

envoycon  
EUROPE

CLOUD NATIVE THEATER

# From ingress-nginx to Envoy Gateway at Zapier

*The benefits, challenges, and migration lessons that matter*



CLOUD NATIVE THEATER



**Kalen Wessel**

Sr. Site Reliability Engineer

*Zapier*

# Why We Needed To Move

## 🕒 Writing on the Wall

- **Ingress-Nginx EOL March 2026** - potential security risk if vulnerabilities emerge
- **Ingress is frozen** - no new features planned
- Gateway API is moving fast - risk of being left behind.

## 📁 Years of Tech Debt

- Patchwork of:
  - Application Load Balancers
  - Network Load Balancers
  - Ingress controls
- Multiple ingress controllers per cluster creating fragmentation
- Mixed annotations across environments led to confusion

## 🗨️ The Pain Points

- Configuration drift between staging and production
- "Works in staging, 503s in production"
- Debugging required deep knowledge of legacy annotation quirks

# Why Envoy Gateway

## Kubernetes API Gateway



Clear separation of responsibilities:

- Gateways for platform teams
- HTTPRoutes for developers



Strongly typed CRDs:

- Schema validation
- IDE support
- Catch errors early in CI/CD pipeline



Built-in advanced routing:

- Header / query matching
- Traffic splitting & mirroring
- Secure cross-namespace delegation

## Envoy Gateway



All-in-One Solution



Strong Kubernetes Gateway API alignment



Fast development cadence



Clear extension path (ext\_authz, policies, filters etc.)

# Migration Strategy

## Baseline First

Preparation & Benchmarking

### Inventory



Inventoried all ingresses backed by GitOps, deployed via ArgoCD

### Performance Metrics



Captured Nginx 4xx/5xx baseline benchmarks before migrating

### Annotation Mapping



Audited and mapped all annotations to Envoy equivalents



## Execution Loop

Batched Migration Process

### Tracking



Tracked status in internal sheets, Jira tickets, and Grafana dashboards

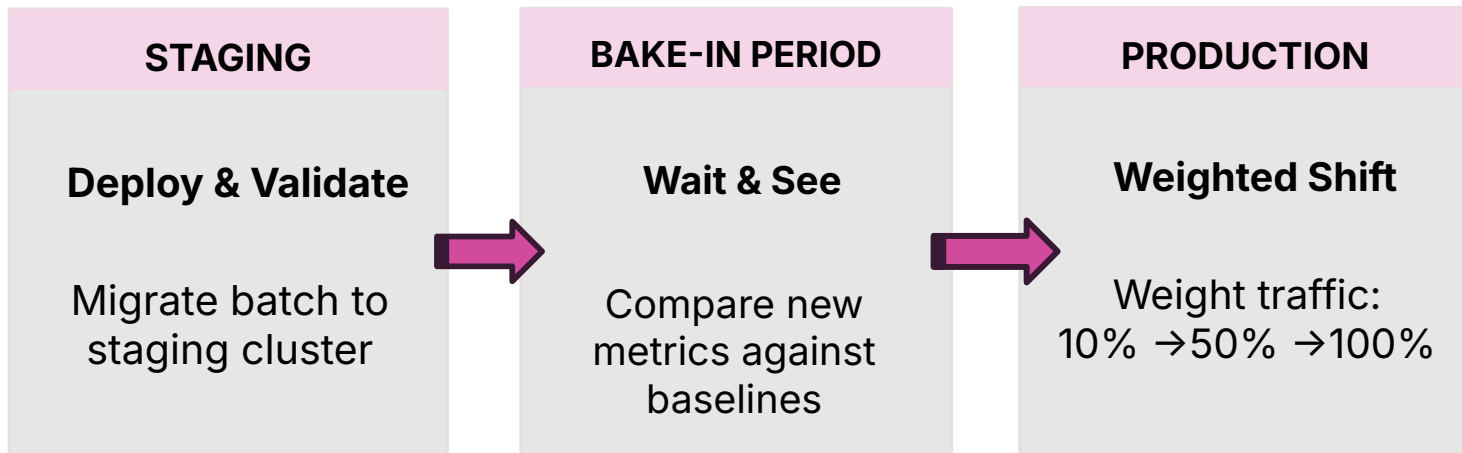
### The Batch Workflow



Prioritized high-impact services.  
Applied the migration loop for each batch:

DEPLOY → VALIDATE → PROMOTE

# Migration Strategy

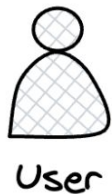


## Why Weighted DNS?

- Our services live on their own domains
- External-DNS with annotations allows us the luxury of setting DNS weighting

# Progressive Rollout

Phase 1 - 100% to Nginx w/DNS



zapier.com



100%



ingress-nginx

0%

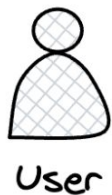


envoy-proxy

 zapier

# Progressive Rollout

Phase 2 - 50/50 split to Envoy w/DNS



zapier.com



50%



ingress-nginx

50%

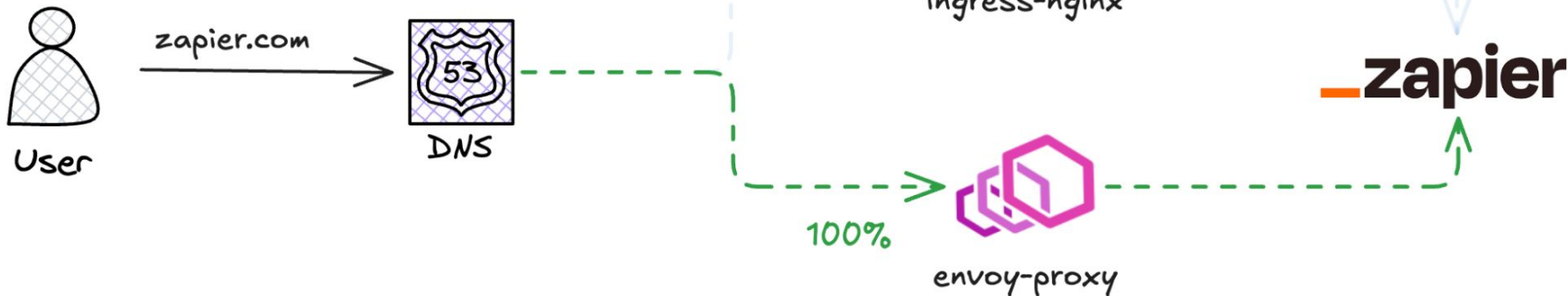


envoy-proxy

The zapier logo, featuring a stylized orange and black icon followed by the word "zapier" in a bold, black, sans-serif font.

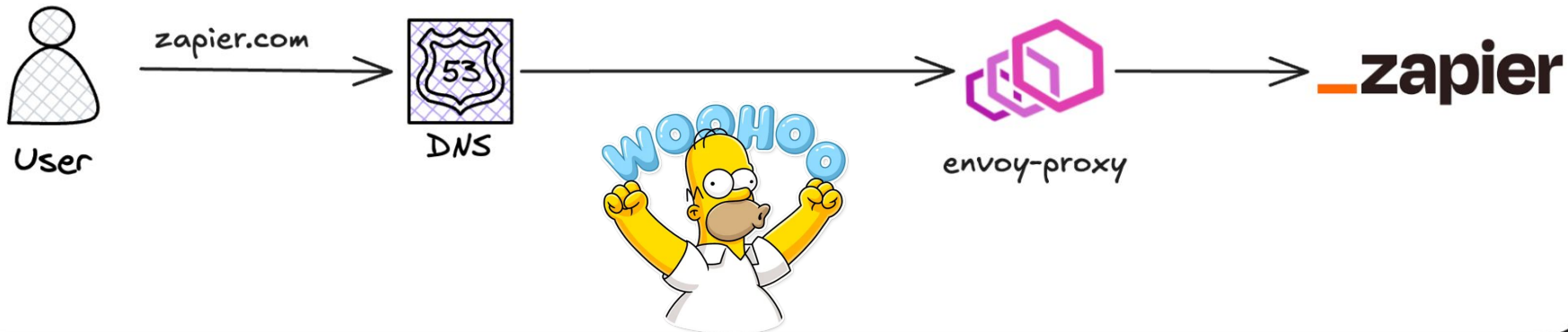
# Progressive Rollout

Phase 3 - 0/100 split to Envoy w/DNS



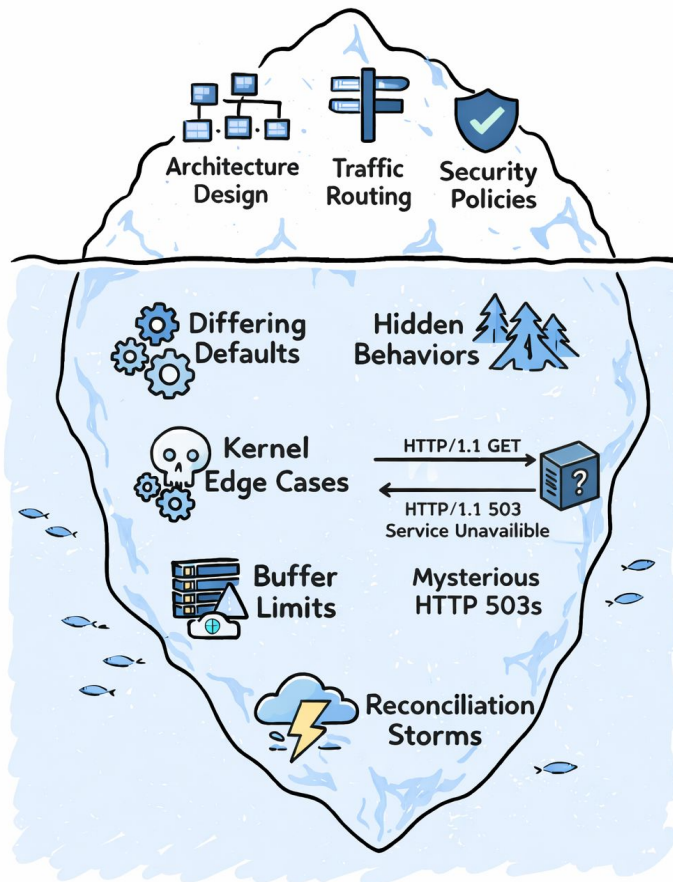
# Progressive Rollout

Phase 4 - Remove ingress-nginx route



# When Theory Meets Reality

- The plan vs. life
- "That shouldn't happen..."
- Are we the cause?
- Get your scuba gear on, cause we're going diving



# The Quiet 503s



## Symptom

- Steady stream of 503 errors on our webhook service
- Low rate but persistent

### KEY METRICS

`upstream_rq_retry{}`  

`retry_or_shadow_abandoned{}` 

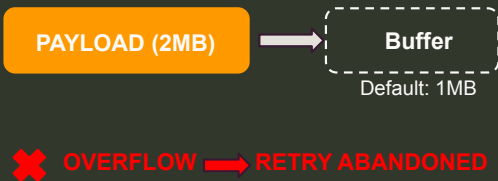
*Retries were being abandoned before they even started*



## Root Cause

- Large Webhook payloads
- Envoy's default buffer is too small for retry

### FAILURE MECHANISM



## Fix

- Increase the `connection.bufferLimit`
- Applied via `ClientTrafficPolicy`

### INCREASE BUFFER

```
kind: ClientTrafficPolicy
spec:
  connection:
    bufferLimit: 12Mi
```

# Reconciliation Woes



## Symptom

Envoy Gateway control plane instability in development cluster



100K+ Events



OOM Killed



## Root Cause

Vault operator refreshing a vault-ca TLS secret in every namespace. Due to the short expiry this was refreshing frequently.

Vault Update



Secret Change



Full Reconcile



## Fix

Remove wildcard from Vault Operator config as this functionality was not needed in our development cluster

```
caNamespaces:  
- "*" # ← dropped wildcard
```

## Takeaway

Be aware of the operators running in your cluster and what resources they're modifying.

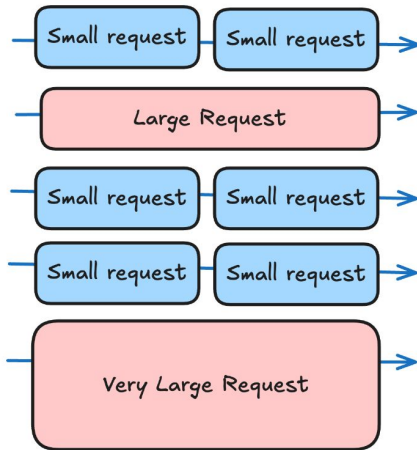
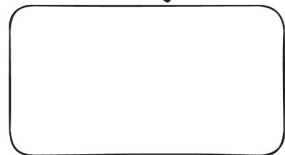
# Connection Reuse



## Symptom

Elevated HTTP response latency

Backlog



Gunicorn Application  
HTTP/1.1



# Connection Reuse



## Symptom

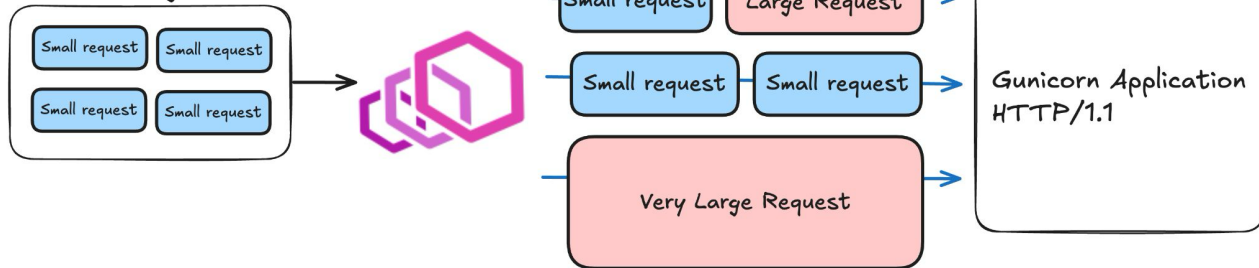
Elevated HTTP response latency



## Root Cause

Envoy connection reuse

## Backlog



# Connection Reuse



## Symptom

Elevated HTTP response latency



## Root Cause

Envoy connection reuse



## Fix

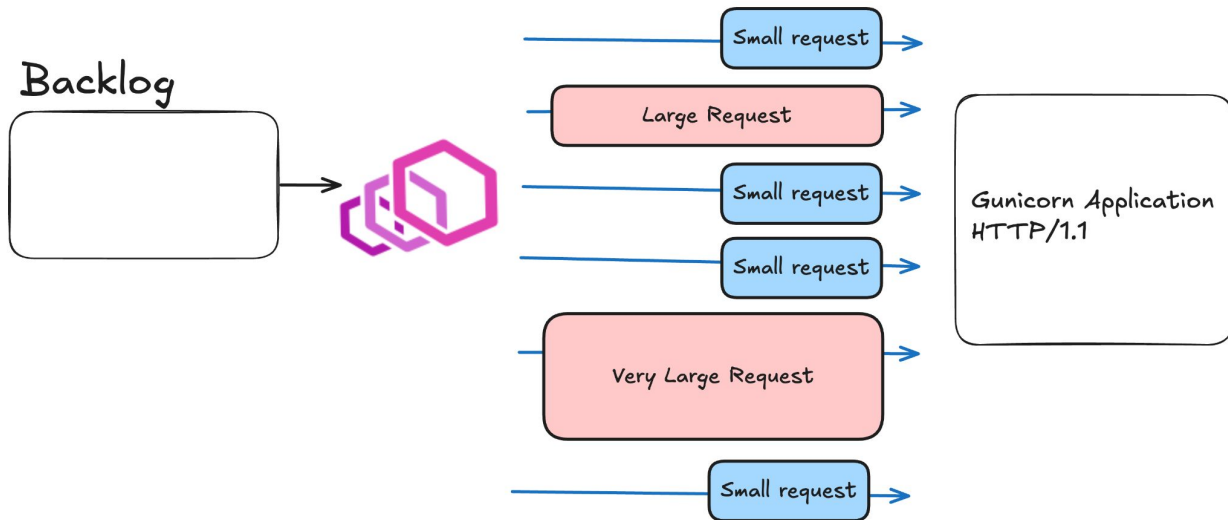
Set max number of requests per connection

```
kind: BackendTrafficPolicy
spec:
  circuitBreaker:
    maxRequestsPerConnection: 1
```

## Takeaway

When running synchronous services keep connection settings in mind

Backlog

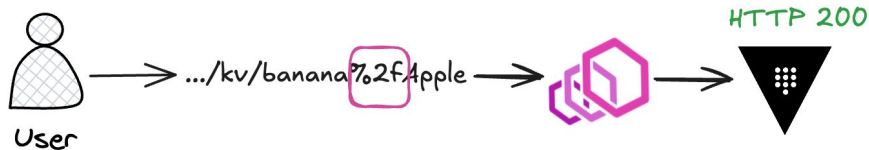
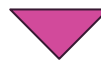


# Differing Defaults

- ingress-nginx by default forwards encoded paths unchanged
- Envoy Gateway by default will see `%2f` and unescape it into `/` and redirect to the new path



```
kind: ClientTrafficPolicy
spec:
  path:
    escapedSlashesAction: KeepUnchanged
```

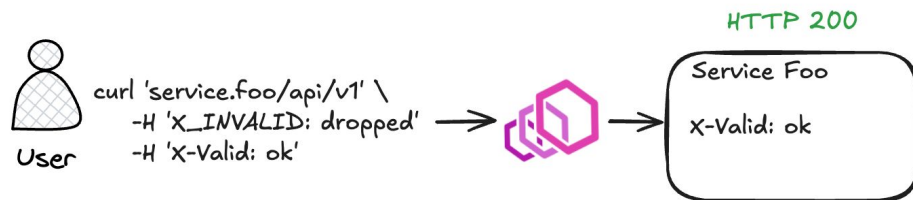
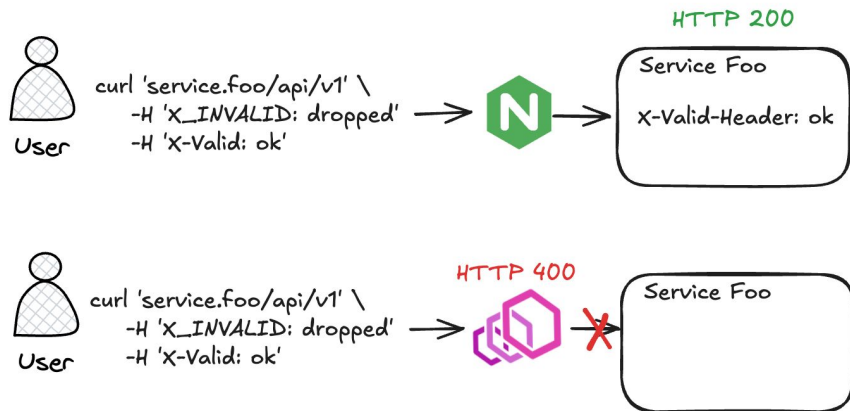


# Differing Defaults

- ingress-nginx by default **DROPS** headers containing underscores
- Envoy Gateway by default **REJECTS** the **requests** with headers containing underscores

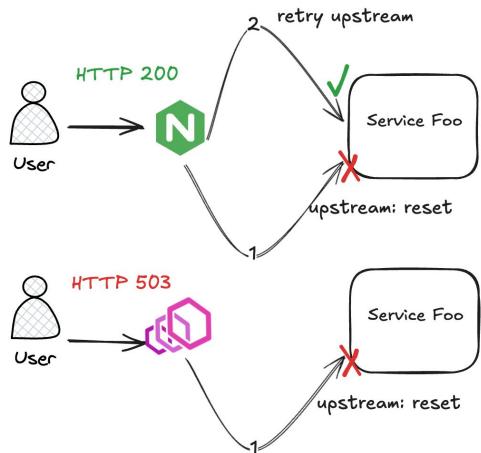


```
kind: ClientTrafficPolicy
spec:
  headers:
    withUnderscoresAction: DropHeader
```

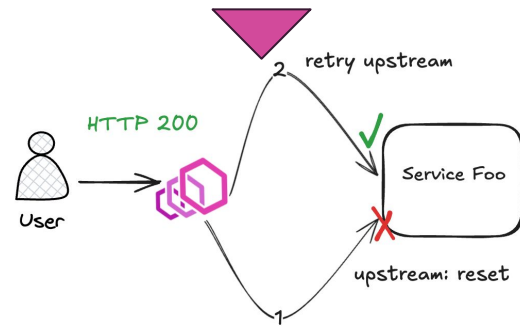


# Differing Defaults

- ingress-nginx default retry behavior includes retrying on **upstream timeouts** via **proxy\_next\_upstream**
- Envoy Gateway **requires** explicit configuration of retry triggers to match that behavior.



```
kind: BackendTrafficPolicy
spec:
  retry:
    numRetries: 3
    retryOn:
      triggers:
        - connect-failure
        - reset
        - refused-stream
```



# Stay Up to Date

## Why it matters:

- Our staging environment churns through hundreds of ephemeral deployments
- Stability in staging was critical for us to feel confident in production

## Memory improvements:

- **Before** EG 1.5.3: ~80% average usage, frequent OOMs
- **After** EG 1.5.3: <20% average usage, zero OOM events

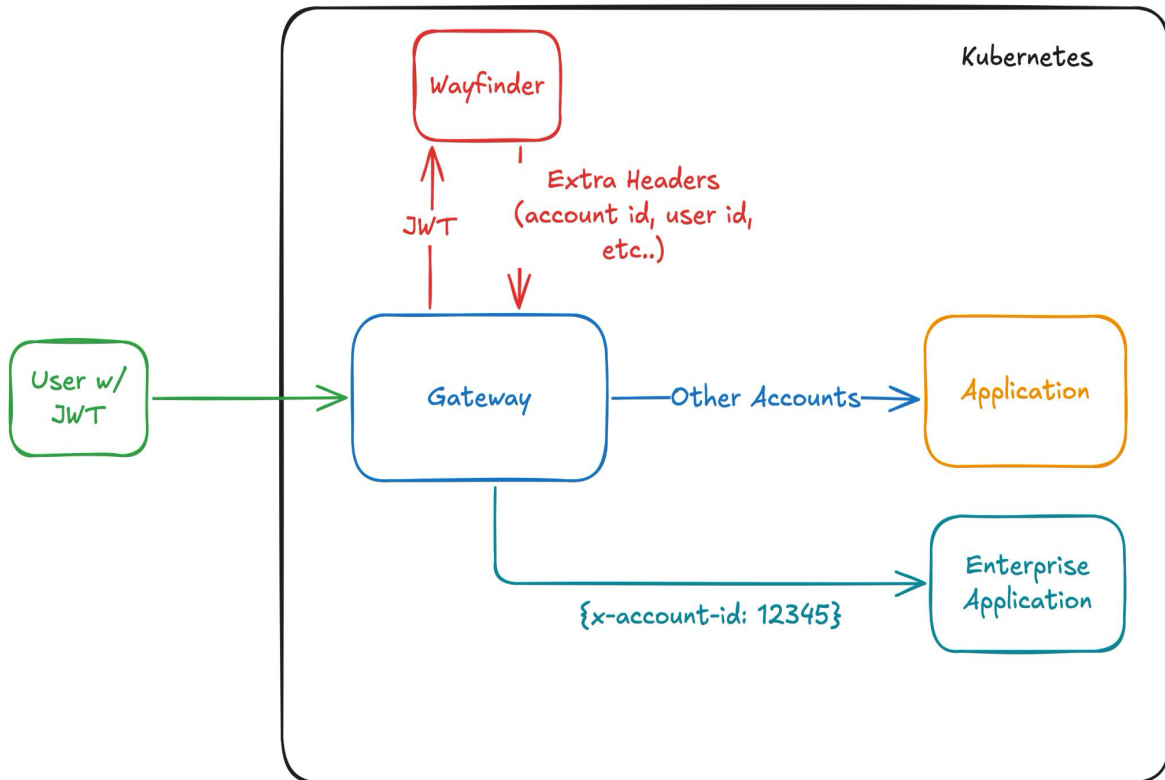
Envoy Gateway Memory Usage %



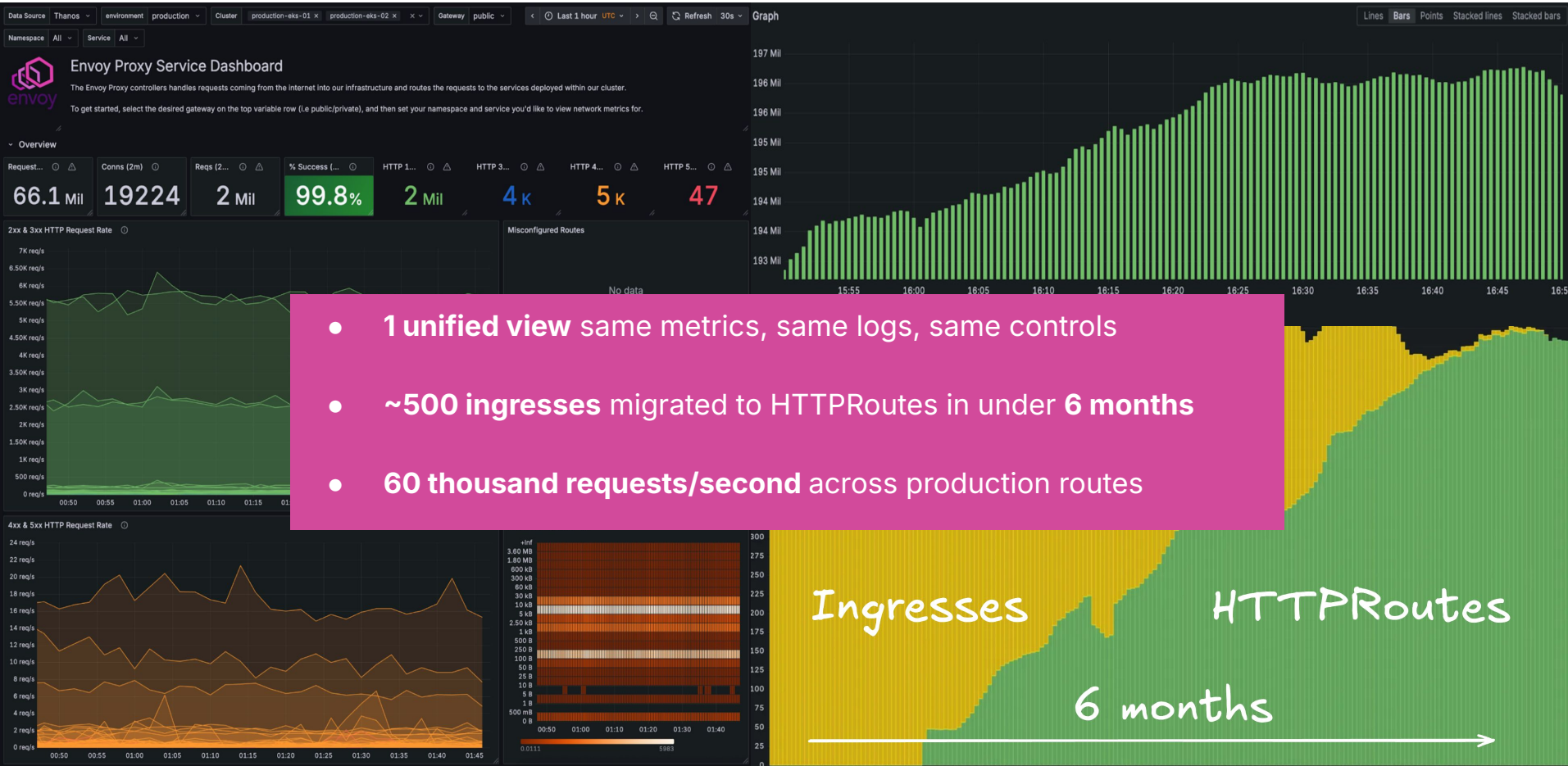
# What We Built On Top

## Custom Authz service (Wayfinder):

- Written in Go, connects via ext\_authz
- Validates JWTs, enriches requests with customer context
- Sub millisecond latency at tens of thousands RPS
- Context flows into observability pipeline



# Where We Are Today



# Thank You

# —zapier



<https://zapier.com/blog/zapier-journey-beyond-ingress-nginx/>



<https://www.linkedin.com/in/kalenwessel/>