How Spotify Migrated Ingress HTTP Systems to Envoy

Alex Sundström,
Erik Lindblad,
Kateryna Nezdolii

Agenda

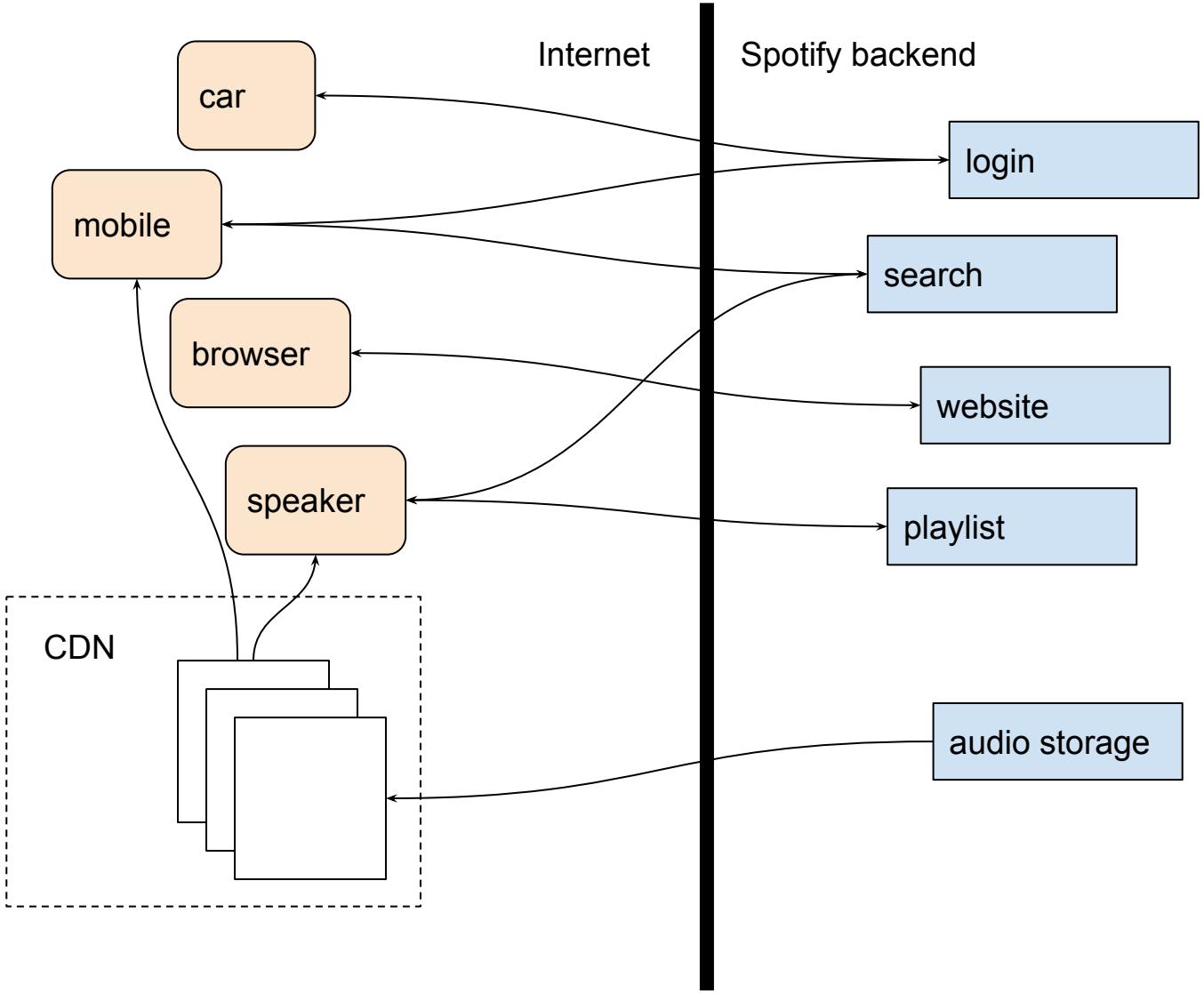
- Spotify’s old perimeter
- Spotify’s new perimeter
- How we migrated HTTP upstreams
- Learnings & problems encountered

About Spotify

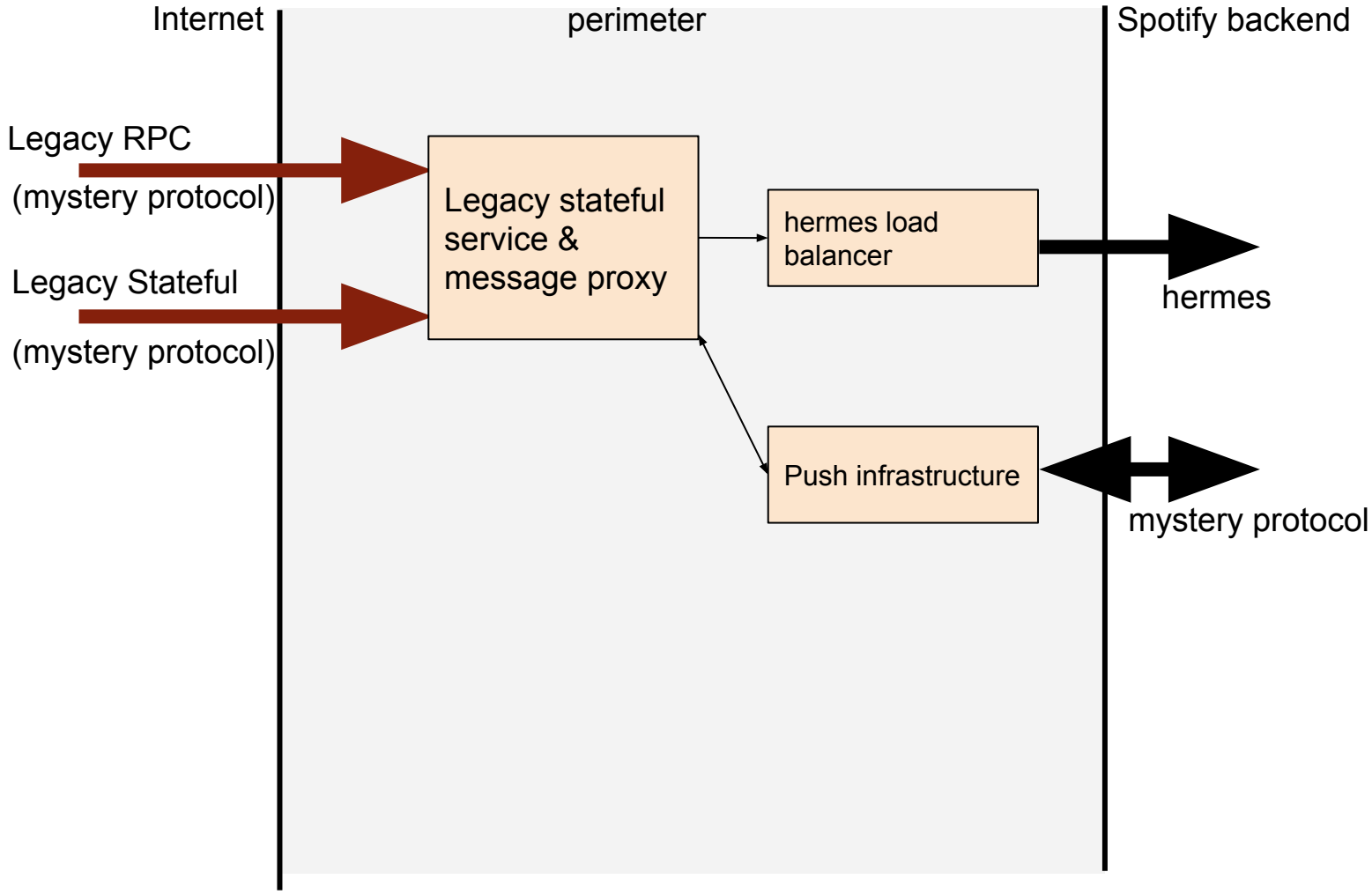
- 248 million MAU*
- Global product
- Hundreds of clients
- Service infra on GCP, 3 regions
- 8 million RPS
- “Micro”service backend

*q3 earnings call

Spotify

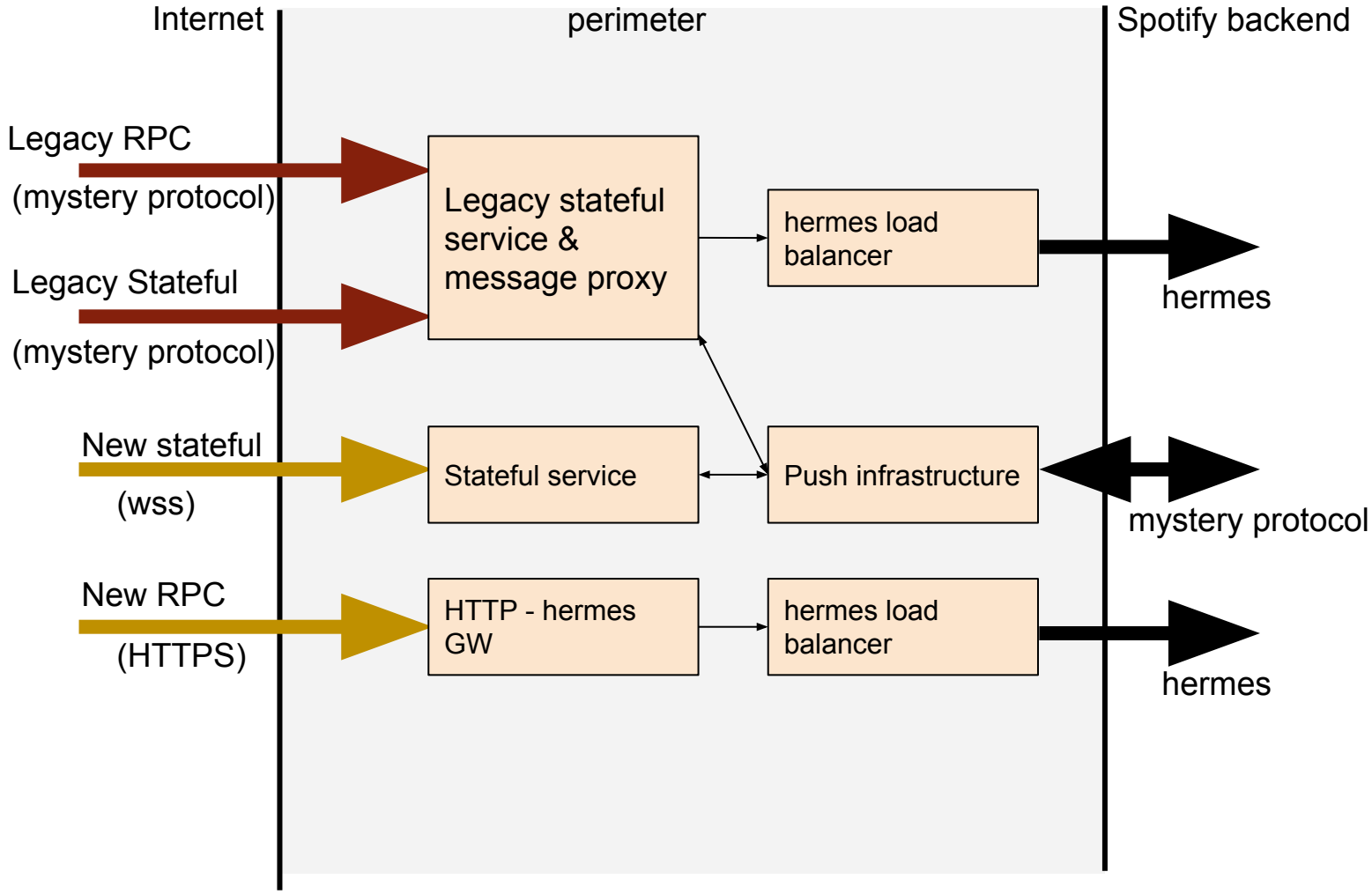

Internet
Spotify backend
car
mobile
browser
speaker
CDN
login
search
website
playlist
audio storage

Really old spotify perimeter



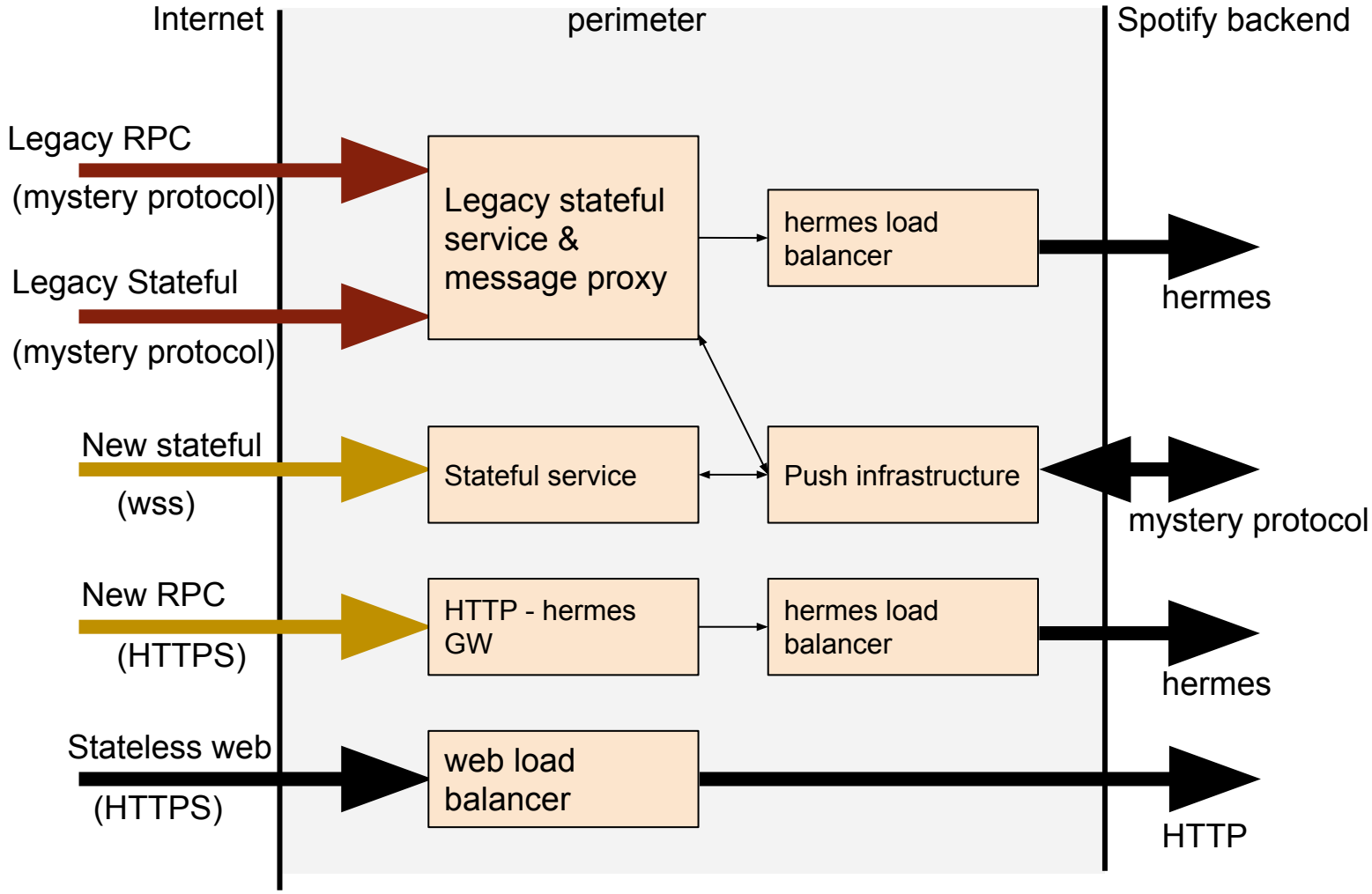
hermes = proprietary & legacy HTTP 1.1 like protocol

Old spotify perimeter



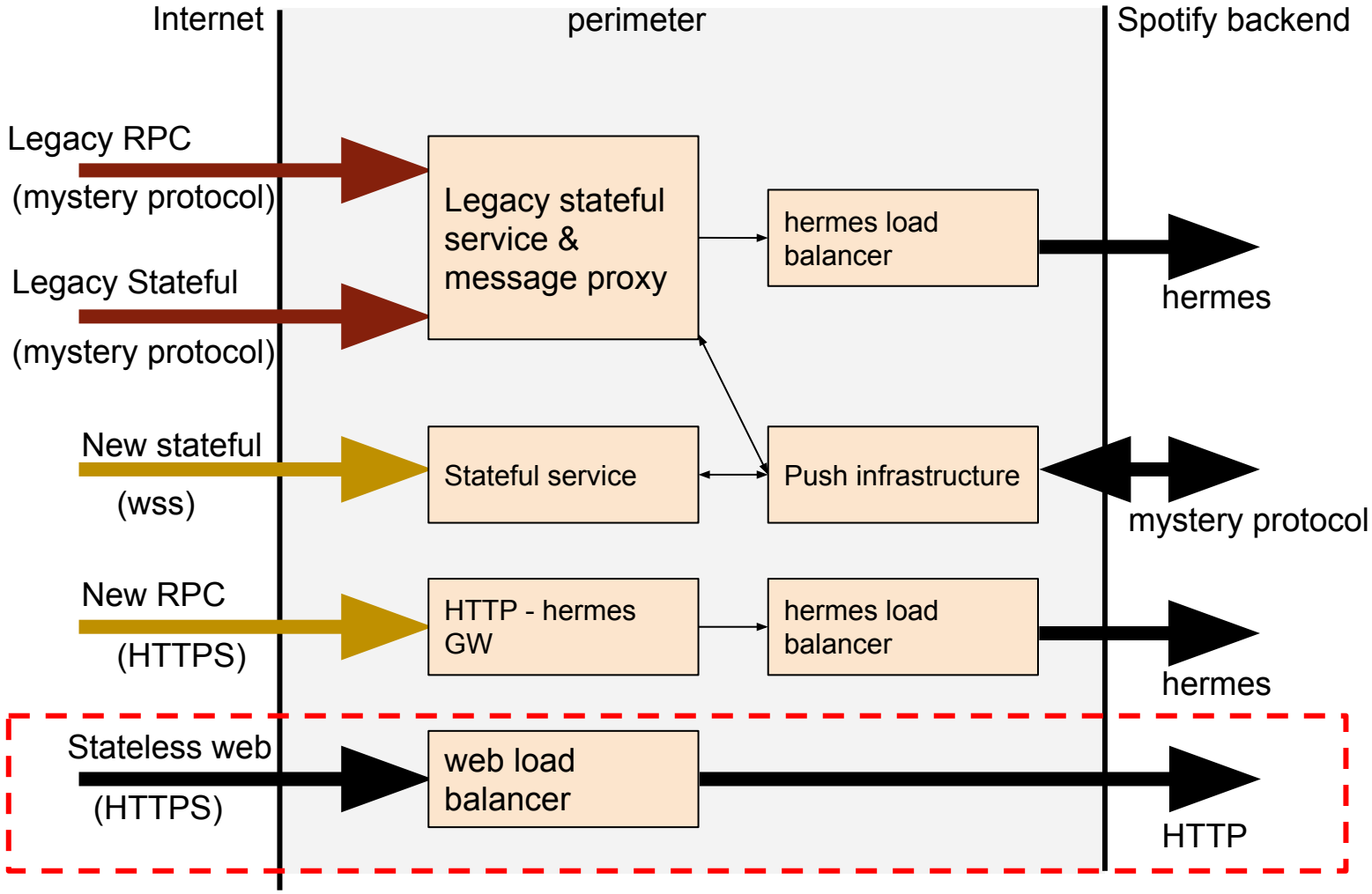
hermes = proprietary & legacy HTTP 1.1 like protocol

Old spotify perimeter



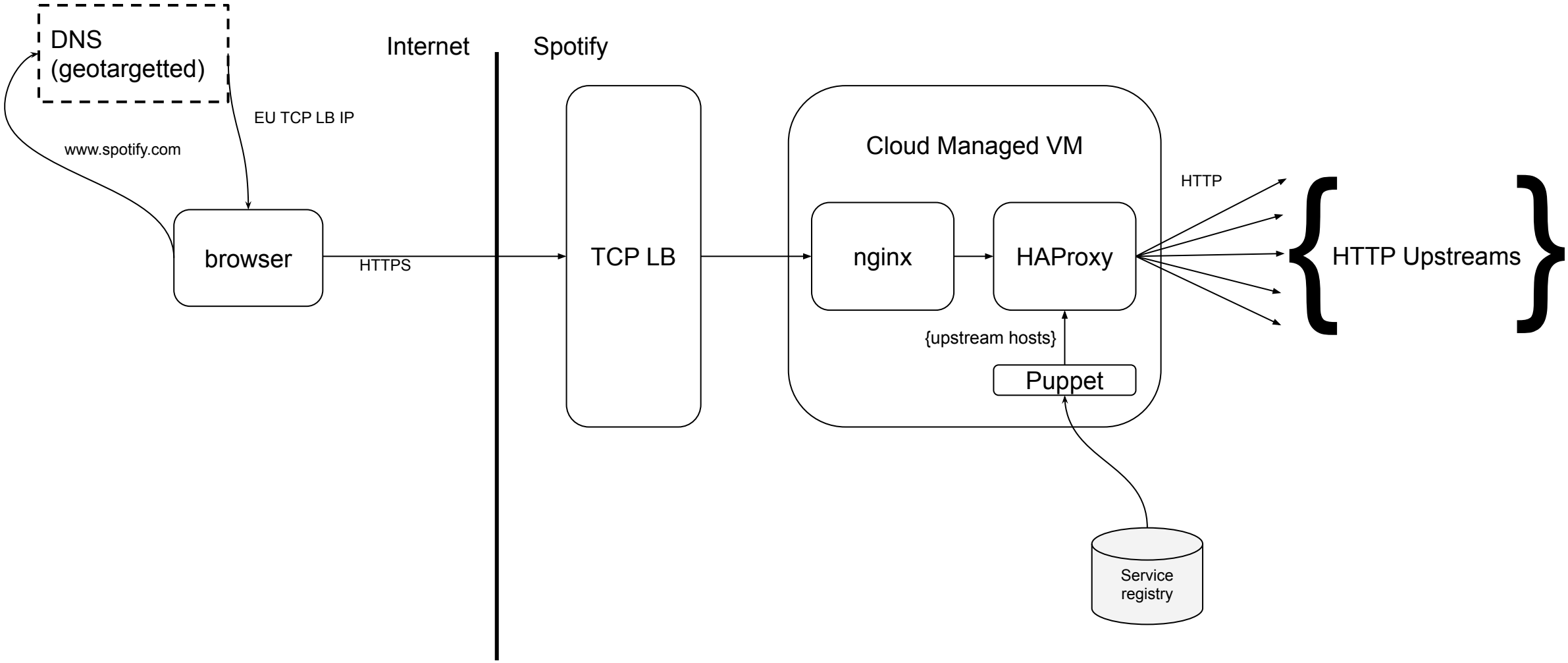
hermes = proprietary & legacy HTTP 1.1 like protocol

Old spotify perimeter



hermes = proprietary & legacy HTTP 1.1 like protocol

Old perimeter - HTTP upstreams



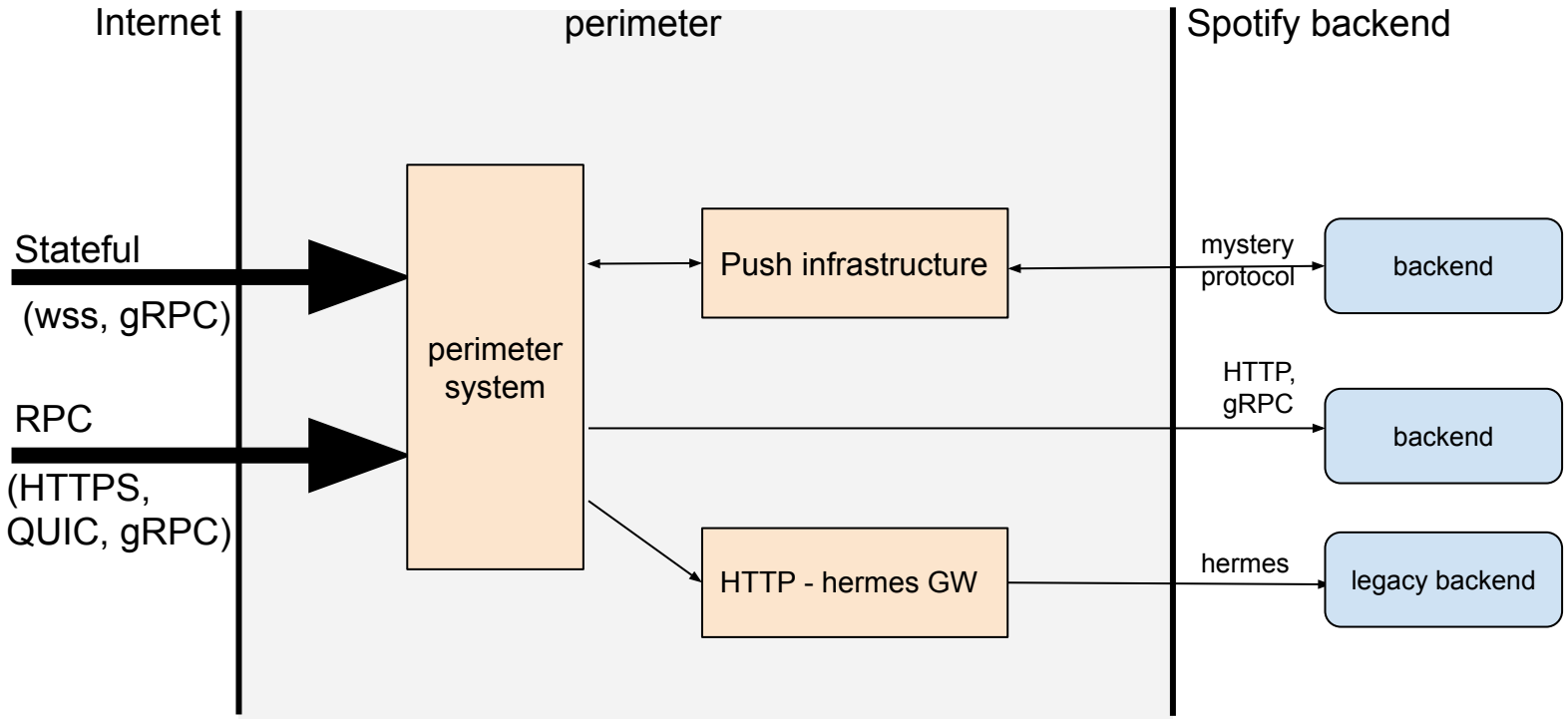
Why migrate?

Design of old system had come to its end-of-life

- Hard to extend
- Puppet based service discovery
- Unintuitive to use
- Investment would only benefit part of the perimeter

A unified perimeter

- One perimeter system doing all the perimeter things!

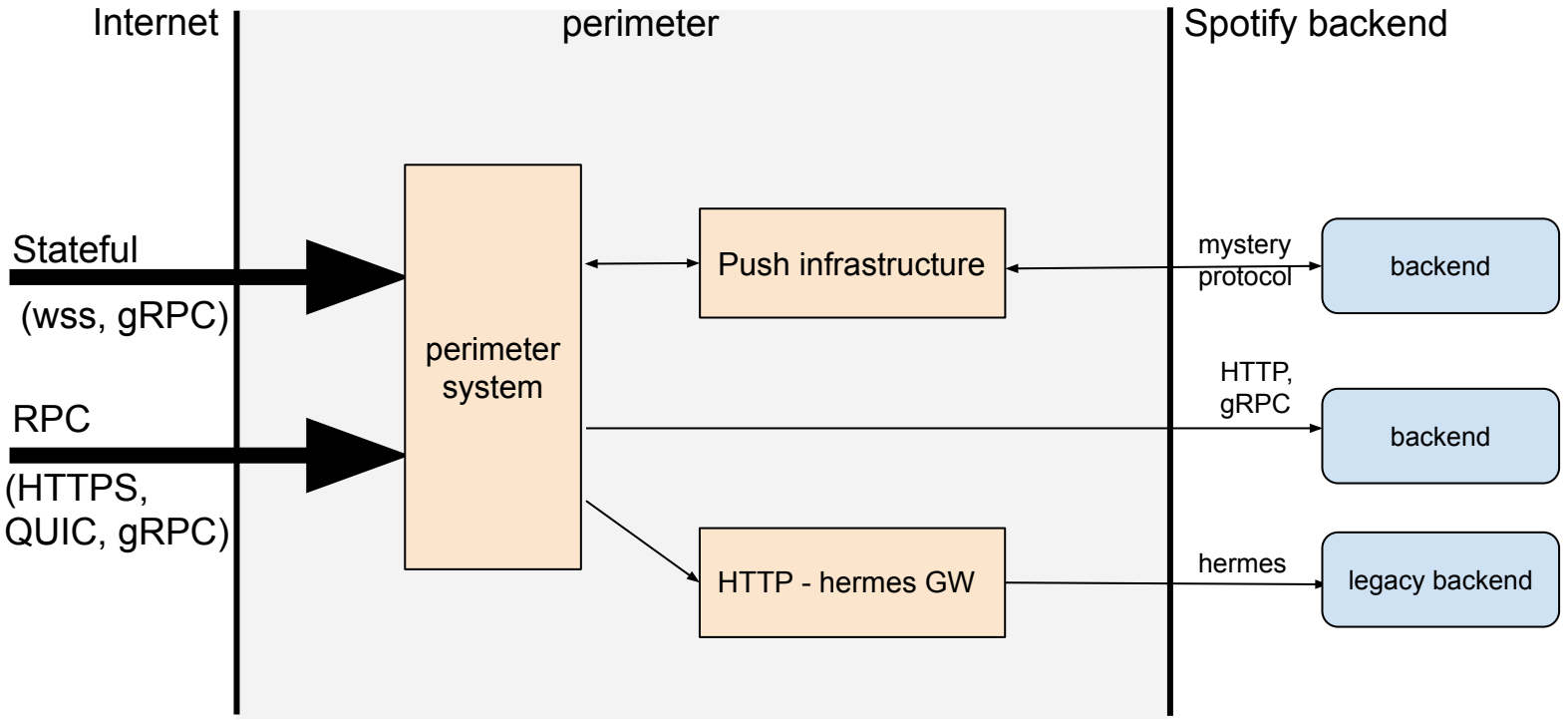


A unified perimeter

- One perimeter system doing all the perimeter things!

e.g.

- Authorization
- Rate limiting
- Metrics
- Protocol support
- Routing
- Security hardening
- Audit trail

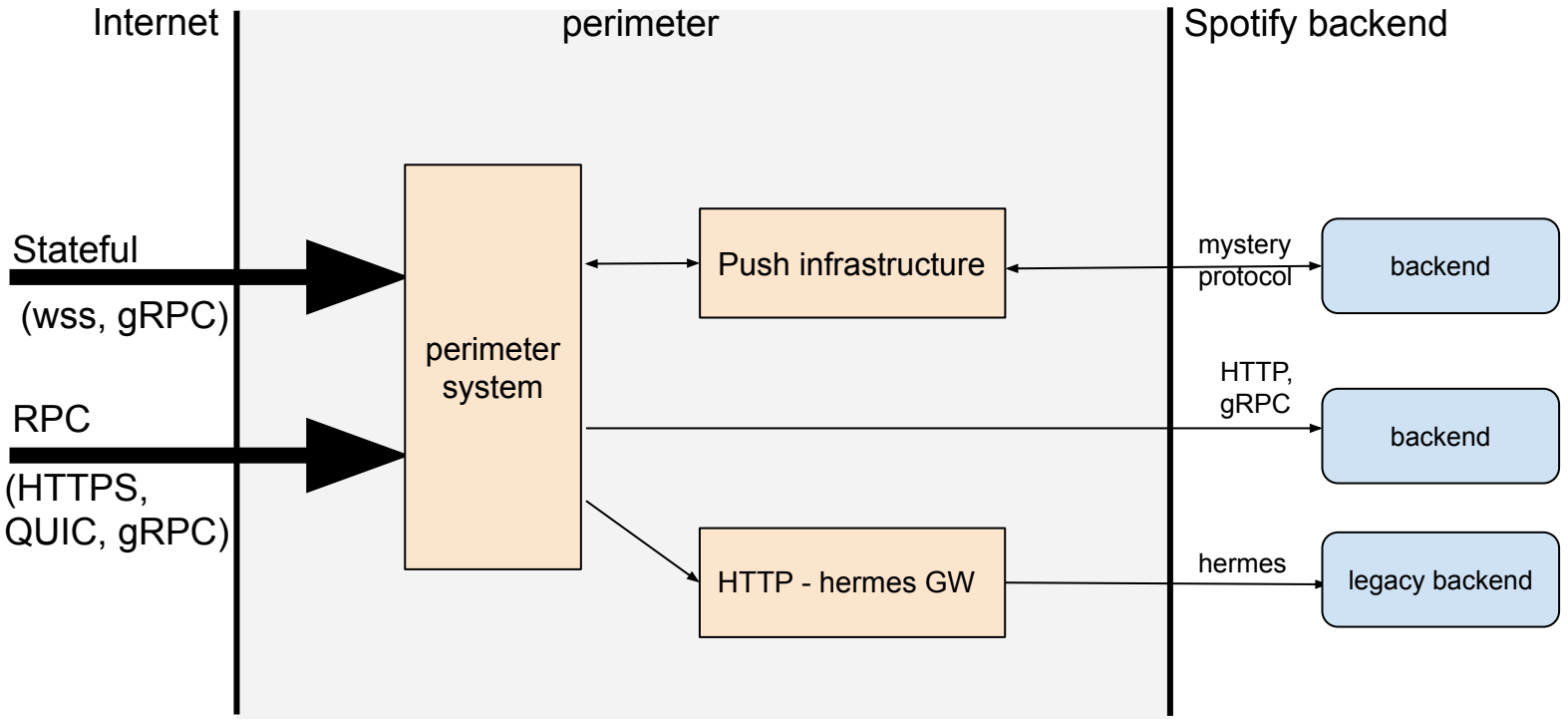


A unified perimeter

- One perimeter system doing all the perimeter things!

e.g.

- Authorization
- Rate limiting
- Metrics
- Protocol support
- Routing
- Security hardening
- Audit trail

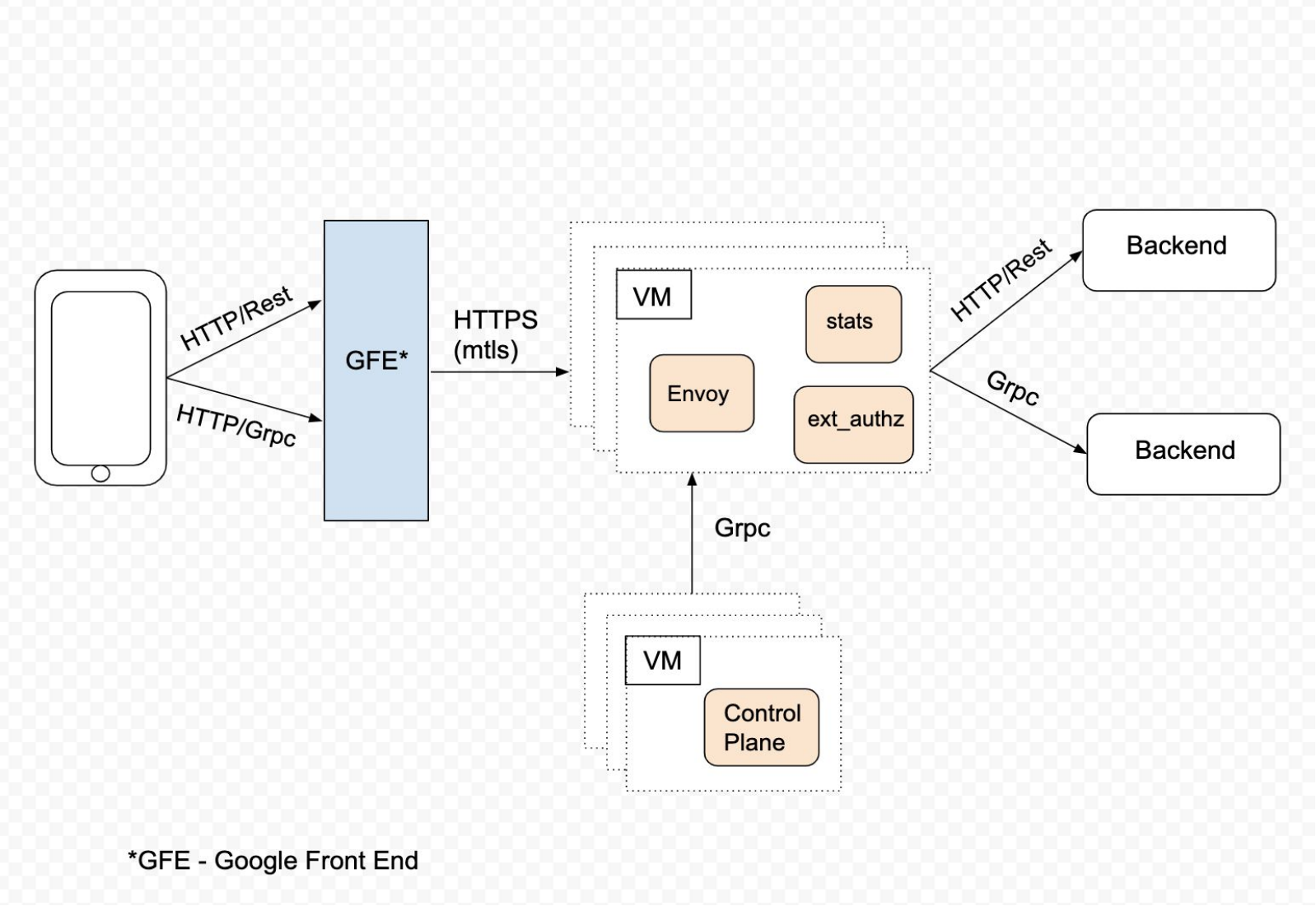


- Avoid fragmentation and feature duplication

Why Envoy

- Moving up the stack + adopting CNCF tech
- XDS
- OSS
- Responsive community

New Edge overview



New Edge risks

- Bad deploys
- Shared resources
- Potential cloud load balancing infra outages

Migration - Goals

- Transparent for up- & downstreams
- Zero downtime
- Gradual migration
- Safety switch

Migration - First step

- Start small
- Service we operate for experiments
- DNS change

Migration - More traffic

- Webplayer next (open.spotify.com)
- Gradual rollout strategy
- L7 Cloud LB to trickle traffic
- Run during office hours

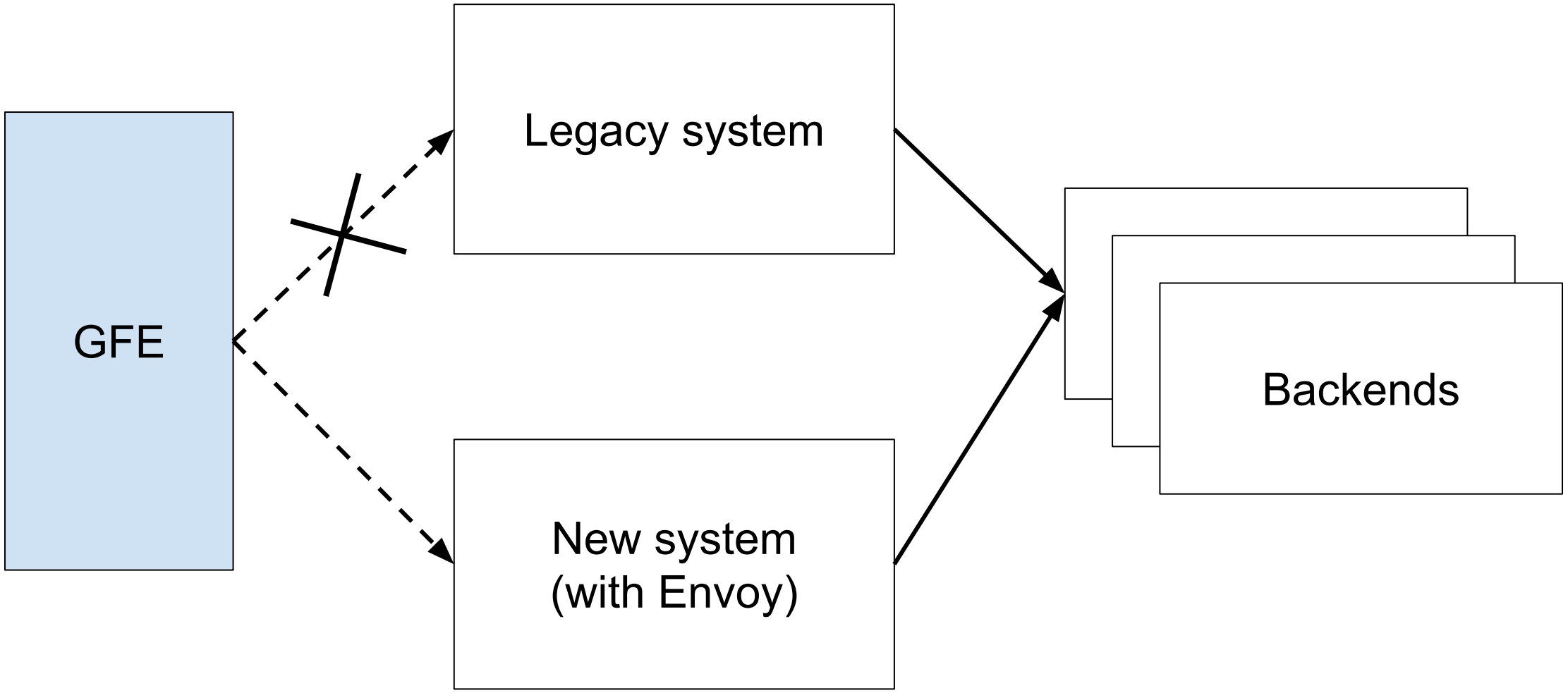
Migration - What was left?

- What features do we need to port?
- Reduce configuration surface?
- Monitoring & alerting

Migration - What we did

- Fewer configuration parameters
- Ported most services with script
- “Either or” rollout of all traffic

Migration - What we did



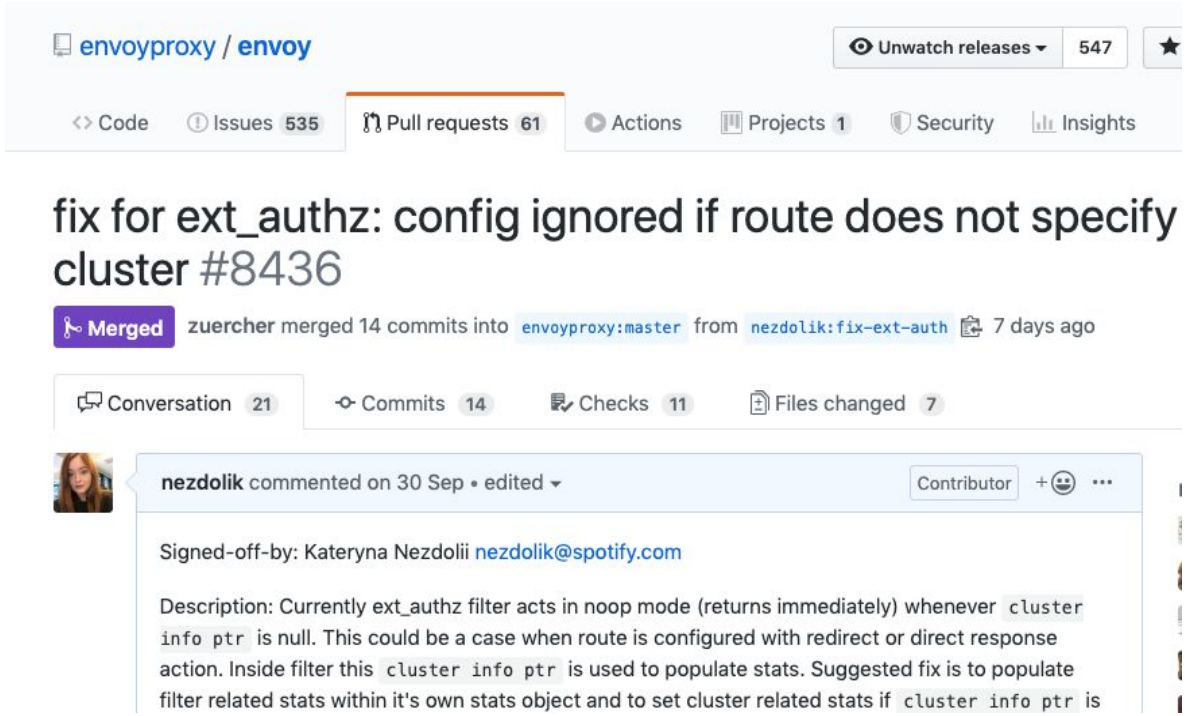
Migration - How it went

- Serve same traffic with fewer cores
- Ready for Spotify hack week
- Fast fallback was very useful

Learnings and Failures

- “Almost zero downtime”
- Grasping failure modes
- Websockets did not work out-of-the-box
- Performance was a surprise

Some Envoy improvements



https://github.com/envoyproxy/envoy/pull/8436

https://github.com/envoyproxy/envoy/pull/8989

Future plans

- Moving rest of traffic behind Edge
- Run on K8s
- Remove header decoration from ext authz
- Performance tuning
- More authz schemes
- Rate limiting
- Explore adaptive concurrency for reliability

Thank you!

envoycon

Please reach out to us at:

Kateryna Nezdolii	Alex Sundström	Erik Lindblad
nezdolik@spotify.com	asundstrom@spotify.com	erili@spotify.com
https://github.com/nezdolik