Service Discovery

Past, Present, Future

Challenges of Change

Chase Childers • November 2019 • Site Reliability Engineering

- **Site Reliability Engineer**
- **Fun Fact**:
  Stayed @ Airbnb's in 12 Countries
- **Boring Fact**:
  I (probably) drank coffee this morning
- **Date of Last Production Incident**:
  (Caused by me) July 15, 2019

Chase Childers (He/Him)

Service Discovery is Hard

Scaling is Hard

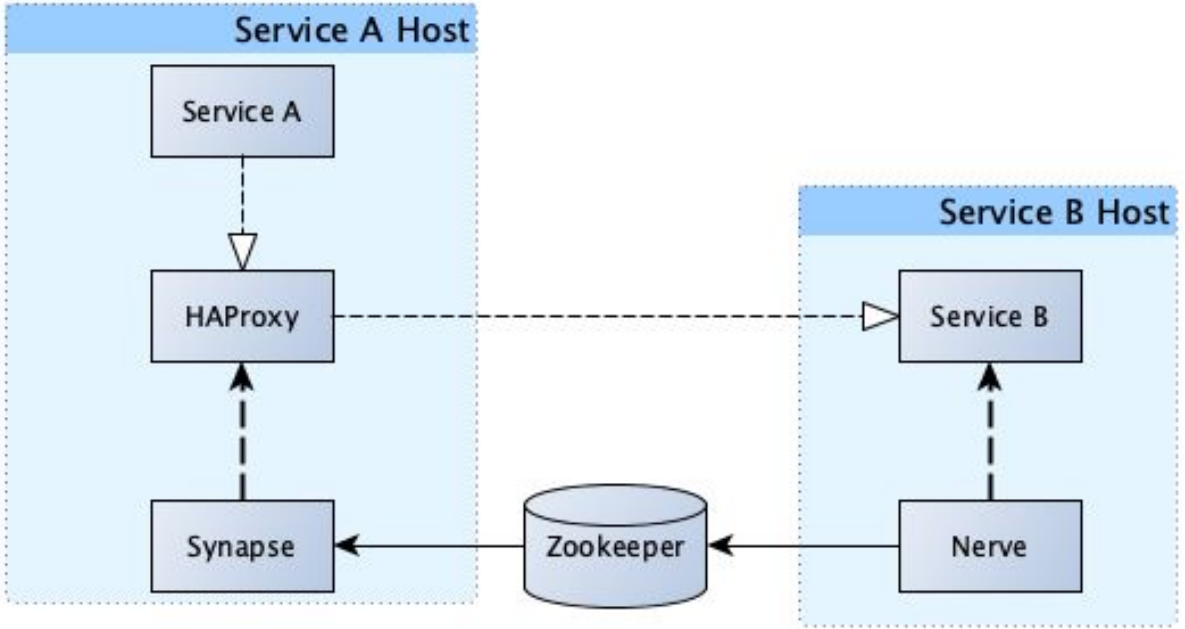
Migrating is Hard

Engineering is Hard


Phase 0: In the Beginning
Phú Quốc, Vietnam
chase.childers@

Smartstack

Distributed Service Discovery

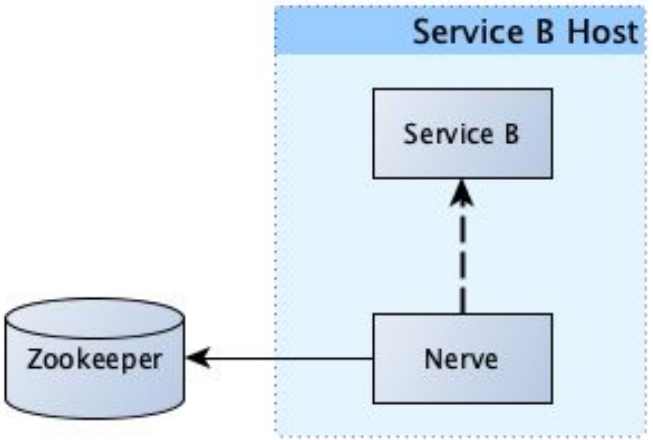


chase.childers@

Nerve

https://github.com/airbnb/nerve

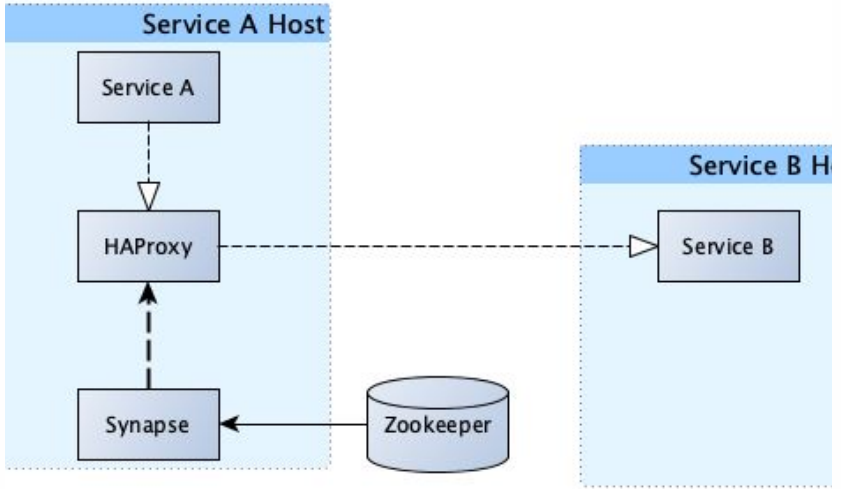
- Executes on service host
- Performs health check on service
- Publishes address, port, and availability to Zookeeper



Synapse

https://github.com/airbnb/synapse

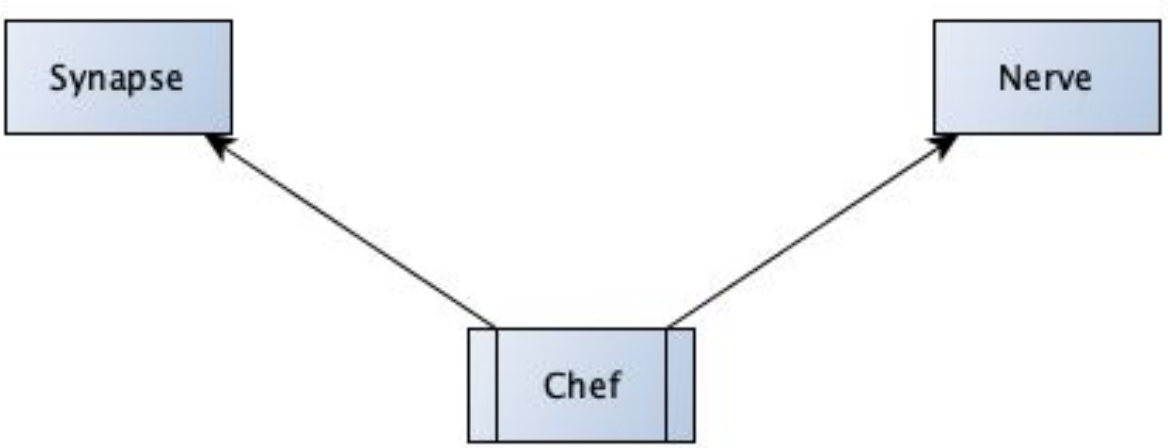
- Executes on client host
- Watches backend addresses and availability from Zookeeper
- Configures and utilizes HAProxy for listening for outbound traffic to the designated service via `*.synapse`
- HAProxy load balances outbound traffic



Setup via Chef

Distributed Service Discovery

- Services declare/reserve ports in Chef
- Clients list dependencies in Chef for Synapse configuration
- Nerve is enabled with a service name for services
- Custom Synapse and Nerve configurations in Chef



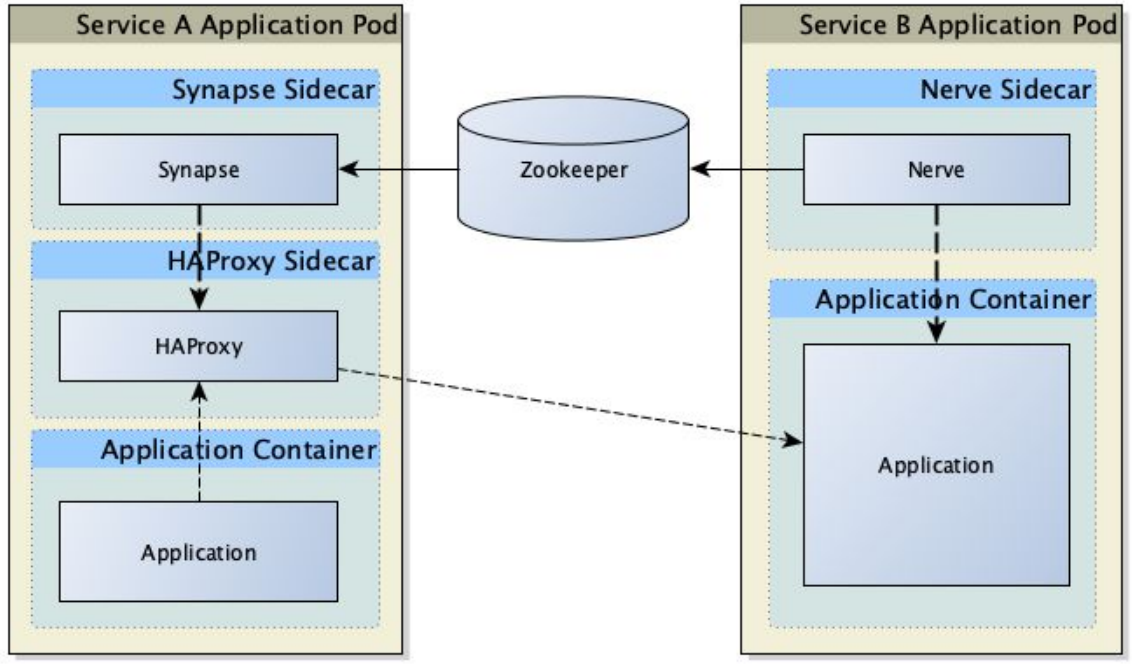
Phase 1

Kubernetes


You get a container!

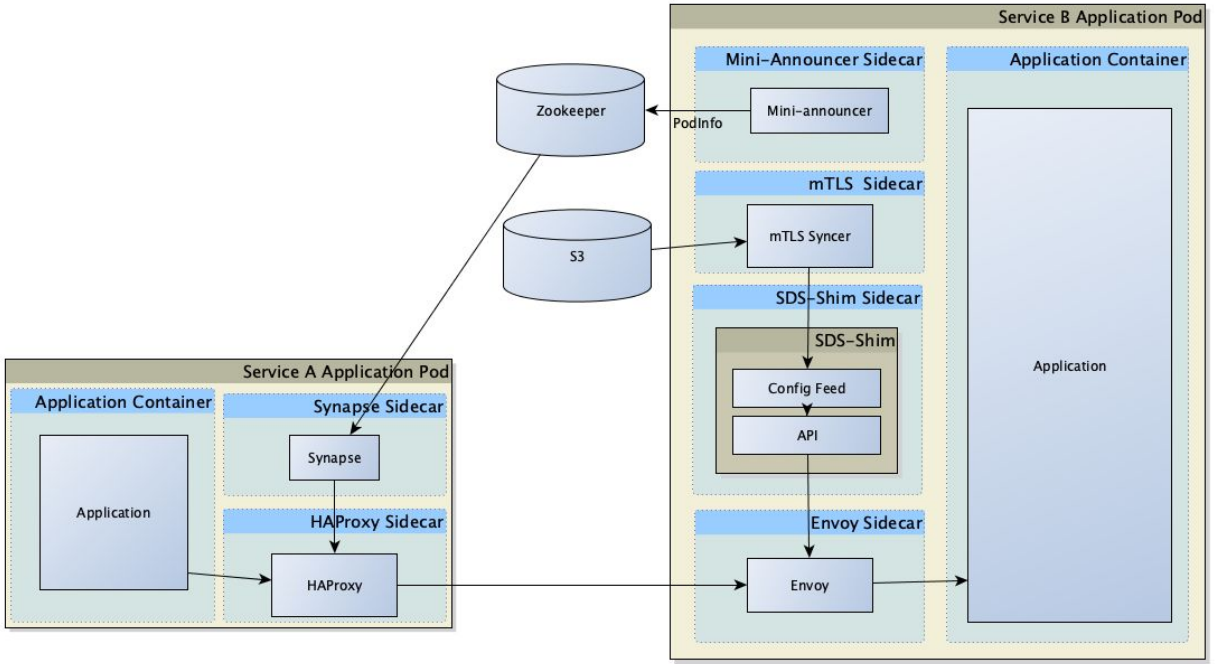
Smartstack

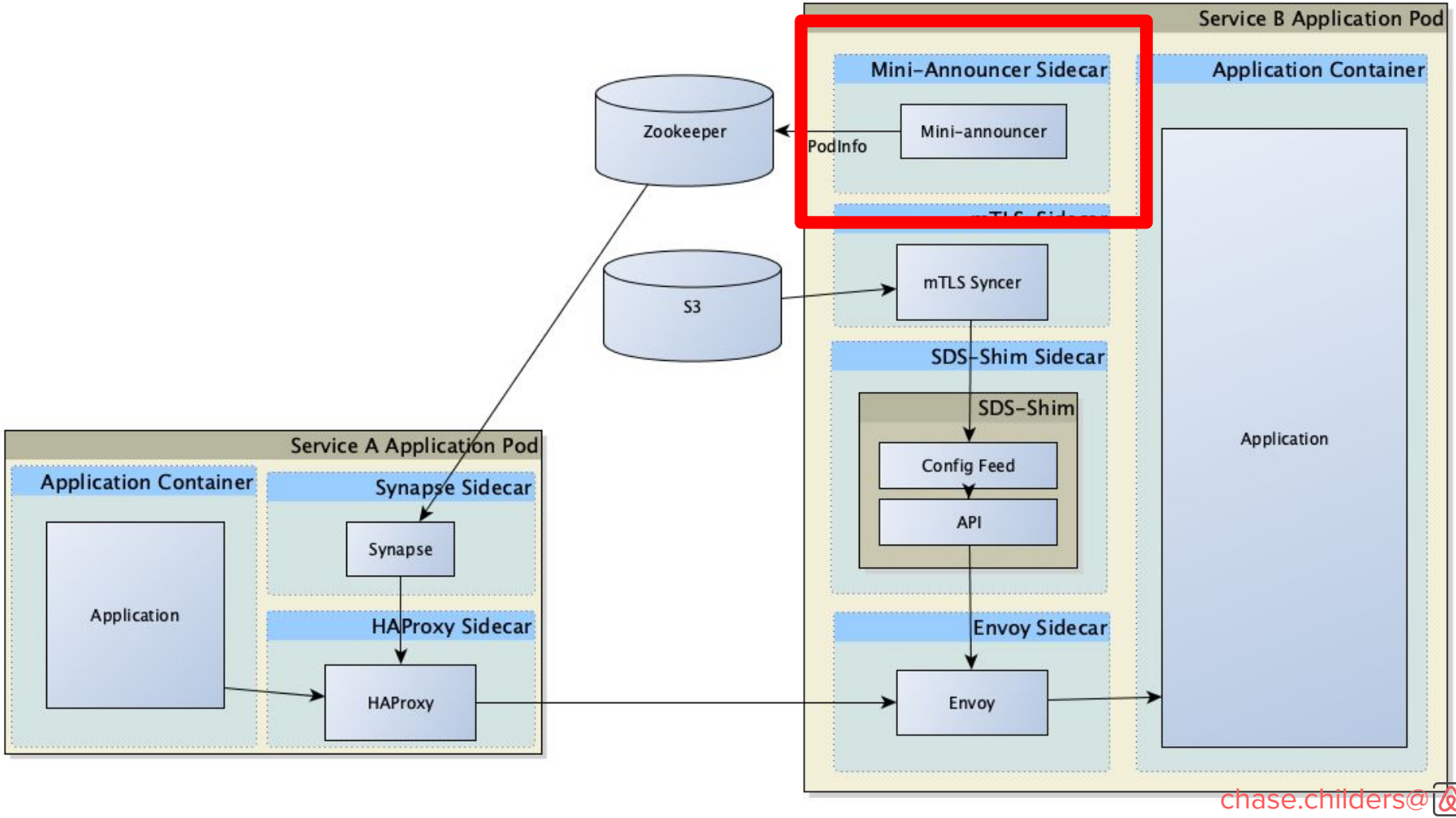
Expectation



Smartstack

Reality





~~Nerve~~

Mini-Announcer

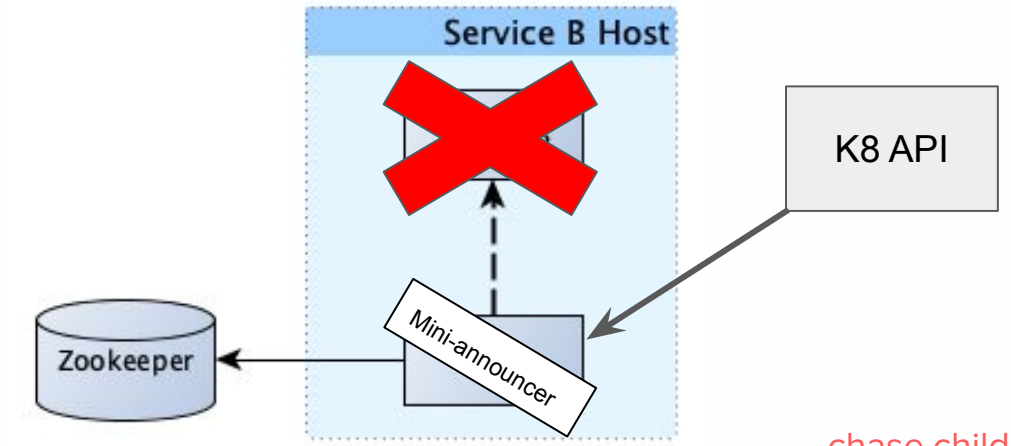
Service Side Availability and Discoverability

chase.childers@

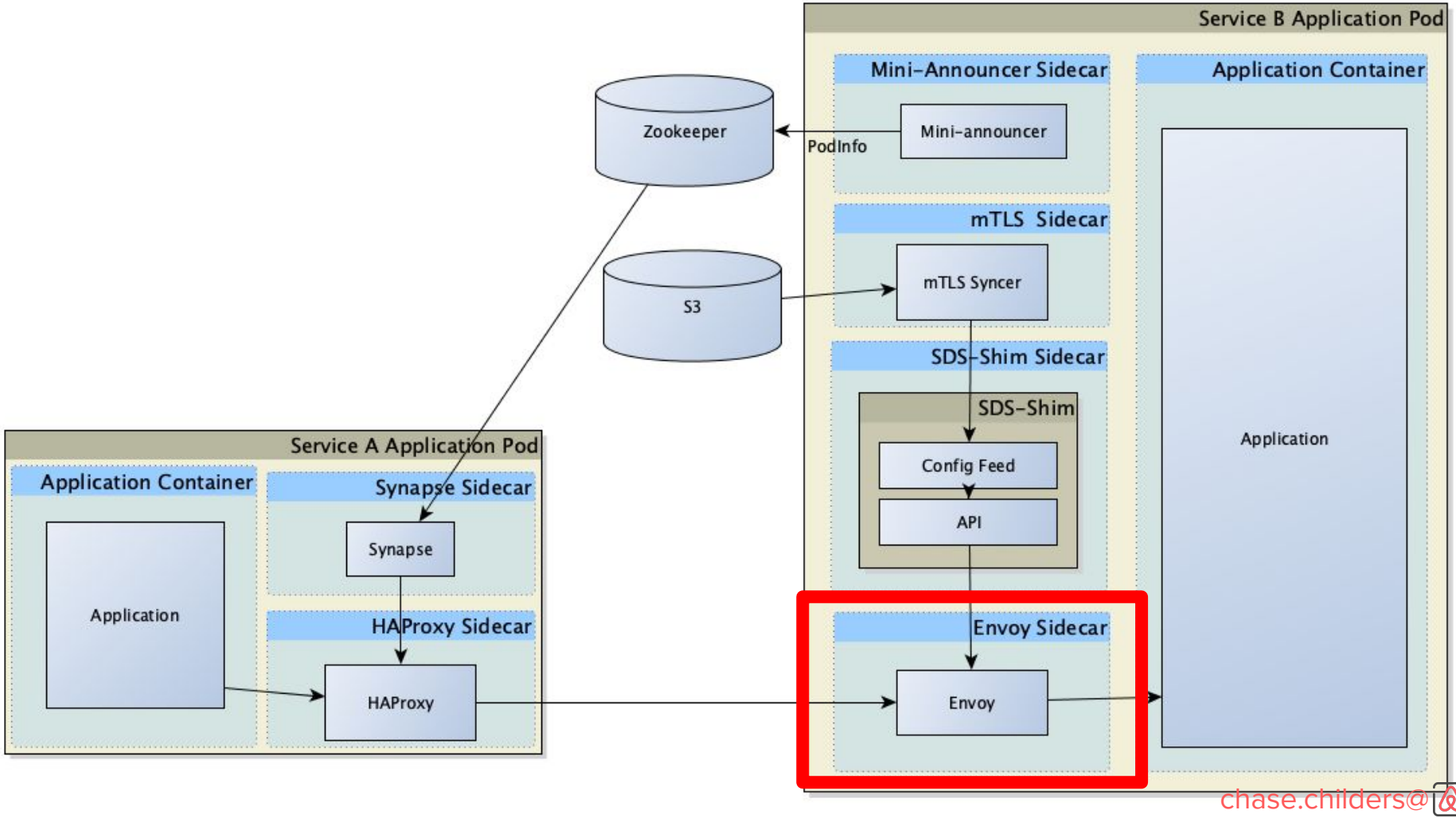
~~Nerve~~ Mini-Announcer

Service Side Availability and Discoverability

- Executes ~~on service host~~ in a container on the pod
- ~~Performs health check on service~~
- Checks if all other containers are ready
- Request IP and port assignment from Kubernetes API
- Publishes IP and port and availability to Zookeeper
- Graceful shutdown via K8 preStop hook

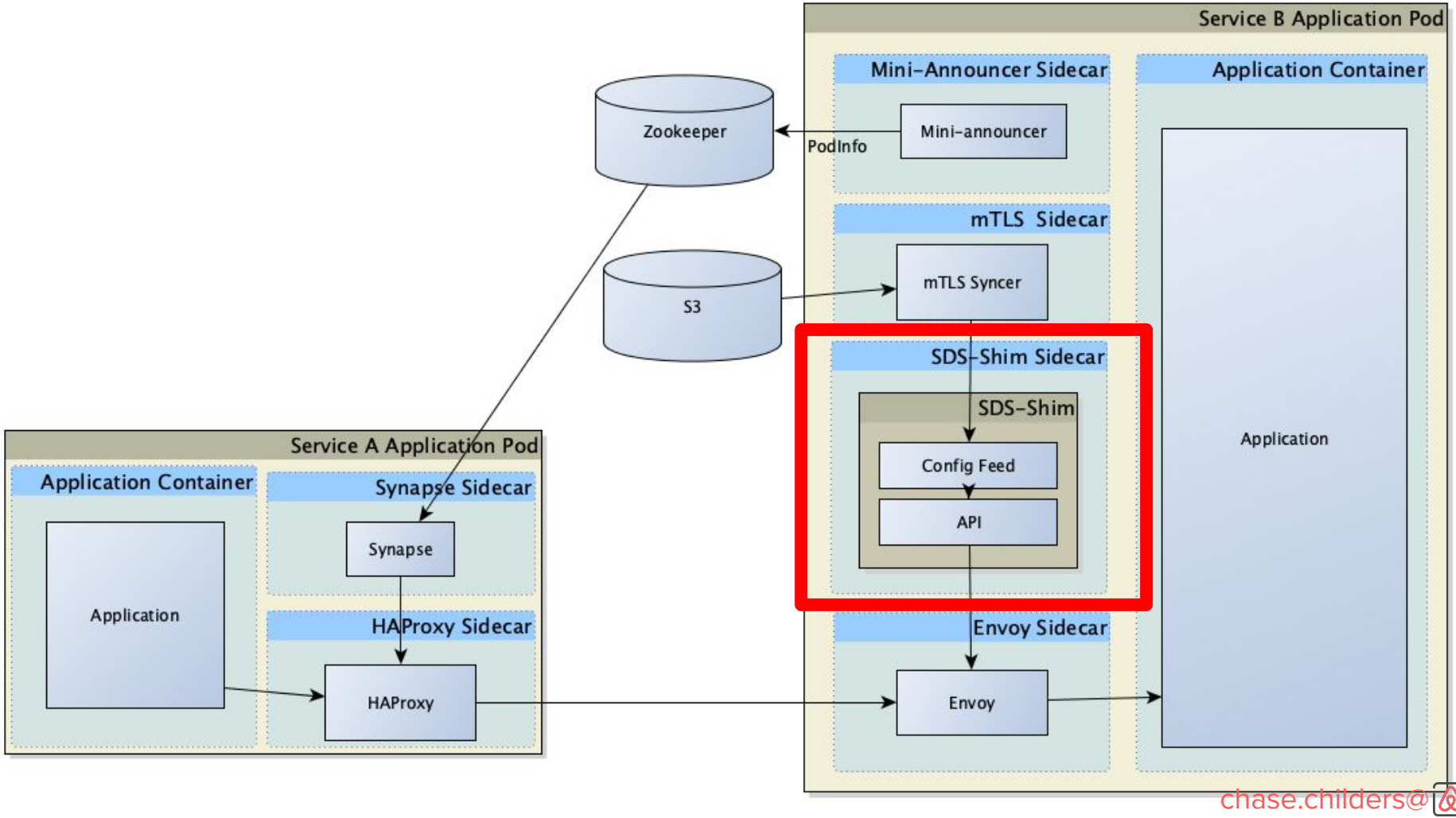


chase.childers@



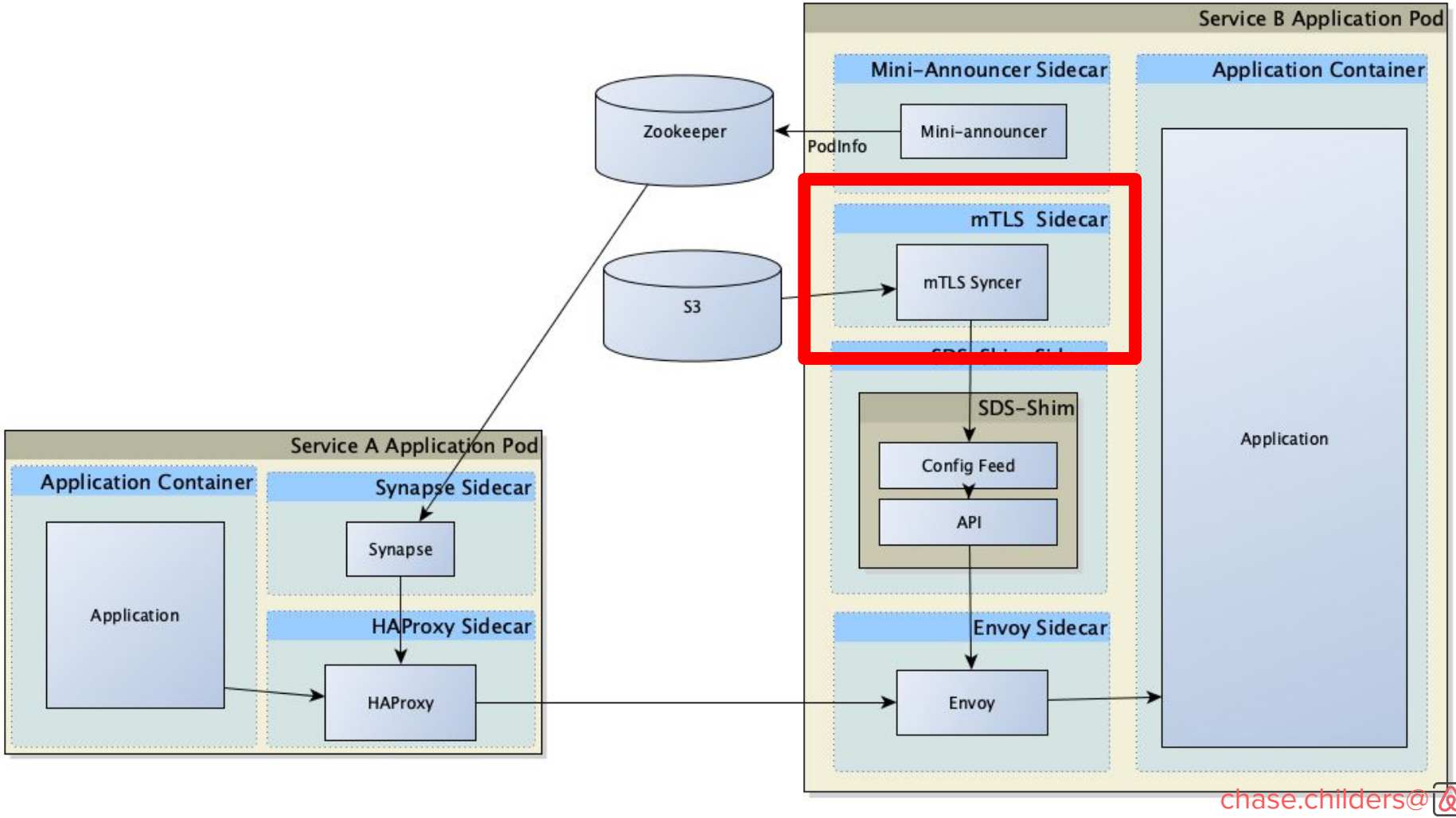
Envoy

- Ingress Proxy
- Bootstrap configure endpoints for local service
- Polls the local SDS-Shim container for updates



SDS-Shim

- Local ‘shim’ to serve the SDS Rest API that Envoy polls
- Merges multiple sources of configuration
- Pulls service discovery configurations from Zookeeper
- Secure service communication from mTLS Syncer
- Can replace Synapse when utilizing Envoy



mTLS Syncer

- On by default for all Kubernetes applications
- Sets up Secure Listeners to receive mTLS connections
- Produces a config that is fed into SDS-Shim

Intermission

Things Fall Apart

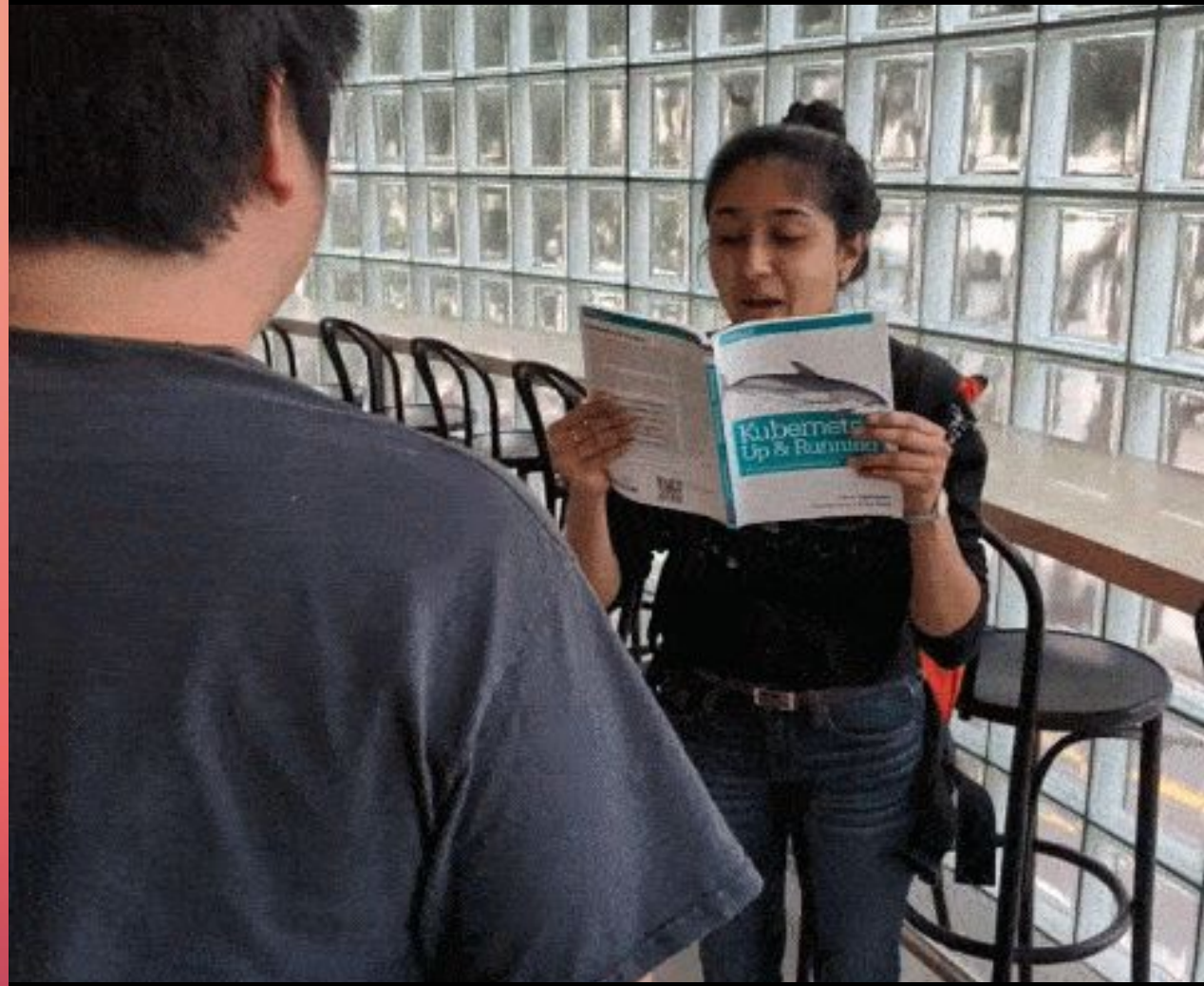
chase.childers@

Growth is Hard

Monolithic Scaling is Harder

HAProxy Config Size

A Hypothetical Scenario on Growth



chase.childers@

One Service Lots of Backends

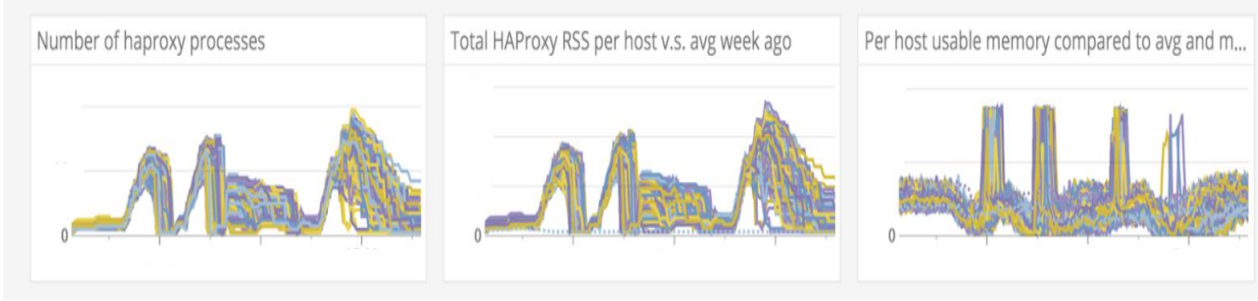
How large is the config?
Backends = 10?

How large is the config?
Backends = 100?

How large is the config?
Backends = 1000?

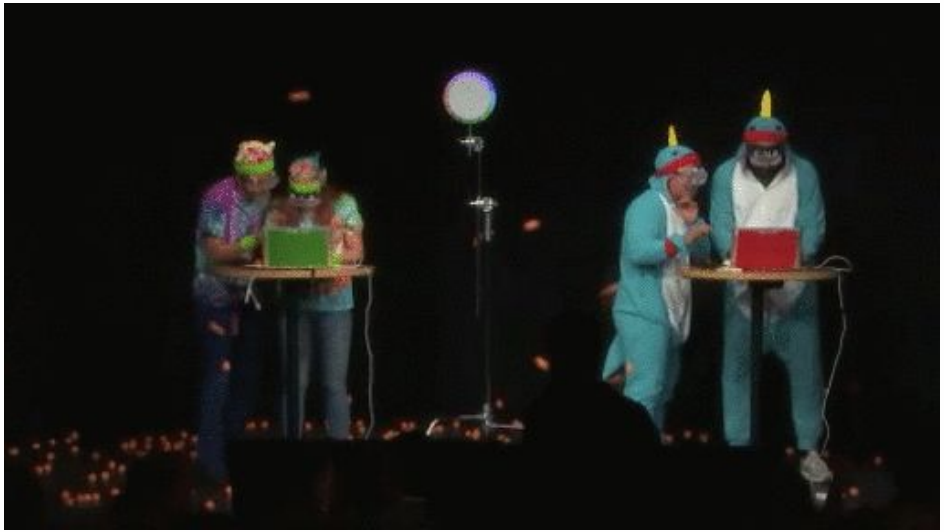
How large is the config?
Backends = 10000?

Memory Exhaustion



- HAProxy reloaded every 2-5 seconds during continuous backend changes (EC2 and K8 Deploys)
- Existing connections are NOT forcefully closed, leaving their lifecycle to be determine by client and server
- Stale HAProxy processes accumulate and consume huge chunks of memory

Network Saturation

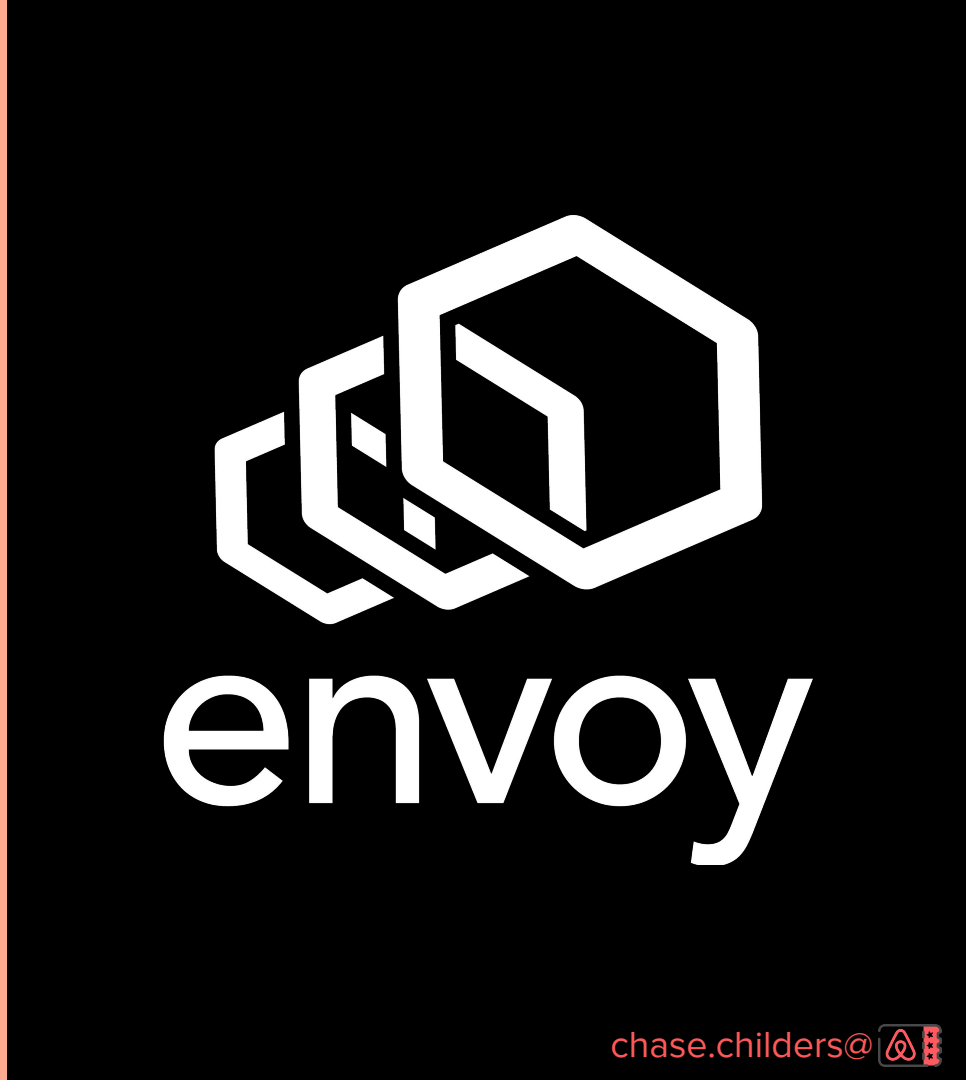


- HAProxy sends initial wave of health checks without jitter or spreading
- During fast reloading, the target instances become overloaded.

What Now?

Phase 2

Envoy



Service Discovery Containers

Recap

Mini-Announcer
Discoverability

mTLS Syncer
Secure Communication

SDS-Shim
Envoy Shim

Envoy
Ingress / Egress Proxy

HAProxy
Egress Proxy

Synapse
HAProxy Configurator

Service Discovery Containers

Recap

Mini-Announcer
Discoverability

mTLS Syncer
Secure Communication

SDS-Shim
Envoy Shim

Envoy
Ingress / Egress Proxy

HAProxy
Egress Proxy

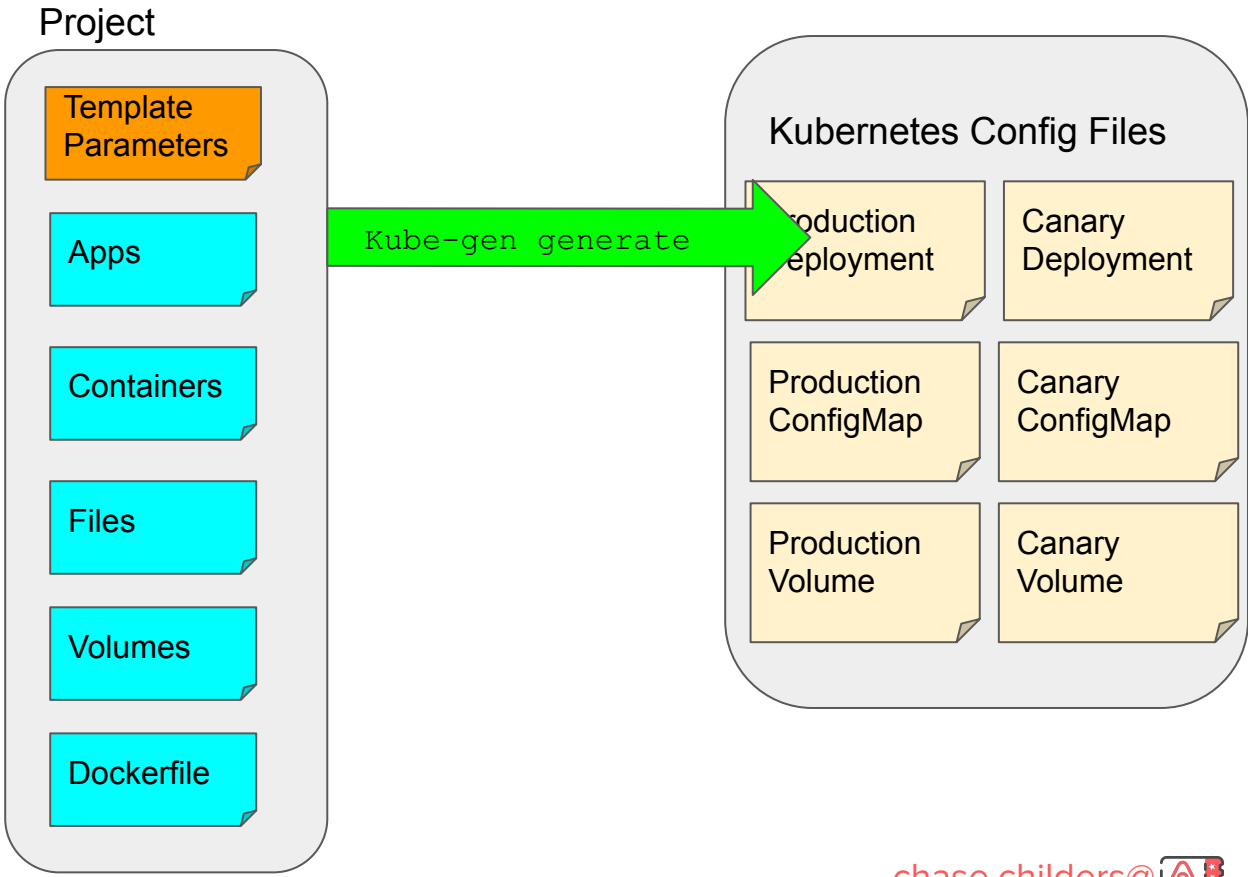
Synapse
HAProxy Configurator

Envoy

- Ingress Proxy / **Egress Proxy**
- Configure endpoints for local service
- Polls the local SDS-Shim container for dynamic backends

Kube-Gen

Tool to convert Airbnb Infra Configuration to Kubernetes Configuration



chase.childers@

Control Plane Containers

How do you decide and how do you configure?

Mini-Announcer
Discoverability

mTLS Syncer
Secure Communication

SDS-Shim
Envoy Shim

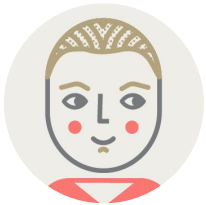
Envoy
Ingress / Egress Proxy

HAProxy
Egress Proxy

Synapse
HAProxy Configurator

Service Discovery Containers

A new challenger approaches!



Mini-Announcer
Discoverability

mTLS Syncer
Secure Communication

SDS-Shim
Envoy Shim

Envoy
Ingress / Egress Proxy

HAProxy
Egress Proxy

Synapse
HAProxy Configurator

SD Configurator
Bootstrap Configuration

SD Configurator

Service Discovery Configuration Management

- Packaged and Released as Ruby Gem
- Wrapped in an Init Container
- Selects service discovery path for dependencies
  - Synapse/HAProxy vs SDS-Shim/Envoy
- Generates bootstrap configs for Synapse / SDS-Shim

- Default service discovery configuration source of truth
- Built and Released independently of kubernetes tooling

Service Discovery Containers

But Seriously? Soooo many containers!

Mini-Announcer
Discoverability

mTLS Syncer
Secure Communication

SDS-Shim
Envoy Shim

Envoy
Ingress / Egress Proxy

HAProxy
Egress Proxy

Synapse
HAProxy Configurator

SD Configurator
Bootstrap Configuration

What about my custom service discovery configs?

Zookeeper

Service Discovery Configs

- Write configs to Zookeeper on converge (via chef)
- Write configs to Zookeeper on deploy (init container)
- Synapse updated to pull these configs from Zookeeper
- Default is to use Chef (EC2) or SD Configurator (K8)

Intermission (2)

Things Keep Falling Apart

chase.childers@

Zookeeper

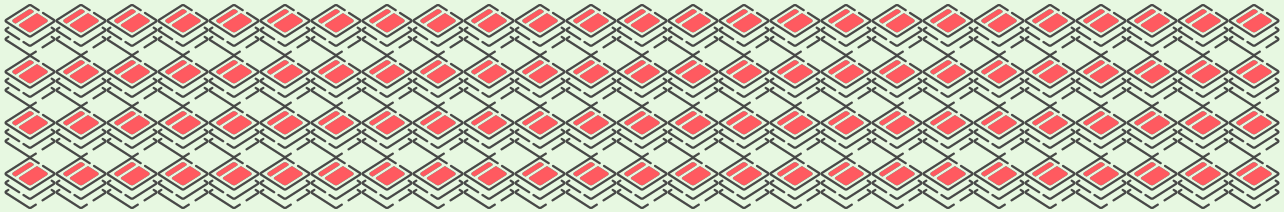
Was that a good idea?

One Service Lots of Backends

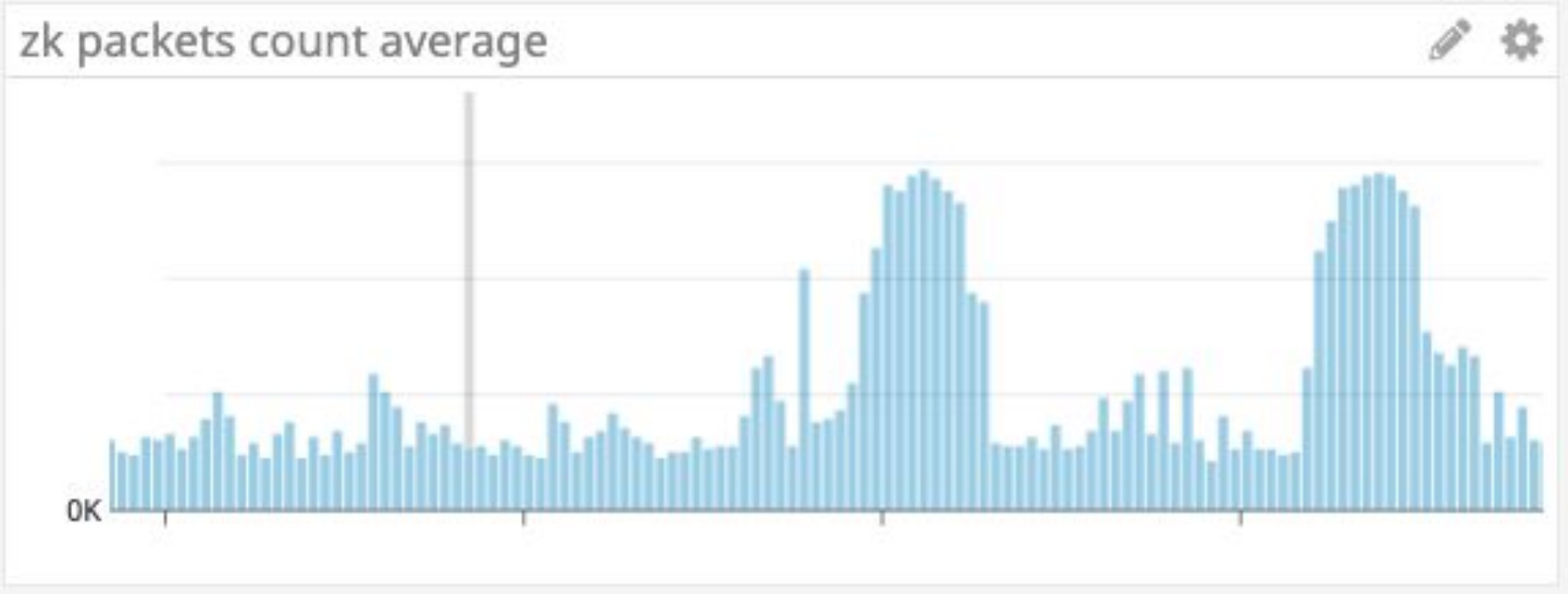
How many packets?
Backends = 10?

How many packets?
Backends = 100?

How many packets?
Backends = 1000?



How many packets?
Backends = 10000?


zk packets count average
0K

avg:system.net.packets_in.count by {host} +
avg:system.net.packets_out.count by {host} > 250000

A machine is under heavy network traffic. Its current PPS is over 250000/sec which is subject to EC2's packets per second (PPS) limit rate. AWS has no official documentation on this as far as we know, but the internet and our experience points to a limit in the 100-150k pps range for classic and 200-300k pps for VPC.

On this topic, AWS has provided the following guidance:

You can always achieve a better rate by selecting a larger instance type which consequently operates on a less busier shared environment.

A machine is under heavy network traffic. Its current PPS is over 250000/sec which is subject to EC2's packets per second (PPS) limit rate. AWS has no official documentation on this as far as we know, but the internet and our experience points to a limit in the 100-150k pps range for classic and 200-300k pps for VPC.

On this topic, AWS has provided the following guidance:

You can always achieve a better rate by selecting a larger instance type which consequently operates on a less busier shared environment.

A machine is under heavy network traffic. Its current PPS is over 250000/sec which is subject to EC2's packets per second (PPS) limit rate. AWS has no official documentation on this as far as we know, but the internet and our experience points to a limit in the 100-150k pps range for classic and 200-300k pps for VPC.

On this topic, AWS has provided the following guidance:

You can always achieve a better rate by selecting a larger instance type which consequently operates on a less busier shared environment.

What happens then?

1. Request Queuing
2. Delayed propagation of service discovery configurations
3. Delayed propagation of backend hosts

A machine is under heavy network traffic. It's current PPS is over 250000/sec which is subject to EC2's packets per second (PPS) limit rate. AWS has no official documentation on this as far as we know, but the internet and our experience points to a limit in the 100-150k pps range for classic and 200-300k pps for VPC.

On this topic, AWS has provided the following guidance:

You can always achieve a better rate by selecting a larger instance type which consequently operates on a less busier shared environment.

Easy. Just Upgrade the Hosts!

Easy. Just Upgrade the Hosts!

- Famous Last Words

What else did we do?

1. Migrate dependencies away from monolith
2. Data encoding in Smartstack
3. Read optimizations; group fetch vs individual fetches (fewer roundtrips)
4. Add jitter and self throttling to Nerve

And Now Back To Our Regularly Scheduled Programming

Service Discovery Containers

Hello Again!

Mini-Announcer
Discoverability

mTLS Syncer
Secure Communication

SDS-Shim
Envoy Shim

Envoy
Ingress / Egress Proxy

HAProxy
Egress Proxy

Synapse
HAProxy Configurator

SD Configurator
Bootstrap Configuration

Service Discovery Containers

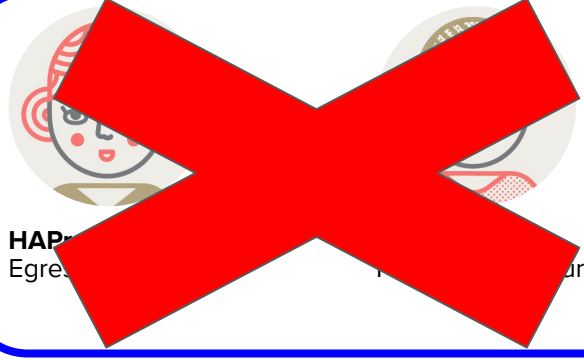
Smartstack is Dead! Long Live Service Discovery!

Mini-Announcer
Discoverability

mTLS Syncer
Secure Communication

SDS-Shim
Envoy Shim

Envoy
Ingress / Egress Proxy



HAP
Egre urator

SD Configurator
Bootstrap Configuration

Phase 2.5

(origin/HEAD) [INPROGRESS] Migrations

You Break Service Discovery

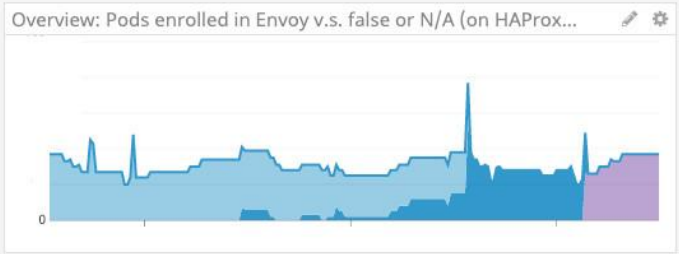
You Break EVERYTHING

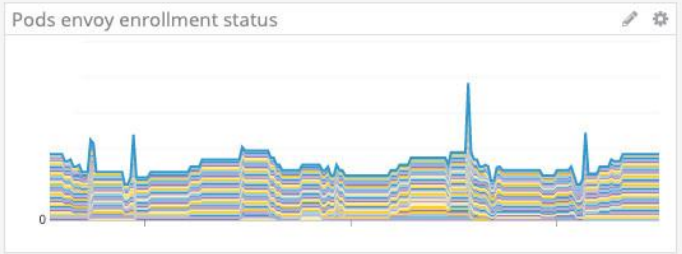
Challenges we face during migration other than scale

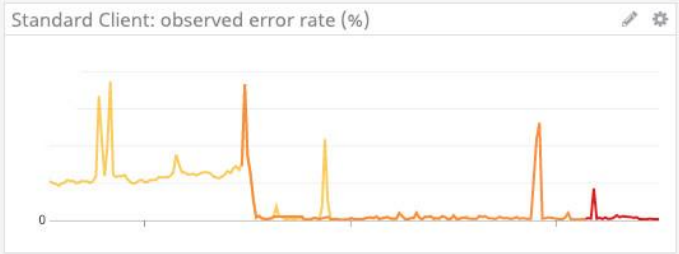
1. Transparency to and for Service Owners

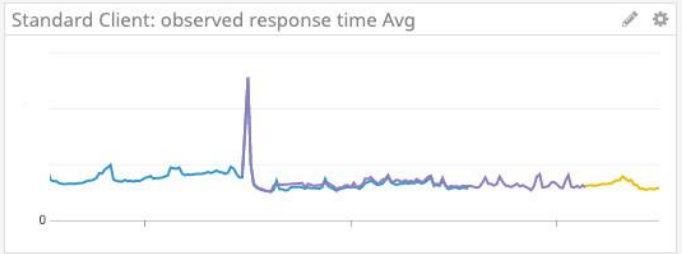
A/B Dashboards

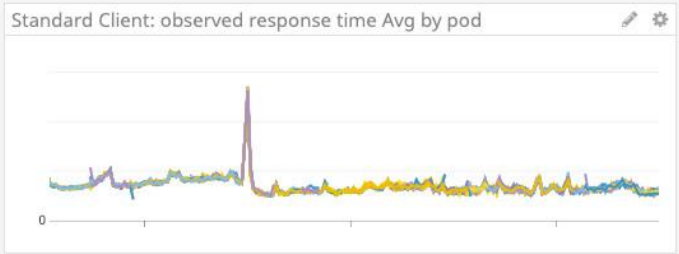
Service Level Metrics

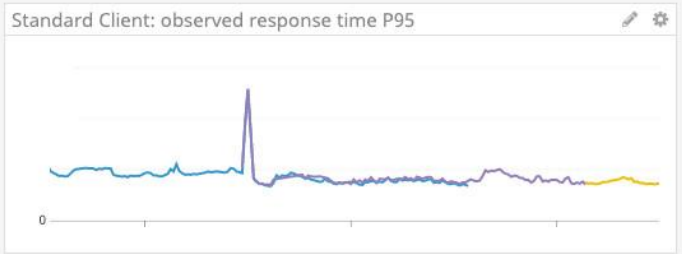


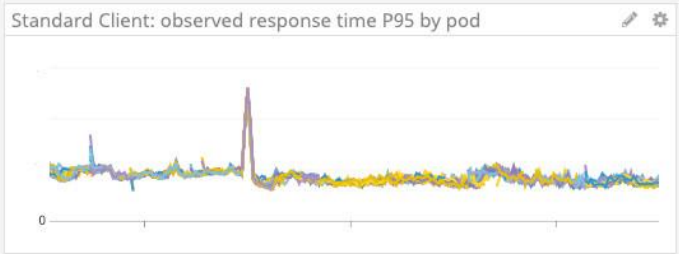


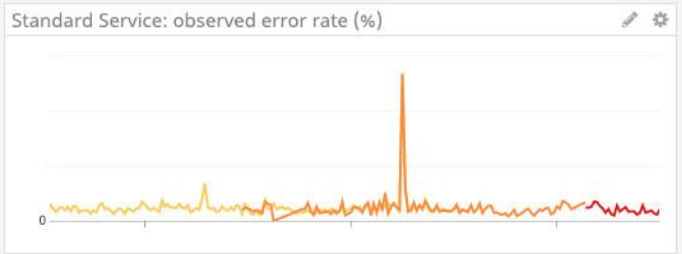












chase.childers@

Challenges we face during migration other than scale

1. Transparency to and for Service Owners

2. **Iterating while migrating**

Challenges we face during migration other than scale

1. Transparency to and for Service Owners
2. Iterating while migrating
3. **Ad-hoc and non-standardized service configurations**

Challenges we face during migration other than scale

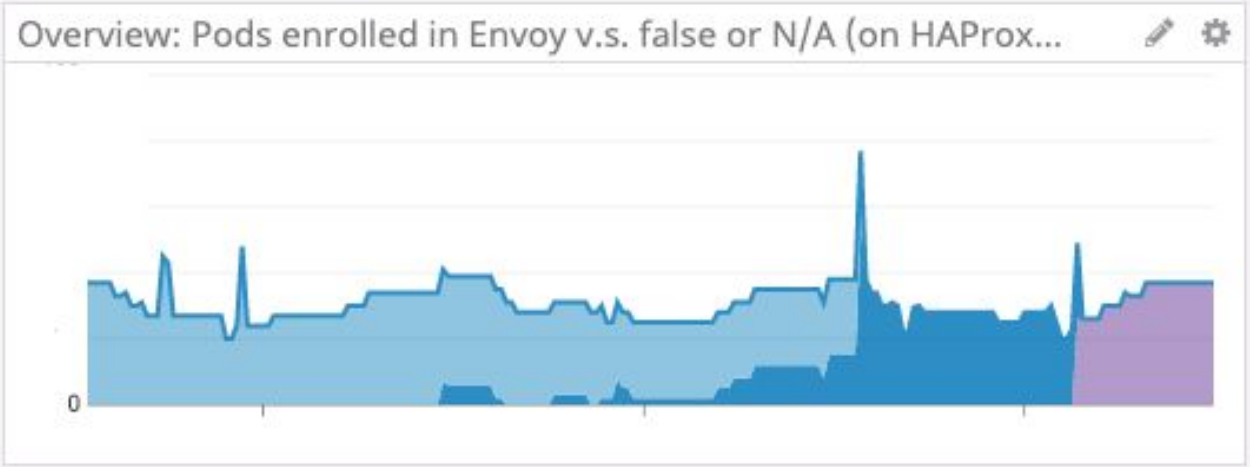
1. Transparency to and for Service Owners
2. Iterating while migrating
3. Ad-hoc and non-standardized service configurations
4. **Distributed service configurations source of truth**

Challenges we face during migration other than scale

1. Transparency to and for Service Owners
2. Iterating while migrating
3. Ad-hoc and non-standardized service configurations
4. Distributed service configurations source of truth
5. **Service discovery version rollouts**

Version Tracking

Via Metric Tagging



Recap / Learnings

Service Discovery is Hard

Service Discovery is Hard

Scaling is Hard

Service Discovery is Hard

Scaling is Hard

Migrating is Hard

Service Discovery is Hard

Scaling is Hard

Migrating is Hard

Engineering is Hard

Dolomites, Italy

Jobs @ airbnb.com/careers
Me @ chasechilders.com

chase.childers@

Sec 20CD
Nov 19
4:25 PM

Scaling Kubernetes to Thousands of Nodes Across Multiple Clusters, Calmly

Ben Hughes

RM 15AB
Nov 20
11:50 AM

Did Kubernetes Make My p95's Worse?

Jian Cheung and Stephen Chan

Dolomites, Italy

Jobs @ airbnb.com/careers
Me @ chasechilders.com

chase.childers@