fastly®

fastly

# ROUTING-STATE OBSERVATION AND MODIFICATION IN A LARGE NETWORK THANKS TO PULUMI-MANAGED API SERVICES IN AWS/ECS

fastly

# AUTOPILOT-ROUTED

# Agenda

- Introduction
- Project overview
- Solution description
- Problems, issues and learnings
- Future
- Closure

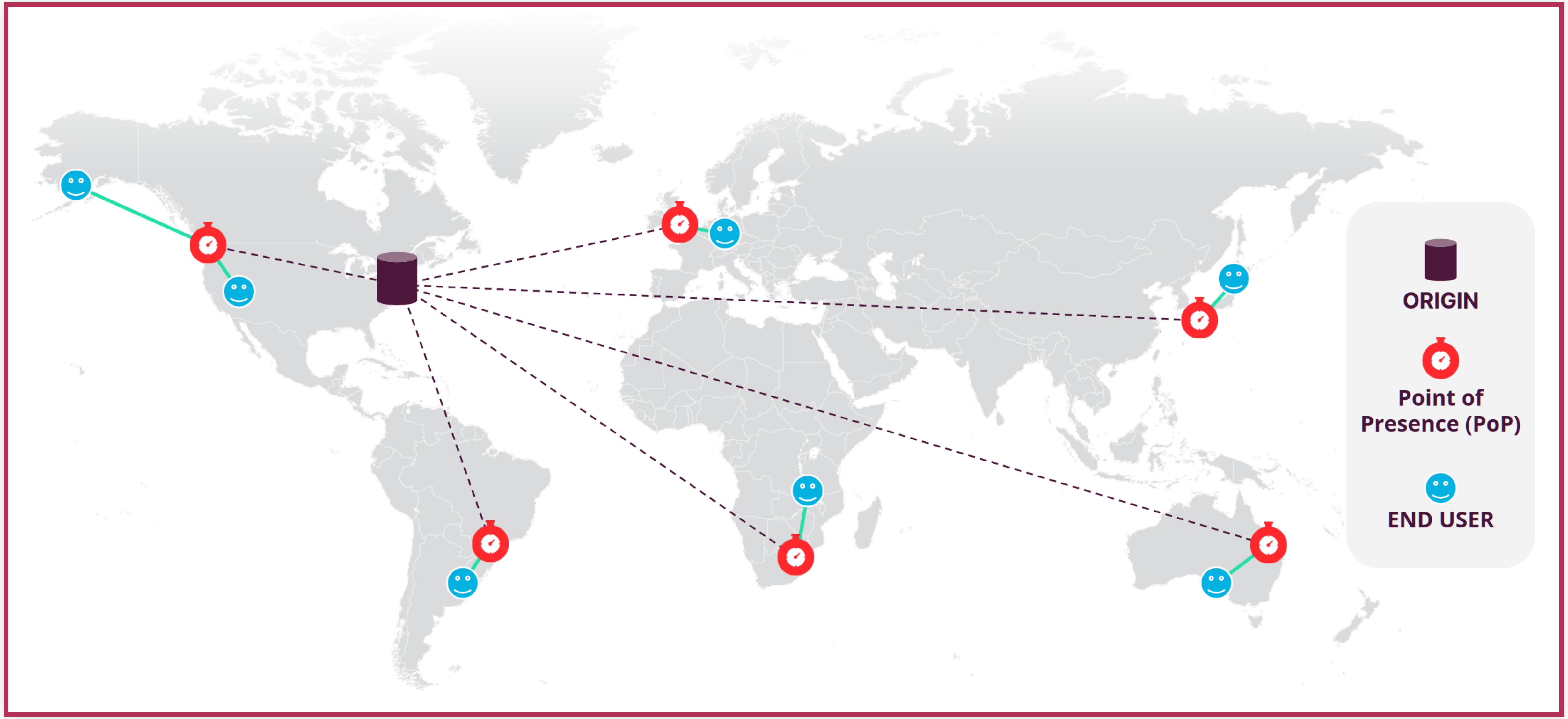
# INTRODUCTION

# Daniel Caballero Rodríguez

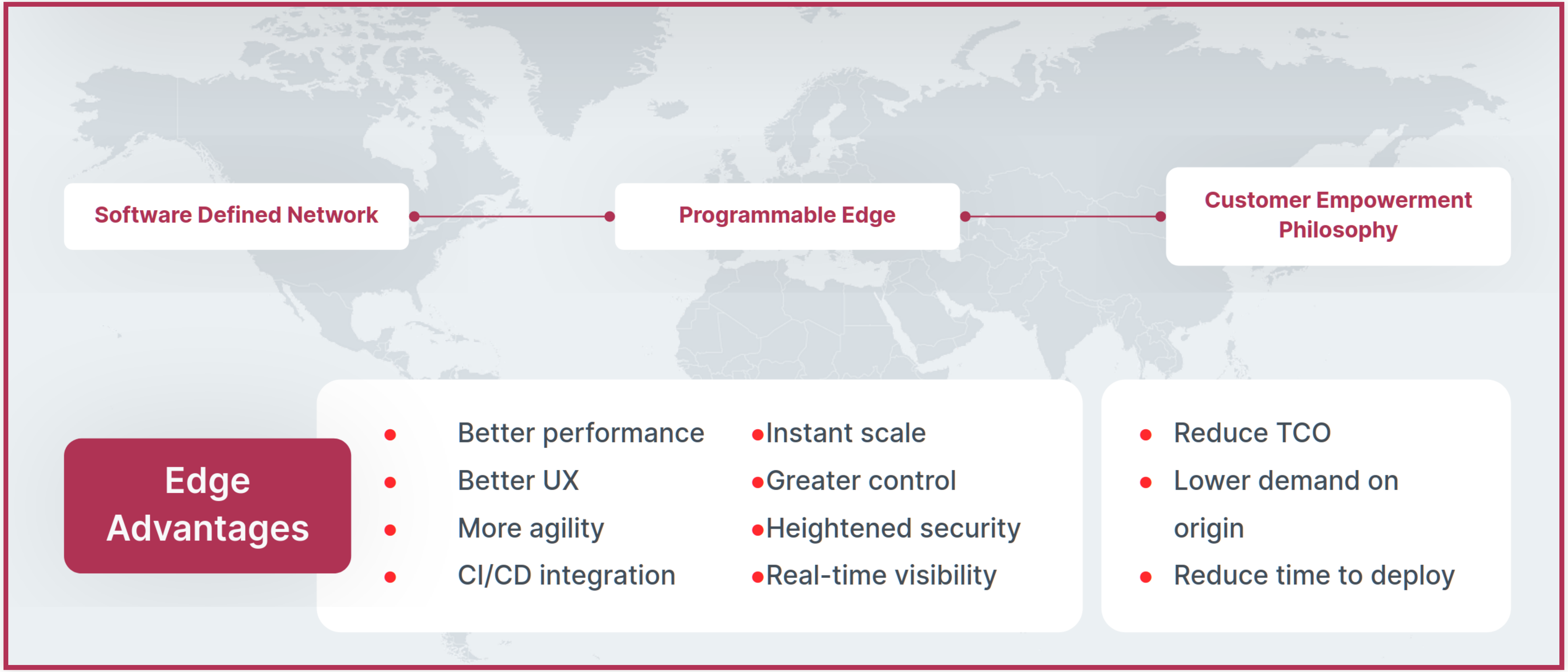
- Principal SREngineer @ **Fastly**
- Previously part time (Devops) lecturer, Devops@**Schibsted** (now Adevinta), **NTT**, **Oracle**…
- *Eventual OSS contributions* (see tcpgoon)

fastly®

# OUR BUSINESS...



# WHAT MAKES US UNIQUE



Software Defined Network

Programmable Edge

Customer Empowerment Philosophy

Edge Advantages

- Better performance
- Better UX
- More agility
- CI/CD integration

- Instant scale
- Greater control
- Heightened security
- Real-time visibility

- Reduce TCO
- Lower demand on origin
- Reduce time to deploy

# FASTLY AT A GLANCE

**>1.4Trillion**

Requests served daily[1]

**215 Tbps**

Edge Capacity[2]

**95%+**

Customer satisfaction score[3]

**13s**

Median Global Deploy Time[4]

**100**

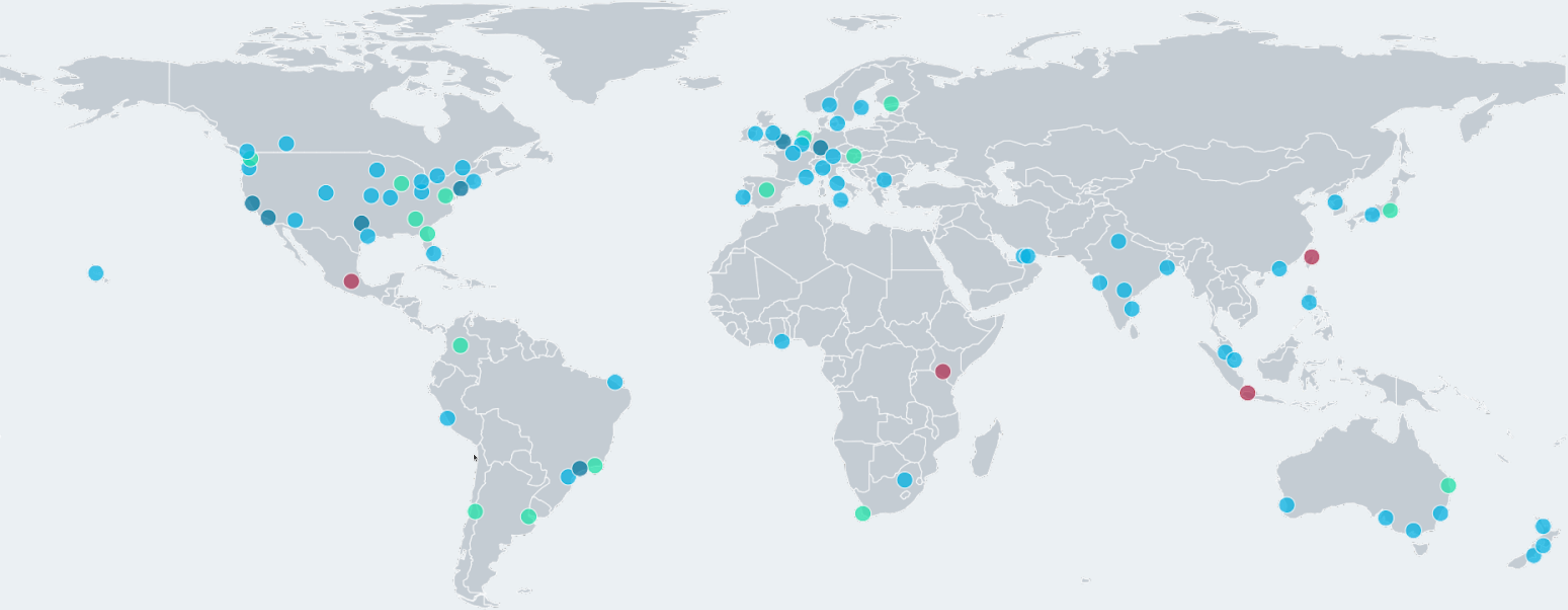
Number of POPs[5]

**>420 Billion**

Images optimized and delivered Per Month[6]

[1] As of January 31, 2022
[2] As of June 30, 2022
[3] As of June 1, 2022
[4] As of March 31, 2019
[5] As of June 30, 2022
[6] As of March 31, 2019

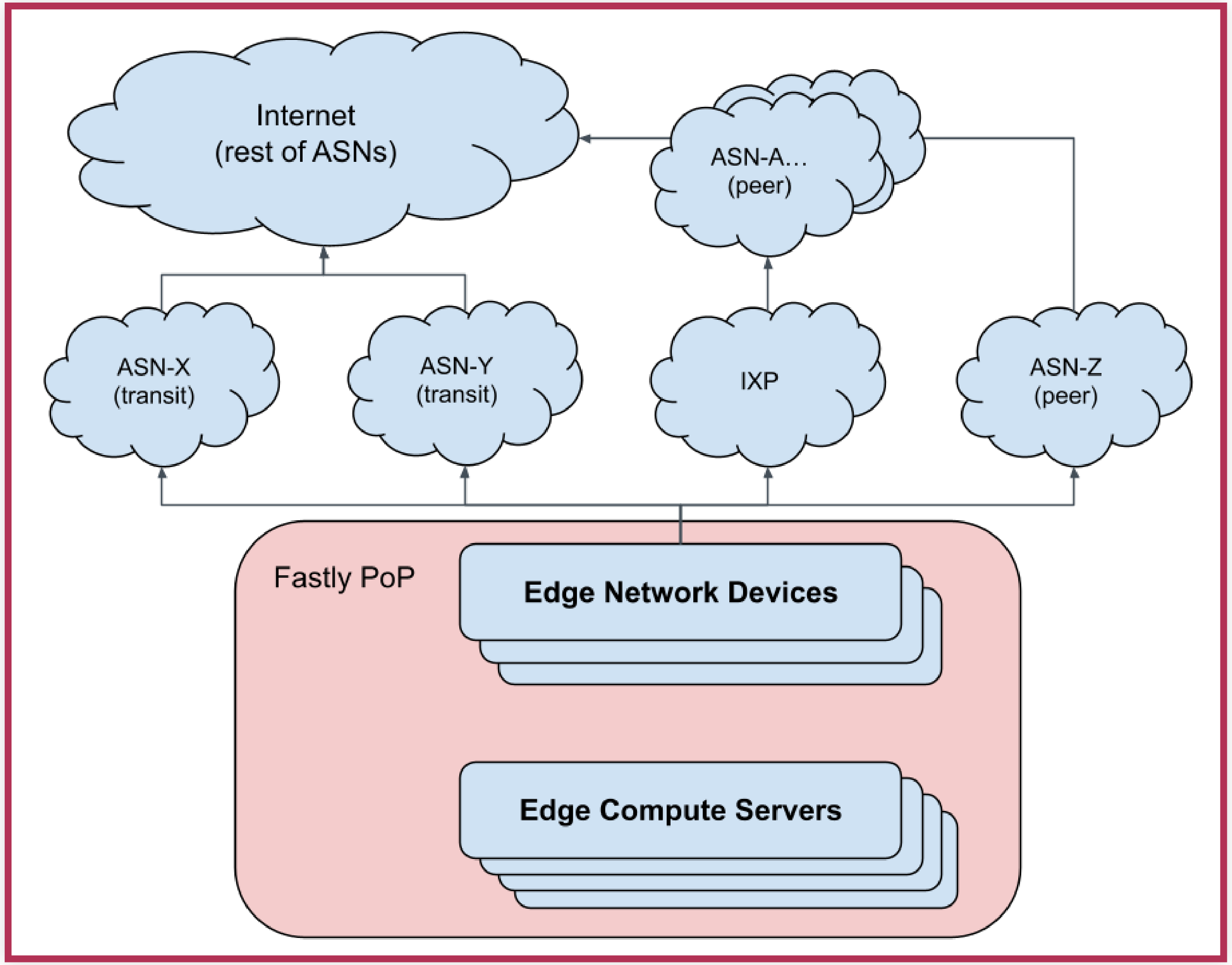
# Fastly Points-of-Presence (POPs)



● Current POP ● Current multi-POP ● Planned new POP ● Planned POP expansion

Map Updated June 30, 2022

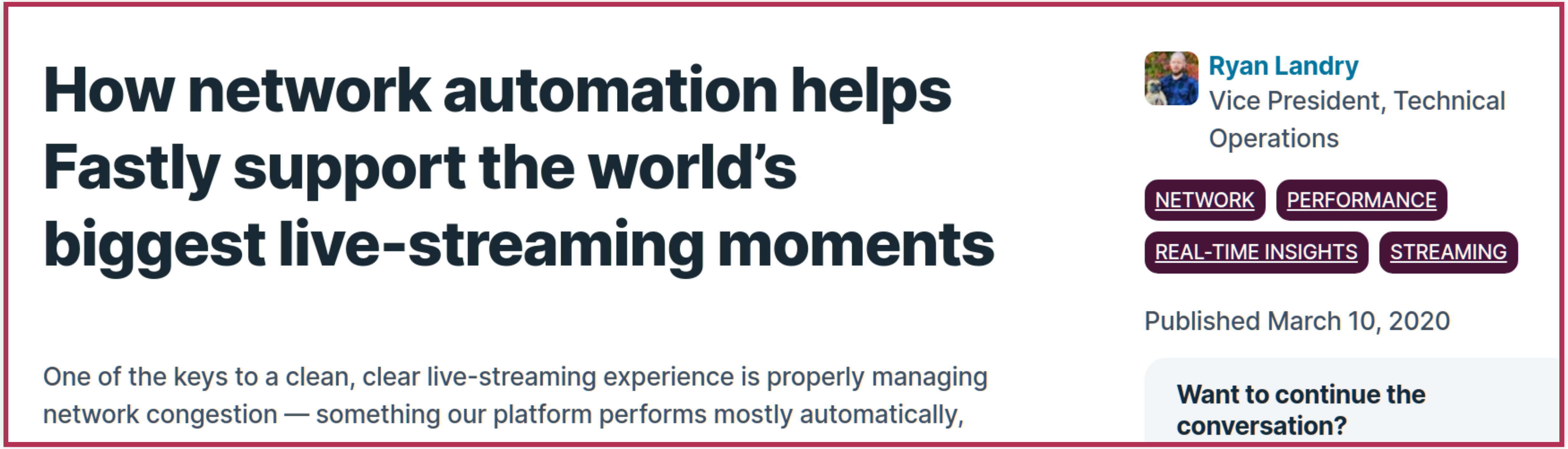
# FASTLY AND EGRESS TRAFFIC ENGINEERING



# WE NEED AUTOMATION

# PROJECT OVERVIEW

# **Automation** in this problem domain is **not a new concept**



## How network automation helps Fastly support the world's biggest live-streaming moments

Ryan Landry
Vice President, Technical Operations

NETWORK PERFORMANCE REAL-TIME INSIGHTS STREAMING

Published March 10, 2020

One of the keys to a clean, clear live-streaming experience is properly managing network congestion — something our platform performs mostly automatically,

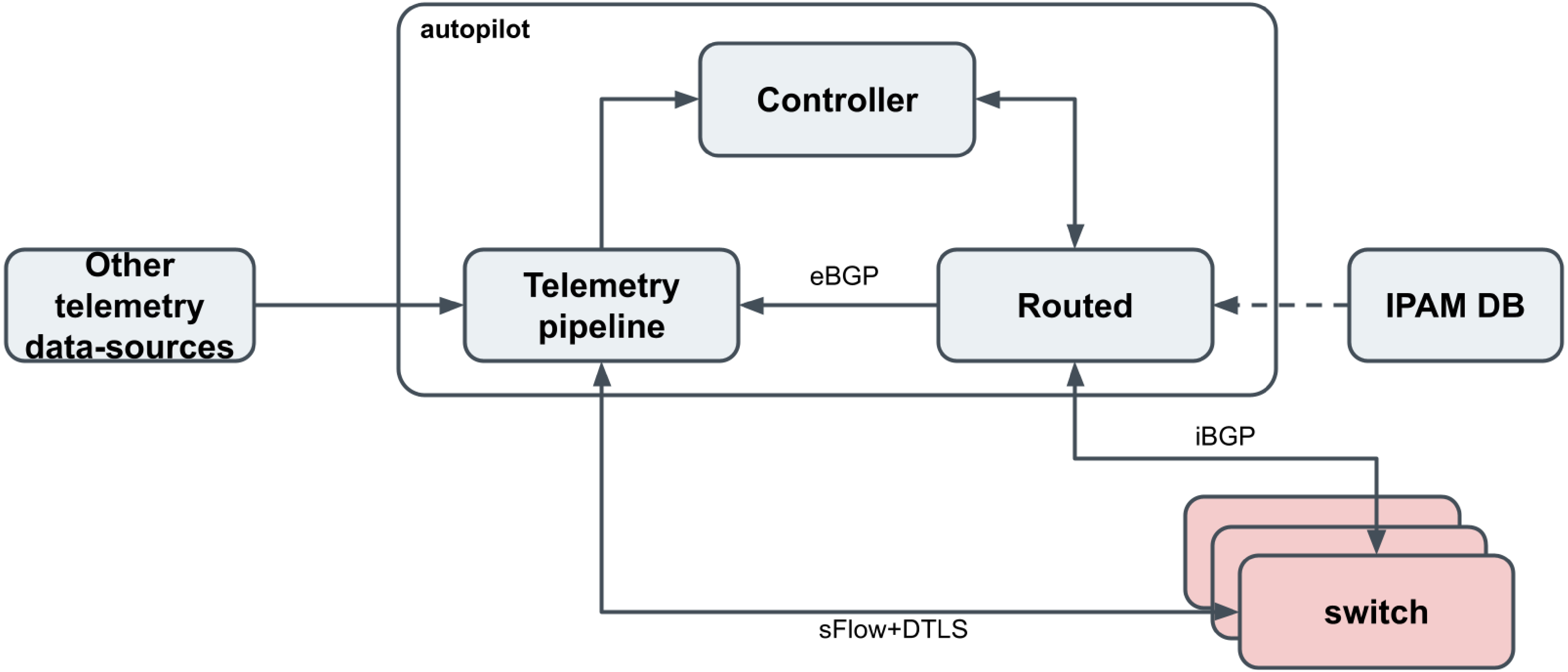
**Want to continue the conversation?**

https://www.fastly.com/blog/network-automation-helps-support-worlds-biggest-live-streaming-moments

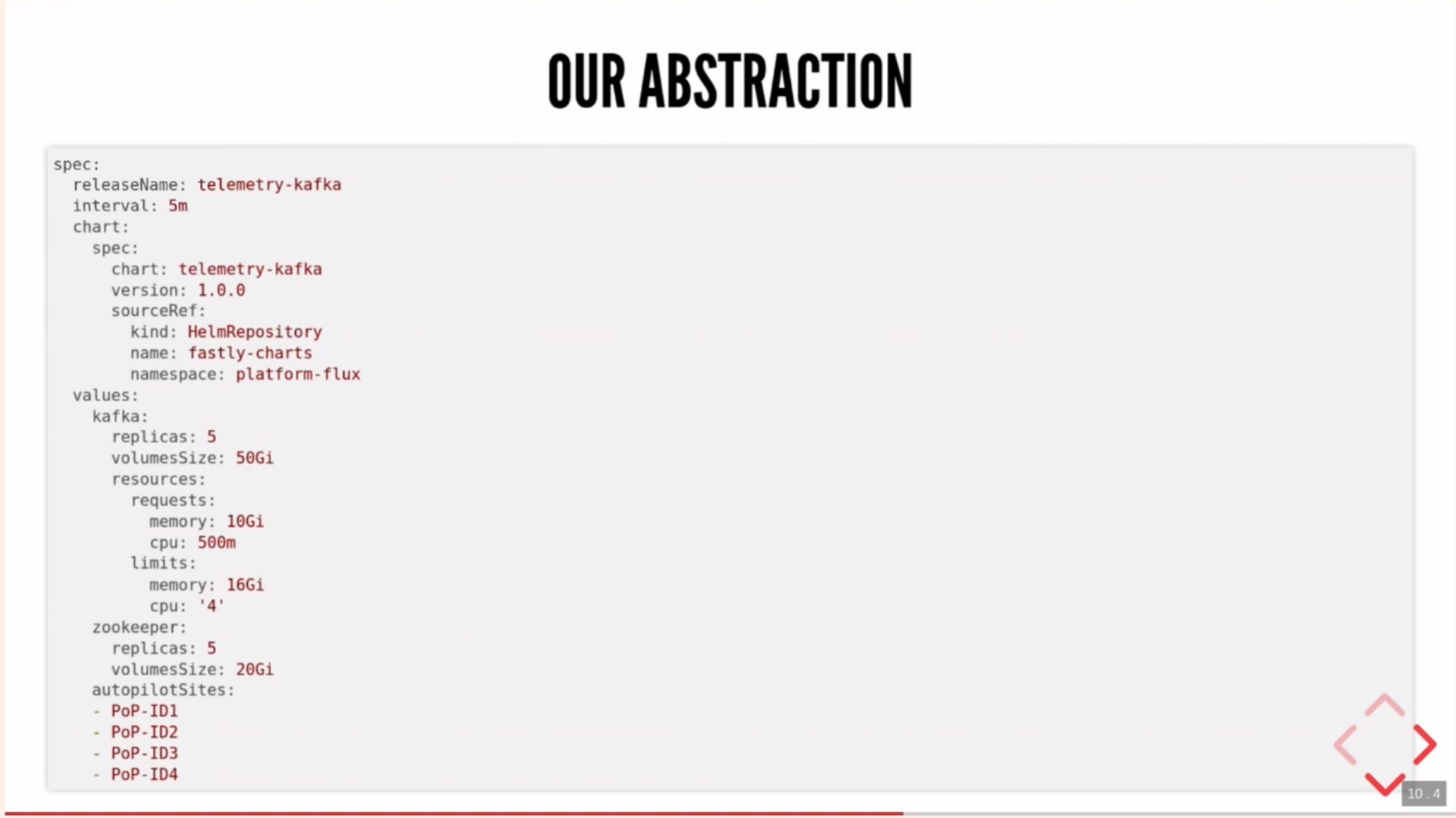
# AUTOPILOT AS THE LAST ITERATION

- Developed and maintained by the *Network Control and Optimization* (**NCO**) team
- Initially around 6 people…
  - with different profiles (Software Engineers, Data Engineers, Network Engineers, SRE)

# AUTOPILOT AT HIGH LEVEL

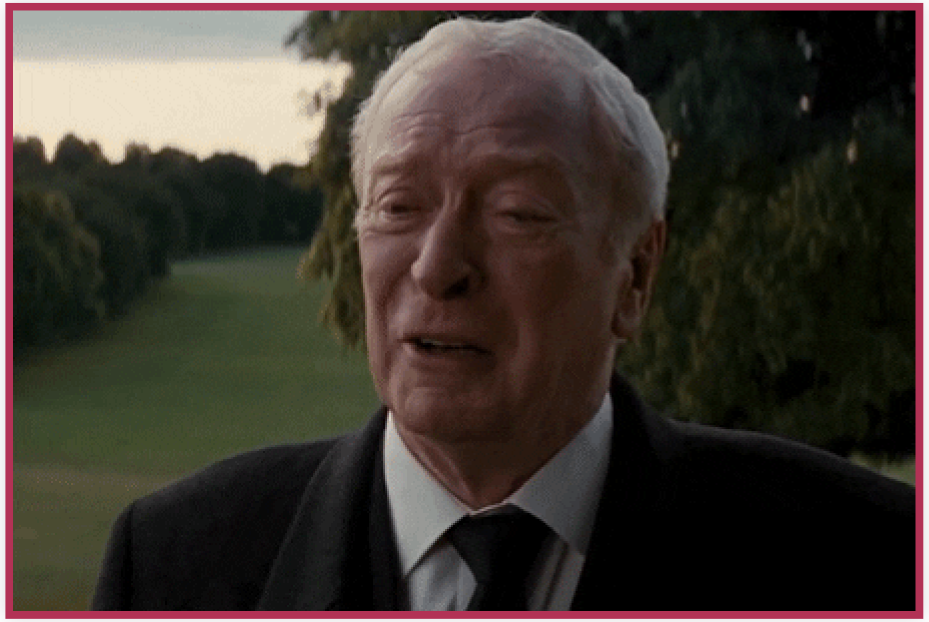


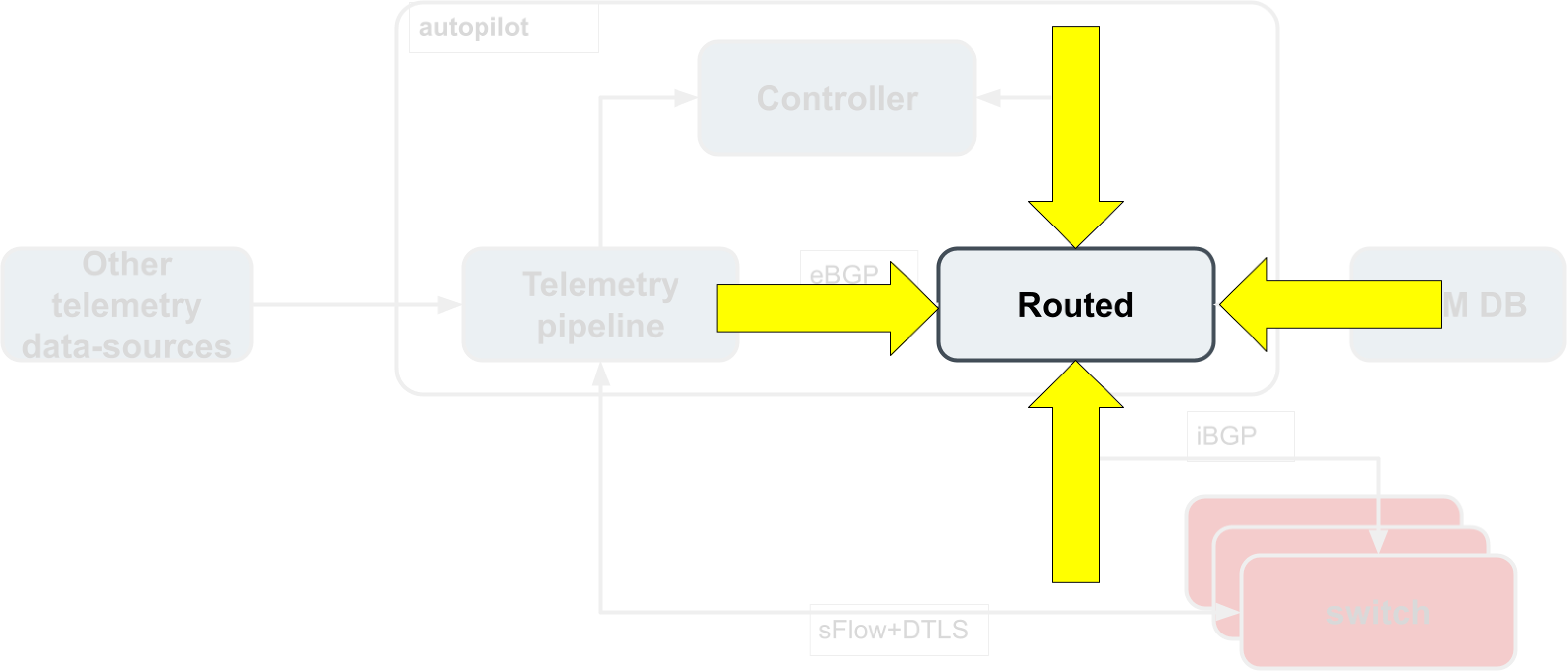
KubeCon CloudNativeCon Europe 2022



# OUR ABSTRACTION

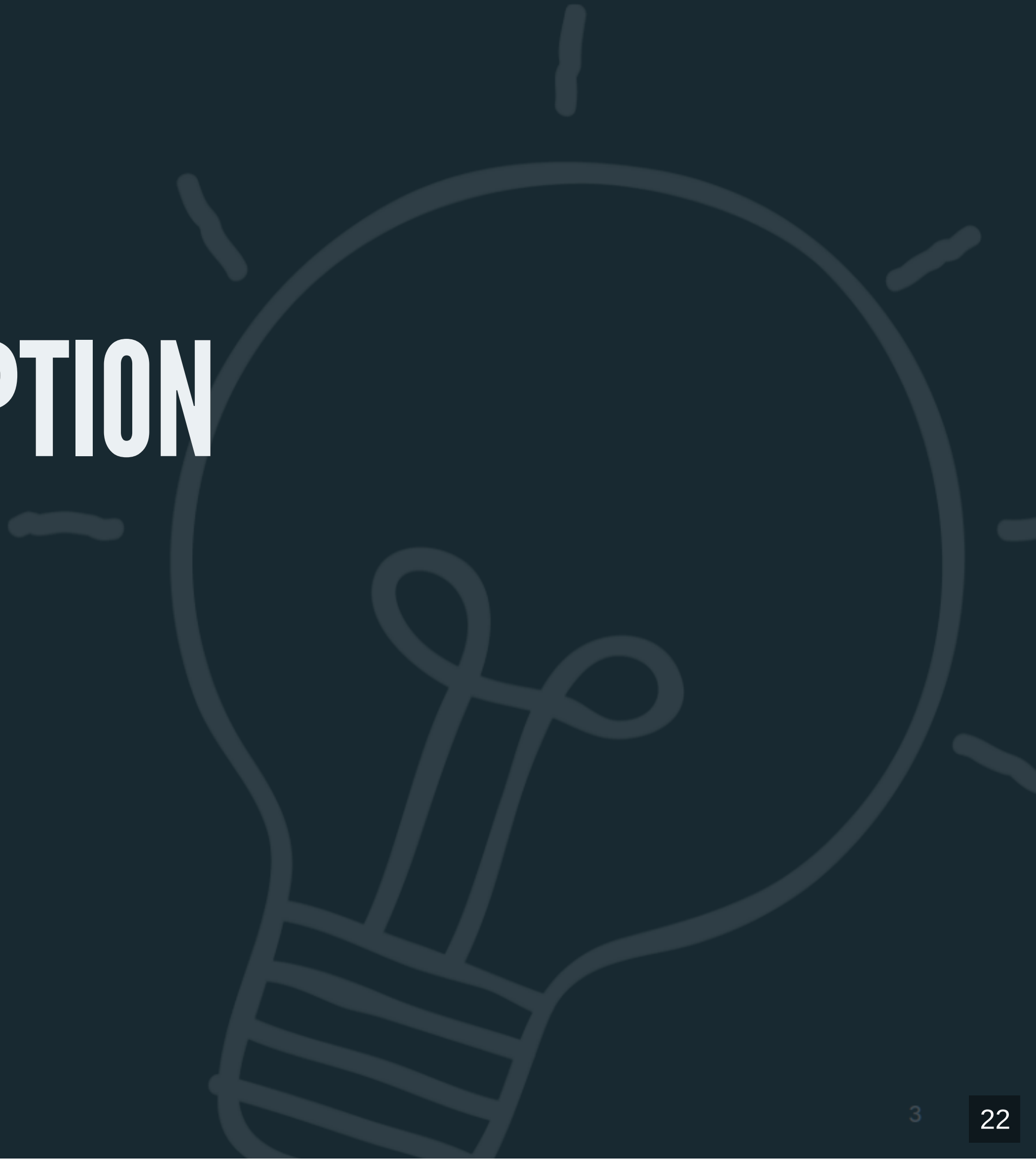
```
spec:
  releaseName: telemetry-kafka
  interval: 5m
  chart:
    spec:
      chart: telemetry-kafka
      version: 1.0.0
      sourceRef:
        kind: HelmRepository
        name: fastly-charts
        namespace: platform-flux
  values:
    kafka:
      replicas: 5
      volumesSize: 50Gi
      resources:
        requests:
          memory: 10Gi
          cpu: 500m
        limits:
          memory: 16Gi
          cpu: '4'
    zookeeper:
      replicas: 5
      volumesSize: 20Gi
    autopilotSites:
    - PoP-ID1
    - PoP-ID2
    - PoP-ID3
    - PoP-ID4
```



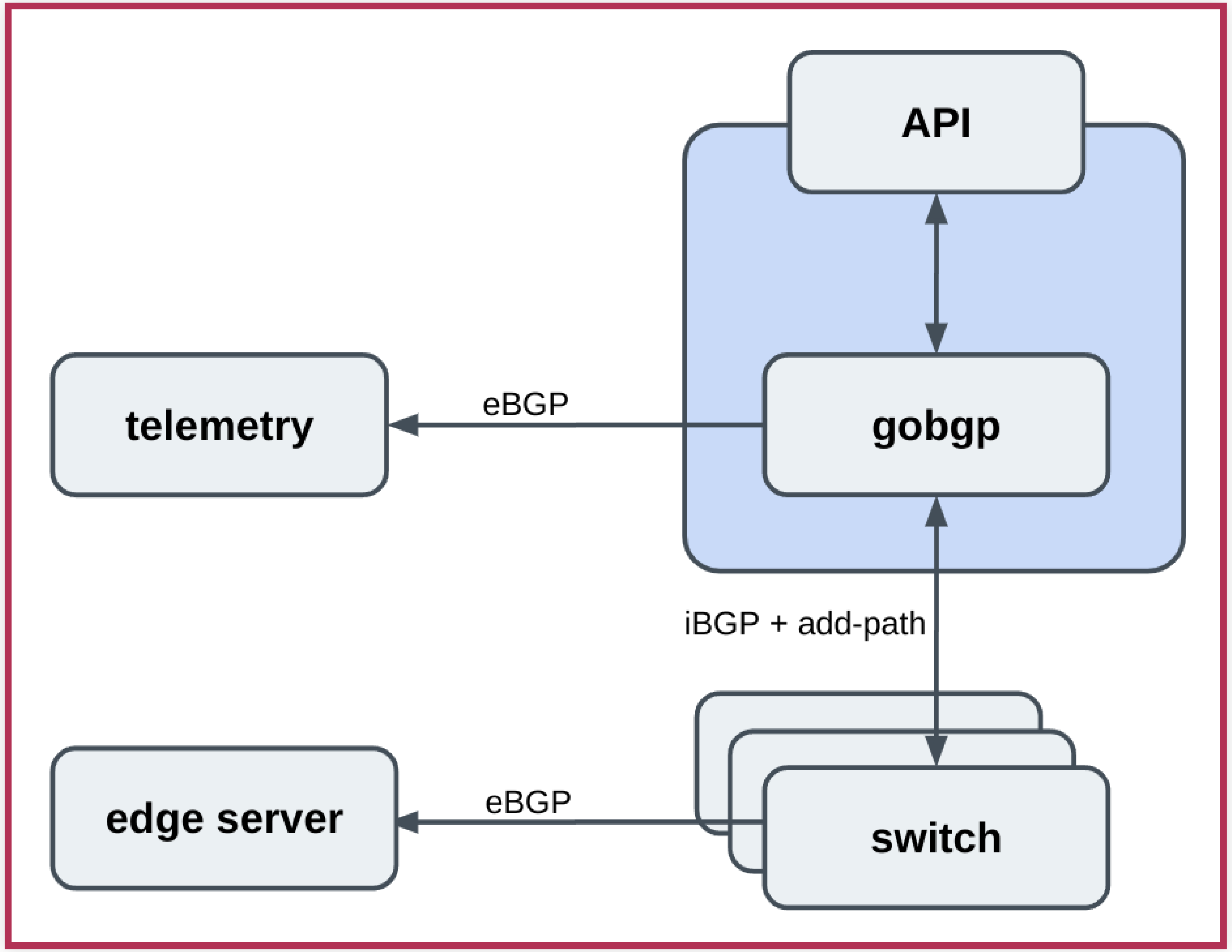

autopilot
Controller
Other telemetry data-sources
Telemetry pipeline
eBGP
Routed
M DB
iBGP
switch
sFlow+DTLS

# THIS TALK IS ABOUT THAT NON-K8S PORTION

# SOLUTION DESCRIPTION



# AUTOPILOT-ROUTED



# WHY NOT K8S?

# *This is our (BGP) routing daemon*

- **bird, networking ACLs, NATs and dynamic IPs** are not good friends
    - And our bird setup also requires **end to end IPv6**

- **bird, networking ACLs, NATs and dynamic IPs** are not good friends
  - And our bird setup also requires **end to end IPv6**

# K8S-APPROACH CHALLENGES

- Dual stack (IPv4/IPv6) was still in early stages…
- Not supported by the tooling that manages our multi-cloud k8s
- No *AWS NAT Gateway* equivalent for IPv6

# OTHER INTERESTING REQUIREMENTS

Critical, and quite large, in-memory DB… that takes a lot (>10m) to start/rebuild its state

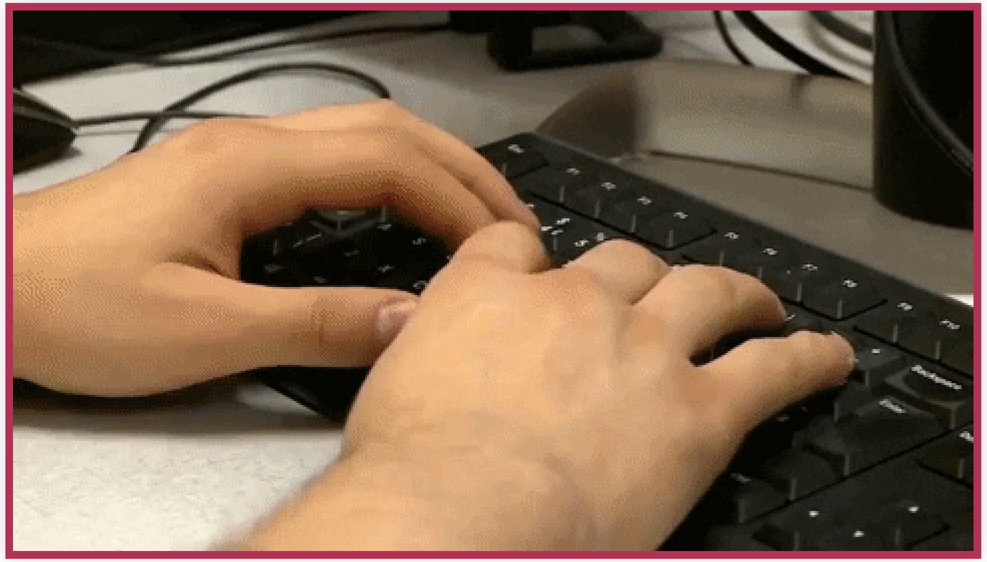
- 200 tasks/pods (2 per Fastly PoP)
- Each requiring about 4 CPU cores and more than 20G of RAM

# So we decided to isolate the complexity in some dedicated autopilot-routed infra

# WHY ECS?

- implicit decision, **AWS**:
  - GCP did not offer end to end **IPv6 with static IPv6 addresses**
- **containerized apps support**, to reduce the gap with the k8s services
- comes with an **app deployment solution**
- **less complex** than a dedicated k8s cluster
- **more flexible than** long running **chef**-managed machines

# And managing this infra/deployment also requires automation…

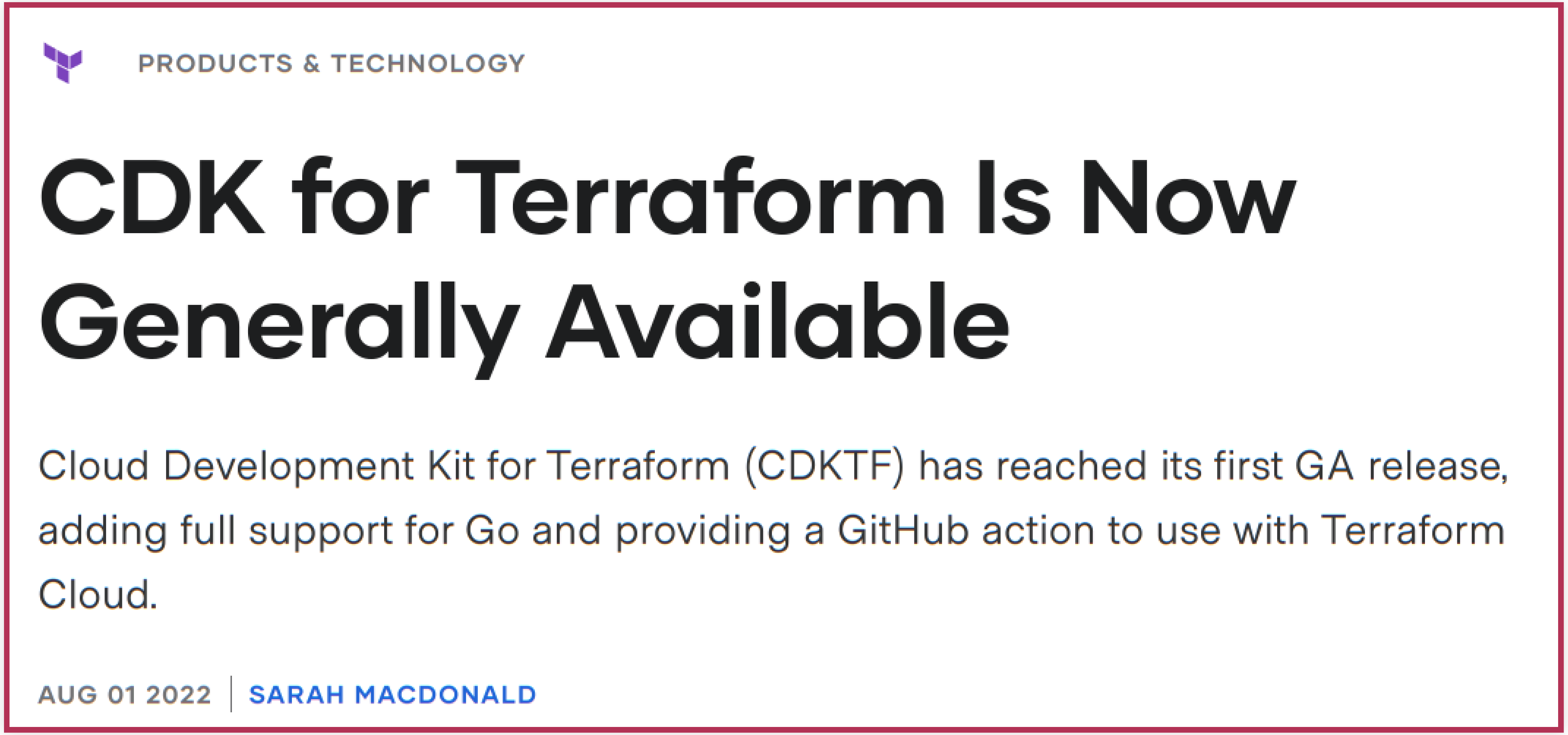


# Summarizing (some) IaC options

| Tool | Provider | Declarative? | Deployment tool? | Config-driven | Programmatic | Language |
|---|---|---|---|---|---|---|
| AWS API | AWS-only | N | N | N | Y | N/A |
| Cloudformation | AWS-only | Y | More or less | Y | N | JSON/YAML |
| Troposphere | AWS-only | Y | N | N | Y | Python |
| CDK | AWS-only | Y | Y | N | Y | Multiple |
| Terraform | Multiple | Y | Y | Y | N | HCL, JSON(v12) |
| Terrascript | Multiple | Y | N | N | Y | Python |
| Pulumi | Multiple | Y | Y | N | Y | Multiple |

# WHY PULUMI?

- Team with software engineering skills and already had some infra managed by Pulumi
- Terraform was not well standardized at Fastly
- Pure-code approach to discover configuration (rather than custom data sources)



PRODUCTS & TECHNOLOGY

# CDK for Terraform Is Now Generally Available

Cloud Development Kit for Terraform (CDKTF) has reached its first GA release, adding full support for Go and providing a GitHub action to use with Terraform Cloud.

AUG 01 2022 | SARAH MACDONALD

# IAC AND STACKS

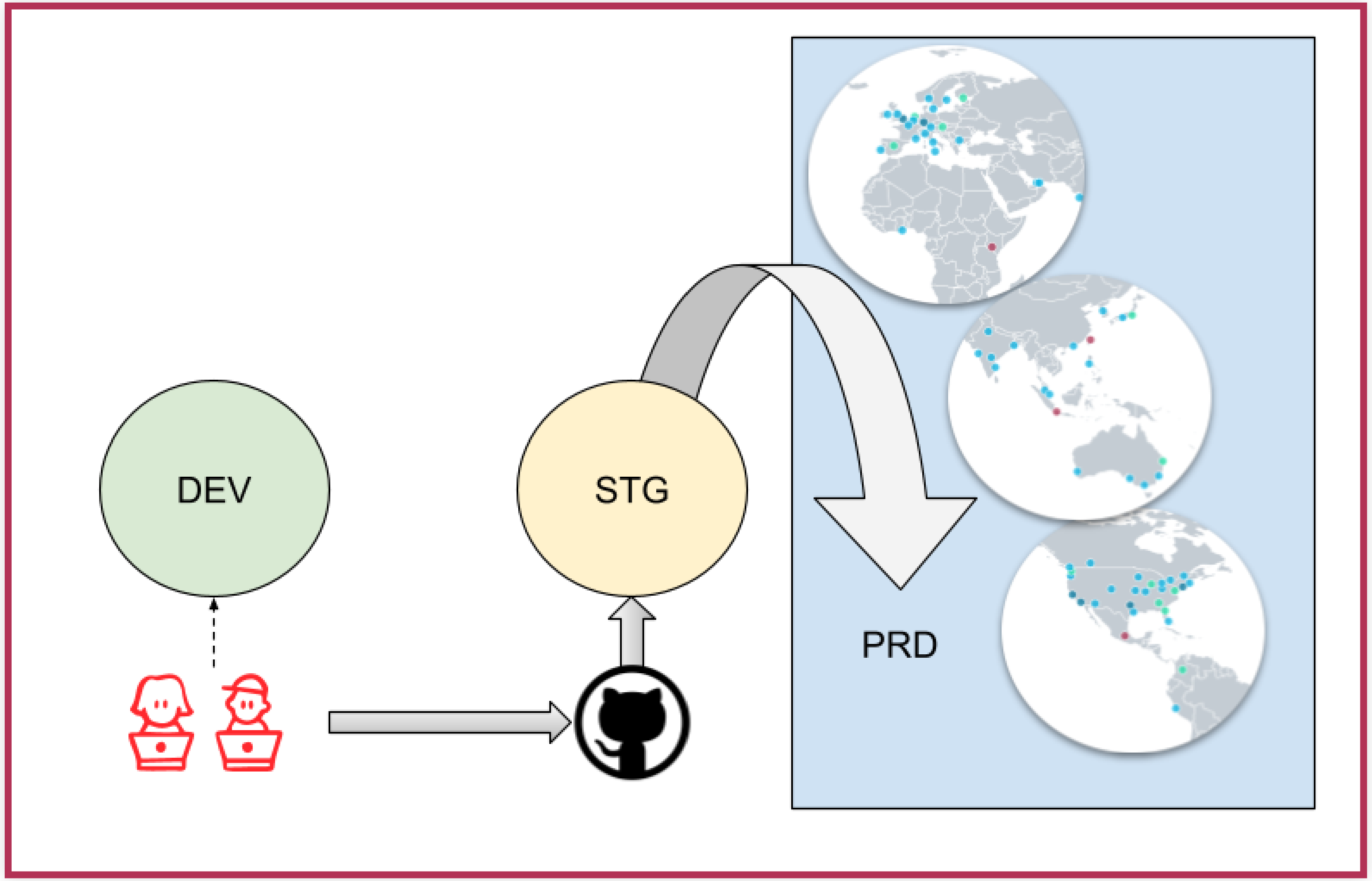
Given an environment (cluster) configuration:

- to support X Fastly PoPs
- in the most convenient AWS region

## Creates:

- ECS cluster and some other GCP resources (ALBs, S3 buckets…)
- N immutable nodes distributed across AZs
  - with “persistent” static IPs (not an ASG)
- and X ECS tasks using host networking

# ENVIRONMENTS AND DEPLOYMENT FLOW



# OBSERVABILITY

fastly®

PromCon 2018

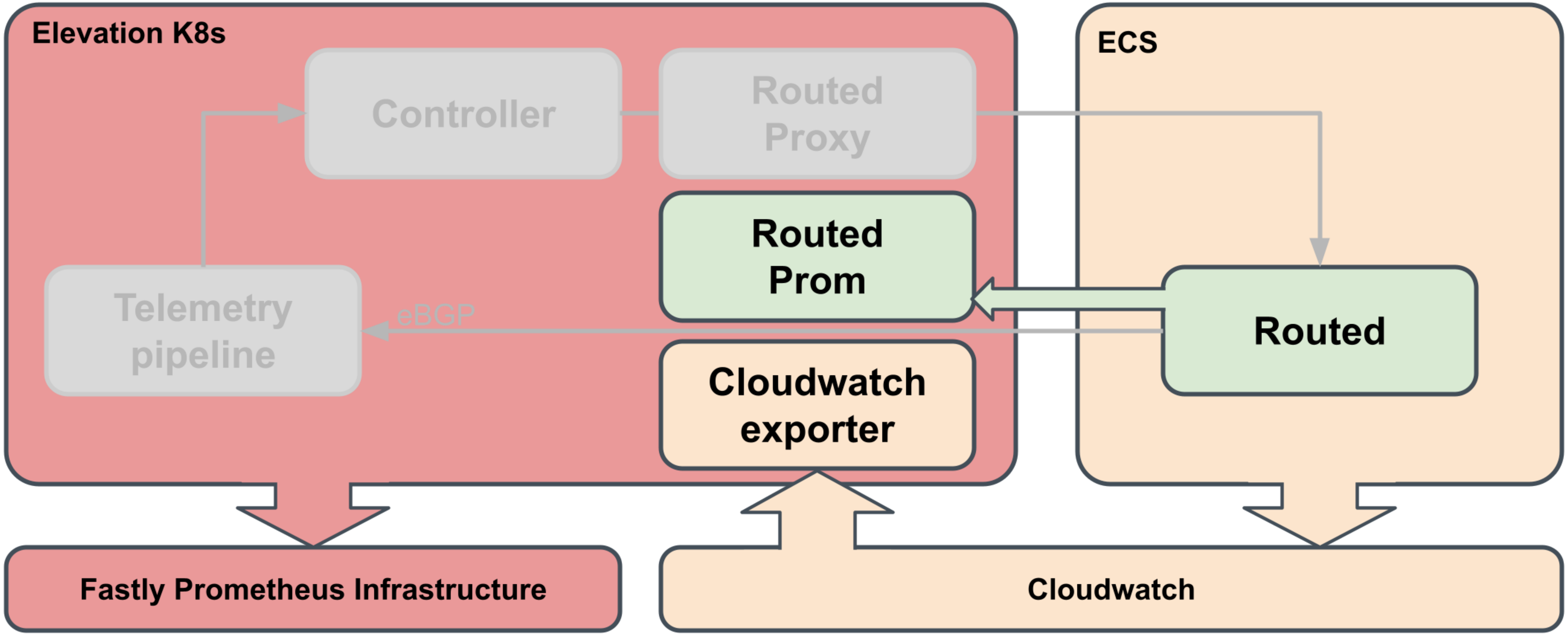
## Monitoring at Scale: Migrating to Prometheus at Fastly

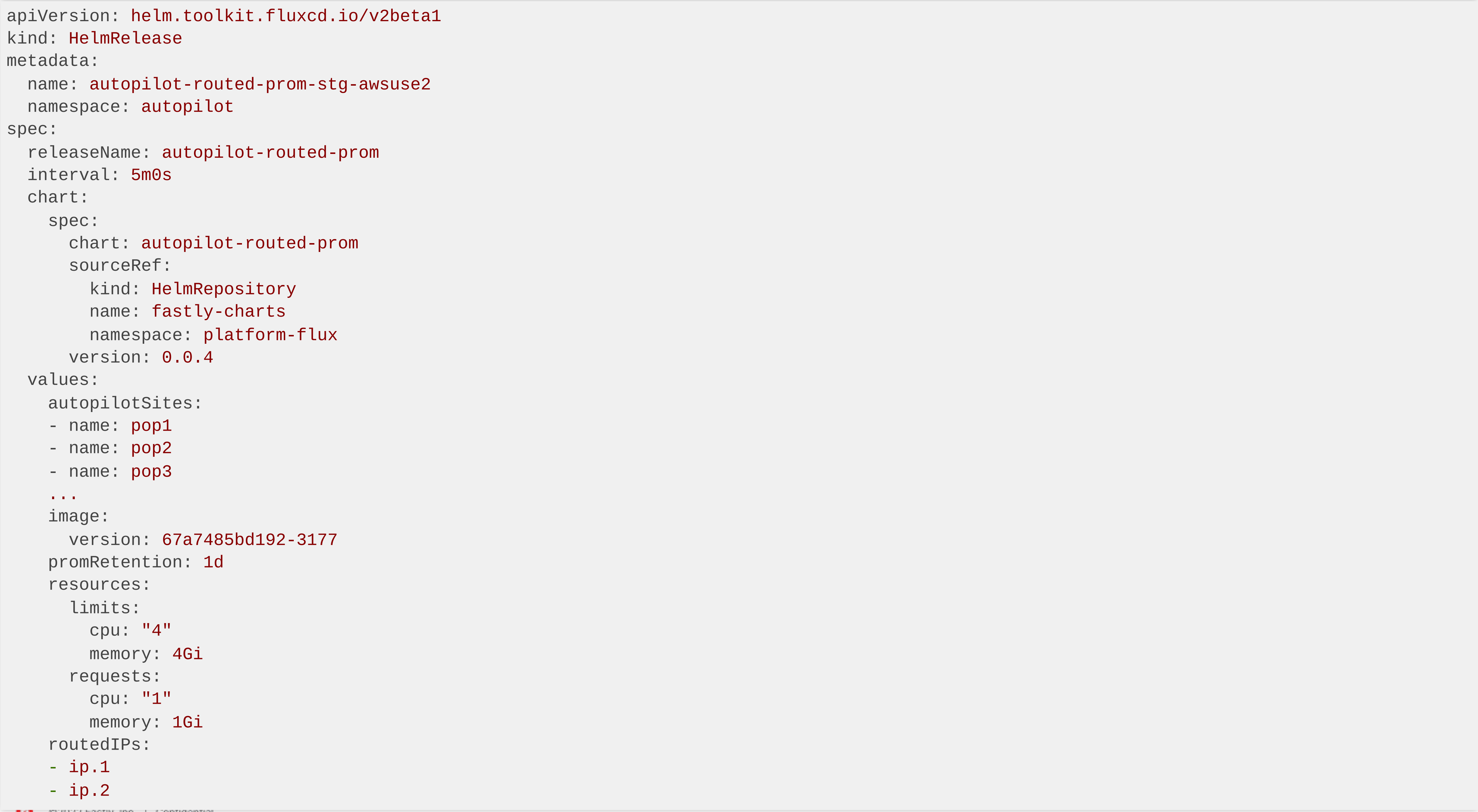
Marcus Barczak
Fastly

DAY 2 - FRIDAY, AUGUST 10

https://prometheus.io
https://promcon.io/2018-munich
@PrometheusIO / #PromCon2018

CLOUD NATIVE
COMPUTING FOUNDATION

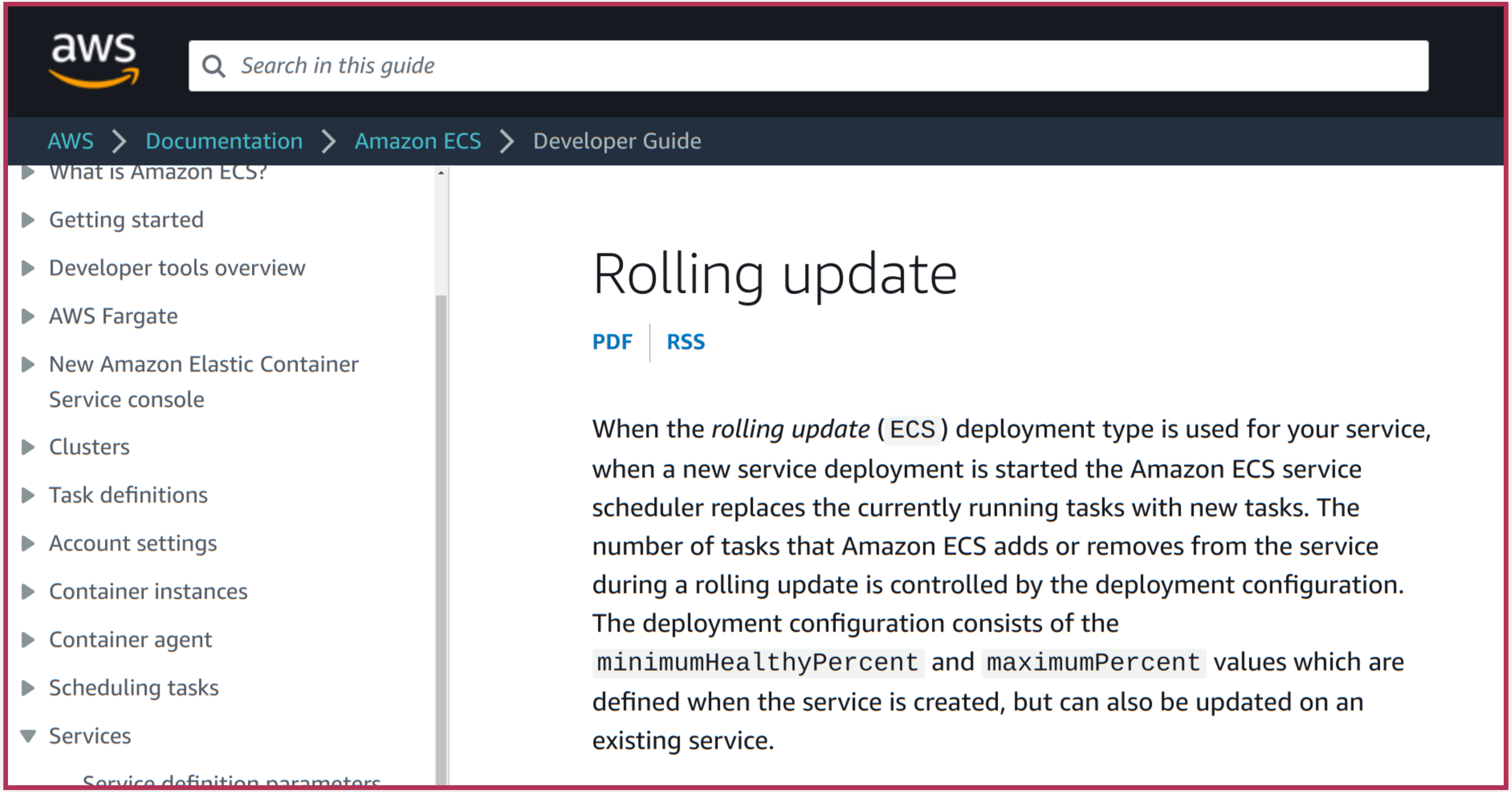

Elevation K8s
Controller
Routed Proxy
Routed Prom
Telemetry pipeline
eBGP
Cloudwatch exporter
ECS
Routed
Fastly Prometheus Infrastructure
Cloudwatch



```yaml
apiVersion: helm.toolkit.fluxcd.io/v2beta1
kind: HelmRelease
metadata:
  name: autopilot-routed-prom-stg-awsuse2
  namespace: autopilot
spec:
  releaseName: autopilot-routed-prom
  interval: 5m0s
  chart:
    spec:
      chart: autopilot-routed-prom
      sourceRef:
        kind: HelmRepository
        name: fastly-charts
        namespace: platform-flux
      version: 0.0.4
  values:
    autopilotSites:
    - name: pop1
    - name: pop2
    - name: pop3
    ...
    image:
      version: 67a7485bd192-3177
    promRetention: 1d
    resources:
      limits:
        cpu: "4"
        memory: 4Gi
      requests:
        cpu: "1"
        memory: 1Gi
    routedIPs:
    - ip.1
    - ip.2
```

# PROBLEMS, ISSUES AND LEARNINGS

# ISSUE 1: NON DISRUPTIVE UPGRADES

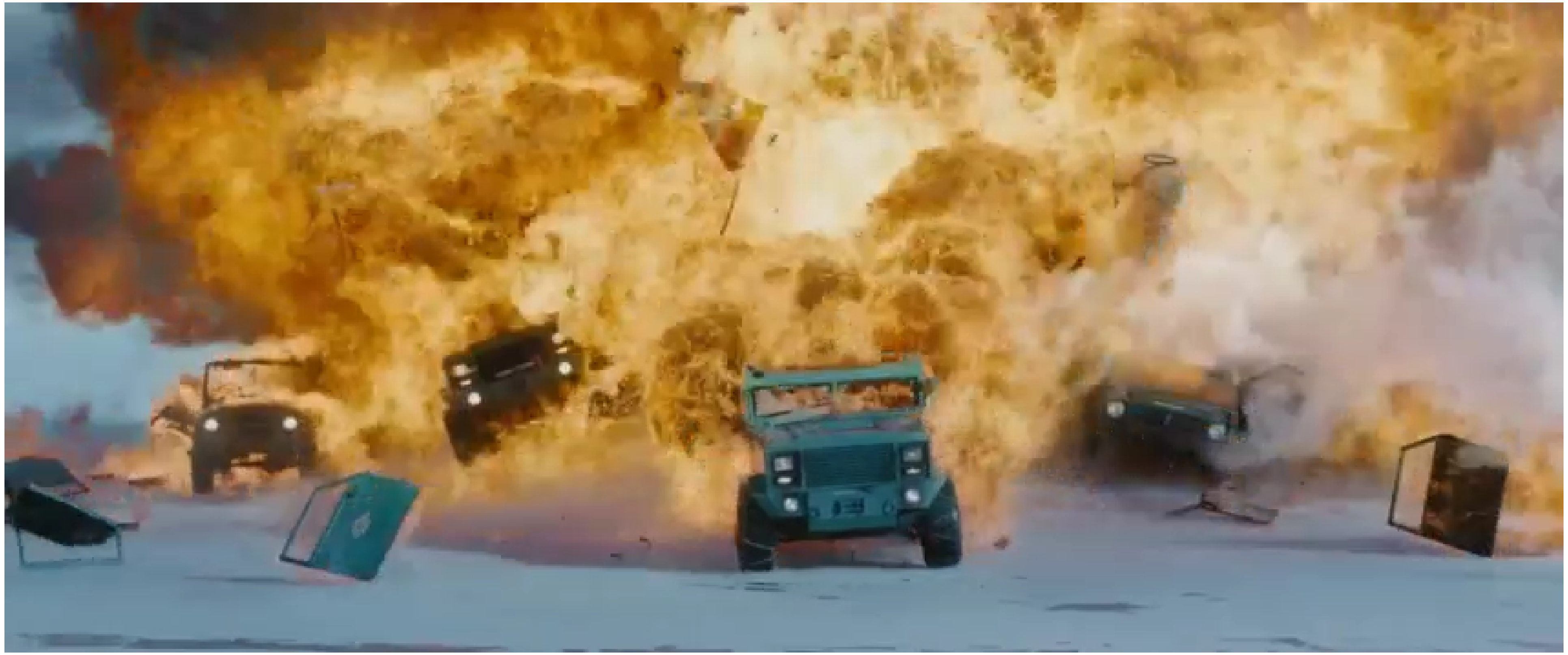


# Rolling update

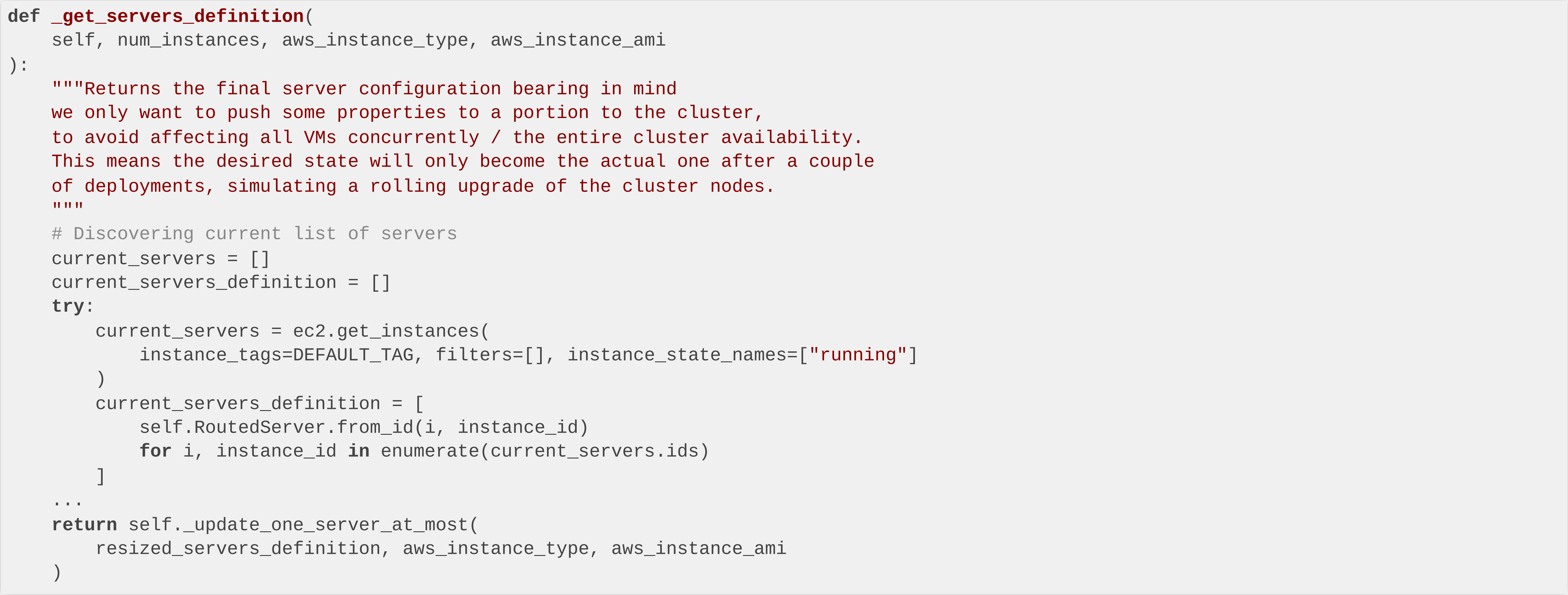
When the *rolling update* (`ECS`) deployment type is used for your service, when a new service deployment is started the Amazon ECS service scheduler replaces the currently running tasks with new tasks. The number of tasks that Amazon ECS adds or removes from the service during a rolling update is controlled by the deployment configuration. The deployment configuration consists of the `minimumHealthyPercent` and `maximumPercent` values which are defined when the service is created, but can also be updated on an existing service.

*Incremental ECS tasks replacement almost out of the box…*

## But node changes (ami, sizing) in absence of ASGs?



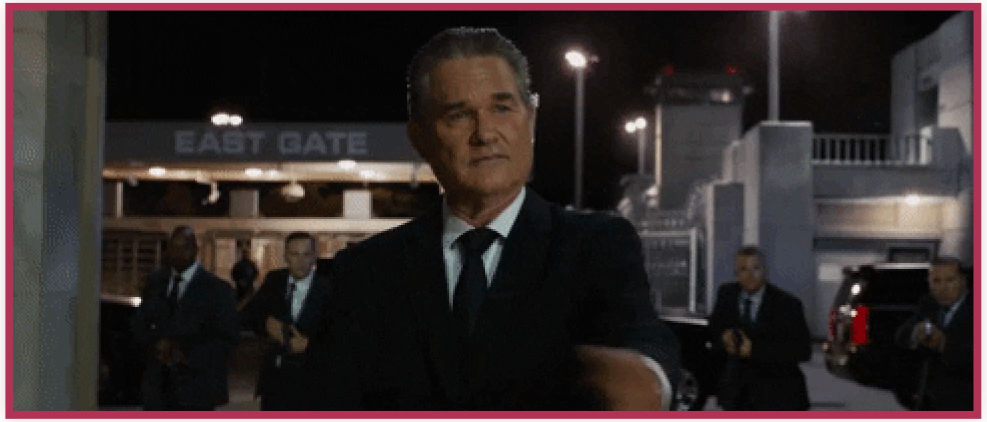
*pulumi (and terraform) pushes desired state to all managed resources*



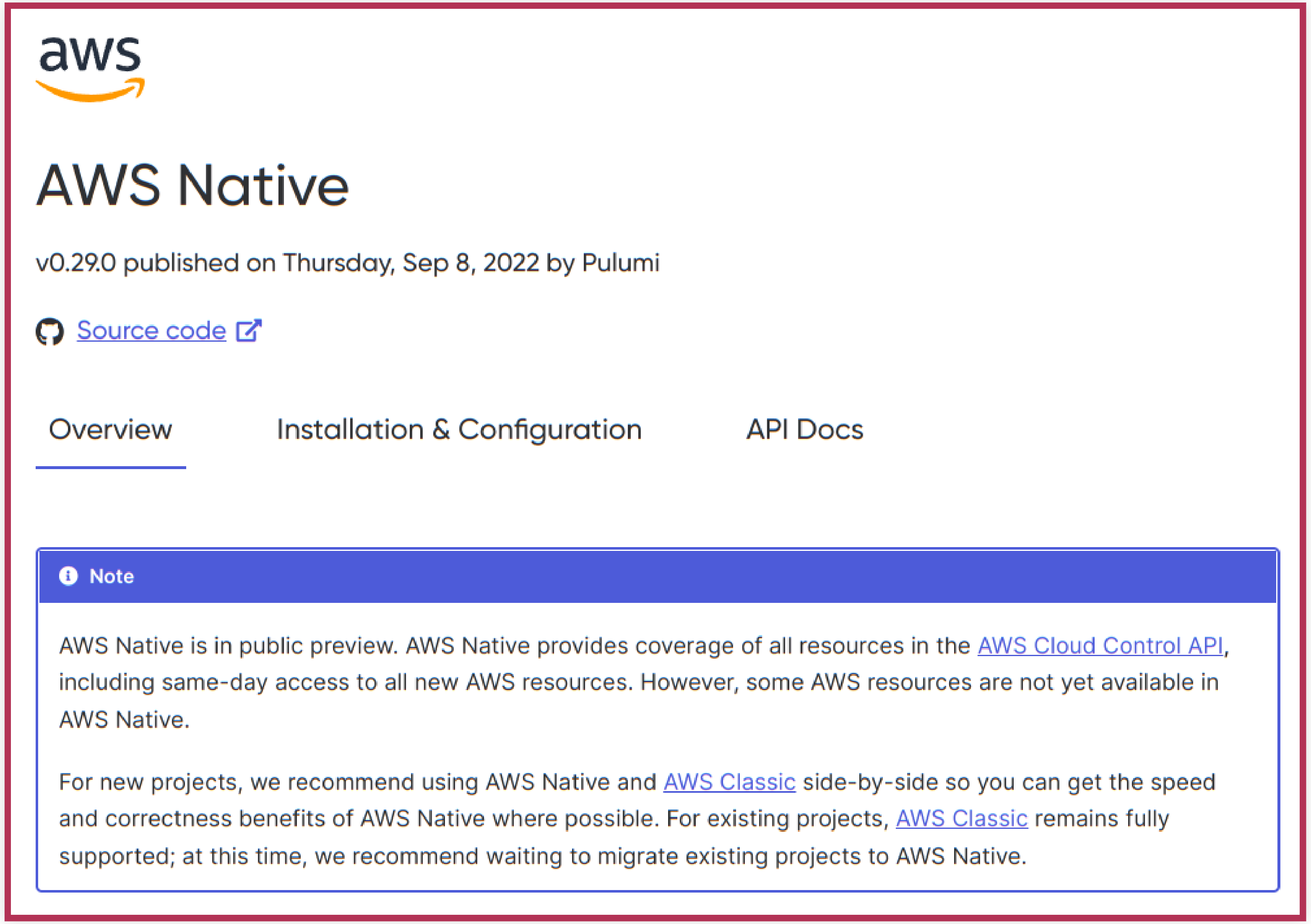
```python
def _get_servers_definition(
    self, num_instances, aws_instance_type, aws_instance_ami
):
    """Returns the final server configuration bearing in mind
    we only want to push some properties to a portion to the cluster,
    to avoid affecting all VMs concurrently / the entire cluster availability.
    This means the desired state will only become the actual one after a couple
    of deployments, simulating a rolling upgrade of the cluster nodes.
    """
    # Discovering current list of servers
    current_servers = []
    current_servers_definition = []
    try:
        current_servers = ec2.get_instances(
            instance_tags=DEFAULT_TAG, filters=[], instance_state_names=["running"]
        )
        current_servers_definition = [
            self.RoutedServer.from_id(i, instance_id)
            for i, instance_id in enumerate(current_servers.ids)
        ]
    ...
    return self._update_one_server_at_most(
        resized_servers_definition, aws_instance_type, aws_instance_ami
    )
```

# ISSUE 2

```
$ ENV=prd-na make deploy
```



# Many ECS tasks and managed resources in a single stack…



# AWS Native

v0.29.0 published on Thursday, Sep 8, 2022 by Pulumi

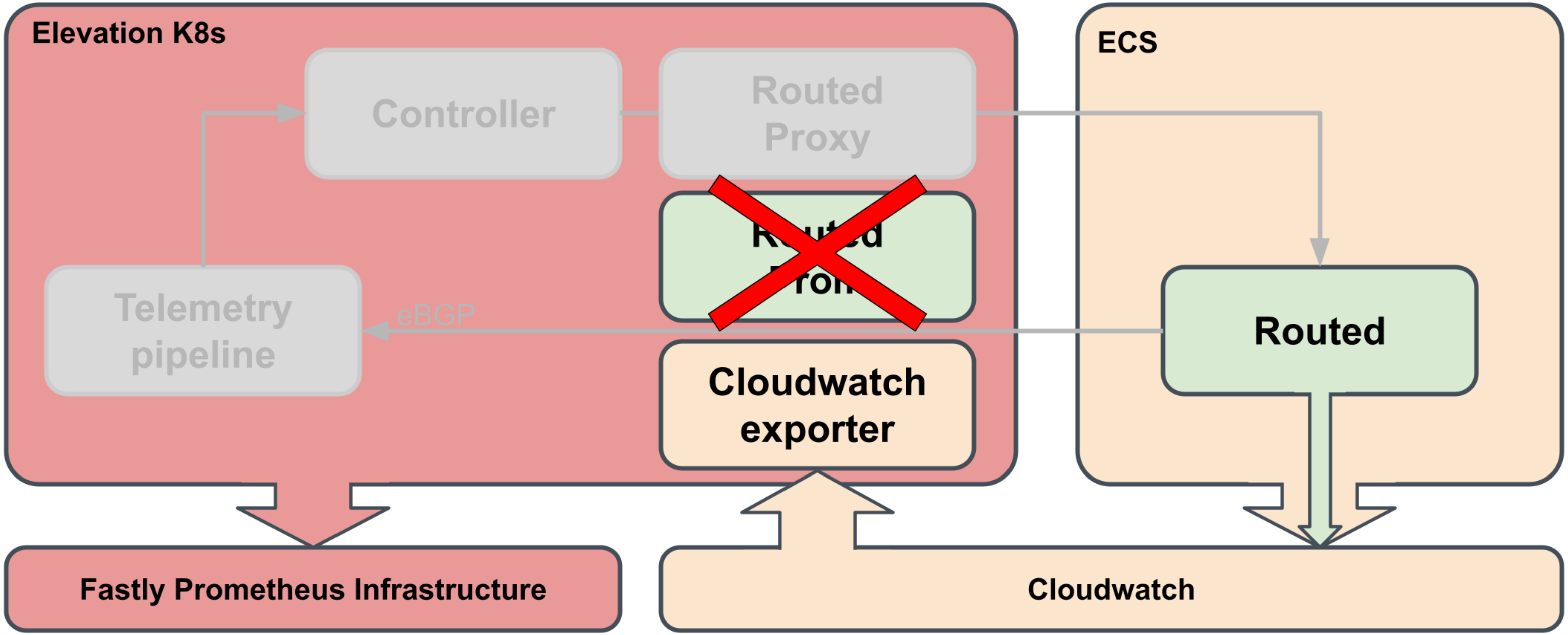
Source code

**Note**

AWS Native is in public preview. AWS Native provides coverage of all resources in the AWS Cloud Control API, including same-day access to all new AWS resources. However, some AWS resources are not yet available in AWS Native.

For new projects, we recommend using AWS Native and AWS Classic side-by-side so you can get the speed and correctness benefits of AWS Native where possible. For existing projects, AWS Classic remains fully supported; at this time, we recommend waiting to migrate existing projects to AWS Native.

# ISSUE 3: APP METRICS TO PROMETHEUS VIA CLOUDWATCH



But:

- Cloudwatch **specific instrumentation or sidecar** for each task
- Scraping custom cloudwatch metrics per task is **expensive**
- **scrape time** was a problem

*We still use `cloudwatch_exporter`, but just to import* ***AWS/ECS infra specific metrics***

# ISSUE 4: SCALABILITY

- Cluster horizontal scalability is complicated given the requirements
- Huge EC2 machines help with vertical scalability
- But less options to pick the right CPU/mem ratio

# ISSUE 5: PERFORMANCE

*we were initially not promoting CPU limits per service, thinking that we would maximize node CPU usage*

That was actually hurting performance

And CPU sets could be even better than shares

# ESSENTIAL CONTAINERS (DAEMON SETS)

*Don’t do this at home…*

Improved node recovery times, but only makes sense:

- we want one task per service and node
- cluster is dedicated for this application

# FUTURE

# CURRENT SETUP IMPROVEMENTS

- Migration to ARM for cost reduction
- Migration from Amazon ECS-optimized AMI to Bottlerocket

# BUT WE STILL WANT K8S

# Unified developer/ops experience

Unified developer/ops experience

Workloads consolidation / costs

Unified developer/ops experience

Workloads consolidation / costs

Less dedicated infra to manage

# HOW?

- Custom NAT gateway
- dynamic BGP peers discovery
- IPv6 over IPv4
- BGP BMP…

# CLOSURE

# SUMMARY

# SUMMARY

Fastly needs **network and systems automation**

# SUMMARY

Fastly needs **network and systems automation**

There may be very specific cases where **custom infra may be required**

# SUMMARY

Fastly needs **network and systems automation**

There may be very specific cases where **custom infra may be required**

**Using Pulumi to define cloud IaC and drive ECS deployments was good**

# QUESTIONS?

# Thank You!

fastly

fastly®