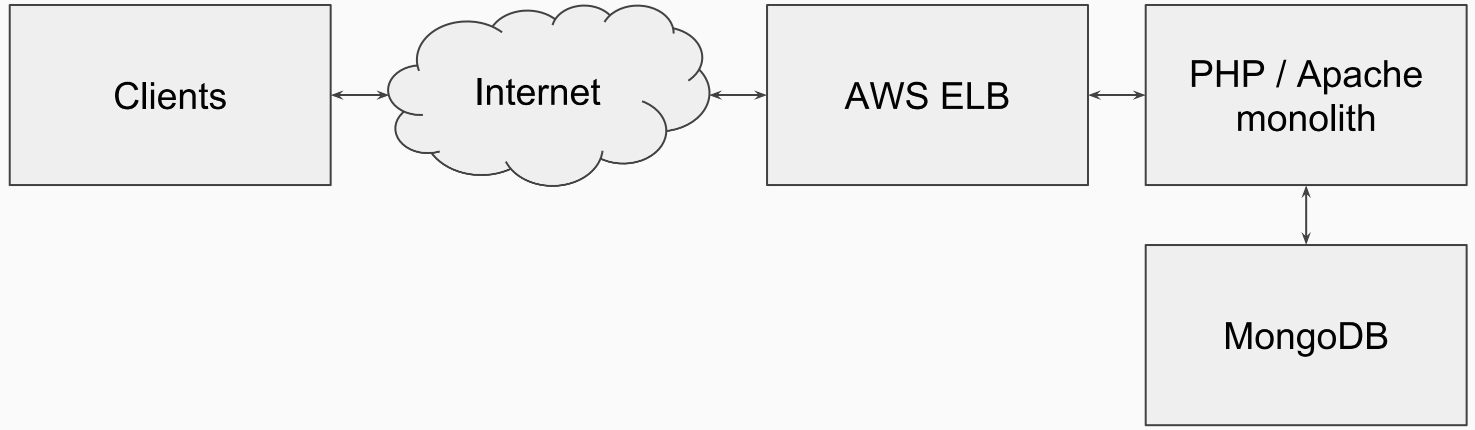
envoy by lyft

Lyft's Envoy: Experiences Operating a Large Service Mesh

SREcon17

Matt Klein / @mattklein123, Software Engineer @Lyft

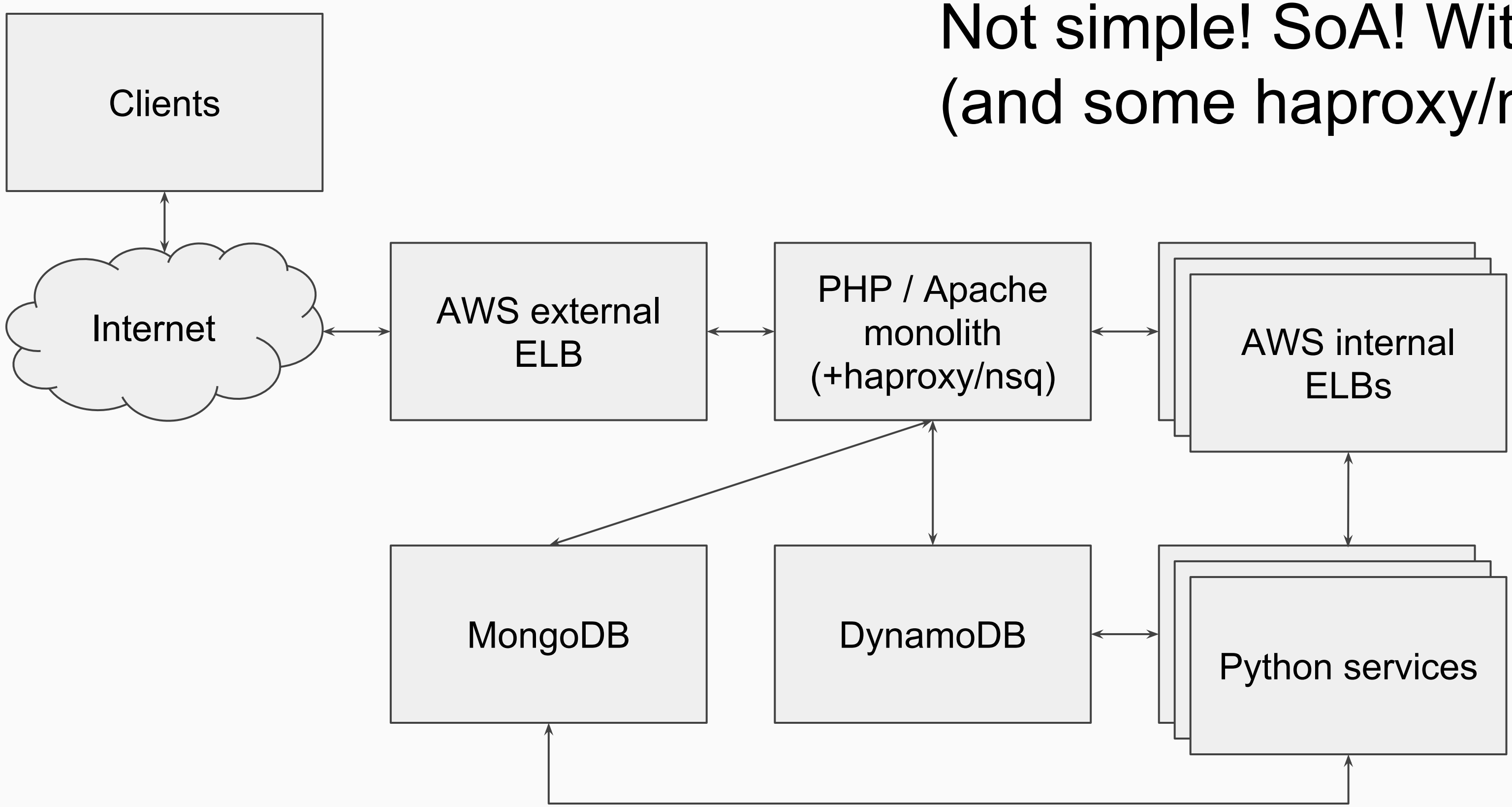
Lyft ~4 years ago



Simple! No SoA! (*but still not that simple*)

Lyft ~2 years ago

Not simple! SoA! With monolith!
(and some haproxy/nsq)



State of SoA networking in industry

- **Languages** and frameworks.
- **Protocols** (HTTP/1, HTTP/2, gRPC, databases, caching, etc.).
- **Infrastructures** (IaaS, CaaS, on premise, etc.).
- Intermediate **load balancers** (AWS ELB, F5, etc.).
- **Observability** output (stats, tracing, and logging).
- Implementations (often partial) of **retry**, **circuit breaking**, **rate limiting**, **timeouts**, and other distributed systems best practices.
- **Authentication** and **Authorization**.
- Per language **libraries** for service calls.

State of SoA networking in industry

A really big and confusing mess...

What is Envoy

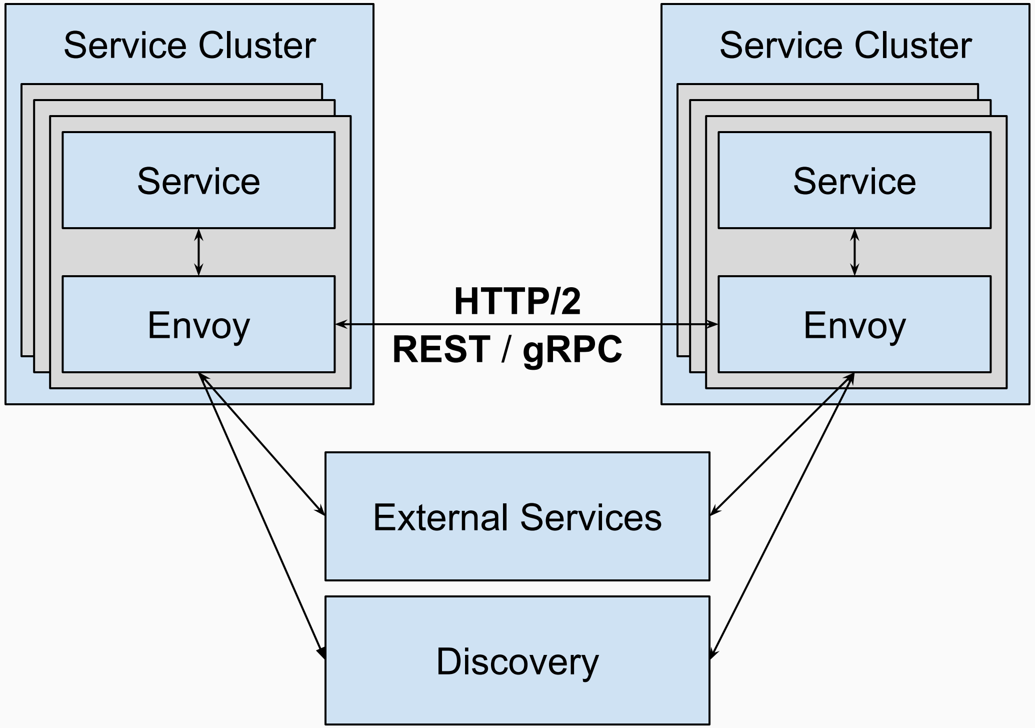
The network should be transparent to applications. When network and application problems do occur it should be easy to determine the source of the problem.

This sounds great! But it turns out it’s really, really hard.

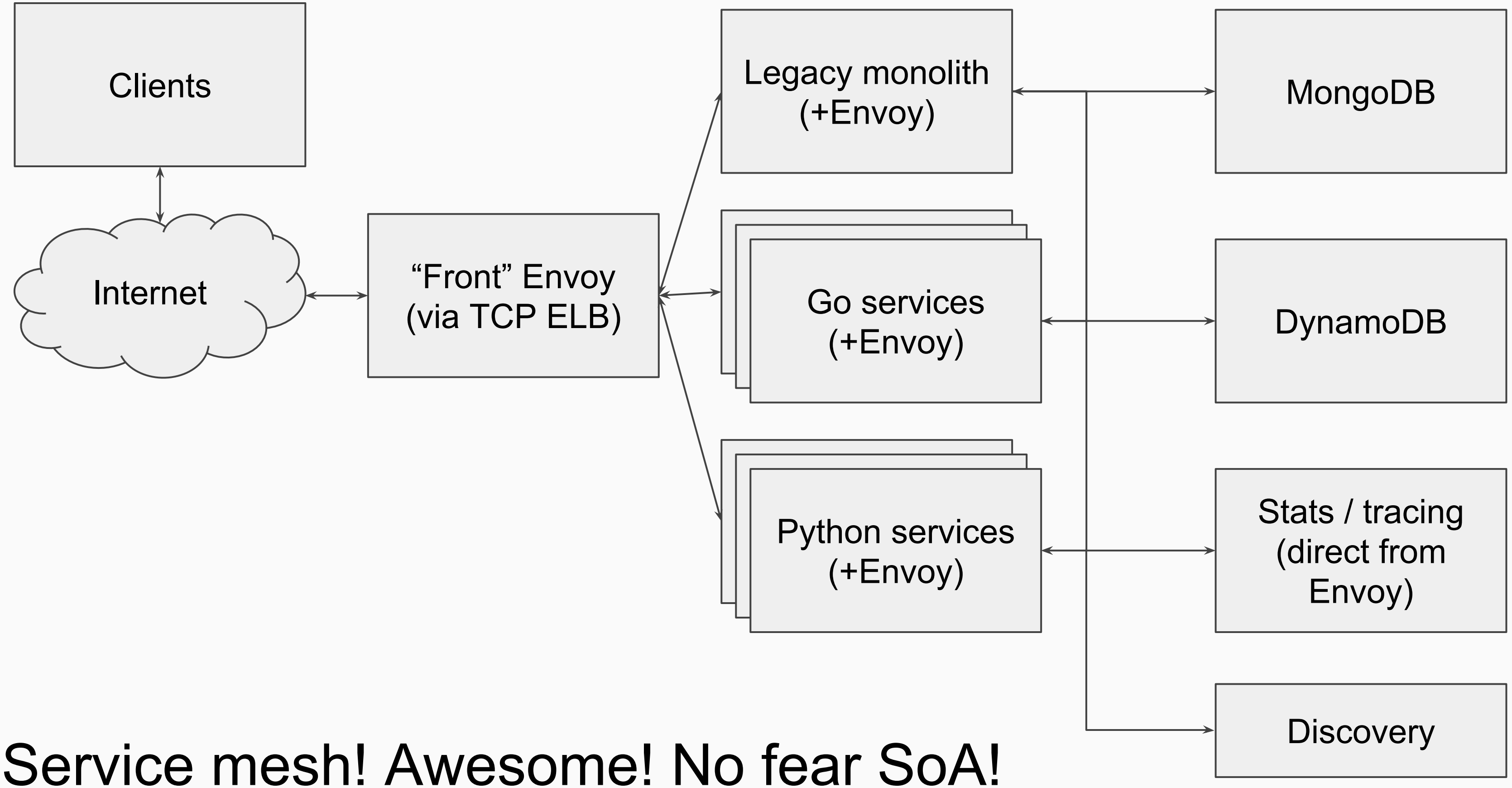
What is Envoy

- **Out of process architecture**: Let’s do a lot of really hard stuff in one place and allow application developers to focus on business logic.
- **Modern C++11 code base**: Fast and productive.
- **L3/L4 filter architecture**: A byte proxy at its core. Can be used for things other than HTTP (e.g., MongoDB, redis, stunnel replacement, TCP rate limiter, etc.).
- **HTTP L7 filter architecture**: Make it easy to plug in different functionality.
- **HTTP/2** first! (Including **gRPC** and a nifty gRPC HTTP/1.1 bridge).
- **Service discovery** and **active/passive health checking.**
- **Advanced load balancing**: Retry, timeouts, circuit breaking, rate limiting, shadowing, outlier detection, etc.
- Best in class **observability**: stats, logging, and tracing.
- **Edge proxy**: routing and TLS.

Envoy service to service topology



Lyft today



Service mesh! Awesome! No fear SoA!

Eventually consistent service discovery

- **Fully consistent** service discovery systems are very popular (ZK, etcd, consul, etc.).
- In practice they are **hard to run at scale**.
- Service discovery is actually an **eventually consistent** problem. Let's recognize that and design for it.
- Envoy is designed from the get go to treat **service discovery as lossy**.
- Active health checking used in combination with service discovery to produce a **routable overlay**.

Discovery Status	**HC OK**	**HC Failed**
Discovered	Route	Don't Route
Absent	Route	Don't Route / Delete

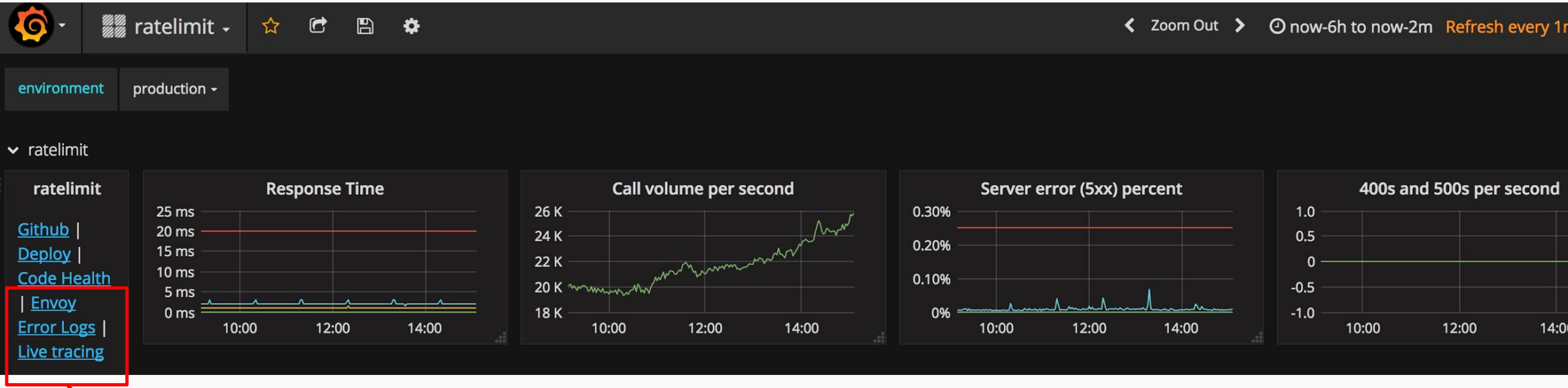
Advanced load balancing

- Different **service discovery** types.
- Zone aware least request **load balancing**.
- **Dynamic stats**: Per zone, canary specific stats, etc.
- **Circuit breaking**: Max connections, requests, and retries.
- **Rate limiting**: Integration with global rate limit service.
- **Shadowing**: Fork traffic to a test cluster.
- **Retries**: HTTP router has built in retry capability with different policies.
- **Timeouts**: Both “outer” (including all retries) and “inner” (per try) timeouts.
- **Outlier detection**: Consecutive 5xx
- **Deploy control**: Blue/green, canary, etc.
- **Fault injection**

Observability

- **Observability** is by far the most important thing that Envoy provides.
- Having all SoA traffic transit through Envoy gives us a single place where we can:
  - Produce consistent **statistics** for every hop
  - Create and propagate a stable **request ID / tracing context**
  - Consistent **logging**
  - Distributed **tracing**

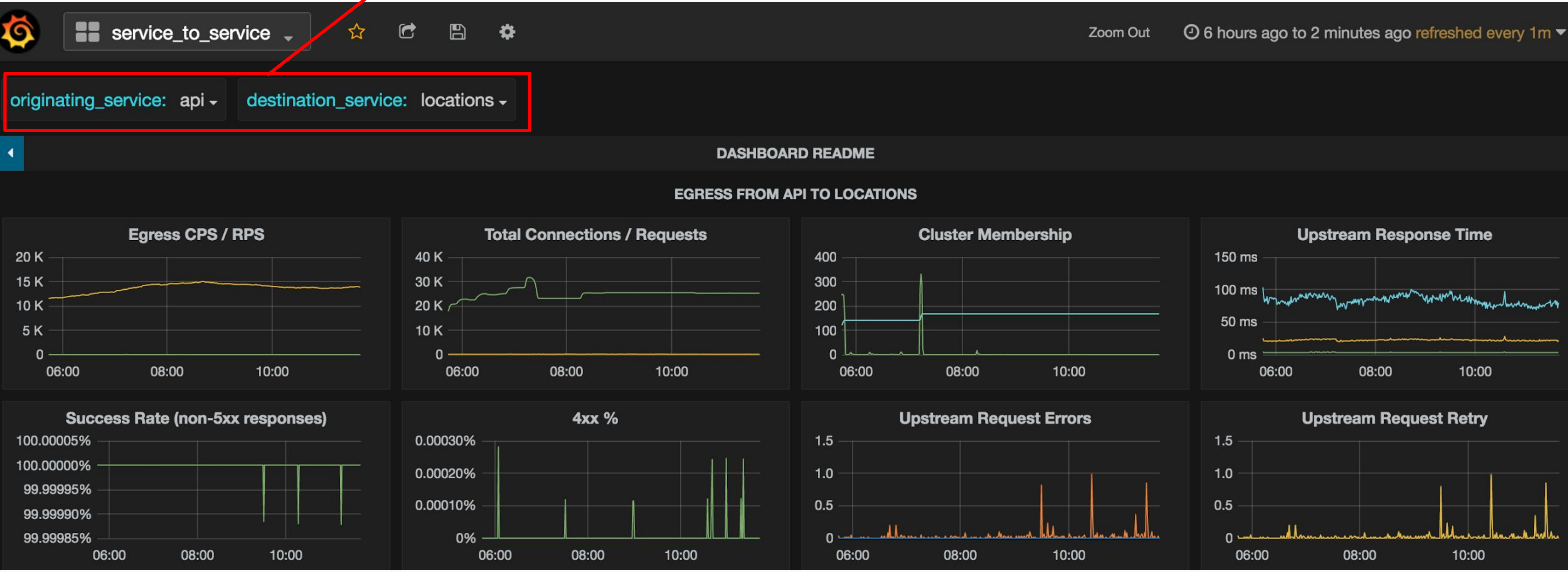
Observability: Per service auto-generated panel



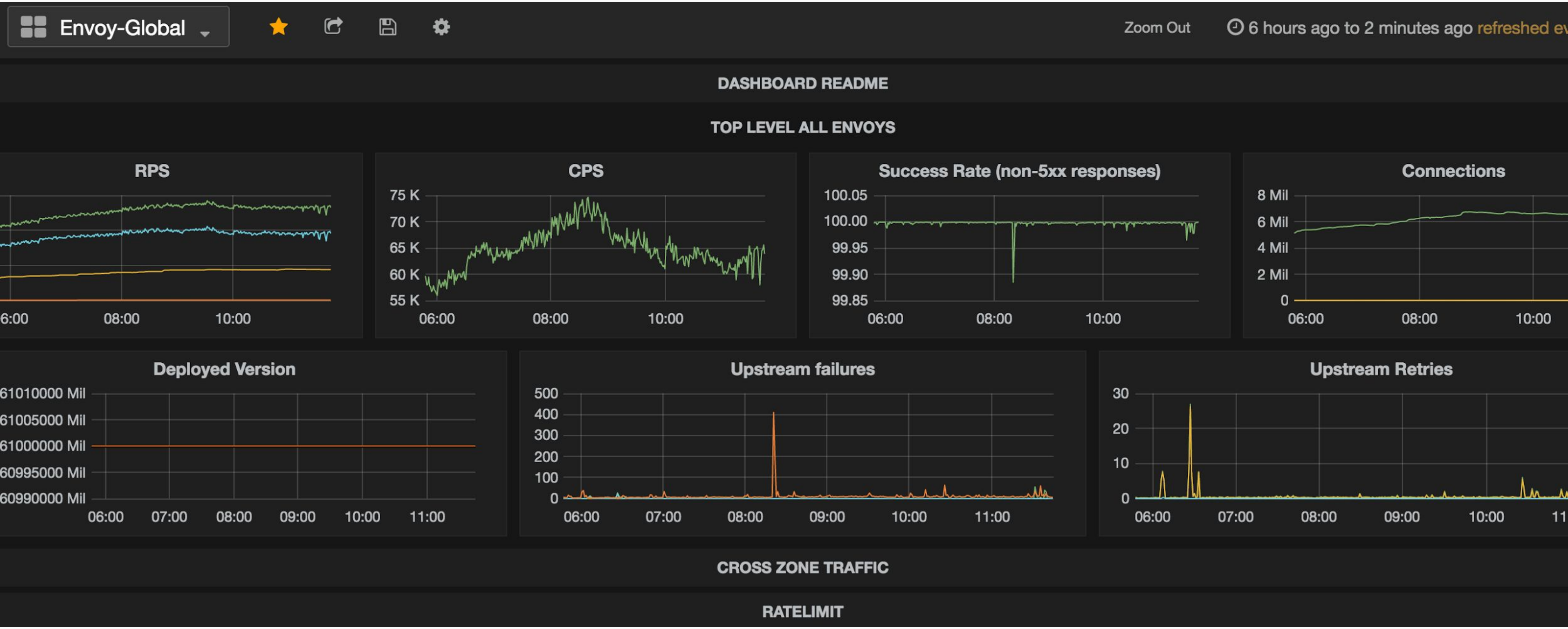
Links to logging and tracing

Observability: Service to service template dashboard

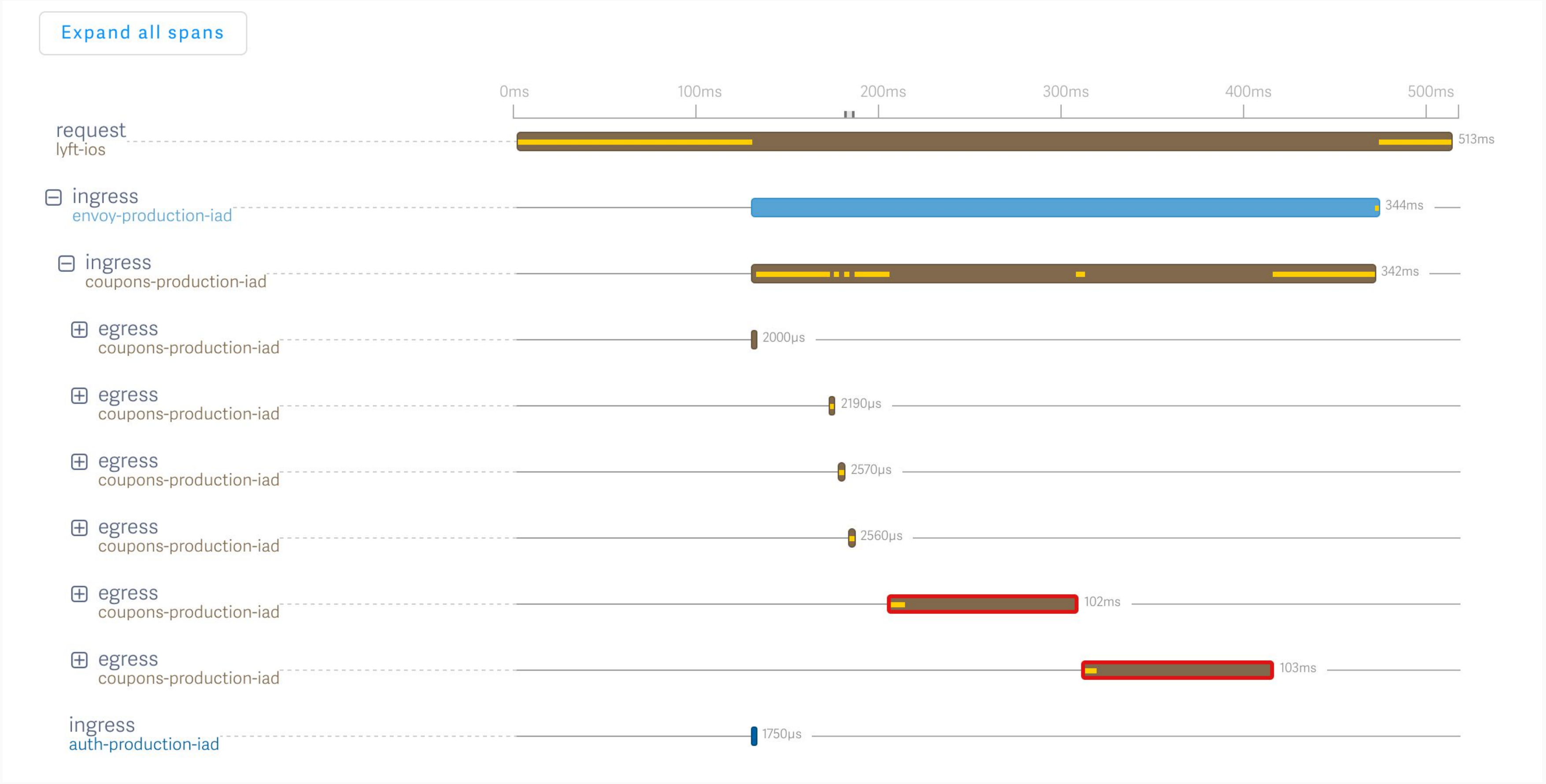
Template with drop down for every service



Observability: Envoy global health dashboard



Observability: Distributed tracing



request
lyft-ios 513ms

ingress
envoy-production-iad 344ms

ingress
coupons-production-iad 342ms

egress
coupons-production-iad 2000μs

egress
coupons-production-iad 2190μs

egress
coupons-production-iad 2570μs

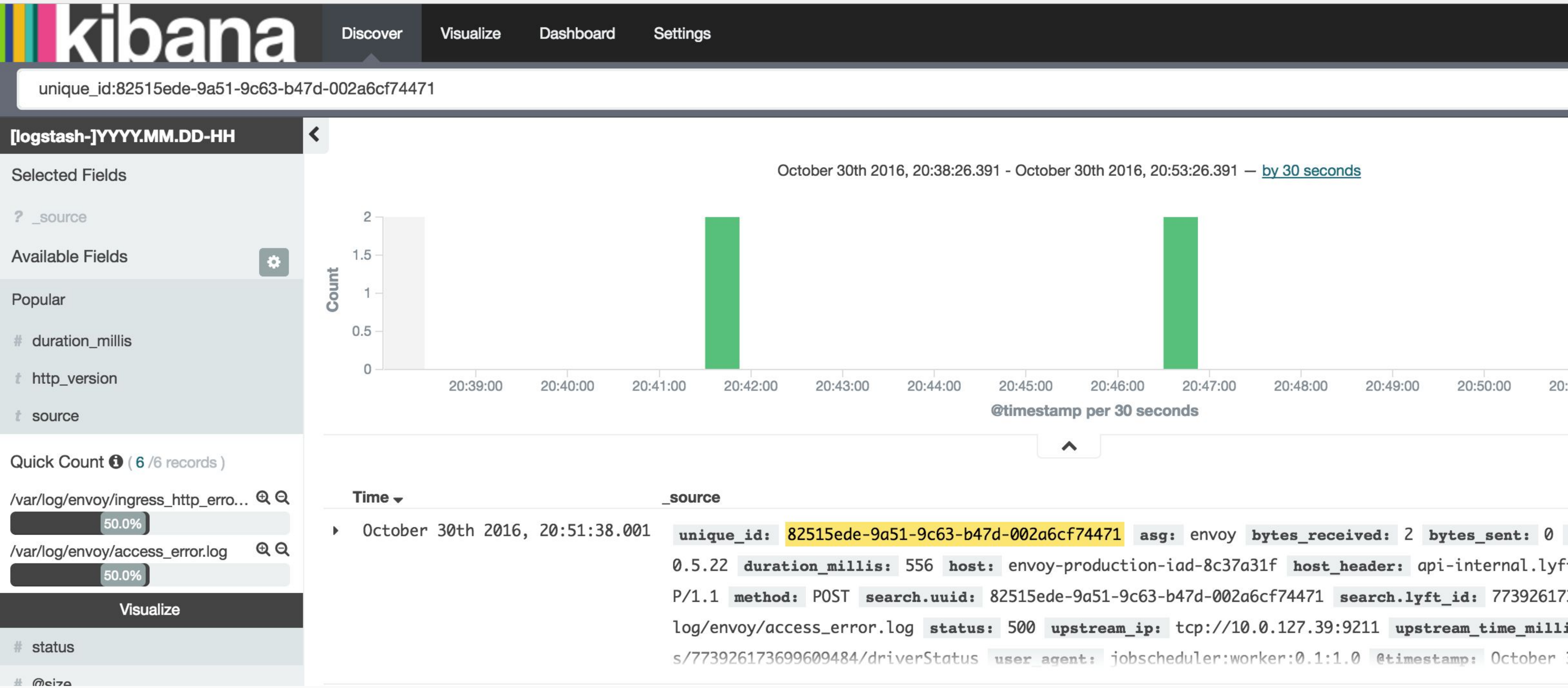
egress
coupons-production-iad 2560μs

egress
coupons-production-iad 102ms

egress
coupons-production-iad 103ms

ingress
auth-production-iad 1750μs

Observability: Logging



unique_id:82515ede-9a51-9c63-b47d-002a6cf74471

[logstash-]YYYY.MM.DD-HH

Selected Fields

? _source

Available Fields

Popular

duration_millis
http_version
source

Quick Count (6 /6 records)

/var/log/envoy/ingress_http_erro… 50.0%

/var/log/envoy/access_error.log 50.0%

status
@size

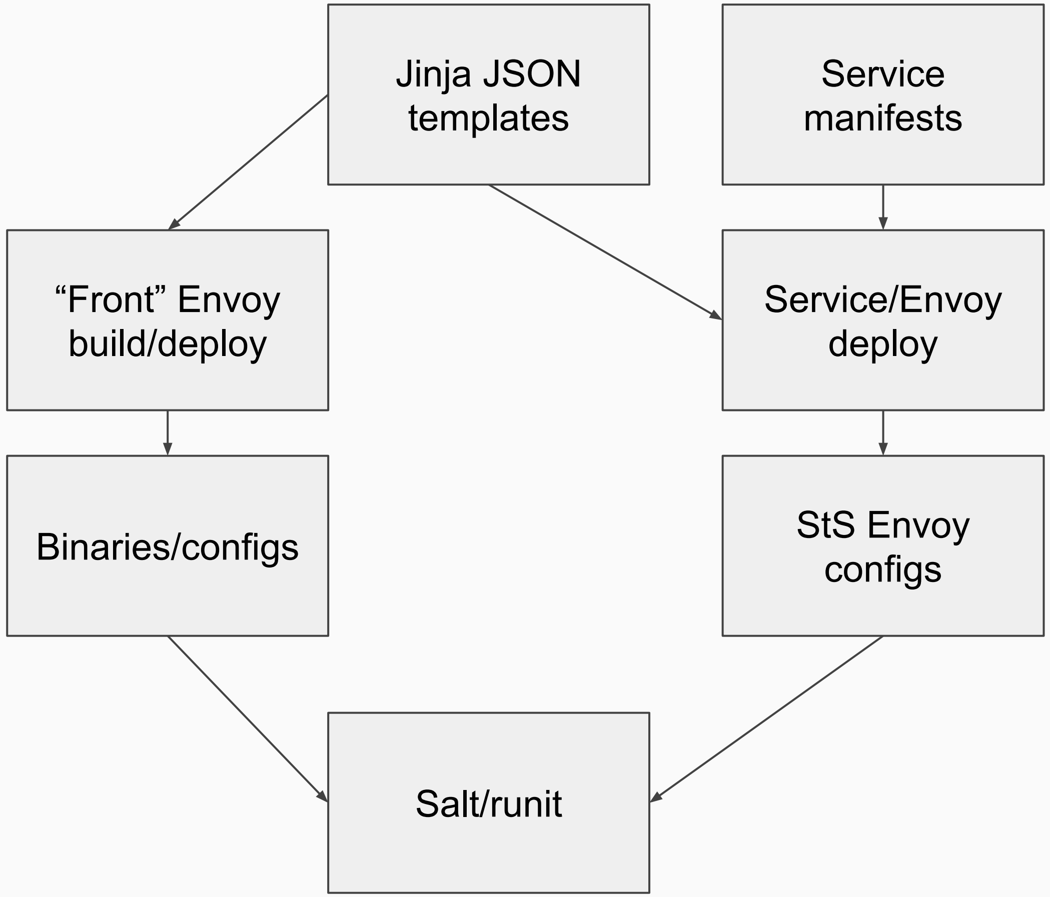
October 30th 2016, 20:38:26.391 - October 30th 2016, 20:53:26.391 — by 30 seconds

Time	_source
October 30th 2016, 20:51:38.001	unique_id: 82515ede-9a51-9c63-b47d-002a6cf74471 asg: envoy bytes_received: 2 bytes_sent: 0 0.5.22 duration_millis: 556 host: envoy-production-iad-8c37a31f host_header: api-internal.lyf P/1.1 method: POST search.uuid: 82515ede-9a51-9c63-b47d-002a6cf74471 search.lyft_id: 77392617 log/envoy/access_error.log status: 500 upstream_ip: tcp://10.0.127.39:9211 upstream_time_milli s/773926173699609484/driverStatus user_agent: jobscheduler:worker:0.1:1.0 @timestamp: October

Performance matters for a service proxy

- For most companies **developer time is worth more than infra costs** (cost vs. throughput).
- However**, Latency** and **predictability** is what matters. And in particular **tail latency** (P99+).
- Virtual IaaS, multiple languages and runtimes, languages that use GC: Niceties that improve productivity and reduce upfront dev costs, but make **debugging really difficult**.
- Ability to **reason about overall performance** and reliability is critical.

Envoy config/process management @Lyft



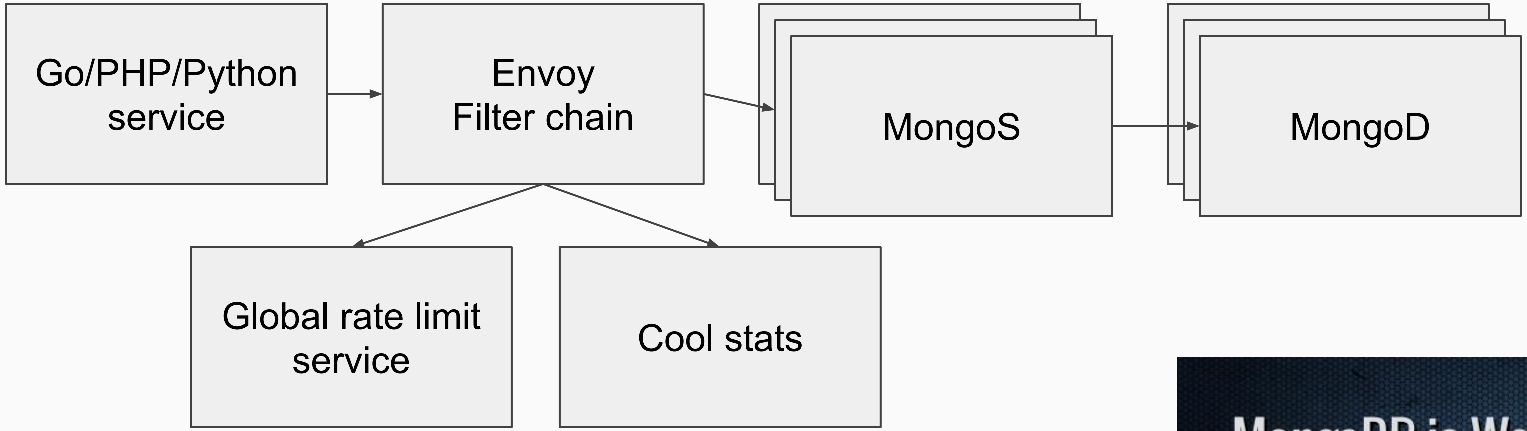
- Combination of static and dynamic configs.
- Service egress, circuit breaking, etc. configs specified in manifest.
- Service configs built on service host at service/envoy deploy time.
- **Next up for Lyft: config service via APIs!**

Envoy thin clients @Lyft

```python
from lyft.api_client import EnvoyClient
switchboard_client = EnvoyClient(
    service='switchboard'
)
msg = {'template': 'breaksignout'}
headers = {'x-lyft-user-id': 12345647363394}
switchboard_client.post("/v2/messages", data=msg, headers=headers)
```

- Abstract away egress port
- Request ID/tracing propagation
- Guide devs into good timeout, retry, etc. policies
- Similar thin clients for Go and PHP

Envoy MongoDB proxy @Lyft



No more death spirals! Web scale!

Filters:

- L4 global rate limit (limit CPS into MongoS)
- Mongo sniffer (cool stats)
- TCP proxy (MongoS load balancing)

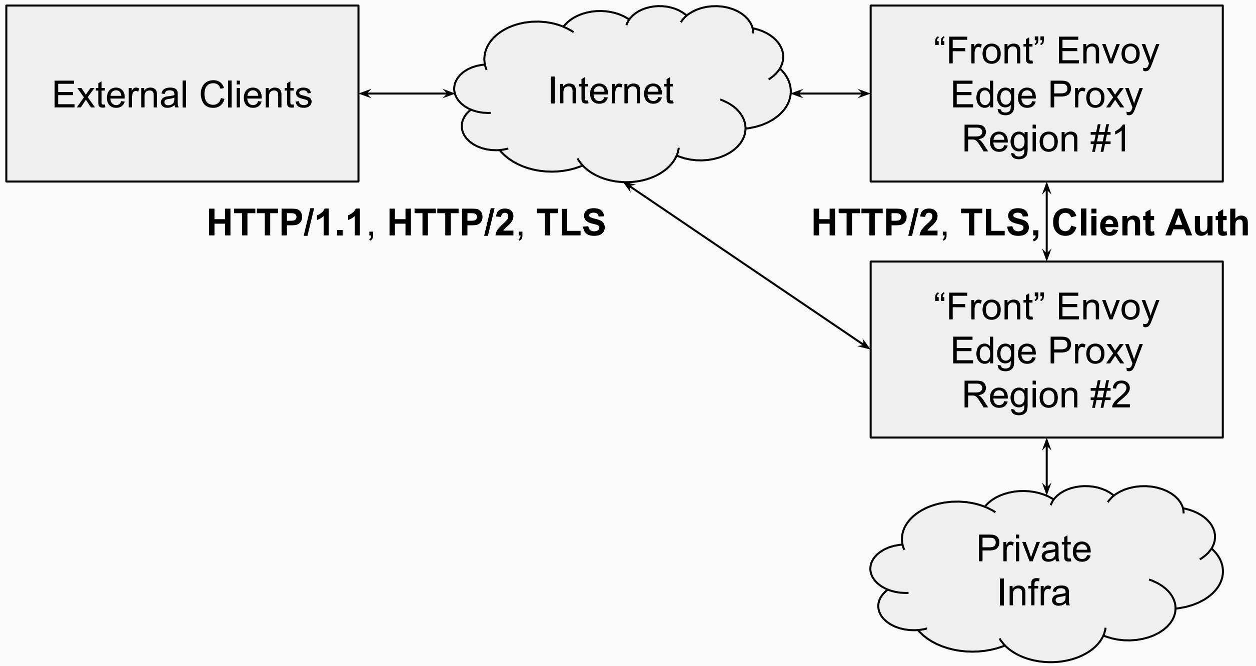
Envoy deployment @Lyft

- > 100 services.
- > 10,000 hosts.
- > 2,000,000 RPS.
- All service to service traffic (REST and gRPC).
- Use gRPC bridge to unlock Python gevent clients.
- MongoDB proxy.
- DynamoDB proxy.
- External service proxy (AWS and other partners).
- Kibana/Elastic Search for logging.
- LightStep for tracing.
- Wavefront for stats (via statsd).

Q&A

- Thanks for coming!
- We are super excited about building a **community** around Envoy. Talk to us if you need help getting started.
- https://lyft.github.io/envoy/
- **Lyft is hiring**: Contact us if you want to work on hard scaling problems in a fast moving company: https://www.lyft.com/jobs

Envoy edge proxy topology


External Clients
Internet
"Front" Envoy Edge Proxy Region #1
HTTP/1.1, HTTP/2, TLS
HTTP/2, TLS, Client Auth
"Front" Envoy Edge Proxy Region #2
Private Infra