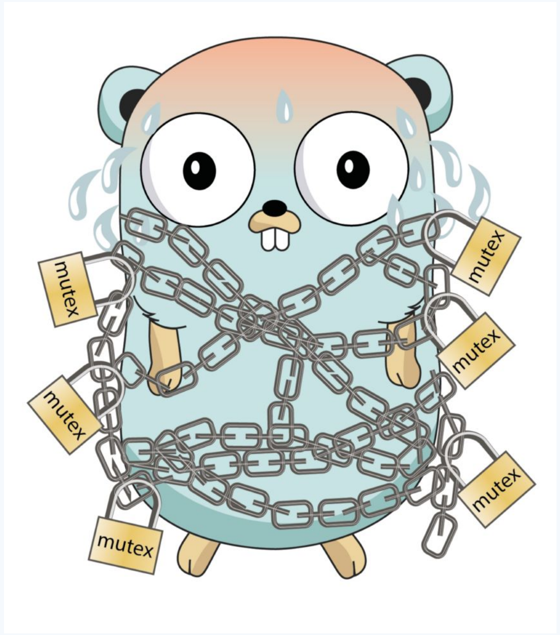
Weird behaviour since go1.9 – up until go1.17

Caused process hugging

Lots of sleepless nights

https://github.com/golang/go/issues/32388

Small patch to fix that



No transactional databases! Why?

Scaling is painful

Multi-region database can be expensive…

… and might not work as expected

In the end, transactional databases are not really needed

Embrace audit logging

Structured logging

Logs are non-blocking

Cloud Logging is your friend...

... but beware of costs!

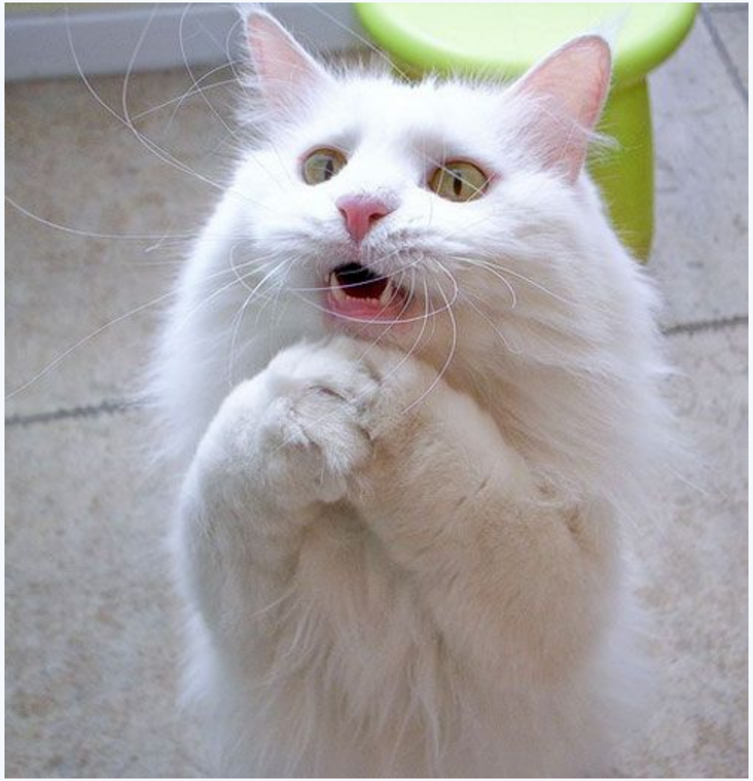
Question: how much it costs per month?

a) Hundreds of thousands euros

b) Tens of thousands euros

c) A few thousands euros

d) Less than a thousand euros



Question: how much it costs per month?

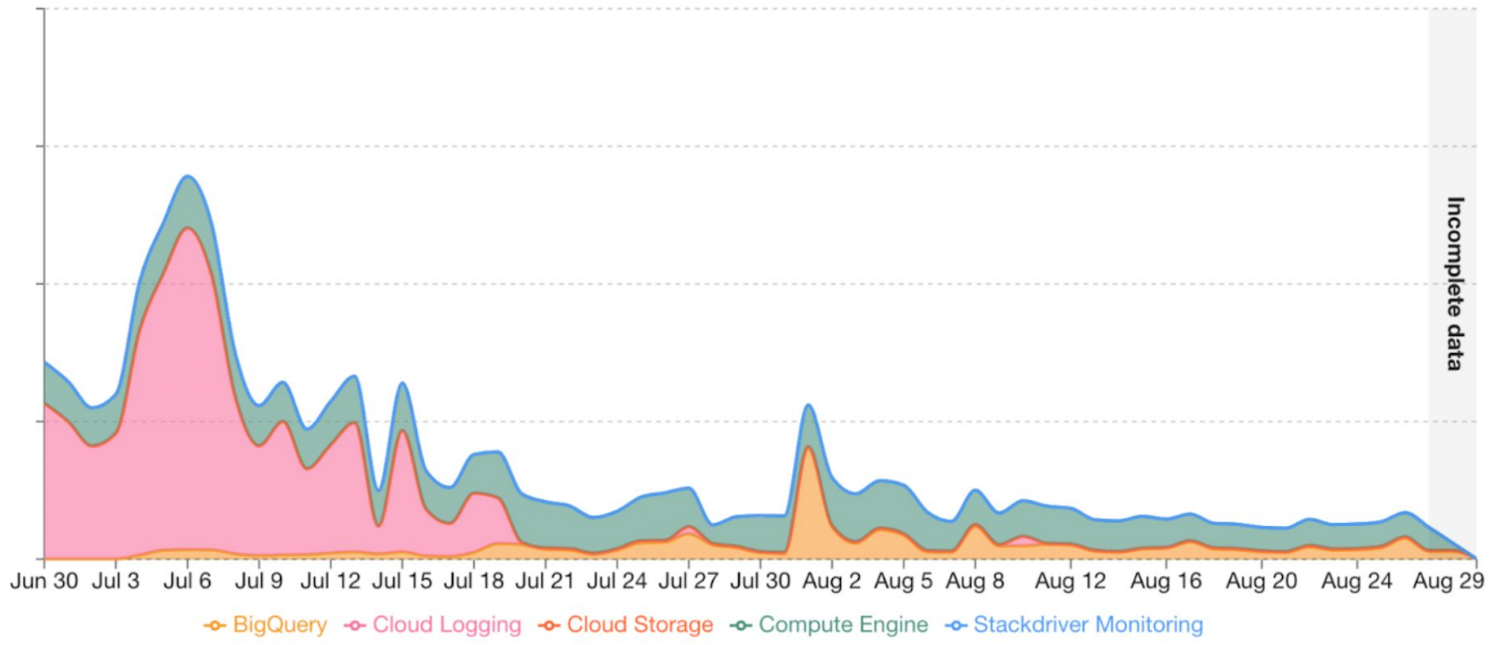
a) Hundreds of thousands euros

b) Tens of thousands euros

c) A few thousands euros ✅

d) Less than a thousand euros

Costs in the last 60 days



Conclusion/main takeaways

Design a system to solve your current problems

Invest time and money scaling when needed

Services doesn’t need to be expensive to do a lot

Issues exists everywhere
(even in Golang’s codebase)

Be careful what you wish for!

Q&A

Ask us anything

Thanks for your attention!