The Tech Evolution of Mailinator.com

Scaling to Millions of Emails/day

Paul Tyma, Ph.D.

What is Mailinator.com?

- A website that accepts email for ANY address @mailinator.com
  - That's ~~billions~~ ~~trillions~~ ~~quadrillions~~ LOTS of possible addresses
- There are no user "accounts"
- Anyone can view any inbox or read any email
- No Signup
- No Login
- Receive-only
- No Attachments
- All email is deleted after a few hours

What is Mailinator.com?

- As relevant to this talk:
  - A website I started in 2003
  - How it evolved into a Saas Service with many thousands of daily users including Privacy, Security, Reliability, and Scalability

What this talk is about

This is not necessarily a talk about the right way to scale a system

This is a talk about how I scaled a system and what I learned along the way

Who am I

- Until Sep 2019, CTO @lendingtree.com
- Founded 5 startups
  - Preemptive Solutions, Pulse.io, Home-account, Manybrain (Mailinator), Refresh
- A few years at Google, Linked
- Ph.D. in Computer Engineering, 2004 - ~compilers

Disclaimer

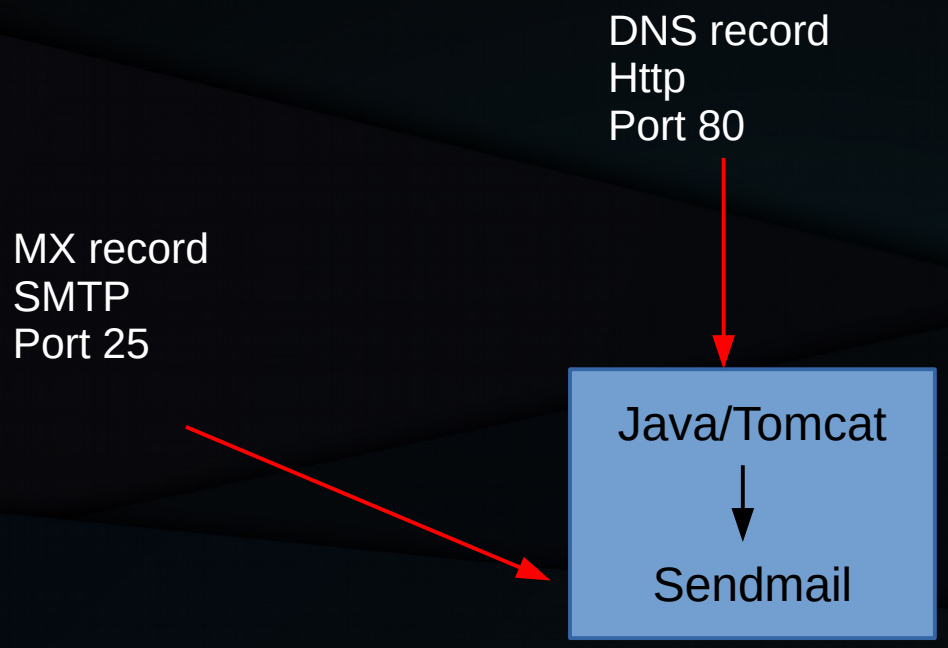
- We'll be showing live (text-only) data
- We don't control the data
- There might be bad words in the data
- There might be a **lot** of bad words in the data
- Only click emails you're sure you want to see

Initial implementation over a weekend

- Configure Sendmail:
  - Vim `virtusertable`

    `@mailinator.com catchallinbox`
- Java (+imap library) / Tomcat
- Code up terrible UI
- Rent Serverbeach:
  - AMD 2Ghx Athlon, 1G ram, 80G HD, 10Mbps Net
- Store only the 20,000 most recent emails (FIFO)

Hardware

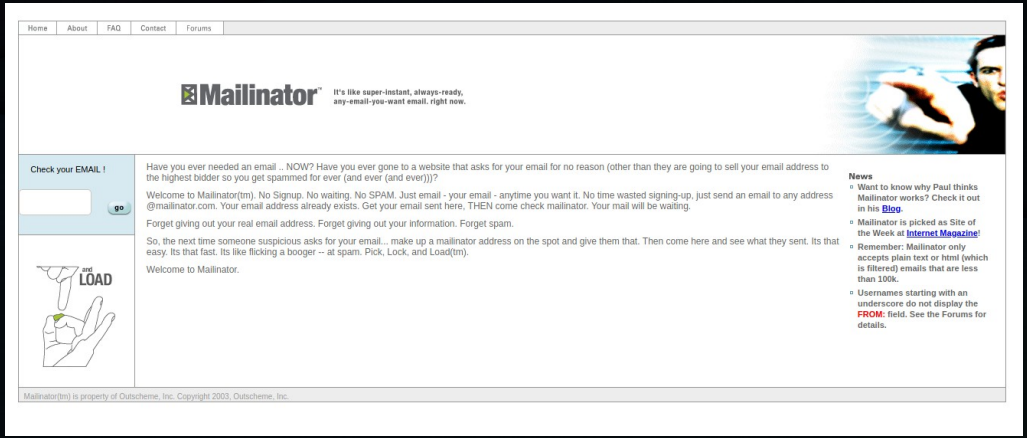


Launch !

- 6000 web visitors first day thanks to Joel

July 23, 2003 by Joel Spolsky

Need a disposable email address so you can sign up for some web site, quick? You've already got one, at Paul Tyma's clever Mailinator. "Just send an email to any address @mailinator.com. Your email address already exists. Get your email sent here, THEN come check mailinator. Your mail will be waiting." More proof that great UI design is done by taking away, not adding things.



Things I learned

- The world tends to think an email address implies identity
  - The world is wrong
- Some Websites that allow signup with an email address figure that creating an email address takes time and that will act as a natural rate-limiter
  - Those websites are wrong
- We write lots of anti-abuse, rate-limiting code

System Limitations

- Both good and bad Software Engineers build systems that eventually crash under load
  - But the good ones can tell you where it'll happen first
- Disk seek time (in early 2000's): ~10ms
- Each incoming email requires a disk write and a disk delete (of the pushed out email)
  - Hand waving over many other disk seeks (directory structures, etc)
- The hard drive alone limited us to ~50 emails/second
- Disk cache was useless given data ingestion pattern

System Limitations (circa 2005)

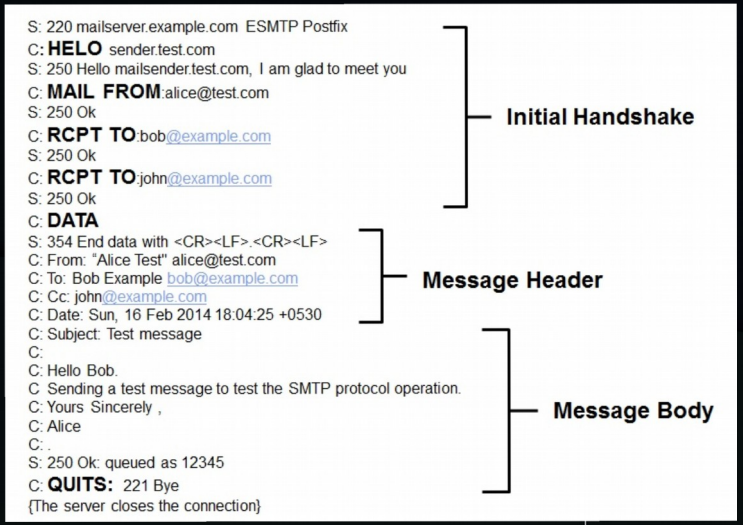
- System crashed on the regular at about 800,000 emails per day (~9.3/second)
- User's reading email caused further disk reads but a trivial amount in the big picture
- As is often the case, disk was the problem
  - Note: SSD seeks times now around ~65microseconds
  - Other Note: Memory access measured in nanoseconds

Rewrite to store data in RAM (circa 2005)

- Google keeps the entire web index in (lots of) RAM
- Why not store all email in RAM?
  - 20,000 emails * ~4k email size = 80M
  - Plus Index inbox lists and email id's
  - Java isn't known for memory frugality
  - Estimate 130M per 20,000 emails

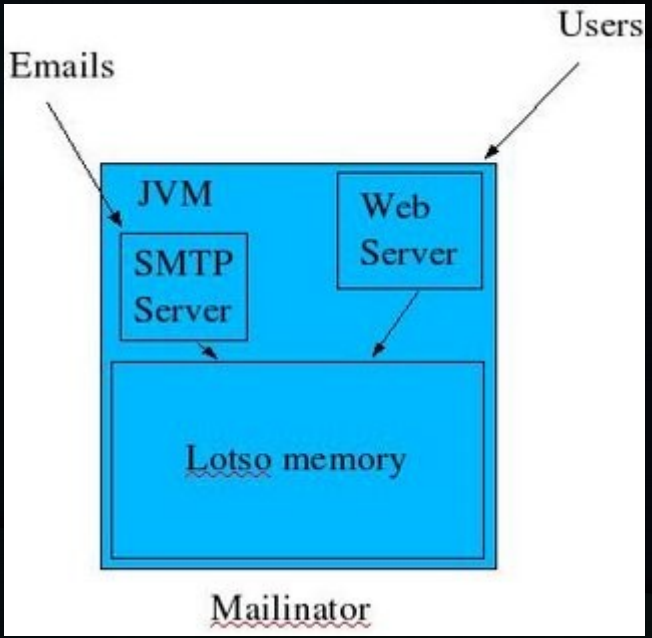
Rewrite to store data in RAM (circa 2005)

- Build a custom Multithreaded SMTP server
  - SMTP is a very chatty protocol
  - thread-per-connection
  - Up to 300 simultaneous threads
    - 512k stack per thread
- Upgrade server to 2GB



Rewrite to store data in RAM (circa 2005)

- Web + SMTP in one
  - Ports 25, 587, 80, 443
- Bouncing the server not only brought the site down, it now lost ALL email
  - Hey. It's a free service.



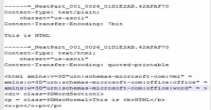
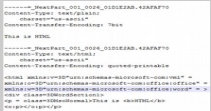
How do we store these emails?

- Compactly. RAM is limited
- Must keep ordering of receipt to expire old emails
- Must be able to lookup emails by inbox
- Must be able to retrieve emails by ID

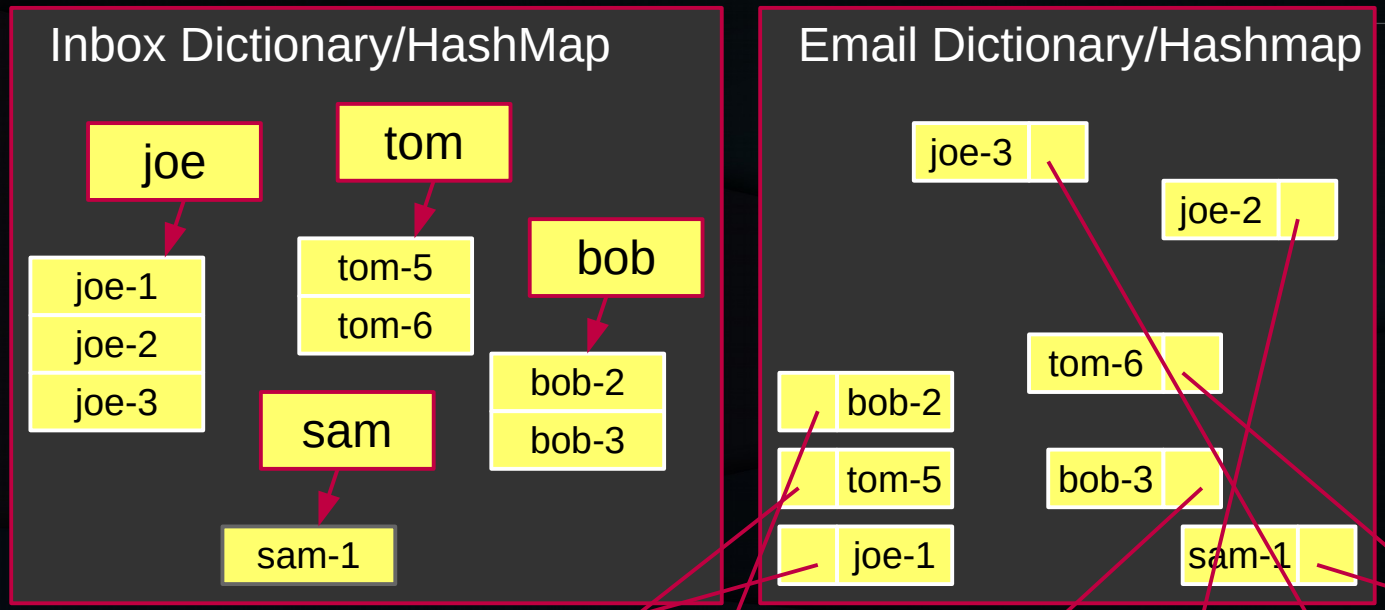
Storage: If this were a database

Index

Index

ID	Received	To	From	Subject	Email Body
joe-3	1/12 3:24pm	joe	elon@tesla.com	Gasoline sucks	
bob-5	1/12 3:25pm	bob	billg@microsoft.com	Try windows	
frank-9	1/12 3:26pm	frank	jeff@amazon.com	Buy now	

How to Organize Memory (V1)



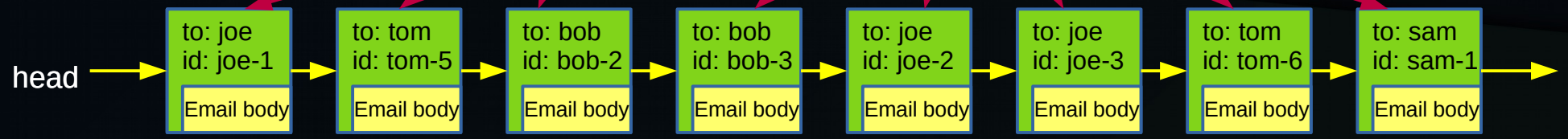
Fetch Inbox: 100's per minute

Fetch Email: 10's per minute

Add Email: 1000's per second

(User) Delete Email: 1's per minute

Need mutex for all 3 structs



Organize Memory

Considerations:

- Inbox Dictionary: hundreds of thousands of entries
- EmailId Dictionary: millions of entries
- Email LinkedList: millions of entries
- Unlike a lot of Dictionary usage, we have a 10/90 read/write ratio

- All 3 structs must be synchronized for mutation
  - i.e. every incoming email
  - Dictionary resizing holds locks a long time
- Replace all structs with: SynchronizedLinkedTreeMap

How about compressing emails?

- CPU intensive
- Became feasible with higher-core machines
- Doesn’t work well
  - Compression algorithms typically work by building a dictionary. Emails are often too short to build a notable one
  - Base64 emails (i.e. big emails) build none at all
  - Tried a email-corpus Huffman encoder
    - Worked on Headers
    - Body data not so much

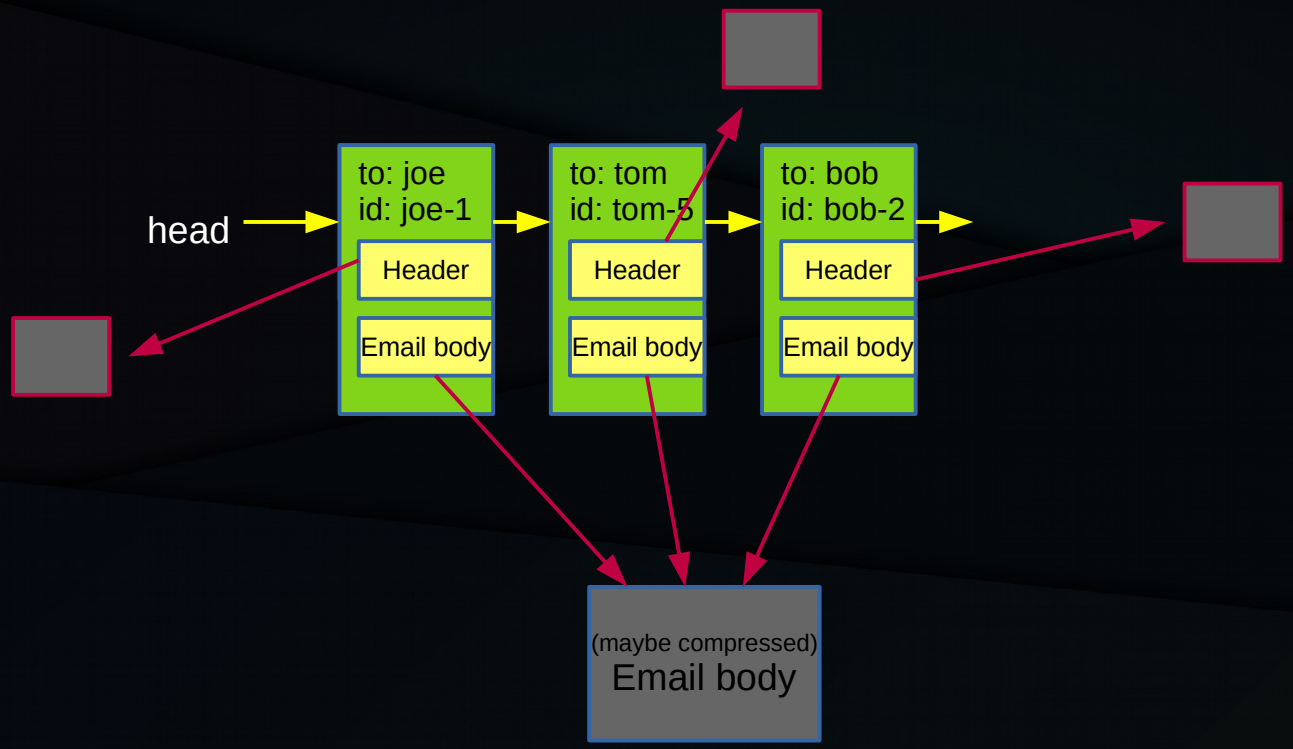
How about reusing emails?

```
Received: from mail-ot1-f43.google.com([209.85.210.43])
        by mail.mailinator.com with SMTP (Postfix)
        for joe@mailinator.com;
        Mon, 06 Jan 2020 20:54:52 +0000 (UTC)
From: Paul Tyma <paul@manybrain.com>
Date: Mon, 6 Jan 2020 15:54:41 -0500
Subject: Hi Joe
To: joe@mailinator.com
Content-Type: multipart/alternative; boundary="000000000000ec3632059b7ede21"

--000000000000ec3632059b7ede21
Content-Type: text/plain; charset="UTF-8"

Hey *Joe*. How are you today? Buy our debt consolidation services !!!!

--000000000000ec3632059b7ede21
Content-Type: text/html; charset="UTF-8"

<div dir="ltr"><div class="gmail_default" style="font-family:tahoma,sans-serif;font-size:large">Hey <b>Joe</b>.</div></div>

--000000000000ec3632059b7ede21--
```

headers

body

Compress and Reuse Message bodies



Everything is Great!

- New System Launch !!!
- System went from crashing at 800,000 to running fine at a consistent 1MM emails/day
  - Rock solid and super fast!
- Disk completely idle
- CPU’s not busy at all

Or is it?

- Kept getting around ~1MM emails/day
- Went on like this for months - it became “too” consistent
- Site seemed pretty slow
- Customers kept complaining email didn’t arrive
  - What did they know

There’s always a new bottleneck

- Then - Serverbeach had a sale
  - Upgraded the 10Mbps NIC to 100Mbps NIC
- Traffic immediately shot up to 3MM emails/day
  - 10Mbps = 1.25MB/s
  - 100Mbps = 12.5MB/s
- Every system has a bottleneck
  - Choose: CPU, RAM, Disk, Network

Things I learned:

- Every user (or email) that hits your system is part of a concerted Denial-of-service attack
  - It’s much harder to DoS your system if their request only hits RAM and CPU
  - It’s really easy to DoS your system if their request hits disk

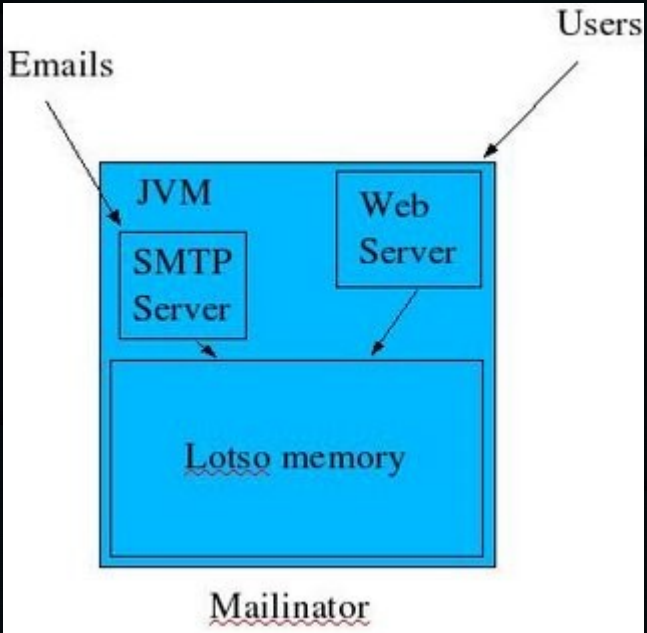
Things I learned:

- Forgot to check domain on incoming emails
  - We don't just accept email for any inbox @mailinator.com – we accept incoming email for any inbox at **any domain** that hits our server
    - **That's a Feature !**
    - Today, ~1900 domains point their MX record at us
    - `https://viewdns.info/reversemx/?mx=mail.mailinator.com`
  - That looks like an open relay
    - We get a lot email from security people telling us our email server is miconfigured
    - Email FROM @mailinator.com always gets classified as Spam
    - But we don't send email?
    - And we don't want to !
    - **Another Feature !**

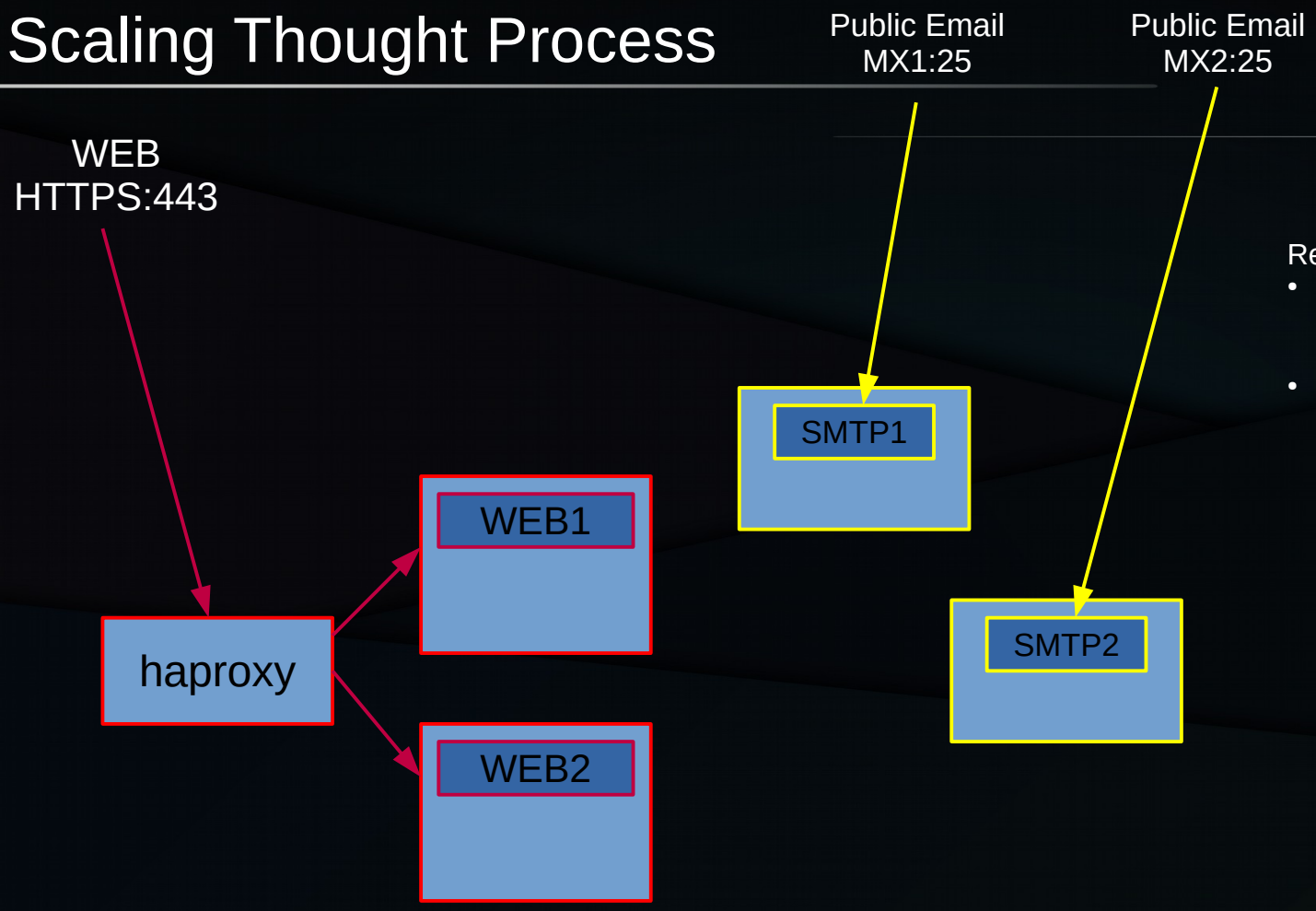
Everything is Great!

- But still running on a single server
  - Few million emails per day
  - Few thousand users per day
- Then I went on vacation for a week
  - Hardware failures always happen when you're on vacation
  - Need to make Infrastructure redundant

Hardware (where we are)



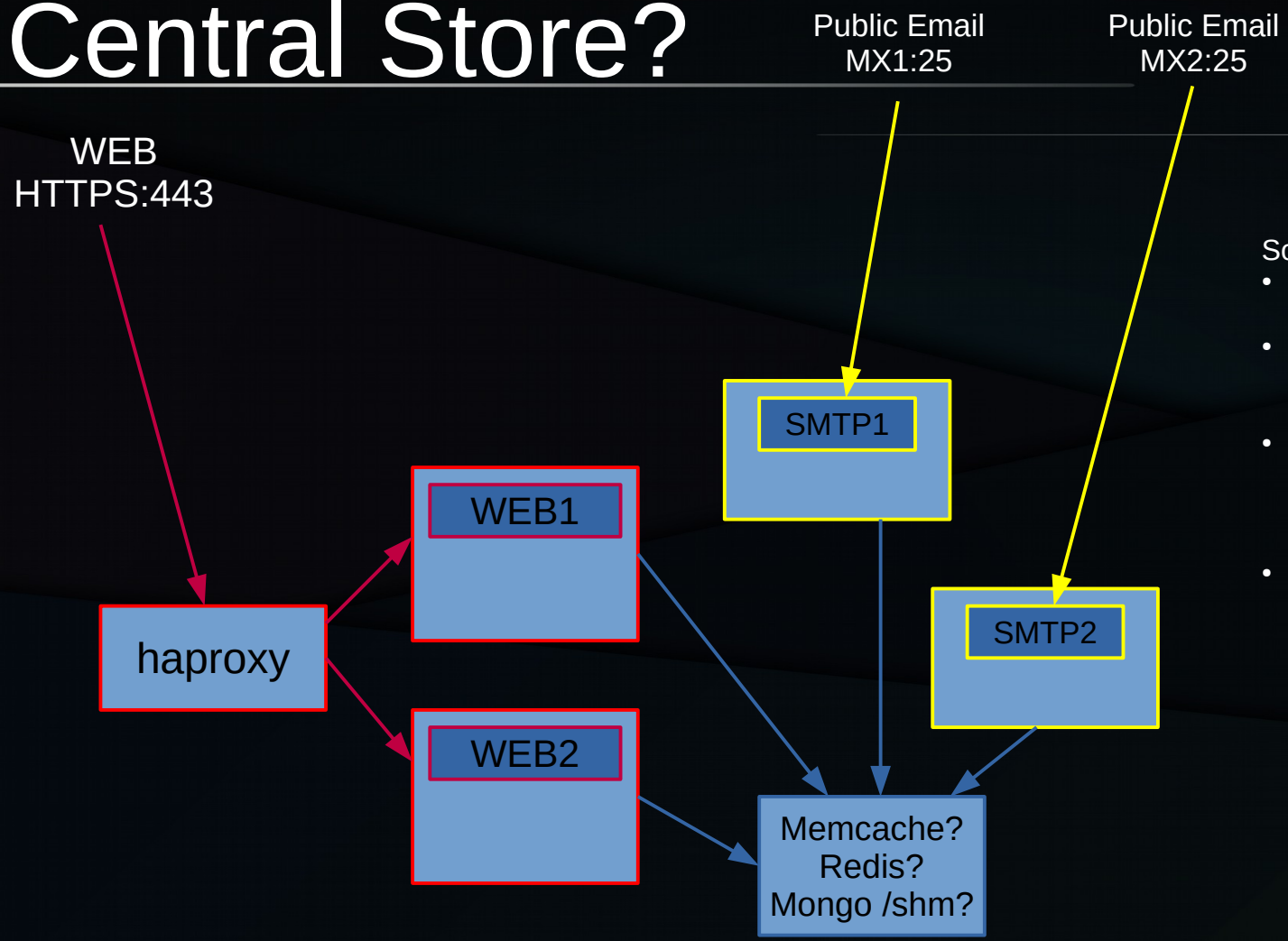
Scaling Thought Process



Remember:

- Each Rectangle is a physical (i.e. virtual) server
- We get WAY more SMTP traffic than Web traffic (but web is prioritized)

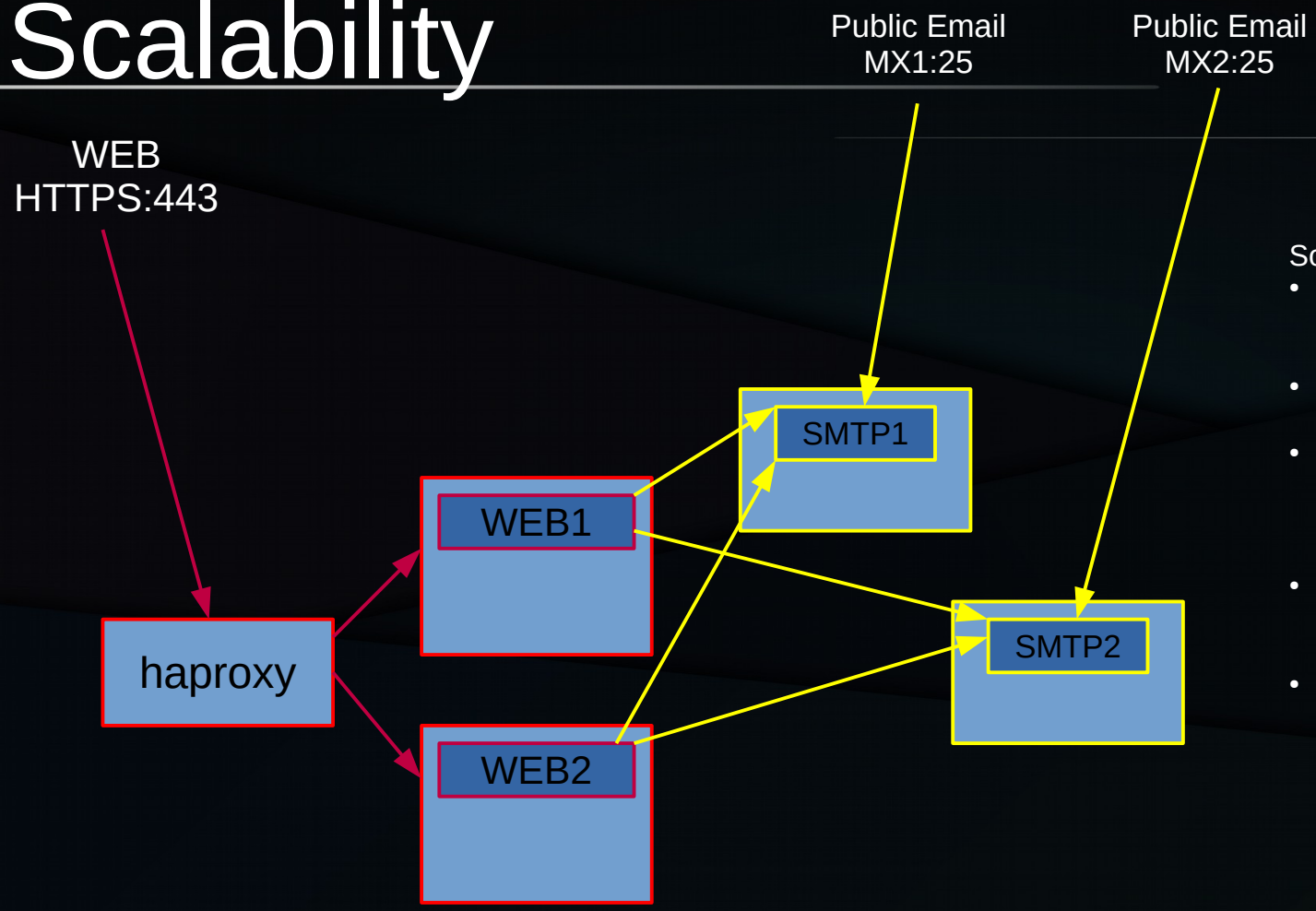
Central Store?



So:

- Central store introduces a bottleneck
- SMTP servers have doubled their bandwidth
- Like most systems in the world, the datastore is surely to be our next bottleneck
- Horizontal scaling of SMTP servers does not help us

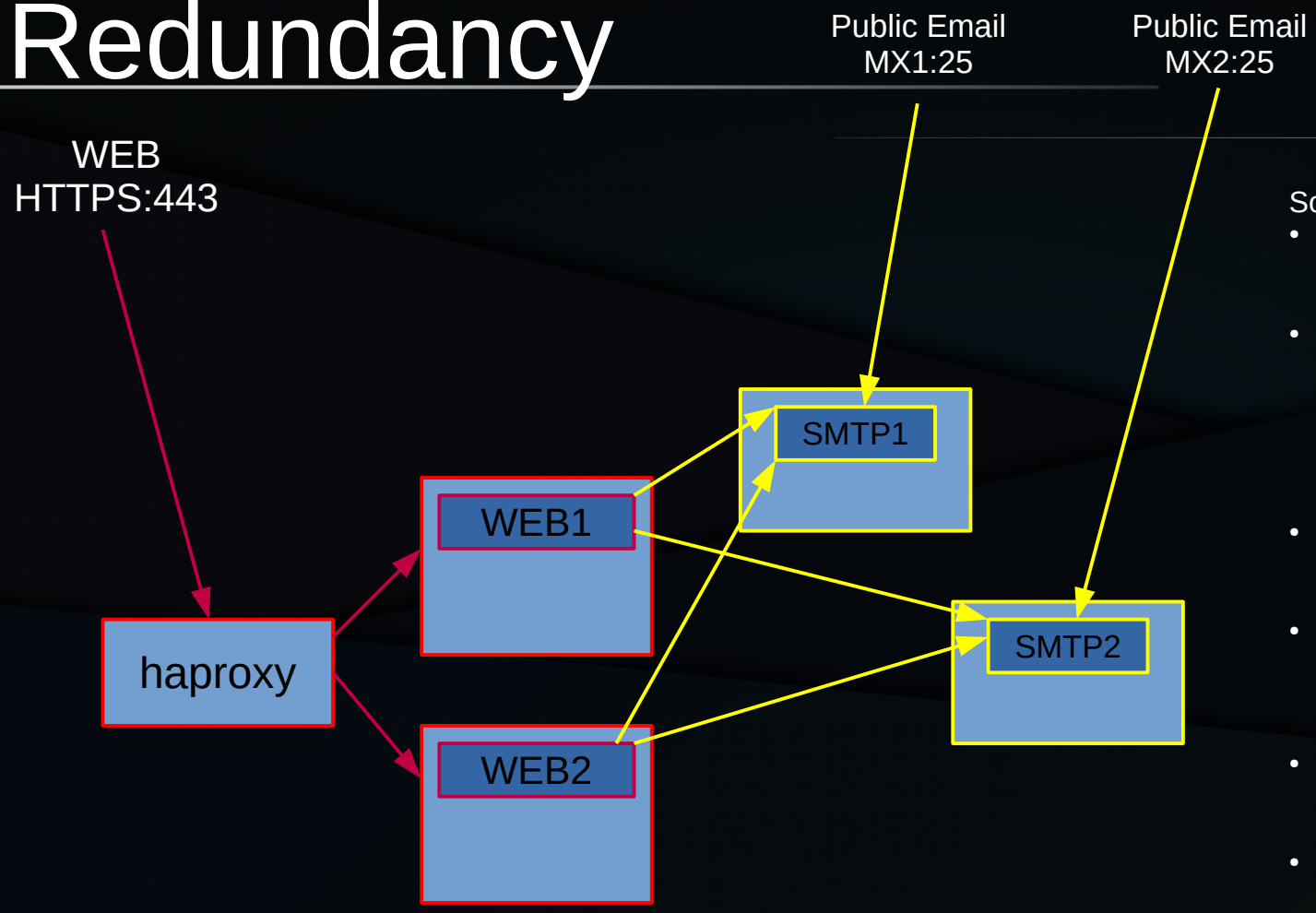
Scalability



So:

- What happens in the SMTP server, Stays in the SMTP server
- MX records scale horizontally
- SMTP Servers don't know about each other – we can continue to add more of these
- Web servers unloaded and Cloudflare protects them from shenanigans
- Can scale them too up to HAPROXY limitations (which is really high)

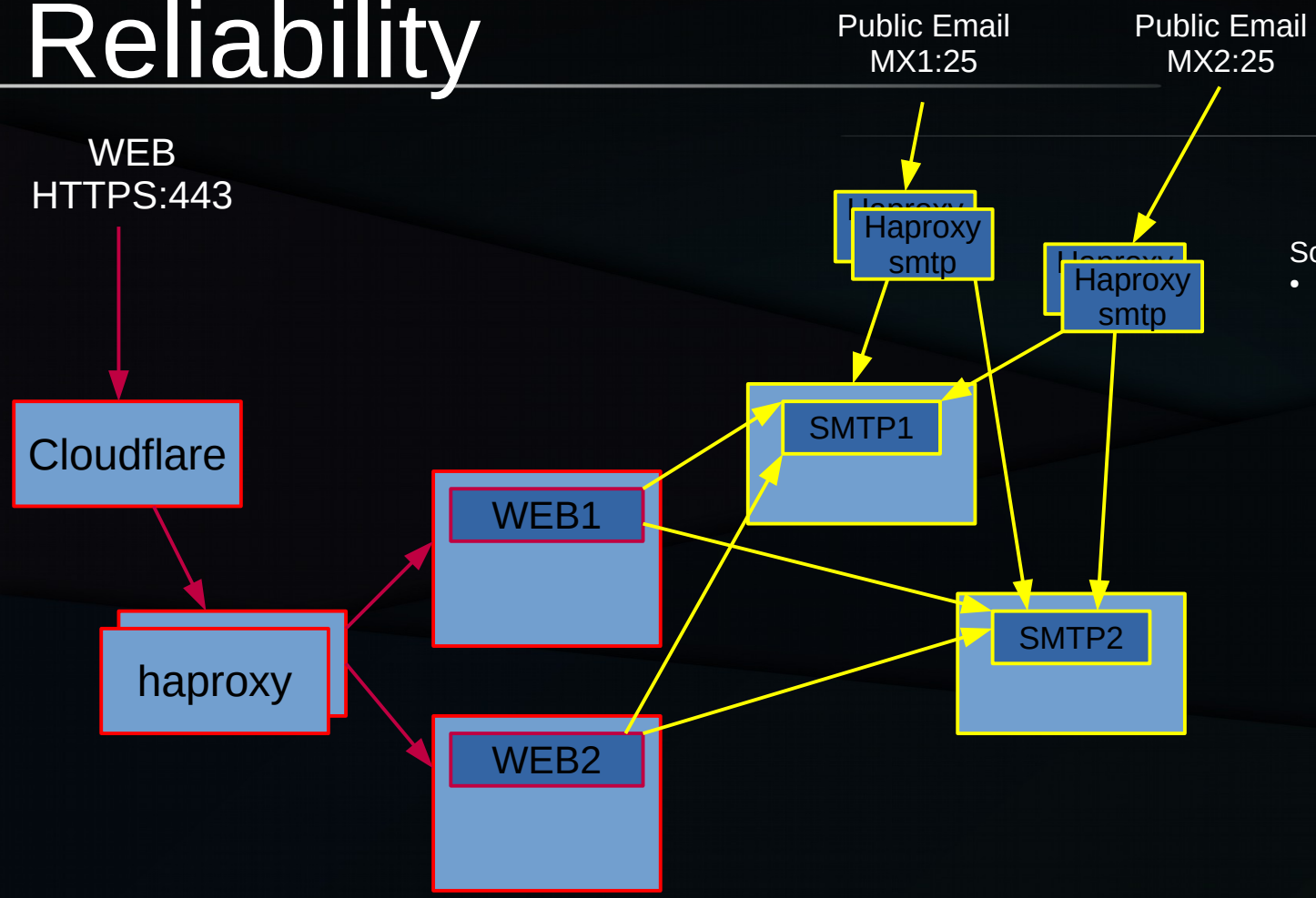
Redundancy



So:

- If a webserver OR Smtpserver goes down. We're fine.
- Web talks to SMTP through proprietary socket protocol (not REST). Note that pure TCP connections have a 3 way SYN, SYN-ACK, ACK just to connect.
- Haproxy never changes and is super stable.
- MX records naturally distribute the load between SMTP servers. But sadly, not very well.
- If a SMTP server goes down, we're still not receiving some email
- We might get DDos'd (hint: we did). Let's get behind a Ddos protector.

Reliability

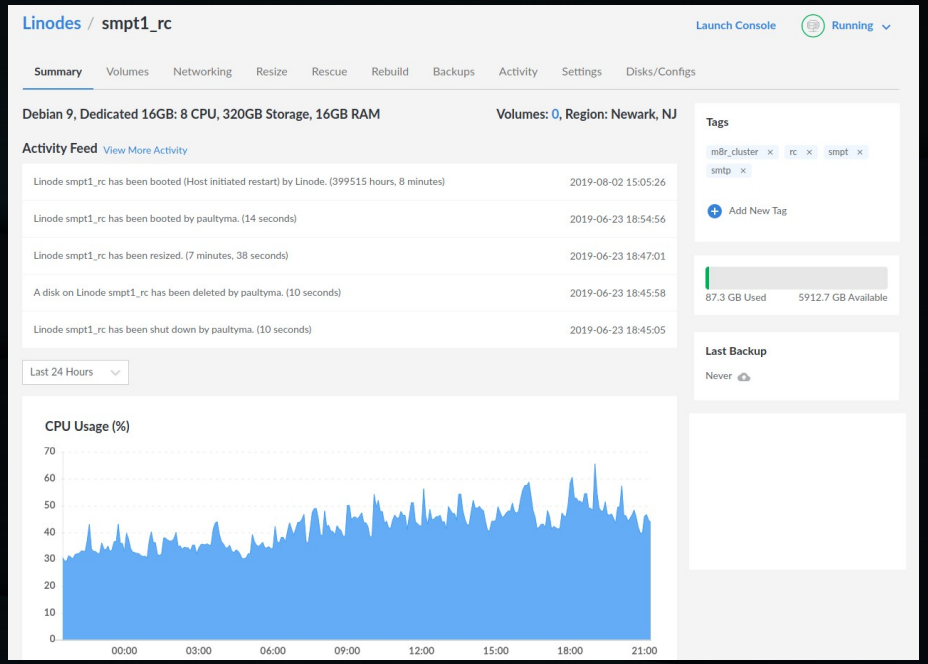


So:

- Great! Super stable. Knock down anything and we're fine. Reboot anything and we're fine. (except Cloudflare)

Multi-server problems

- Service Discovery
  - Linode API with Tags
- Jgroups
  - Leader Election
    - Batch Jobs!
  - Fabric
    - Cross-server shared heap



Everything is Great !

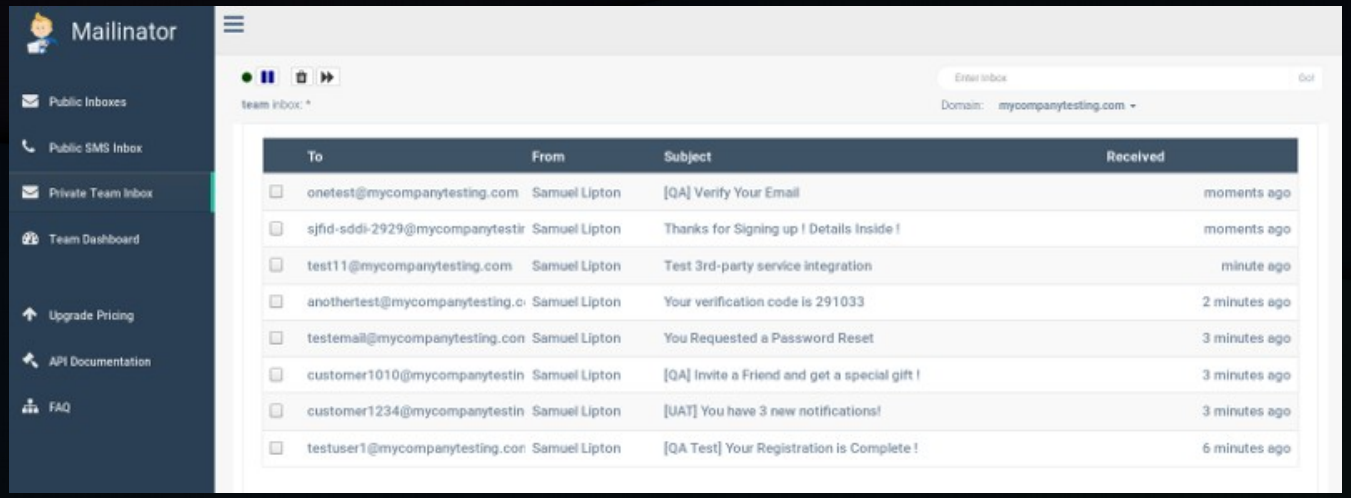
- System running like clockwork
- Only thing that brings system down is me tinkering with the code
- Support emails start to increase
  - But not for system issues
  - Mostly business users asking for features and enhancements

Time to go Pro

- You know - We could charge for this
- QA Teams are using this like crazy
  - Having an infinite number of email addresses is darn handy for testing

Subscriber Features

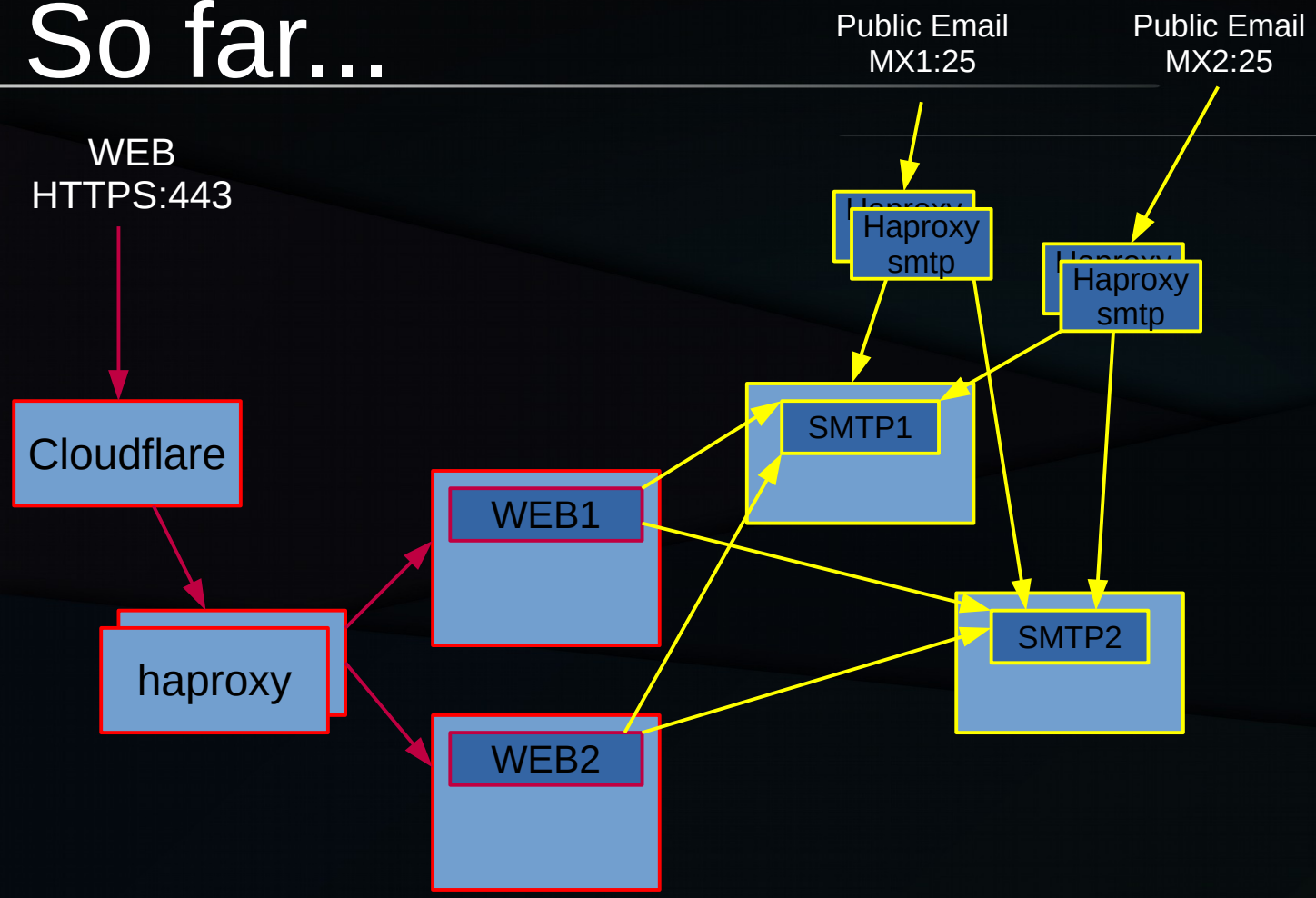
- Your own “Private Domain” - i.e. @mycompanytesting.com
  - Email is private to your team
  - Private emails aren’t stored in RAM – they stick around as long as you need them
  - Web Interface now shows all inboxes for an **entire domain**
  - Search inboxes with wildcards test*@mycompanydomain.com



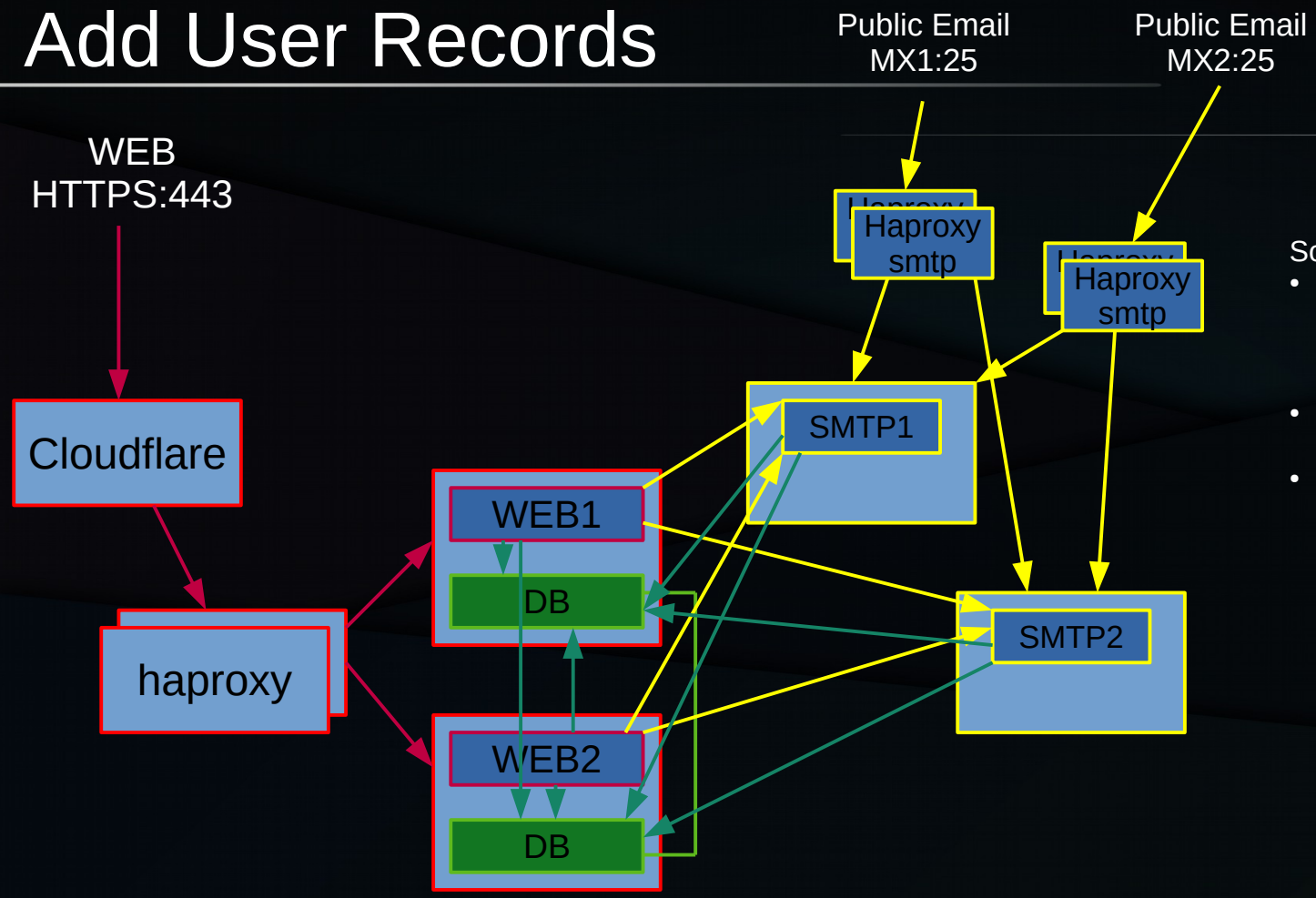
Subscriber Features

- API Access
- Private SMS testing numbers
  - Incoming SMS's arrive in "inboxes" of the phone number
- Rule System
  - Auto-route emails. Send them to Slack. Forward them. Auto-clicking of links.
- We need SignUp, and Login, and "forgot your password", and Uptime guarantees, Security, and Support
  - Sheesh… free services are way easier

So far...



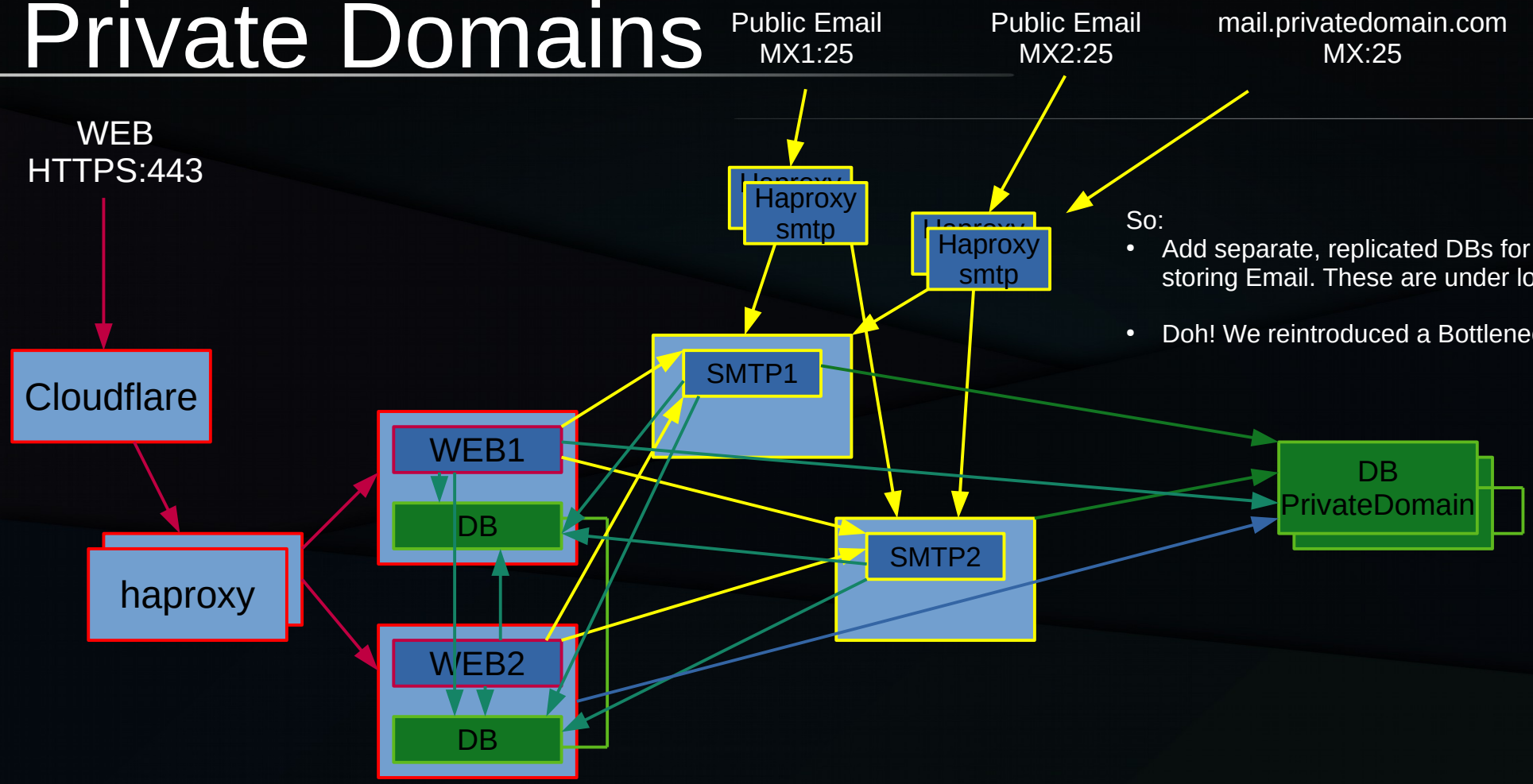
Add User Records



So:

- Add a DB cluster (redundant) for User records – signup, private domains, etc.
- Low traffic. Just logins and such.
- Connect it to, well, everything

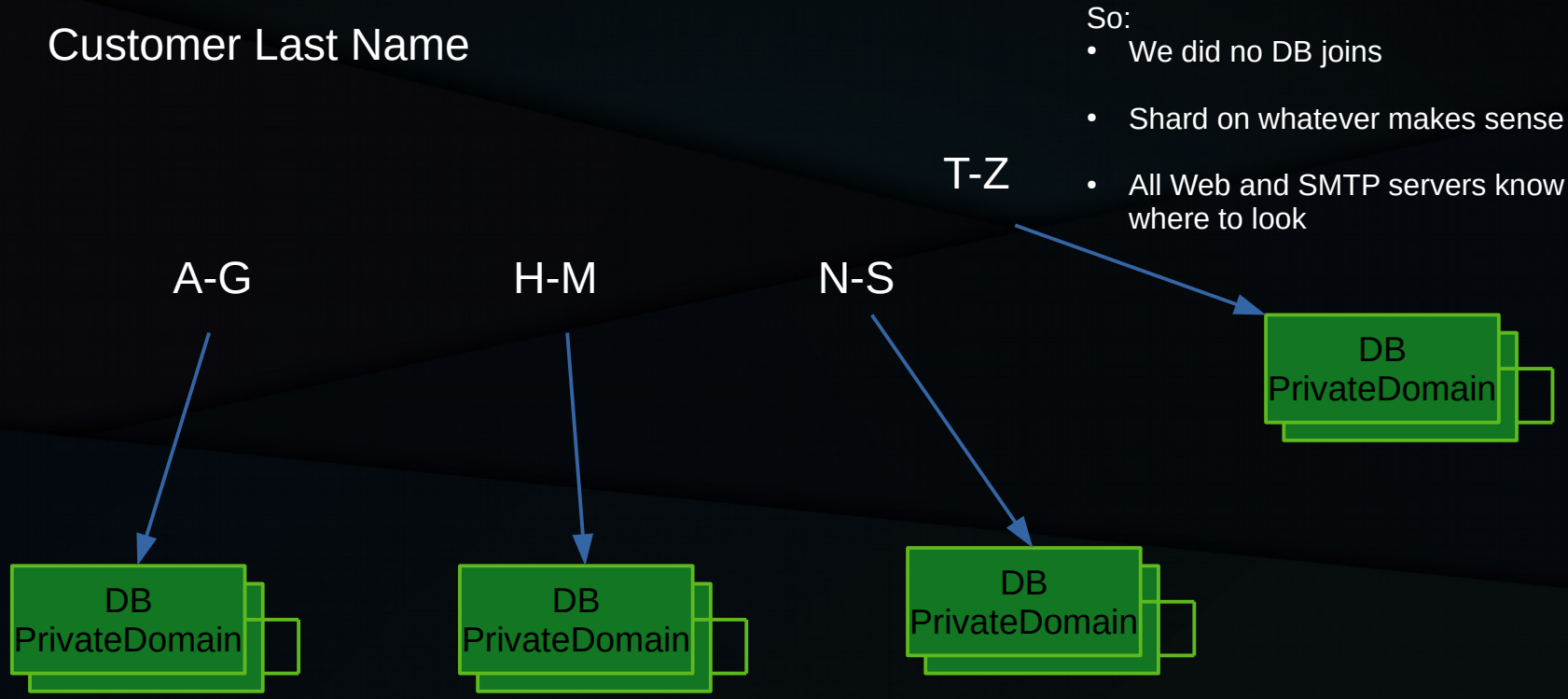
Private Domains



So:

- Add separate, replicated DBs for storing Email. These are under load
- Doh! We reintroduced a Bottleneck?

We Can Shard



So:

- We did no DB joins
- Shard on whatever makes sense
- All Web and SMTP servers know where to look

We Can Shard

Customer Usage Patterns

So:

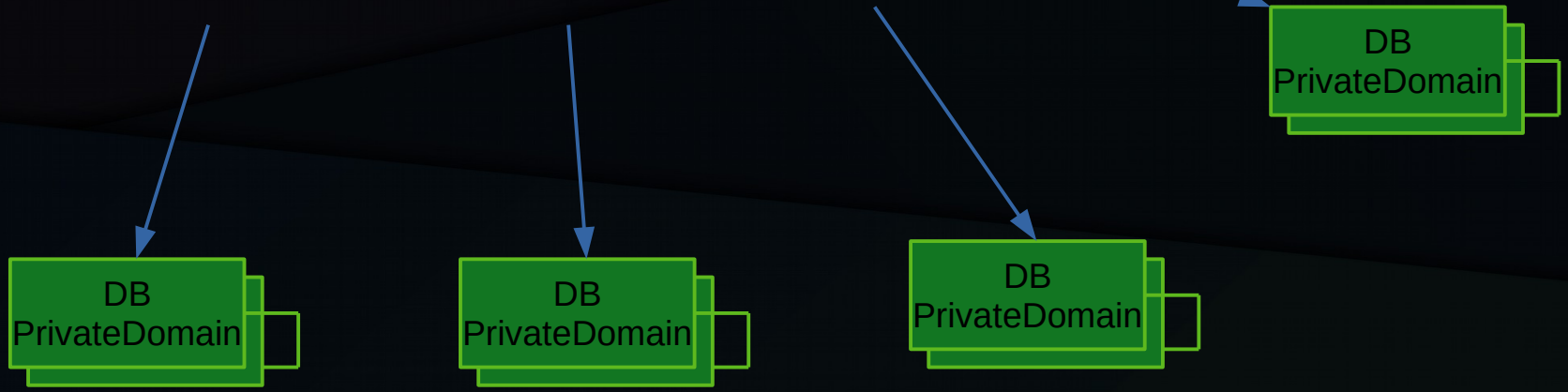
- We did no DB joins
- Shard on whatever makes sense
- All Web and SMTP servers know where to look

The Big Customer > 5MM/day

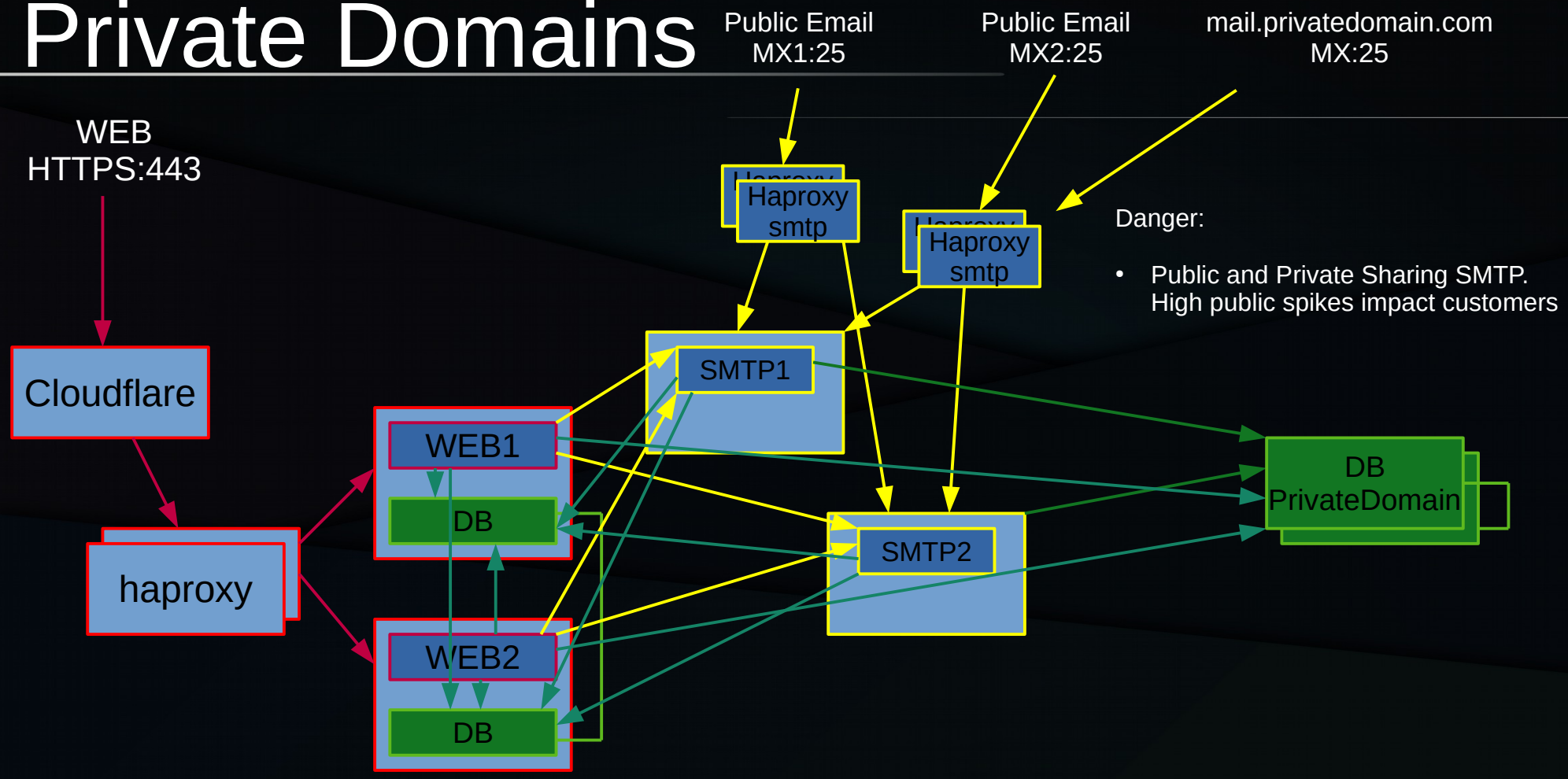
<1000 emails/day

1k-10k/day

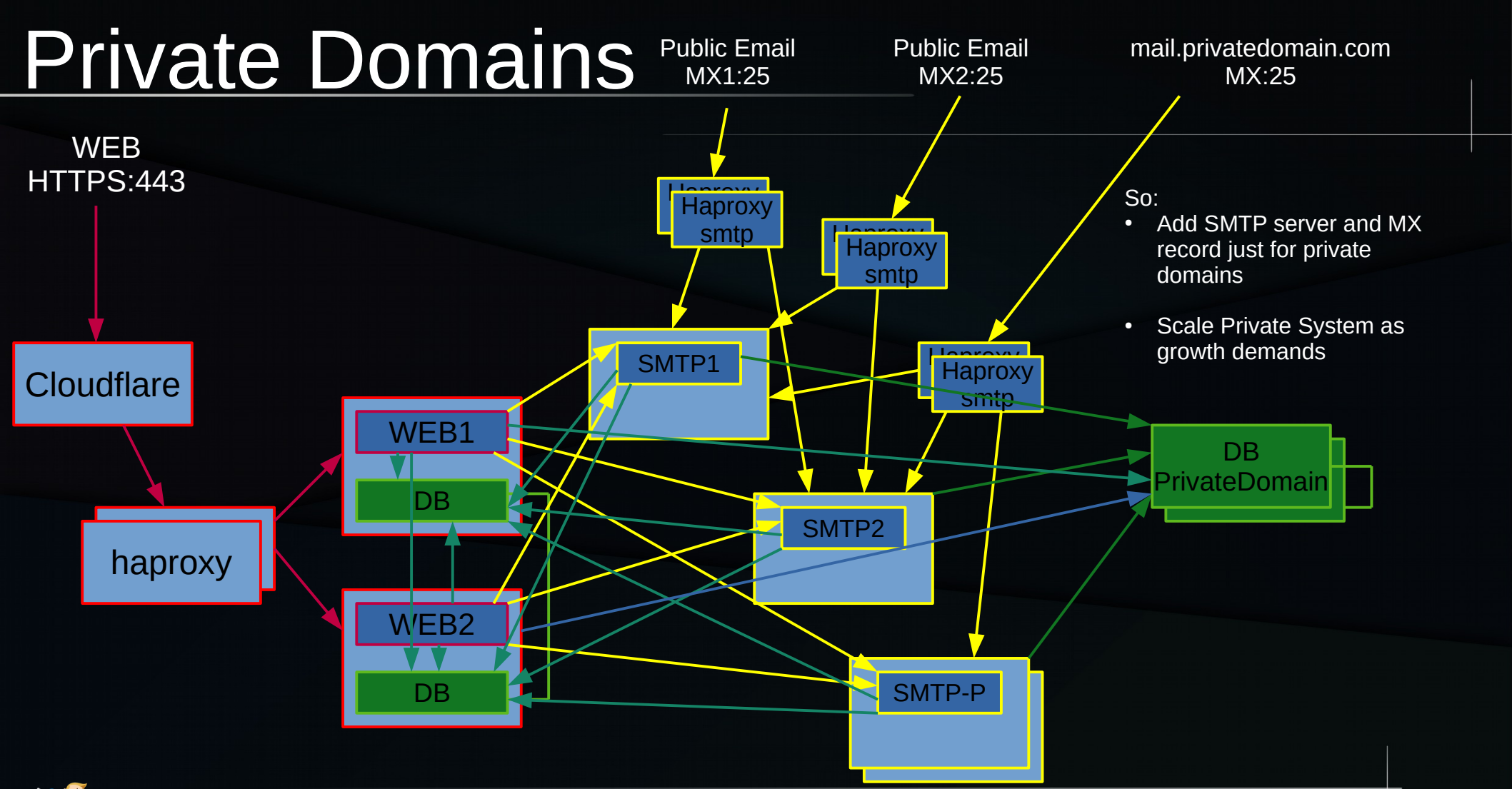
>100k/day



Private Domains



Private Domains



So:

- Add SMTP server and MX record just for private domains
- Scale Private System as growth demands

Everything is Great!

- Subscribers get their own Private Mailinator
- Public users still get their public system
- Danger: Every incoming email (i.e. thousands per second) must check database to see if THIS domain is a private one
  - Solution: Replicate Private Domain DB in memory

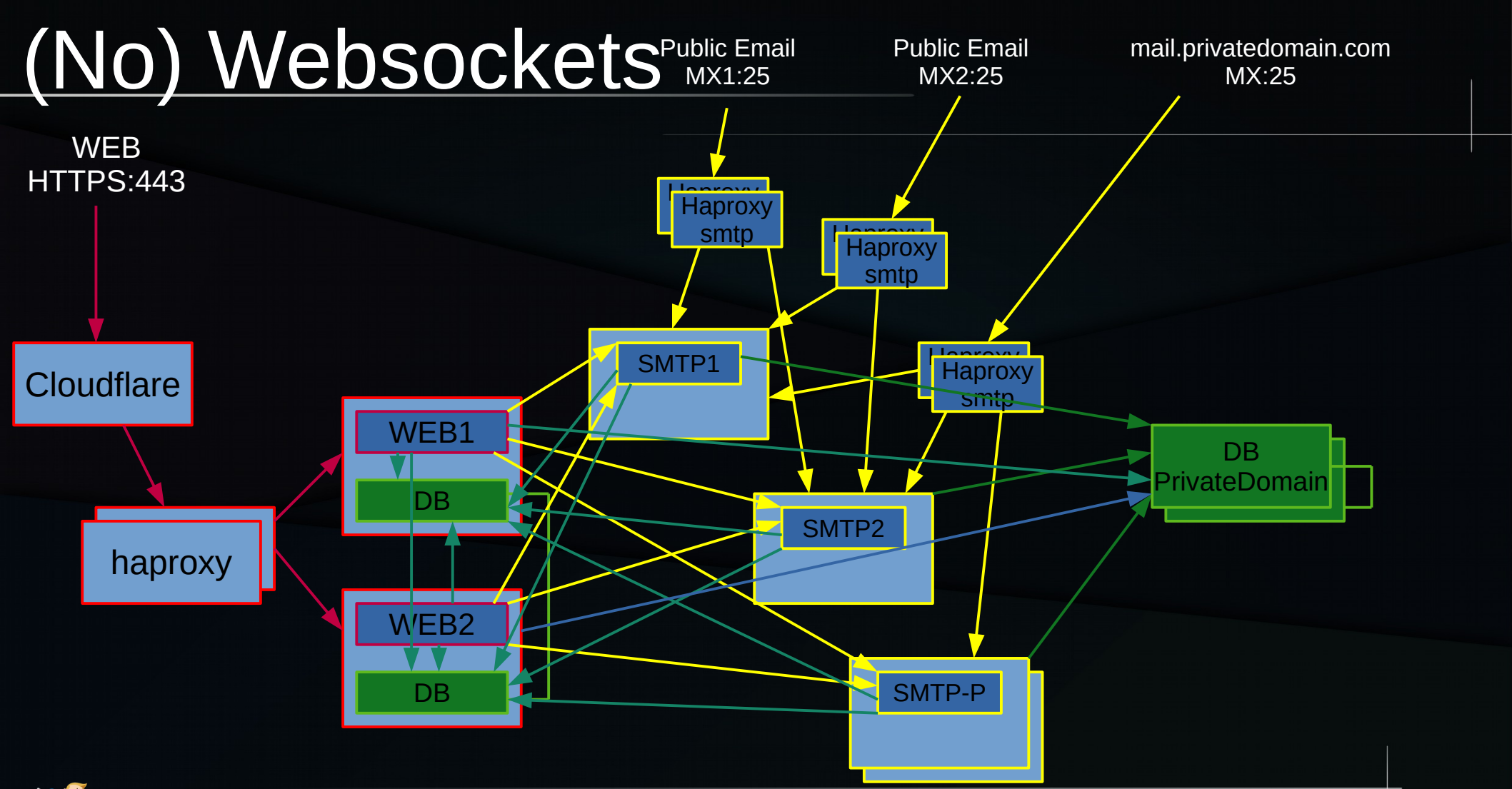
Big Idea: Websockets

- Currently
  - All clients “poll” the server for updates to the inbox (i.e. has new email arrived?)
  - If we have say, 2000 concurrent users who each poll an inbox every 5 seconds
    - 24,000 web hits a minute. 400 per second.
    - That’s a lot of useless web traffic
- Incoming email is a “stream”. Why not “tap into” that stream by specifying an inbox.
- Think of Mailinator as a “Email Twitter” where you “subscribe” to Inboxes
- Arriving email will INSTANTLY arrive in the web UI
  - Should we write our own pub/sub system? (Of course we should – and we do)
  - But eventually decide Redis on localhost has better features

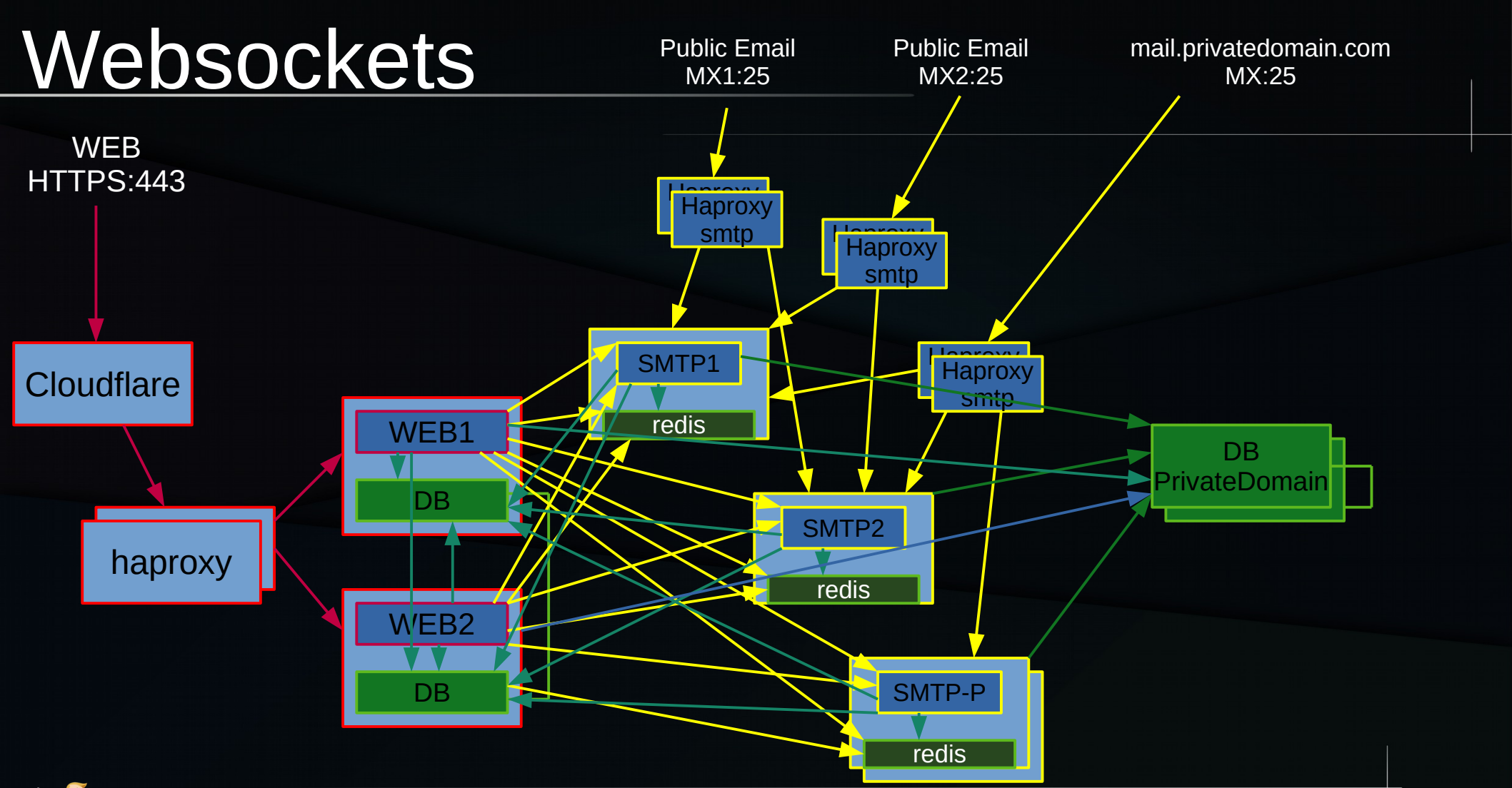
RealTime Inboxes !

- Each SMTP server gets it’s own Redis and publishes all “inbox snippets” of incoming email to Redis
  - To, From, Subject, Date
  - No email-body
  - No data leaves the SMTP server unless it’s subscribed to
  - We can abuse Redis as needed
    - Very robust software
    - Running on localhost – no bandwidth!

(No) Websockets



Websockets


Public Email
MX1:25
Public Email
MX2:25
mail.privatedomain.com
MX:25
WEB
HTTPS:443
Haproxy
smtp
Haproxy
smtp
Haproxy
smtp
Cloudflare
SMTP1
redis
WEB1
DB
DB
PrivateDomain
haproxy
SMTP2
redis
WEB2
DB
SMTP-P
redis

Everything is Great!

- I think?
- I mean, it seems like things are running good
- Actually I have no idea what's going on
  - We have zero system monitoring
- Need full system & application monitoring

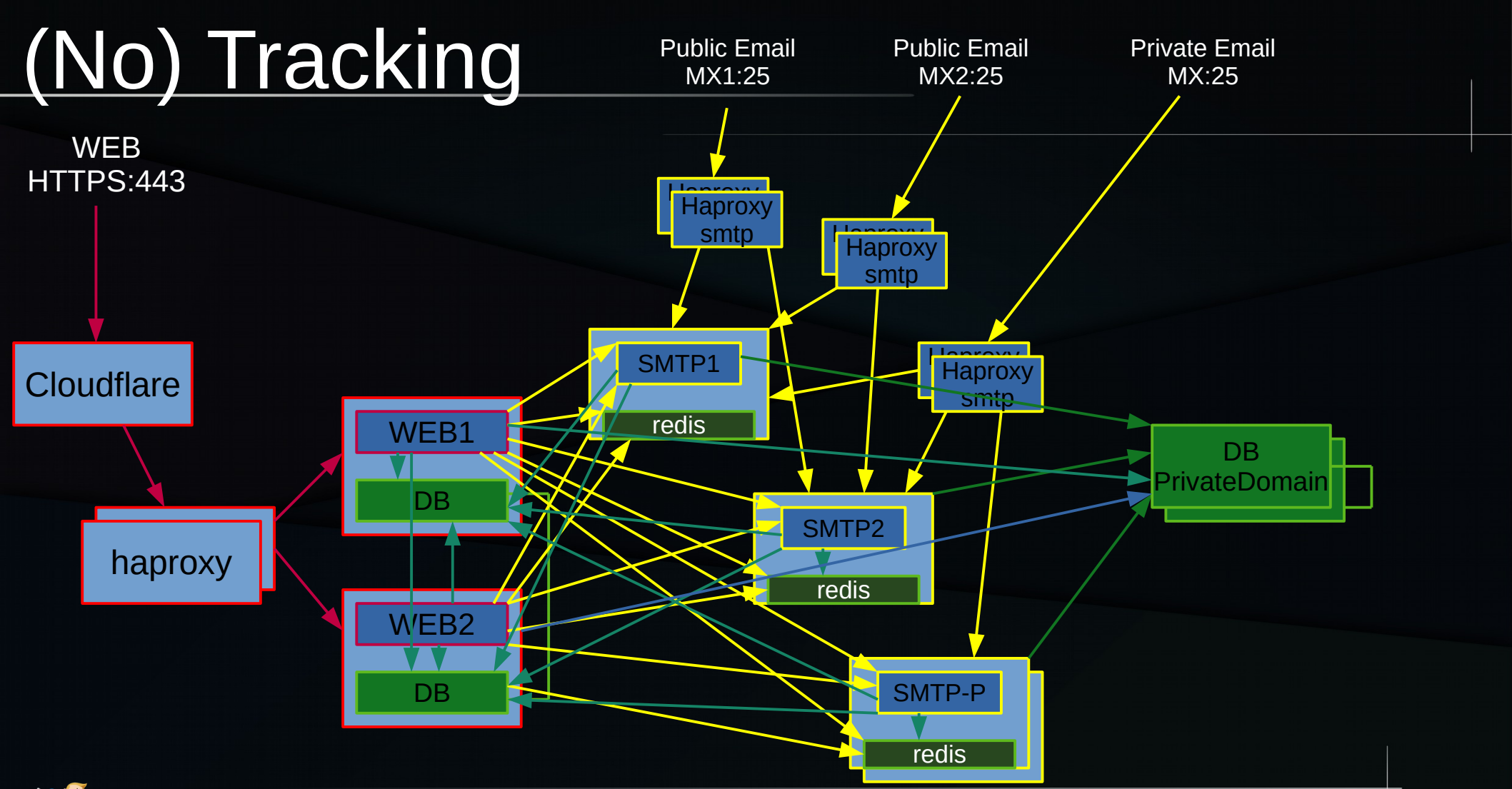
ELK

- Elasticsearch
- Logstash
- Kibana (and Grafana)

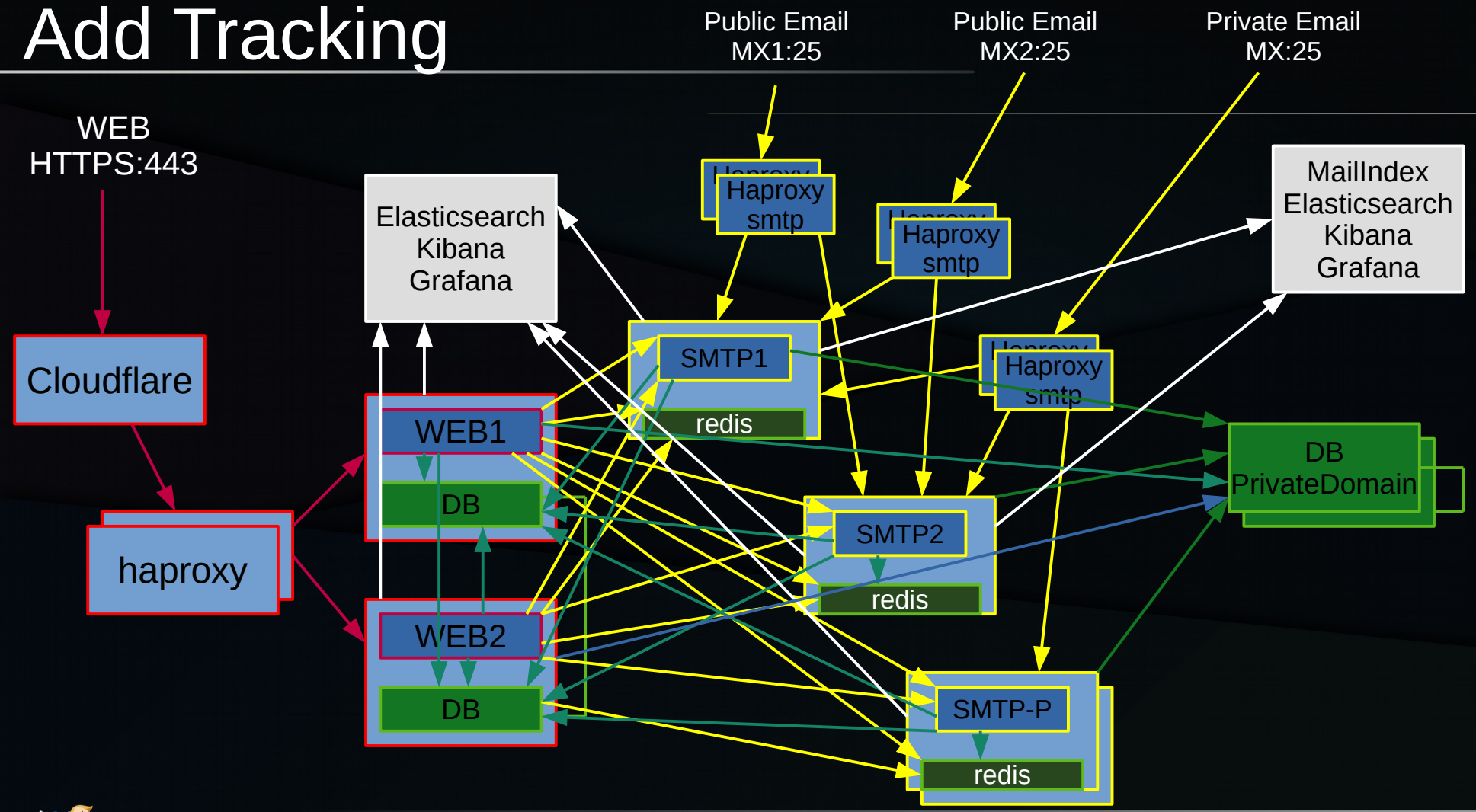
ELK-G

- ElasticSearch
  - A very cool, mostly free search engine for JSON data
- Logstash
  - A system where you take your structured program data, unstructure it into loglines, and logstash will re-structure it for you into JSON
  - That sounds dumb. Let’s not use that
- Kibana
  - Search engine front-end. Like Google with cool graphs (Elasticsearch GUI)
- Graphana
  - Like Kibana, but only graphs. Very powerful (another GUI for Elasticsearch)

(No) Tracking



Add Tracking



EKG

- Cycle all analytic data every few days
- Let’s see the data!

Mailinator Today

- Team of engineers, sales, marketing, support
- ~3-10MM emails/day
  - This is way down from 20-30MM a few years ago
  - Biggest Public spike seen:
    - ~200MM emails/day
    - 2000-3000 emails/second
      - Very hand wavey – how big of emails? How many per connection? How many SMTP commands?
- ~30,000-60,000 unique users/day
  - >100K when we hit the news

Mailinator Today

- Thousands of corporate users
- Millions of API hits/day
- All Java – few frameworks. SMTP is raw socket code
- Web: Raw Servlets/JSPs, Angular 1.0 + MailinatorFrontEnd™,Websockets

Disposable Email Games

(more stuff we learned)

Banning Mailinator.com emails

- What if they ban you?
  - They did… and how
  - At first – we took it personal and created “alternate domains”
    - Remember? Mailinator accepts email for any domain
    - @notmailinator.com
    - @reallymyemail.com
    - @veryrealemail.com
    - @putthisinyourspamdatabase.com
- “Hi, could I get a list of all your alternate domains?”
  - No.
  - We never showed the entire list
- Sites would scrape us every day to get a list of the alternate domains

Banning Mailinator.com emails

- Soon however, we realized this wasn't a "war" we wanted to win
- It's your site. If you want to block domains, that's your business
- But note, It's probably not completely possible
  - There's dozens/hundreds of disposable email sites now
  - Here's a list of ~23,000 domains to block to get you going

    `https://github.com/ivolo/disposable-email-domains/blob/master/index.json`

A lesson in defensive programming

- Your boss tells you to ban all @mailinator.com signups, so you write something like:

```
if (email.getDomain().equals(“mailinator.com”)) {
    // reject the signup attempt
}
```

A lesson in defensive programming

- Remember we said the Mailinator SMTP server accepts email ANY domain?
  - That includes subdomains of mailinator.com
  - So Public Mailinator not only accepts email for
    - <anything>@mailinator.com
  - It also accepts email for:
    - <anything>@<anything>.mailinator.com

A lesson in defensive programming

```
if (email.getDomain().endsWith(“mailinator.com”)) {
      // reject the signup attempt
   }
```

Healthcare.gov

- Many companies that specialize in signing people up for healthcare.gov use Mailinator addresses
  - i.e. non-technical people may not even have an email address nor know how to get one
  - Spike in November each year
  - Some subscribe for the month for privacy and to keep ownership of domain

hhhhhhhhhhhhh@mailinator.com

- May 3, 2017 – someone creates a Google App store application called “Google Docs”
  - Hint: it wasn’t google. It was just called that
- Then emails hundreds of people this:

---------- Forwarded message ----------
From: [redacted]
Date: Wed, May 3, 2017 at 1:29 PM
Subject: [redacted] has shared a document on Google Docs with you
To: hhhhhhhhhhhhhhhh@mailinator.com

[redacted] has invited you to view the following document:

Open in Docs

hhhhhhhhhhhh@mailinator.com

- They then get a message, “Google Docs wants to view your Contacts. Ok?”
- When they click OK, it emails all their contacts the same thing
- It spread across the Internet like wildfire

hhhhhhhhhhhh@mailinator.com

- Note that when it emailed their contacts, it put their entire contacts list in the BCC field
- But the program needed something in the “to” field.
  - They chose hhhhhhhhhh@mailinator.com
- That inbox got an email every time the virus spread
  - i.e. It received hundreds of thousands of emails that day

FIN