Migrating Ingress Controllers: The Six-Month Journey

Martin Beránek - ShipMonk's Traefik evangelist

shipmonk

Introduction

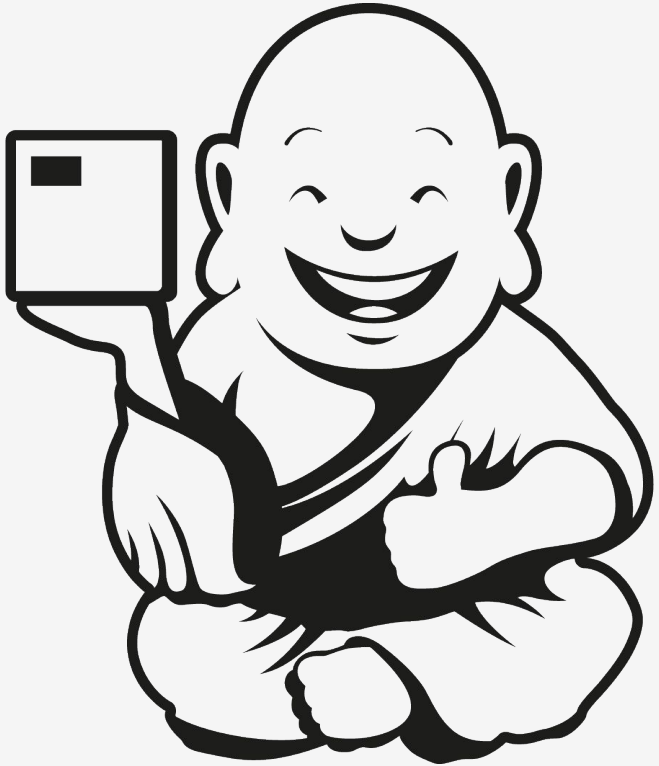
shipmonk

shipmonk

The history of ingress setup

Introduction

- ~2019 introduction of K8s instead of DO droplets
- Routing done through AWS ALB
  - Used AWS Cognito for private services
- Later changed to NLB with Nginx
- It worked!
  - Minor issues with latencies
  - Configs in configmap
  - Access logs - disabled after a while



Retrospective

Introduction

- NGINX does not offer [good] metrics
  - NGINX plus does
- High utilization under high connection count
  - Caused outage during the peak
  - No action item, nothing in logs
- Upstream setup causing unknown issues
  - Which was there cause of http 503

shipmonk

Service Upstream

By default the Ingress-Nginx Controller uses a list of all endpoints (Pod IP/port) in the NGINX upstream configuration.

The `nginx.ingress.kubernetes.io/service-upstream` annotation disables that behavior and instead uses a single upstream in NGINX, the service's Cluster IP and port.

This can be desirable for things like zero-downtime deployments . See issue #257.

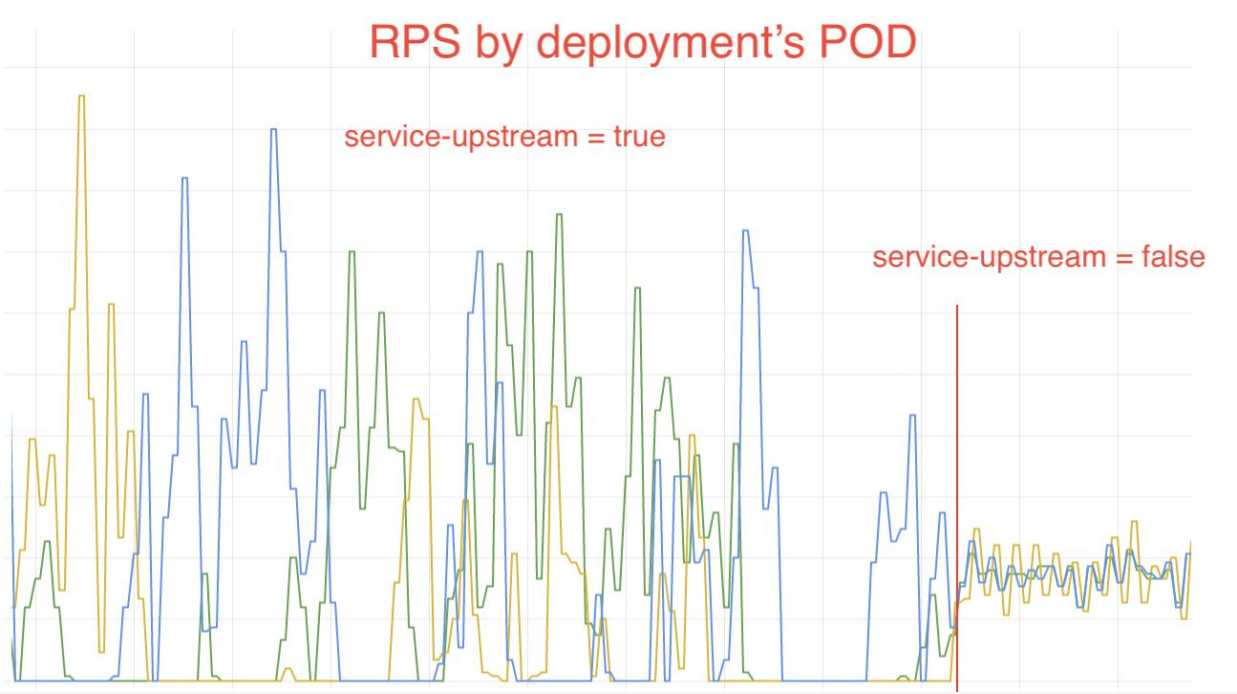
Known Issues

If the `service-upstream` annotation is specified the following things should be taken into consideration:

- Sticky Sessions will not work as only round-robin load balancing is supported.
- The `proxy_next_upstream` directive will not have any effect meaning on error the request will not be dispatched to another upstream.

From docu

shipmonk



Issue #257, @narqo comment

Upstream setup causing unknown issues

Introduction

shipmonk

- To avoid HTTP 503 we switched service-upstream=true
- After 2-3 months, we noticed that backends were not utilized evenly
- Overall making HPA unusable and creating more HTTP 503

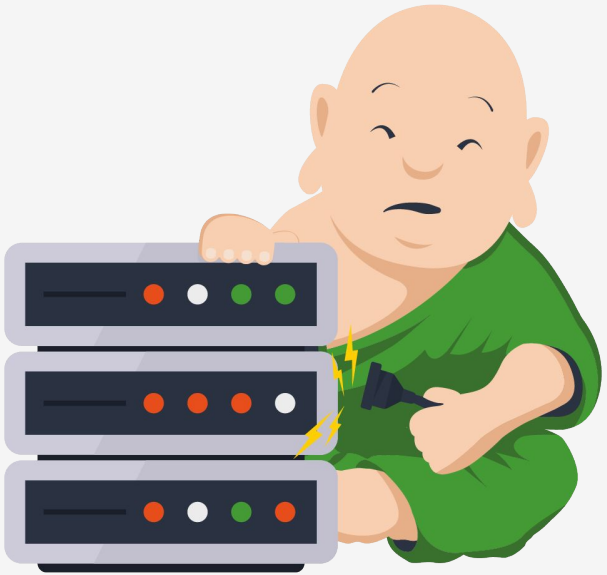
```HTTP
503 Service Unavailable
```

shipmonk

Requirements

Introduction

- Use a similar setup, no new CRDs
- Have the same setup on production and dev
- Use something which the team already partially knows
- It has to have [at least] metrics in Prometheus
- Be stable



shipmonk

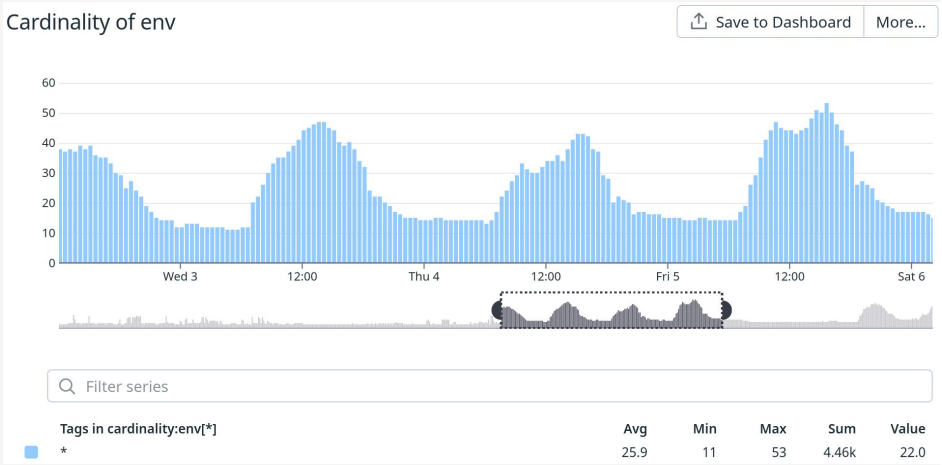
Devs

Introduction

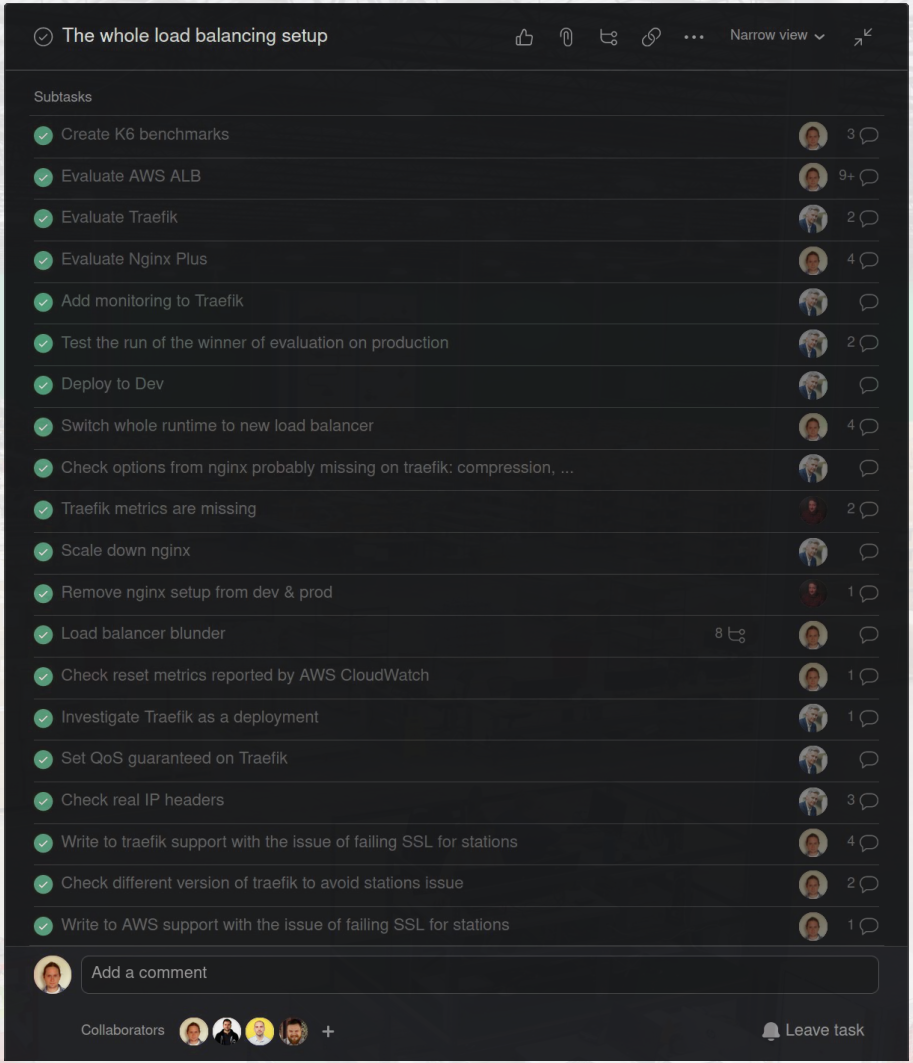
- Anyone can deploy from their own MR
- Environments live for four hours
  - Can be resurrected by redeploy
- Each environment contain about 8 ingresses
  - Overall 32 rules

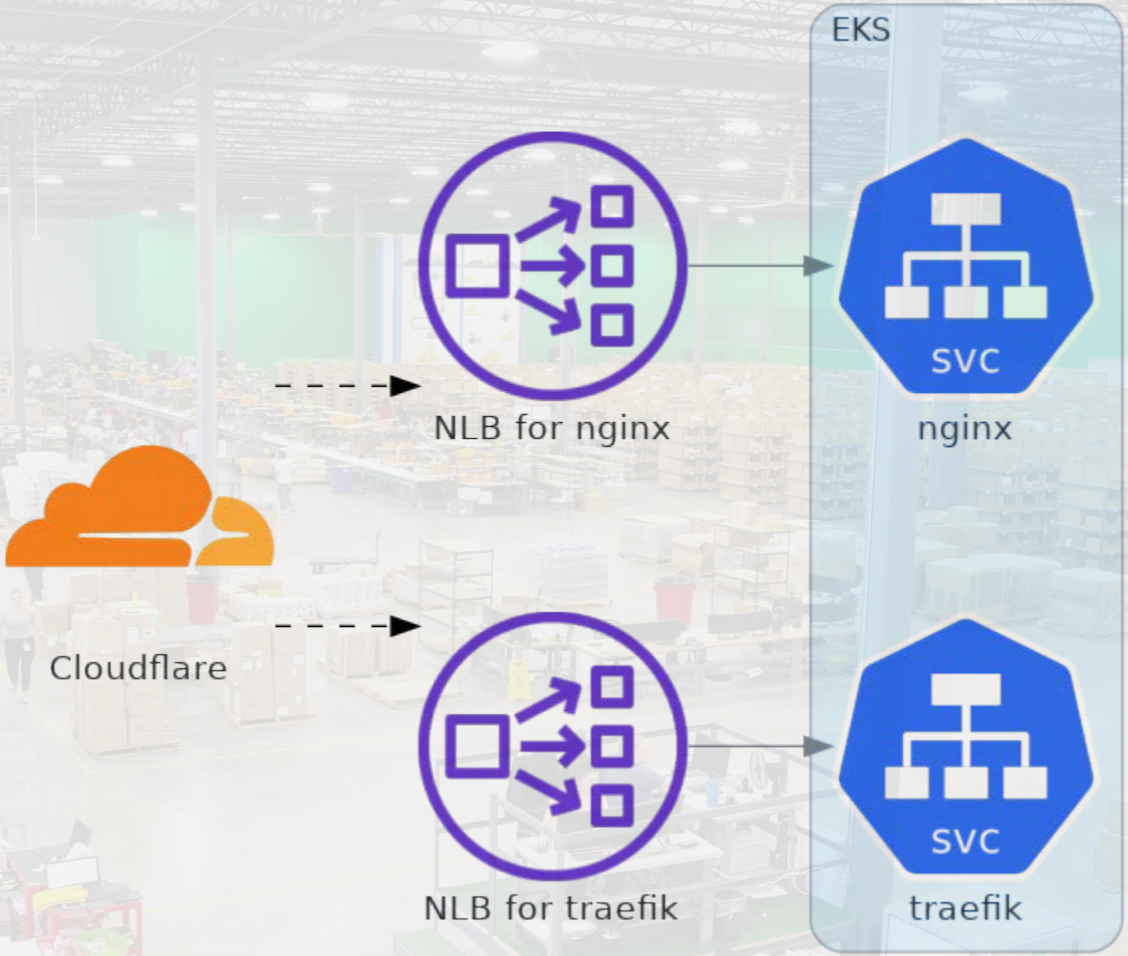
The overall cost of AWS ALB ~700 USD due to rule count

For AWS NLB only ~50USD



Plan:


The whole load balancing setup
Narrow view
Subtasks
Create K6 benchmarks 3
Evaluate AWS ALB 9+
Evaluate Traefik 2
Evaluate Nginx Plus 4
Add monitoring to Traefik
Test the run of the winner of evaluation on production 2
Deploy to Dev
Switch whole runtime to new load balancer 4
Check options from nginx probably missing on traefik: compression, ...
Traefik metrics are missing 2
Scale down nginx
Remove nginx setup from dev & prod 1
Load balancer blunder 8
Check reset metrics reported by AWS CloudWatch 1
Investigate Traefik as a deployment 1
Set QoS guaranteed on Traefik
Check real IP headers 3
Write to traefik support with the issue of failing SSL for stations 4
Check different version of traefik to avoid stations issue 2
Write to AWS support with the issue of failing SSL for stations 1
Add a comment
Collaborators
Leave task


EKS
SVC
nginx
NLB for nginx
Cloudflare
NLB for traefik
SVC
traefik

What took us so long?

- TLS suites issue — 0-1 days
- EPROTO error in Nodejs — 80 days
- Non TLS traffic on TLS endpoint — 120 days
- Usual suspect: Traefik — ∞ days

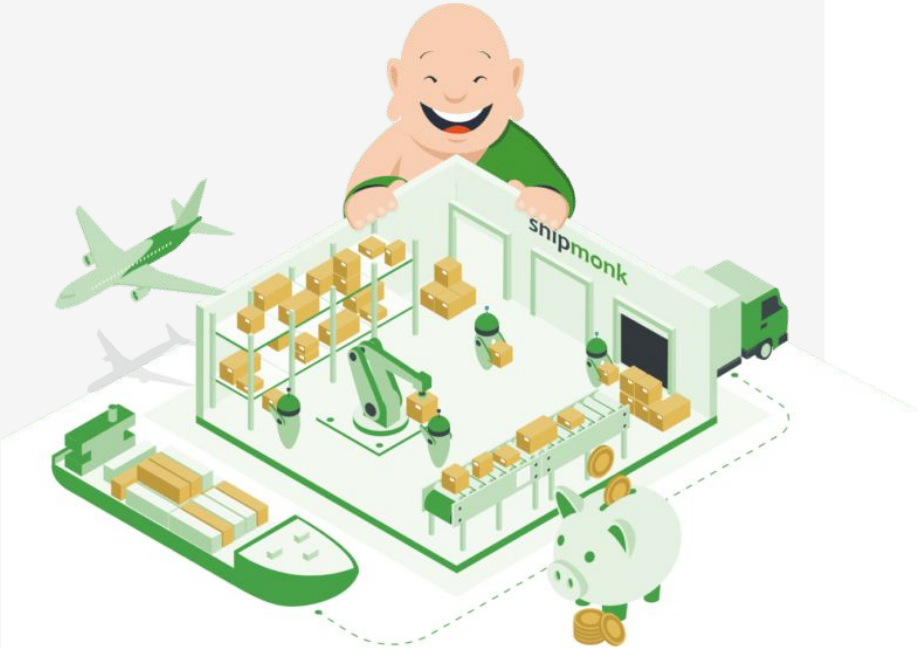
shipmonk

shipmonk

Partners are old

TLS suites issues

- Most of the times, you are cooperating with old, very old software
- ICM_HTTP_SSL_ERROR connecting to the app.shipmonk.com
- Failed connection to Locus robots
- Failed to connect to SAP



shipmonk

Solution? Downgrade

TLS suites issues

Cipher Suites

 # TLS 1.3 (suites in server-preferred order)

TLS_AES_128_GCM_SHA256 (0x1301) ECDH x25519 (eq. 3072 bits RSA) FS

TLS_AES_256_GCM_SHA384 (0x1302) ECDH x25519 (eq. 3072 bits RSA) FS

TLS_CHACHA20_POLY1305_SHA256 (0x1303) ECDH x25519 (eq. 3072 bits RSA) FS

 # TLS 1.2 (suites in server-preferred order)

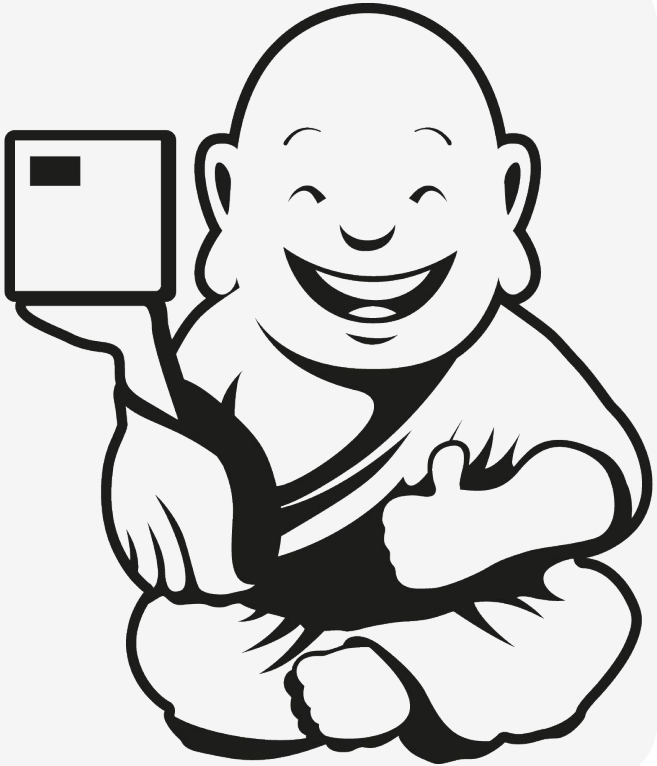
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8) ECDH secp521r1 (eq. 15360 bits RSA) FS

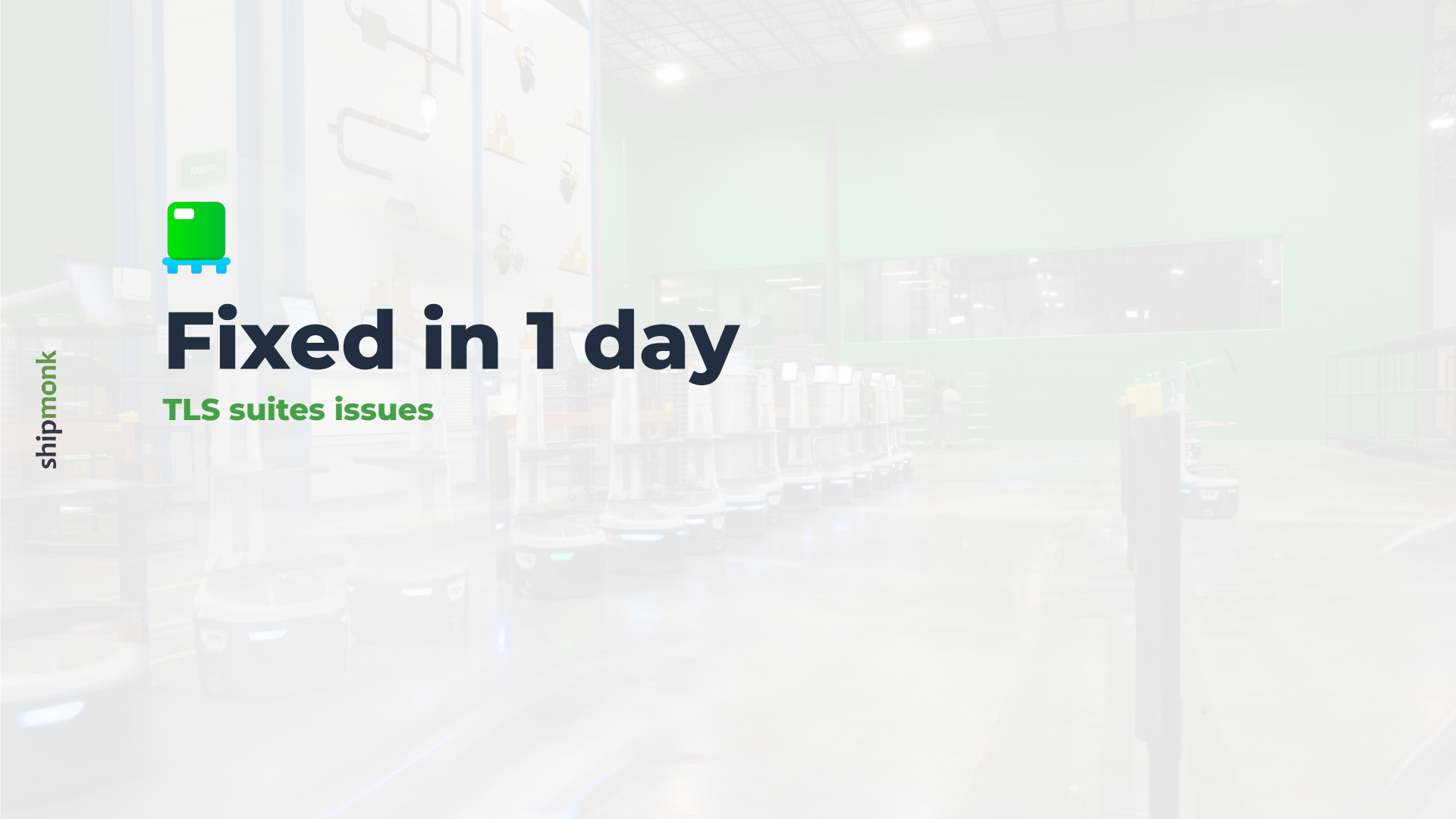
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f) ECDH secp521r1 (eq. 15360 bits RSA) FS

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) ECDH secp521r1 (eq. 15360 bits RSA) FS

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027) ECDH secp521r1 (eq. 15360 bits RSA) FS **WEAK**

(P) This server prefers ChaCha20 suites with clients that don't have AES-NI (e.g., Android devices)





Fixed in 1 day

TLS suites issues

shipmonk

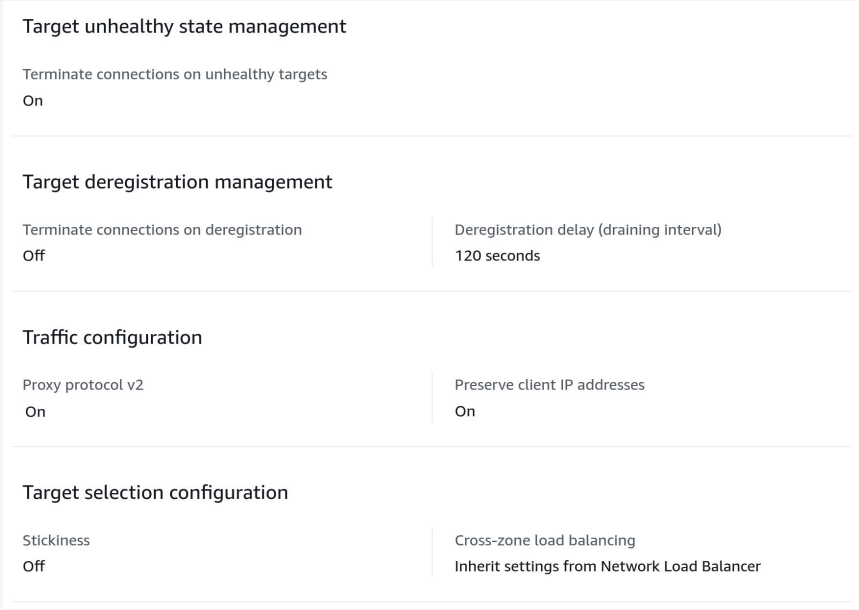
shipmonk

Proxy protocol v2

EPROTO error in Nodejs

- Protocol for transporting connection information
- Should it be on or off?

Let’s keep it on!



shipmonk

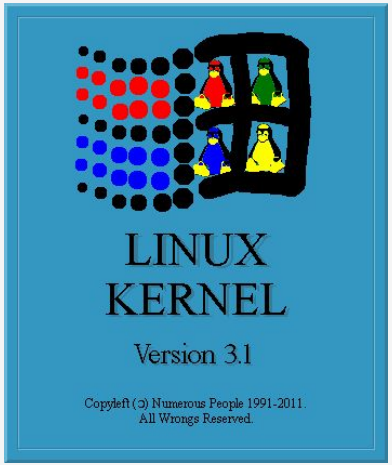
```
[24. 5. 2023 8:30:05] Request failed: FetchError: request to https://shipmonk-dev-eu-west-1-tfk-
pub-6526a9003aa92c35.elb.eu-west-1.amazonaws.com/api/v1/wms/orders/images/refresh-from-remote failed,
reason: write EPROTO 139976298616768:error:1408F10B:SSL routines:ssl3_get_record:wrong version
number:../deps/openssl/openssl/ssl/record/ssl3_record.c:332:

request to https://shipmonk-dev-eu-west-1-tfk-pub-6526a9003aa92c35.elb.eu-west-1.amazonaws.com/api/v1
/wms/orders/images/refresh-from-remote failed, reason: write EPROTO 139976298616768:error:1408F10B:SSL
routines:ssl3_get_record:wrong version number:../deps/openssl/openssl/ssl/record/ssl3_record.c:332:

EPROTO
```

shipmonk

Warehouse stations sentry issue

EPROTO error in Nodejs

- Frontend developers reported SSL issues from nodejs app
- The app is very old
    - Old SSL/TLS suite?
    - Old OS?
    - ...
- This can’t be us, can be?



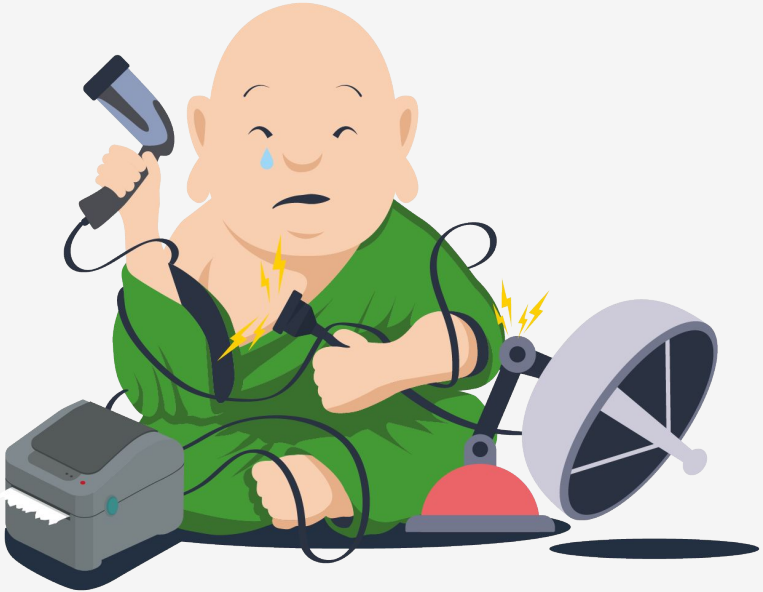
Linux Kernel 3.1 proposed logo

shipmonk

Error replication

EPROTO error in Nodejs

- Using the same node version, the same packages replicates the error
- Using the same code in different languages does not cause the issue



❌ Nodejs → Golang ✅

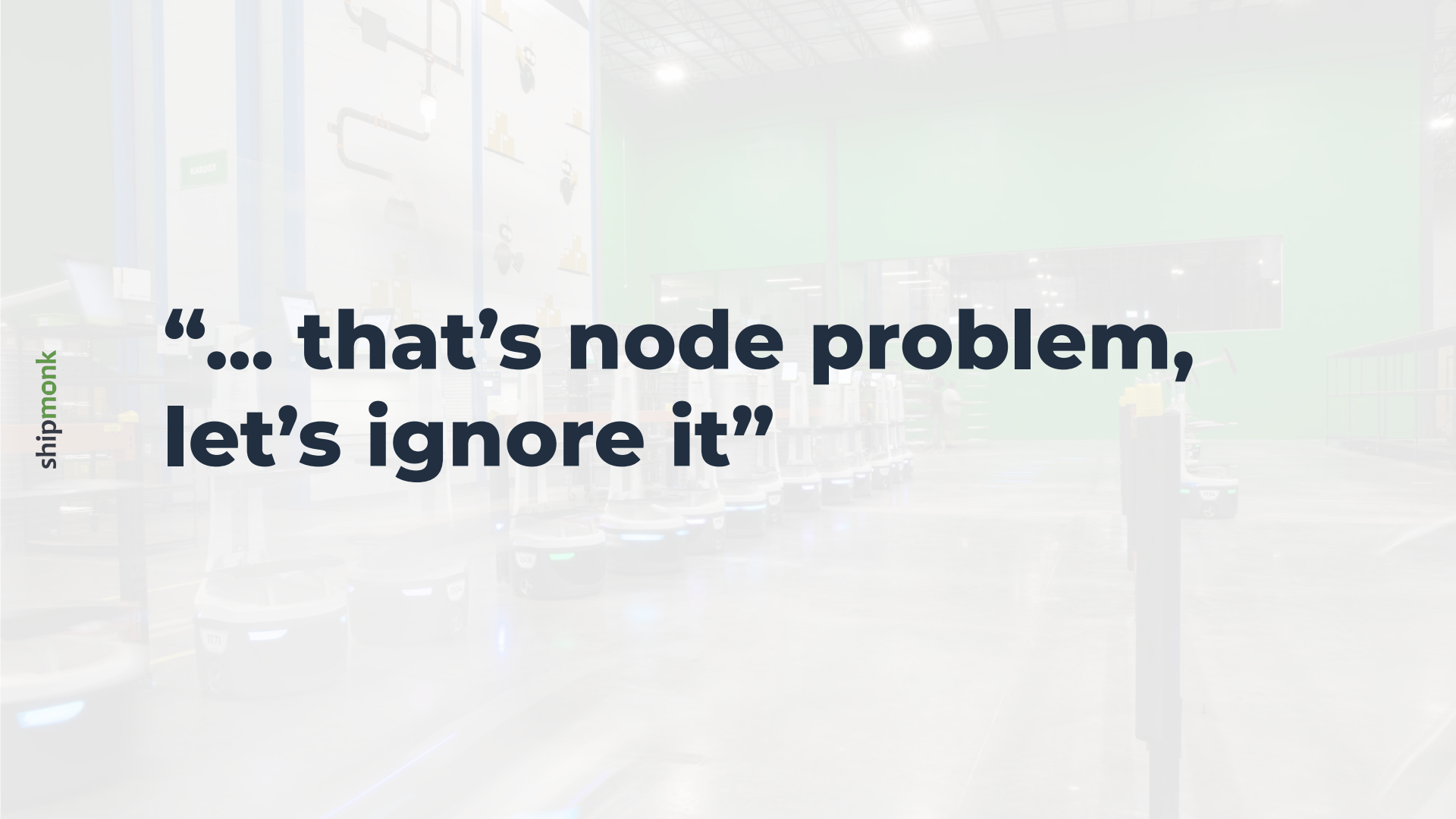
shipmonk

```nodejs
const fetch = require('node-fetch');
const https = require('https');

const url        = "https://shipmonk-dev-eu-west-1-tfk-pub-6526a9003aa92c35.elb.eu-west-1.amazonaws.com/api/v1/wms/orders/images/refresh-from-remote"
const hostHeader = "dev30783.shipmonk.dev"

let running = true;

process.on('SIGTERM', () => {
  running = false;
});

async function run() {
  while (running) {
    try {
      const res = await fetch(url, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          Host: hostHeader,
        },
        port: 443,
        agent: new https.Agent({ rejectUnauthorized: false }),
        body: JSON.stringify({
          fulfillmentId: 51292698,
        })
      });

      const date = new Date().toLocaleString();
      const status = res.status;
      const body = await res.text();
      if (![202, 400].includes(status)))
        console.log(`[${date}] Response status: ${status}\nBody: ${body}`);
    } catch (err) {
      console.error(`[${new Date().toLocaleString()}] Request failed: ${err}\n${err.message}\n${err.code}`);
    }

  }
}


for (let i = 0; i < 10; i++)
  run();
```

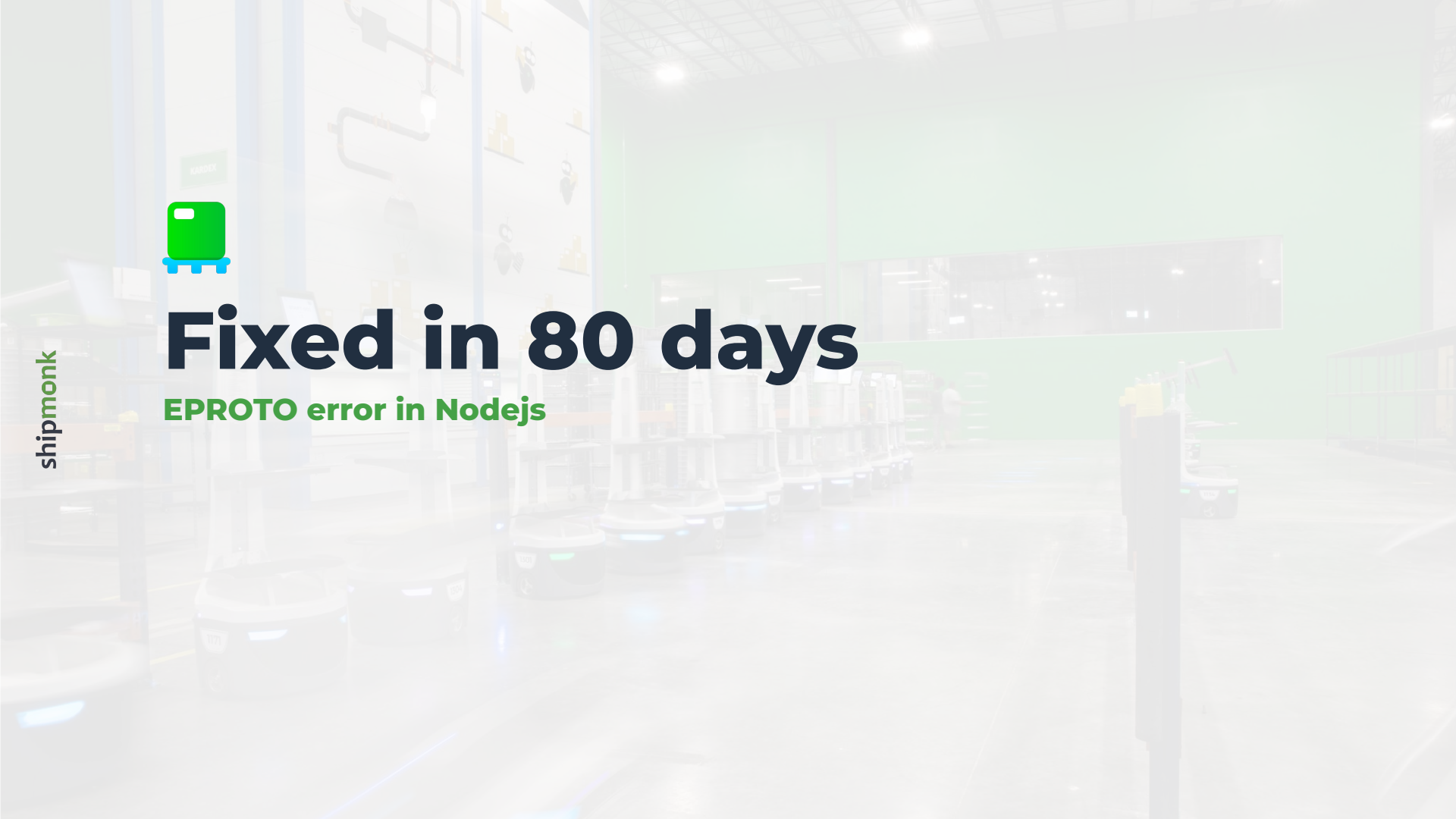
~60 lines

```golang
package main

import (
    "bytes"
    "crypto/tls"
    "encoding/json"
    "fmt"
    "io"
    "net/http"
    "os"
    "os/signal"
    "sync"
    "time"
)

const (
    url        = "https://traefik-lb-dev-pub.shipmonk.dev/api/v1/wms/orders/images/refresh-from-remote"
    hostHeader = "dev30783.shipmonk.dev"
    interval = 1 // time between requests in seconds
)

var (
    running = true
    wg      sync.WaitGroup
)

type request struct {
    FulfillmentID int64  `json:"fulfillmentId"`
    AppSecret     string `json:"appSecret"`
}

func main() {
    signalChan := make(chan os.Signal, 1)
    signal.Notify(signalChan, os.Interrupt)

    go func() {
        <-signalChan
        running = false
    }()

    for i := 0; i < 60; i++ {
        wg.Add(1)
        go run()
    }

    wg.Wait()

    fmt.Println("Exiting...")
}

func run() {
    defer wg.Done()

    AppSecret := os.Getenv("APP_SECRET")

    transport := &http.Transport{
        TLSClientConfig: &tls.Config{InsecureSkipVerify: true},
    }
    client := &http.Client{Transport: transport}
    for running {
        reqBody := request{
            FulfillmentID: 51292698,
            AppSecret:     AppSecret,
        }
        reqBytes, err := json.Marshal(reqBody)
        if err != nil {
            fmt.Printf("[%s] Request body encoding failed: %s\n", time.Now().Local().String(), err.Error())
            continue
        }
        req, err := http.NewRequest("POST", url, bytes.NewBuffer(reqBytes))
        if err != nil {
            fmt.Printf("[%s] NewRequest failed: %s\n", time.Now().Local().String(), err.Error())
            continue
        }
        req.Header.Set("Content-Type", "application/json")
        req.Host = hostHeader

        resp, err := client.Do(req)
        if err != nil {
            fmt.Printf("[%s] Request failed: %s\n", time.Now().Local().String(), err.Error())
            continue
        }
        defer resp.Body.Close()

        date := time.Now().Local().String()
        if resp.StatusCode != 202 {
            body, err := io.ReadAll(resp.Body)
            if err != nil {
                fmt.Printf("[%s] Error reading response body: %s\n", date, err.Error())
                continue
            }
            fmt.Printf("[%s] Response status: %d\nBody: %s\n", date, resp.StatusCode, string(body))
        }
    }
}
```

~100 lines



“... that’s node problem, let’s ignore it”



Fixed in 80 days

EPROTO error in Nodejs

shipmonk


shipmonk
“A few days later...”

shipmonk

Robert Josiek (CZ, WMS Product Manager) 6 months ago

https://metabase.shipmonk.cloud/ down again

✅ 1

55 replies

Richard Felkl (CZ DevOps engineer) 6 months ago

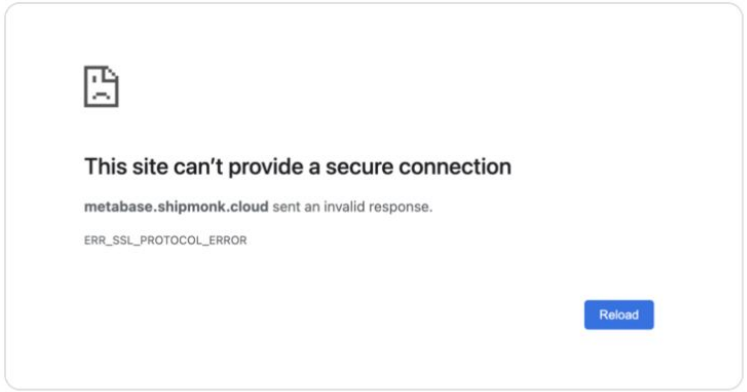
not for me

Tomáš Hejátko (CZ DevOps Engineer) 6 months ago

works for me too

Robert Josiek (CZ, WMS Product Manager) 6 months ago

image.png



Traefik returned non TLS content on TLS endpoint

Non TLS traffic on TLS endpoint

- What was that content about?
    - What the Wireshark session got me:

```
HTTP/1.1 400 Bad Request
Content-Type: text/plain; charset=utf-8
Connection: close
400 Bad Request
```

shipmonk

shipmonk

What we knew?

Non TLS traffic on TLS endpoint

- Everytime we switched 100% of traffic to Traefik, issue started after 16pm Prague time, 9am EST
- But we benchmarked everything with k6, is the load an issue?
- This never happened with 10% or 50% of traffic

Let’s bomb it till it starts to talk!

shipmonk

- It's not easy to force app **not** reuse connections
- Even if they are not reused, they can be cleanup anytime
- Let's create the largest amount of concurrent connections we can
- Just never ...Body.Close()

```
...

func main() {
    ...

    var wg sync.WaitGroup

    for i := 0; i < numConns; i++ {
        wg.Add(1)
        go func() {
            defer wg.Done()
            for true {
                req, err := http.NewRequestWithContext(traceCtx, http.MethodGet, url, nil)
                if err != nil {
                    log.Fatal(err)
                }
                _, err = httpClient.Do(req)
                if err != nil {
                    log.Fatal(err)
                }
                // _.Body.Close()
            }
        }()
    }
    wg.Wait()
}
```

shipmonk

Healthy

```
while true; do netstat --all --program
2>/dev/null | grep `pgrep main` 2>/dev/null | wc
-l; sleep 1; done
0
0
0
555
2994
3279
4540
4540
4540
5127
5518
5518
6110
6405
^C%
```

Not Healthy

```
while true; do netstat --all --program
2>/dev/null | grep `pgrep main` 2>/dev/null | wc
-l; sleep 1; done
0
0
0
555
2994
0
0
0
^C%
```

Solution? Dunno

Non TLS traffic on TLS endpoint

shipmonk

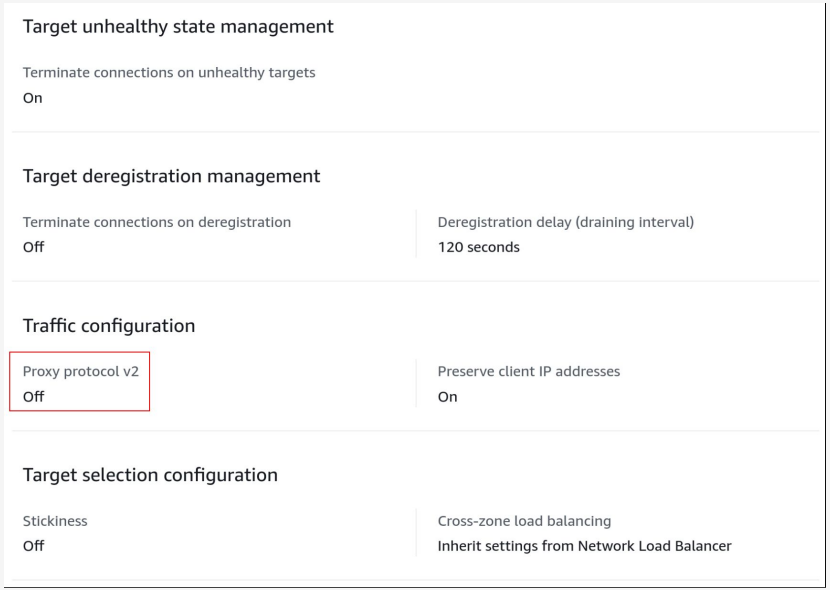
We tried:

- Opening ticket to AWS
- Opening ticket to Traefik partner
- Rewrite ingress to IngressRoute (Traefik CRD)
- Different builds AMD64 instead of ARM
- Moving versions up and down
- Scaling turned off & on, pre scaled to ~100 of Traefics

- Opened an issue on GitHub

Solution? Disable Proxy v2

Non TLS traffic on TLS endpoint



shipmonk

shipmonk

Fixed in 180 days

Non TLS traffic on TLS endpoint

It got fixed recently

Non TLS traffic on TLS endpoint

juliens commented 2 weeks ago Member

Upon conducting additional analysis, it appears that the issue is linked to a read header timeout. Upon further inspection, I discovered that this timeout was not configured in the PROXY protocol library we are utilizing. Consequently, I implemented a solution.

A big thank you to **@jb-abbadie** for providing a reproducible case; it was immensely helpful.

shipmonk

Usual suspect: Traefik

shipmonk

So can we confirm Traefik is causing it?

It works for me but not for @Petr Morávek (CZ SW Engineer), is traefik working correctly?

Thanks everyone for identifying the root cause
@Martin Beránek (CZ INF Eng. lead) this was another issue caused by the switch to Traefik. Can we implement a better process for catching similar errors?
You saw it in #major-system-issues, correct?
cc @Matěj Smíšek (CZ VP of Engineering)

This was another issue with Traefik, @Martin Beránek (CZ INF Eng. lead) we have to think of a way how to test these changes directly after they are done so we don't disrupt production and operations

I know but where can we have a discussions? Should we create a traefik possibly related issue temp channel?

shipmonk

@Martin Beránek (CZ INF Eng. lead) can we revert Traefik and check with them if it works again?

Traefik was deployed yesterday, maybe it's causing issues?

Previously it was caused by the switch to Traefik, co you check if it's still the case?

Locus sync is unstable, not sure if Traefik would cause it, it's happening from last night
https://shipmonk.slack.com/archives/C04GQ6FD01F/p1690355324640869

Could it be Traefik?

Probably a short issue when removing Traefik https://shipmonk.slack.com/archives/C03E989J37F/p1683014931176319

> **Richard Felkl (CZ DevOps engineer)**
> @channel Today from 11AM to 12AM CET (5AM - 6AM EST / 2AM - 3AM PST) we're going to switch 50% of production load to Traefik load balancer again. There should be no outage. If any problem happens let us know to this thread.
>
> Thread in # devops_status | May 2nd | View message

3 replies

Can we turn off traefik balancing for the backend app and see what effect it has on the latency? https://app.datadoghq.c ... Show more (edited)

Why would this happen? What does Traefik do differently? It shouldn't cause unreachable apps

shipmonk

People tend to blame every issue on Traefik

Usual suspect: Traefik

- With 90% of accuracy
- But those 10% hurt a lot
- Even in cases there was only Nginx running
- Or after Traefik was running without issues for months

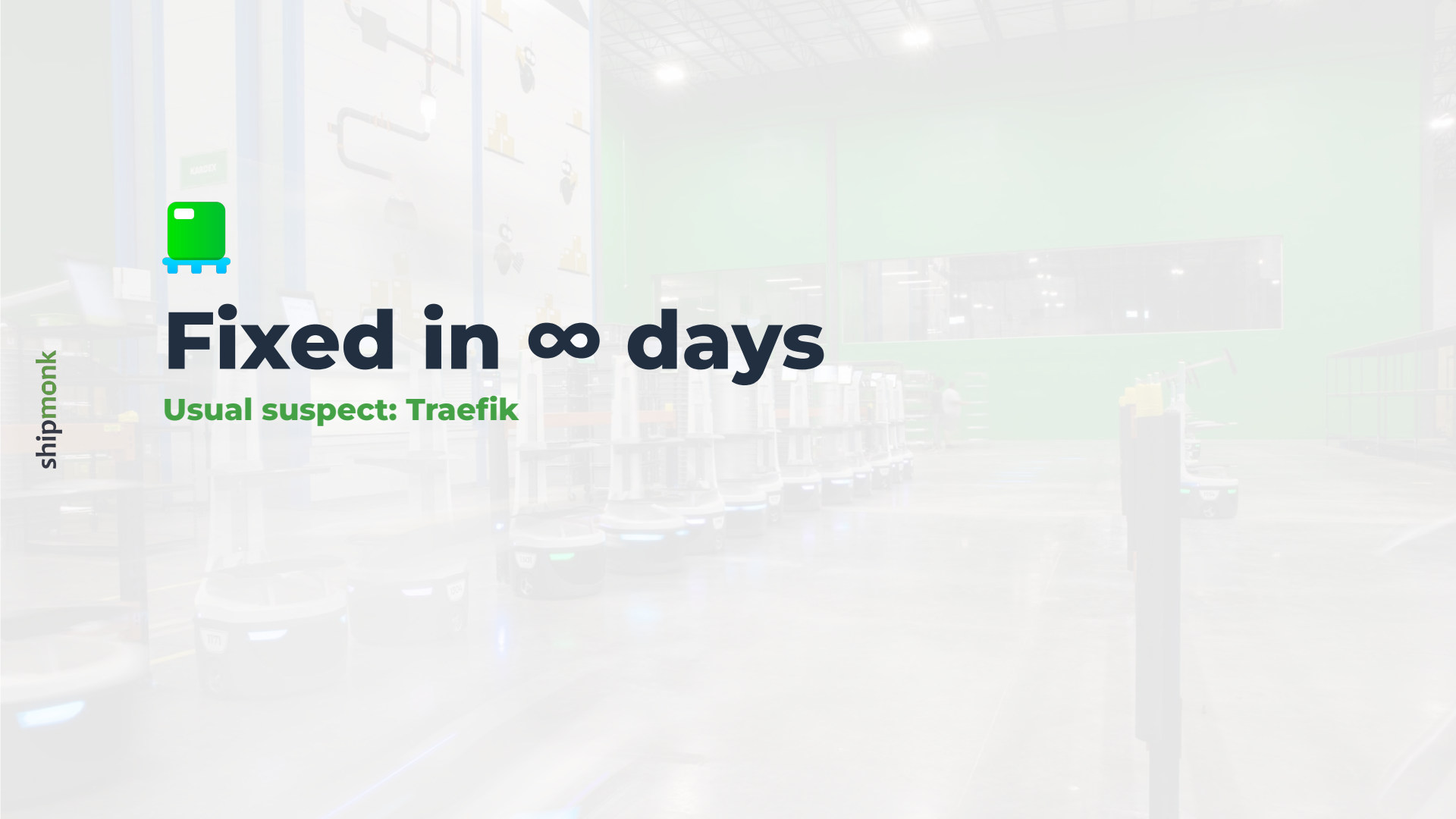
The only cure is the overall stability

The flow

Usual suspect: Traefik

- Evaluate multiple options (benchmark, check docu, ...)
- Announce the change as detailed as possible
- Deploy to reasonable amount of traffic: 10%, 50%, ...
- Evaluate, do reviews
- Announce the final deployment
- Deploy and evaluate again
- Cleanup of old setup

shipmonk



Fixed in ∞ days

Usual suspect: Traefik

shipmonk

Review

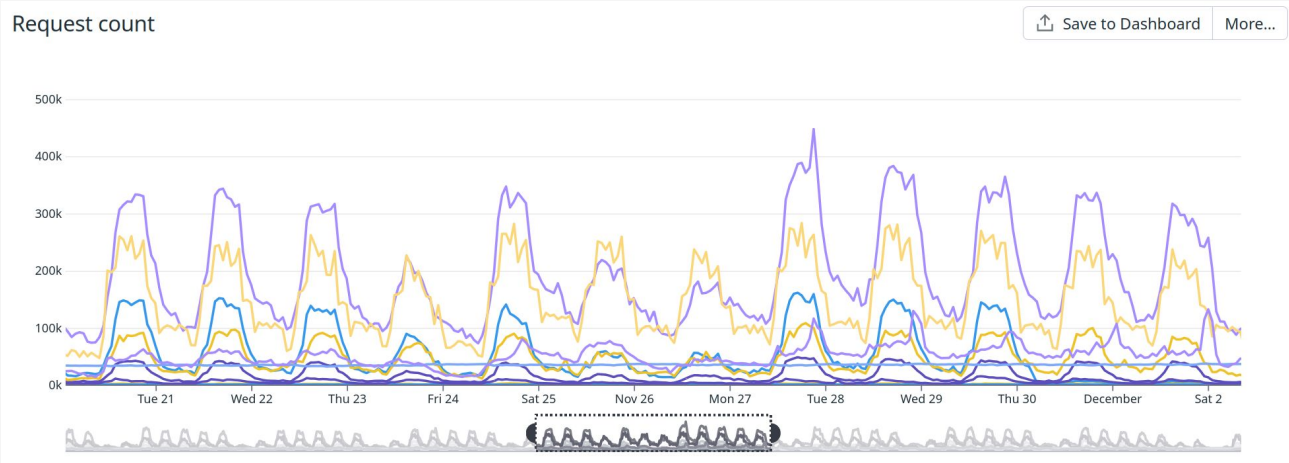
shipmonk

24.11.23 the Peak, Black Friday

Review

shipmonk

- No utilization issues
- Only 4 replicas
- About 0.1 CPU load for each
- Around 1 GB of RAM
- Average day



We had ~10 minutes downtime last year cause of NGINX issues

Sizing

Review

shipmonk

- Original setup used on production
  - 8 pods
  - Each pod 1 CPU and 2 GiB
- Much larger scale on dev
  - Each pod 500 mCPU and 256 MiB
  - As DaemonSet

- Traefik has on prod:
  - 4 pods
  - Each pod 1.3 CPU and 2.4 GiB
- On dev:
  - 2 pods
  - Each pod 0.7 CPU and 1.2 GiB

Rightsizing

Review

shipmonk

Summary

Type	Usage	Requests	Limits
CPU	8.02%	1.30 Cores	1.30 Cores
Memory	25.70%	2.34 GB	2.34 GB

Recommendation

Type	Recommended Requests
CPU	0.17-0.21 Cores
Memory	1.00-1.20 GB

Requirements

Introduction

- Use a similar setup, no new CRDs ✅
- Have the same setup on production and dev ✅
- Use something which the team already partially knows ✅
- It has to have [at least] metrics in Prometheus ✅
- Be stable ✅

shipmonk

Thank You

Questions?

shipmonk

We are hiring

Try our challenge in this docker image

shipmonkdevops/shipmonk-test

shipmonk