How To Easily Deploy And Manage Your Microservices With Traefik And Maesh

London Devops #50 - February 25 2020

Whoami

Manuel Zapf

- Head of Product Open Source @ Containous
- Maintainer on Traefik
- @mZapfDE
- SantoDE

Containous

https://containo.us

- We Believe in Open Source
- We Deliver Traefik, Traefik Enterprise Edition, Maesh
- Commercial Support
- 30 people distributed, 90% tech

THE EVOLUTION OF

SOFTWARE ARCHITECTURE

1990's

SPAGHETTI-ORIENTED
ARCHITECTURE
(aka Copy & Paste)

2000's

LASAGNA-ORIENTED
ARCHITECTURE
(aka Layered Monolith)

2010's

RAVIOLI-ORIENTED
ARCHITECTURE
(aka Microservices)

WHAT'S NEXT?

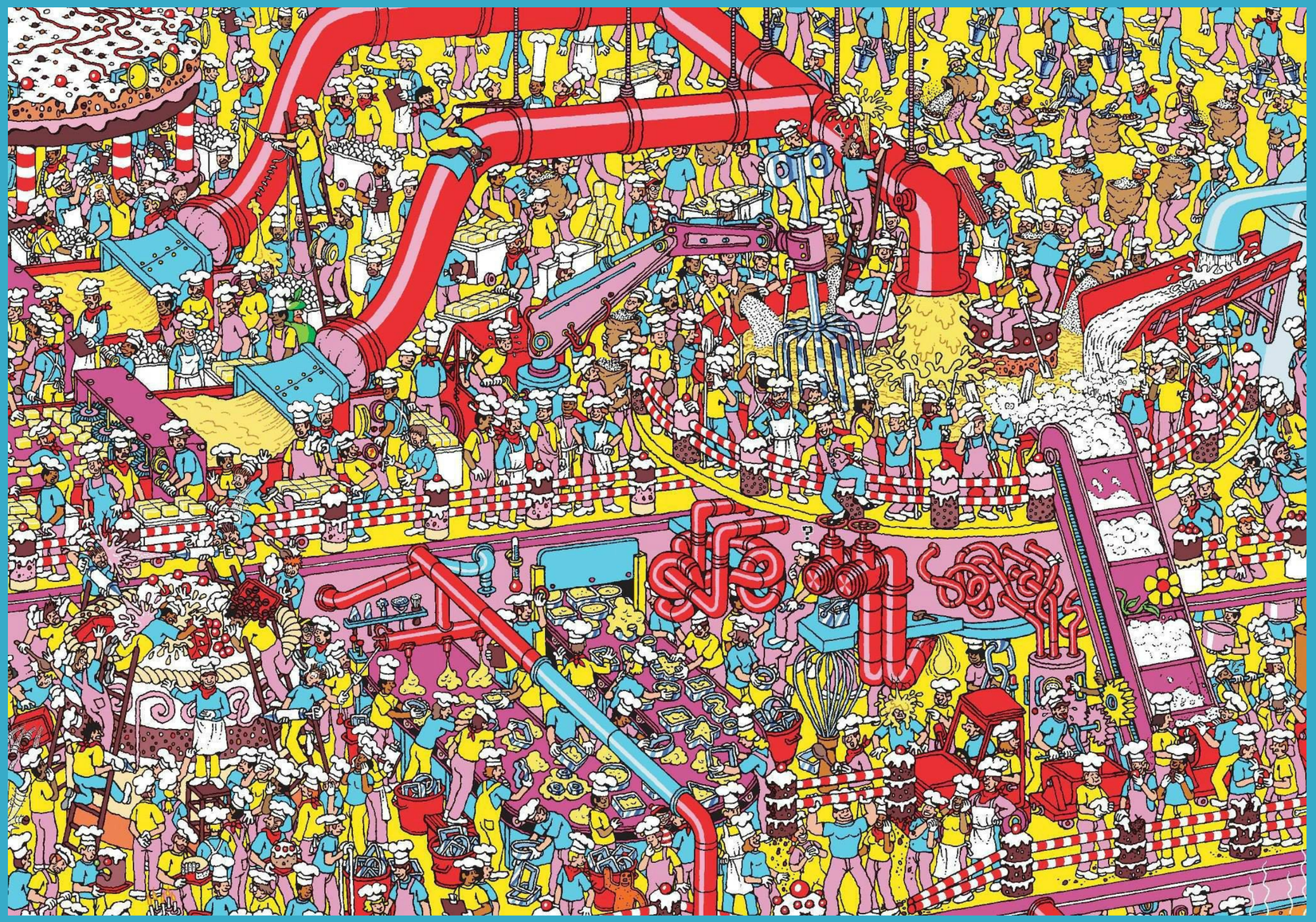
PROBABLY PIZZA-ORIENTED ARCHITECTURE

By @benorama

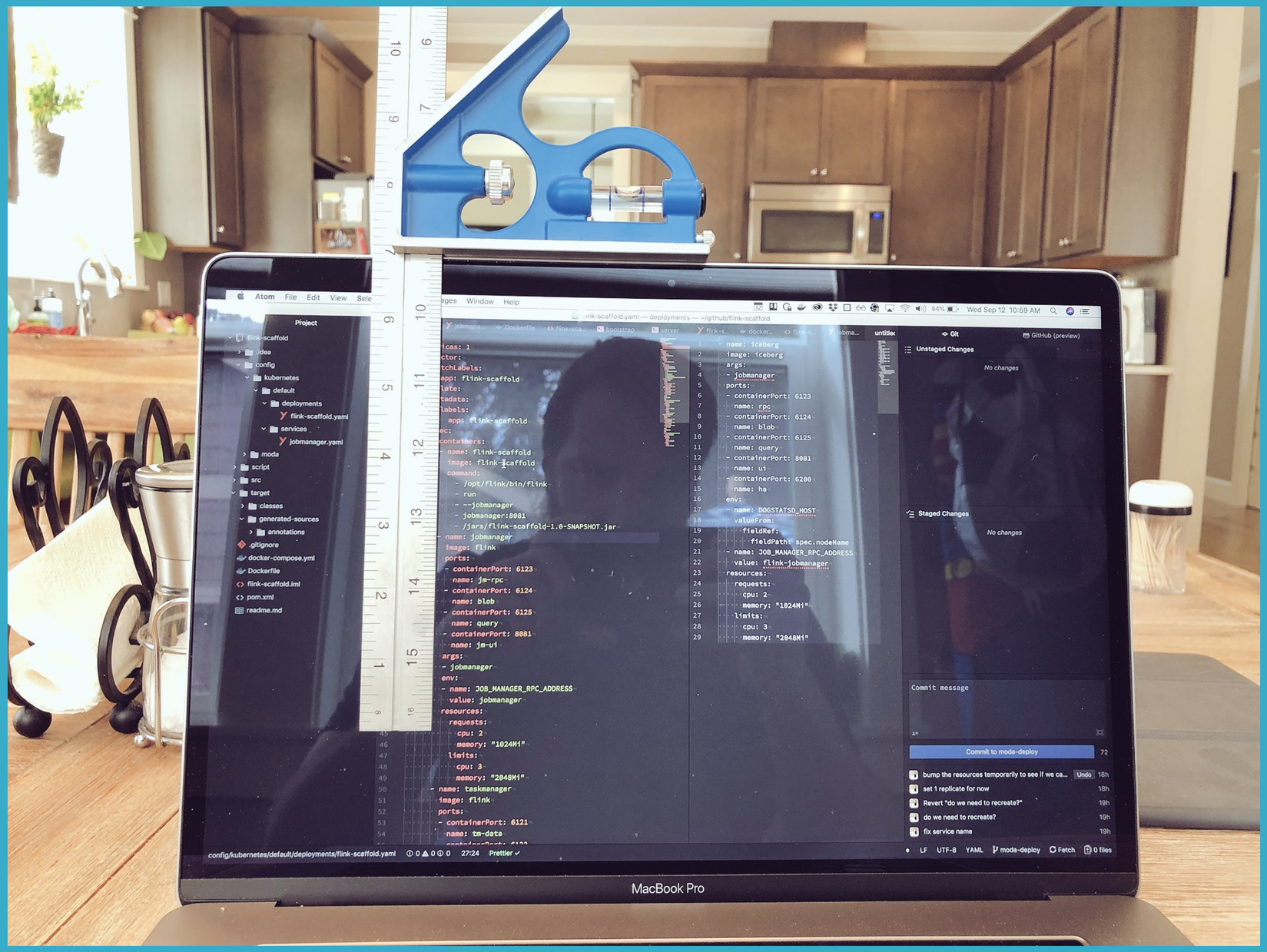
The Premise Of Microservices…

···And What Happens

Where’s My Service?



Tools Of The Trade

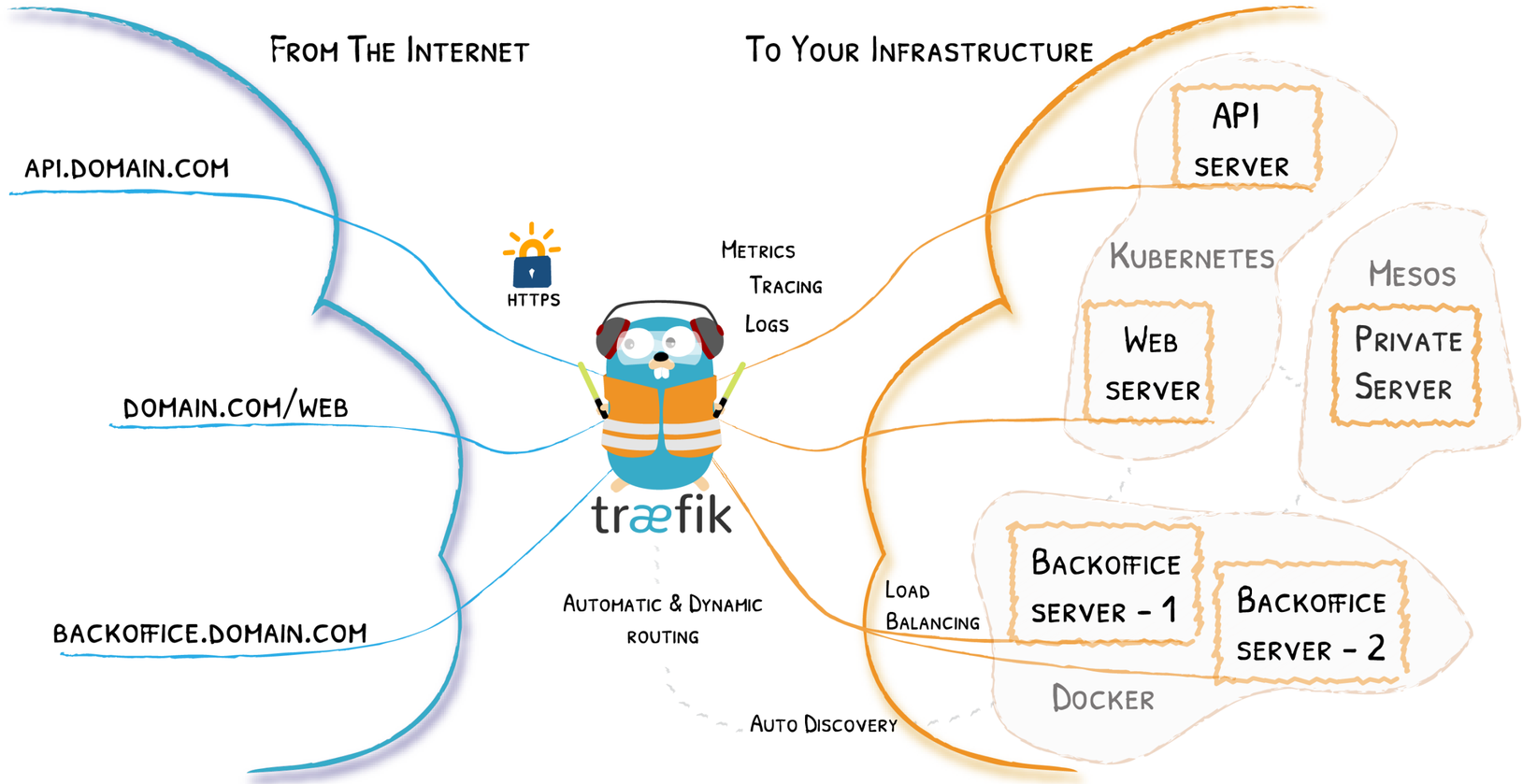


Source: https://twitter.com/Caged/status/1039937162769096704

What If I Told You?

That You Don’t Have to Write This Configuration File…?

Here Comes Traefik!



Traefik Project

27,000+
stars on GitHub

1.4 B+
downloads

450+
contributors

TOP 15
golang project

100K +
living
instances

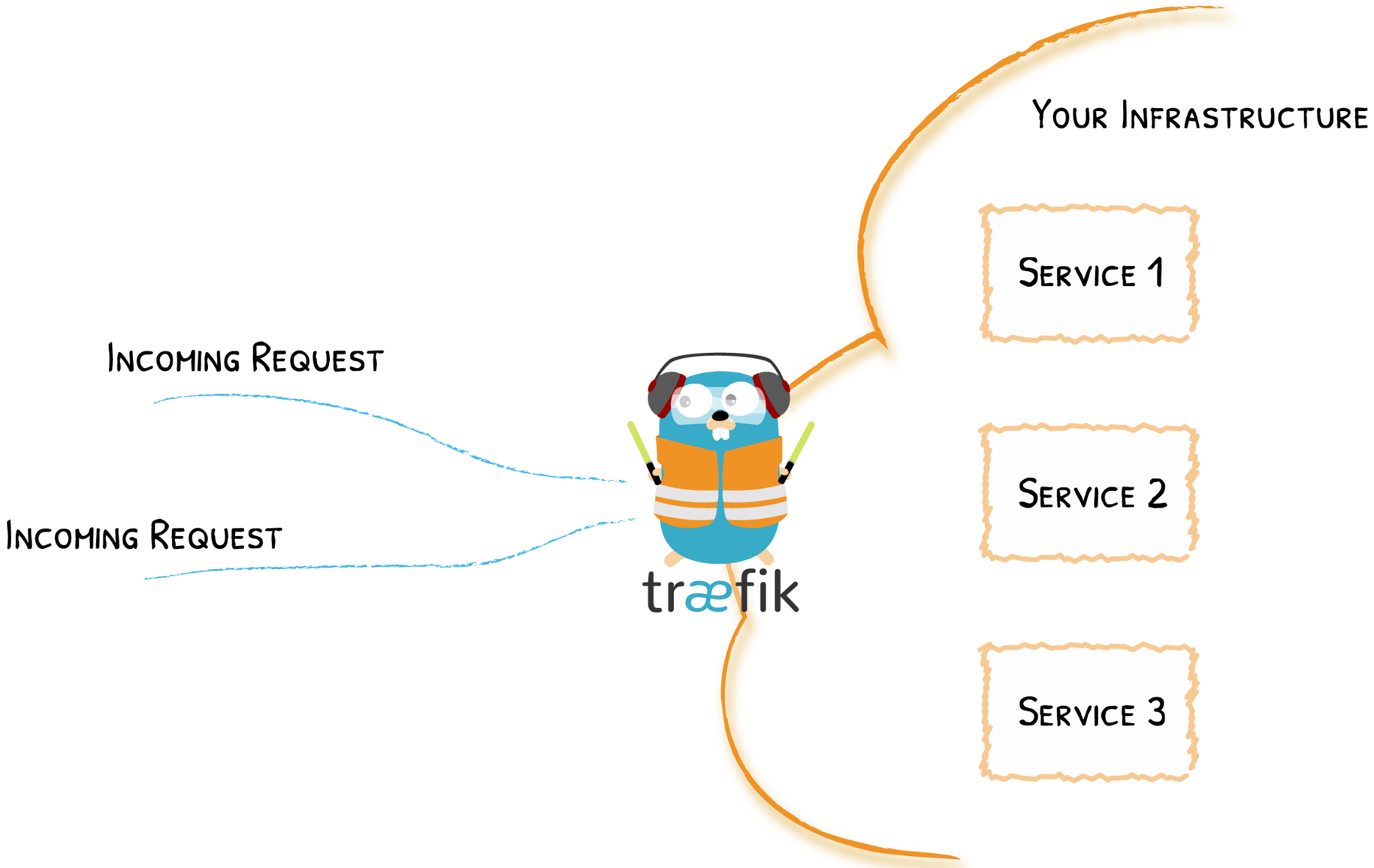
Traefik 2.0 Quick Overview

- Revamped Documentation
- Clarified Concepts
- Expressive Routing Rule Syntax
- Middlewares
- TCP Support
- Canary / Mirroring
- And so Much More...

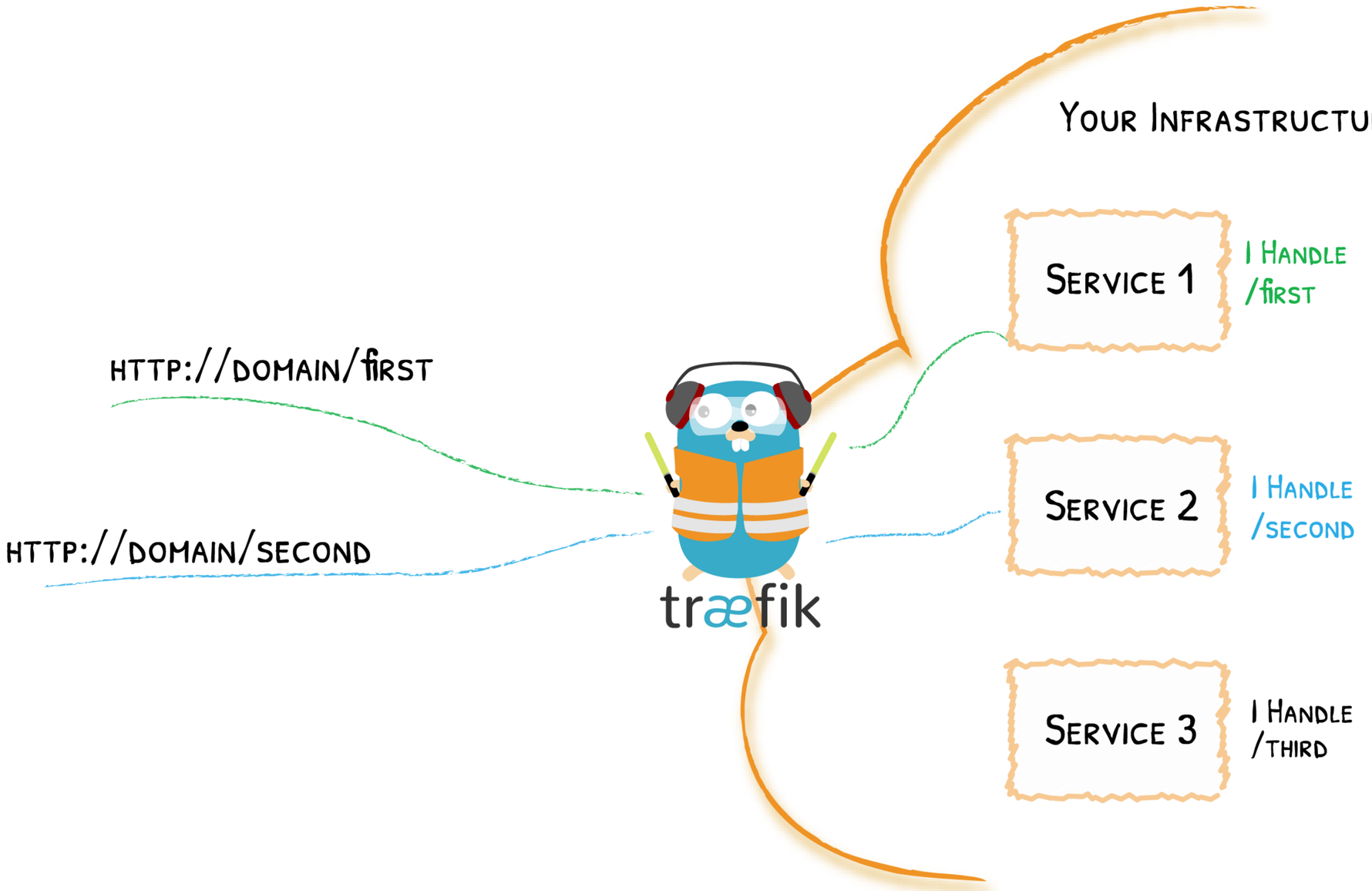
Learn more on the blog post

Traefik (V2.0) Core Concepts

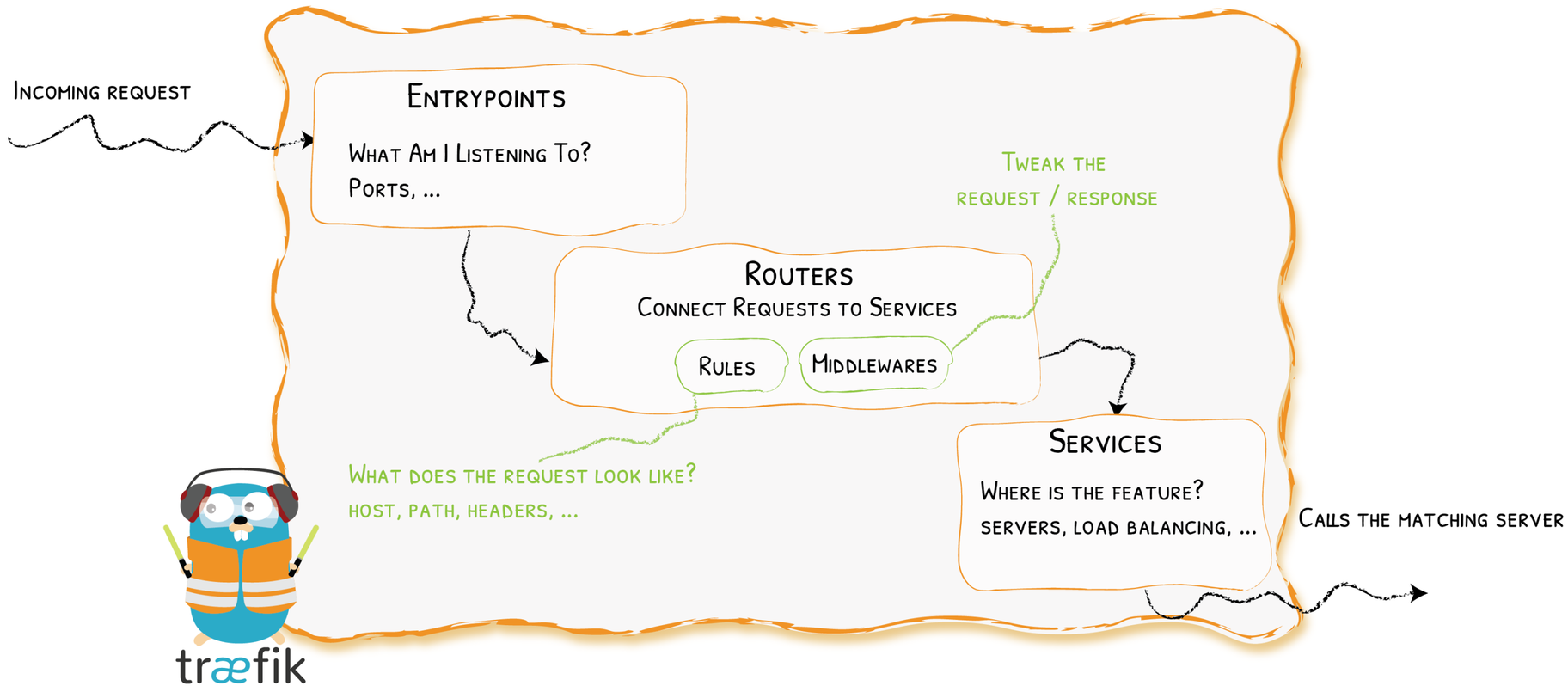
Traefik Is An Edge Router



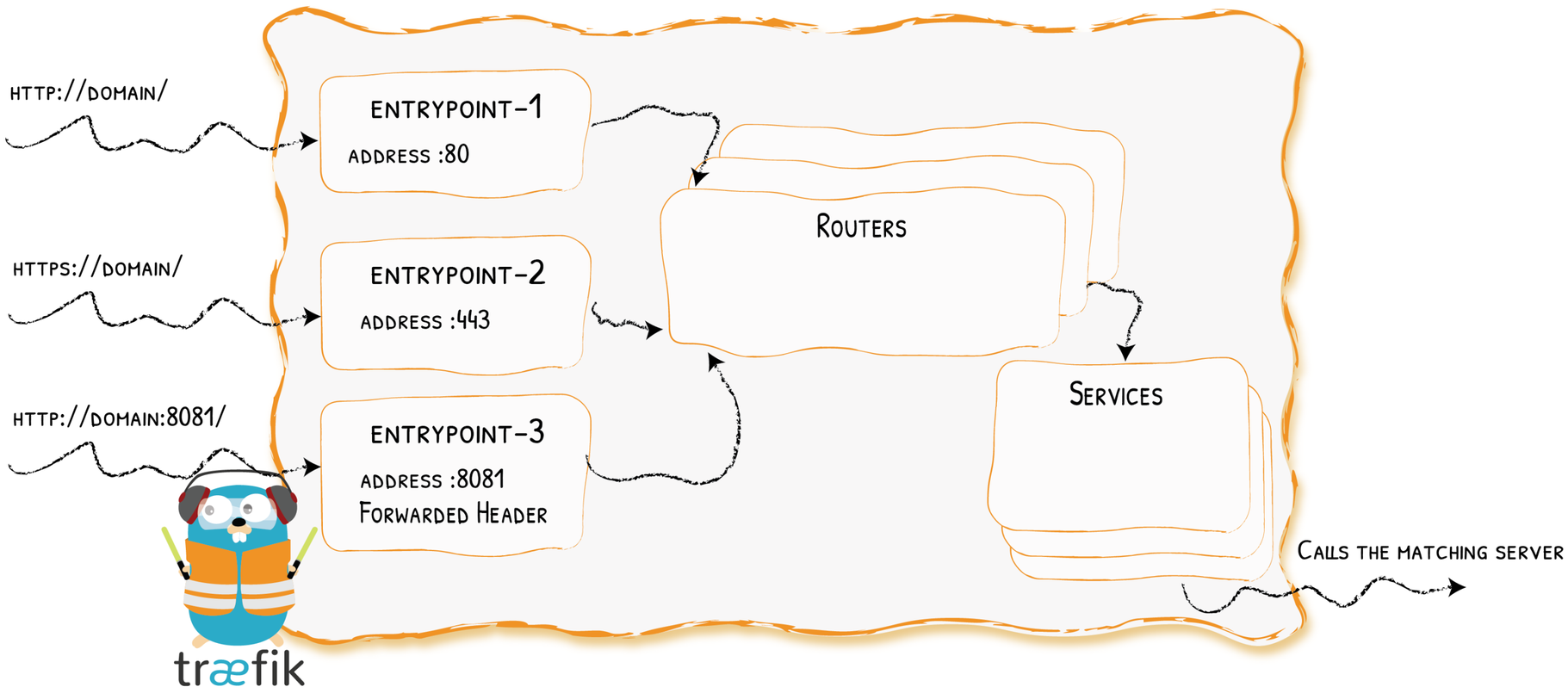
Dynamically Discovers Services



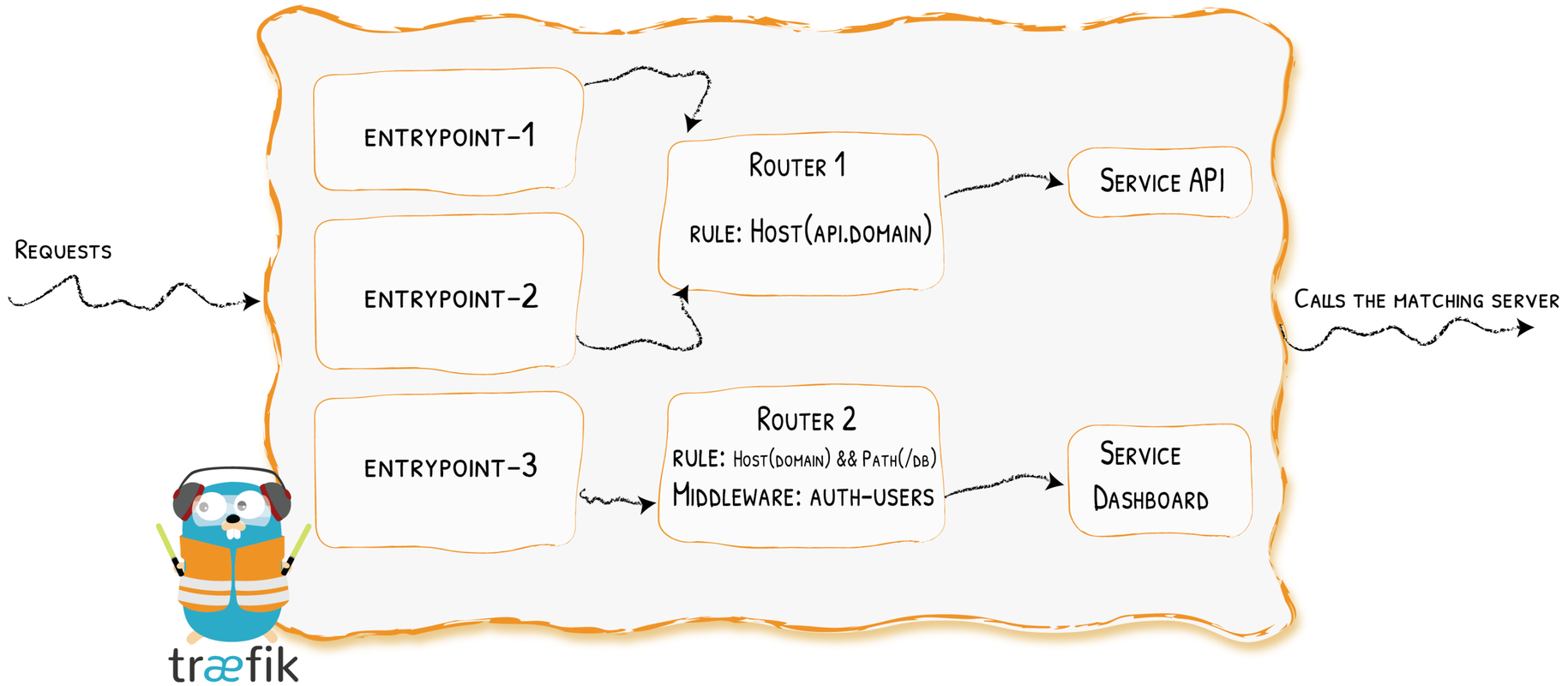
Architecture (V2.0) At A Glance



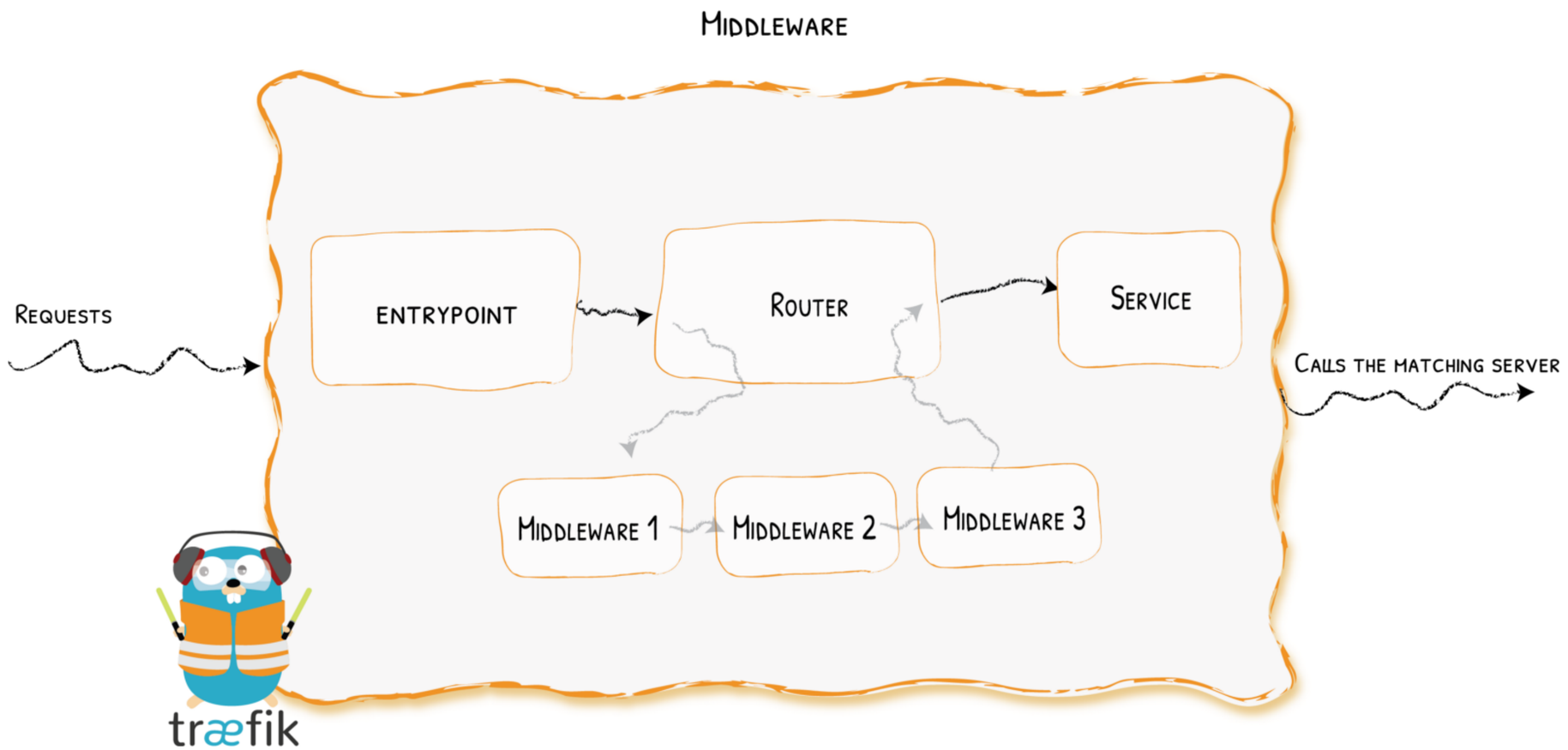
Entrypoints



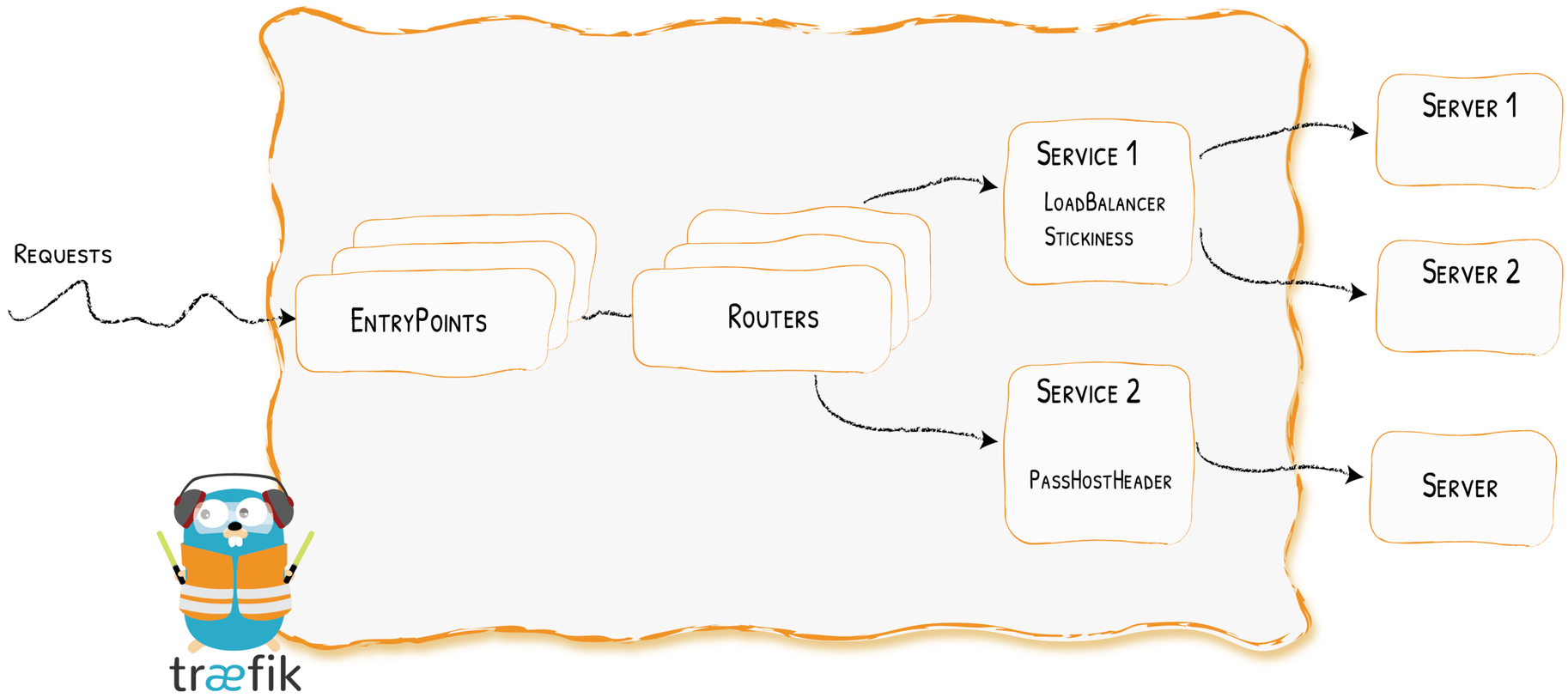
Routers


ENTRYPOINT-1
ROUTER 1
RULE: HOST(API.DOMAIN)
SERVICE API
REQUESTS
ENTRYPOINT-2
CALLS THE MATCHING SERVER
ENTRYPOINT-3
ROUTER 2
RULE: HOST(DOMAIN) && PATH(/DB)
MIDDLEWARE: AUTH-USERS
SERVICE
DASHBOARD
træfik

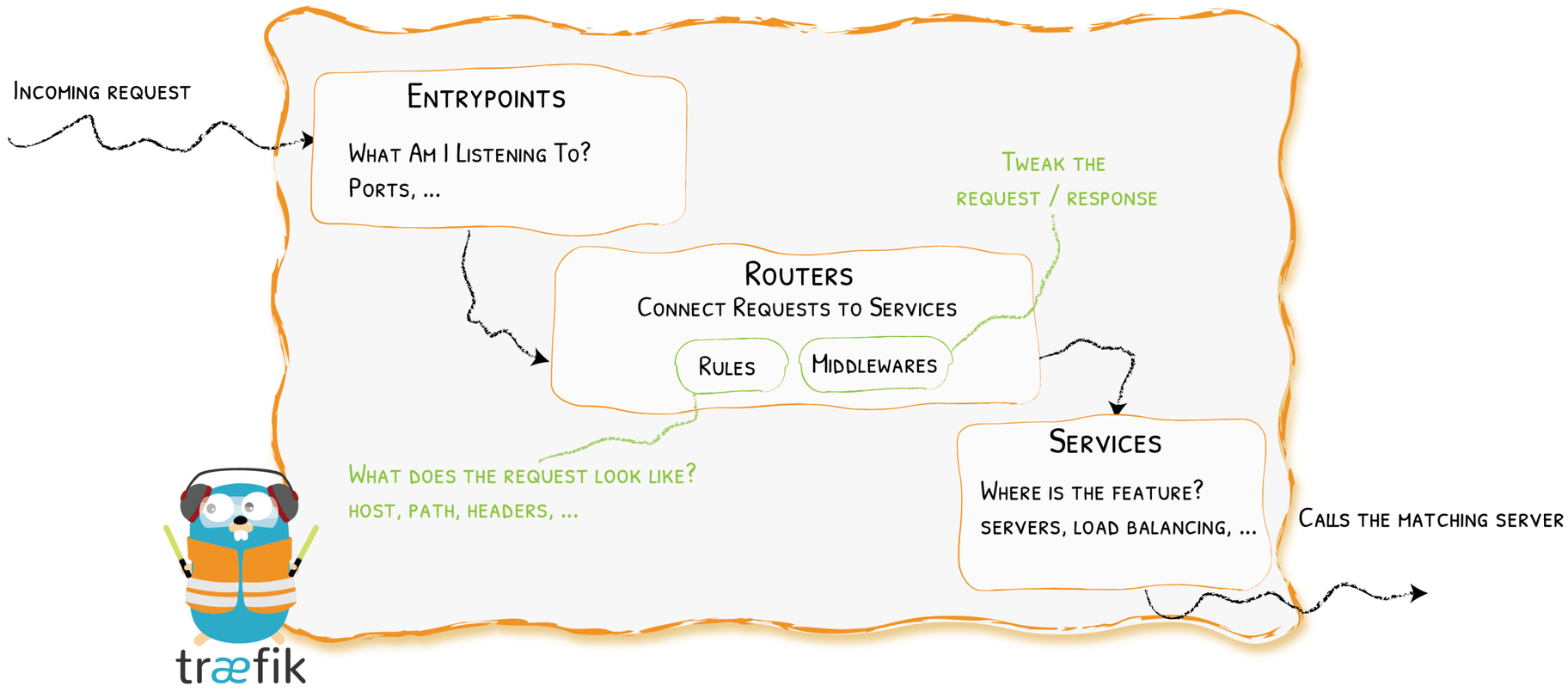
Middlewares



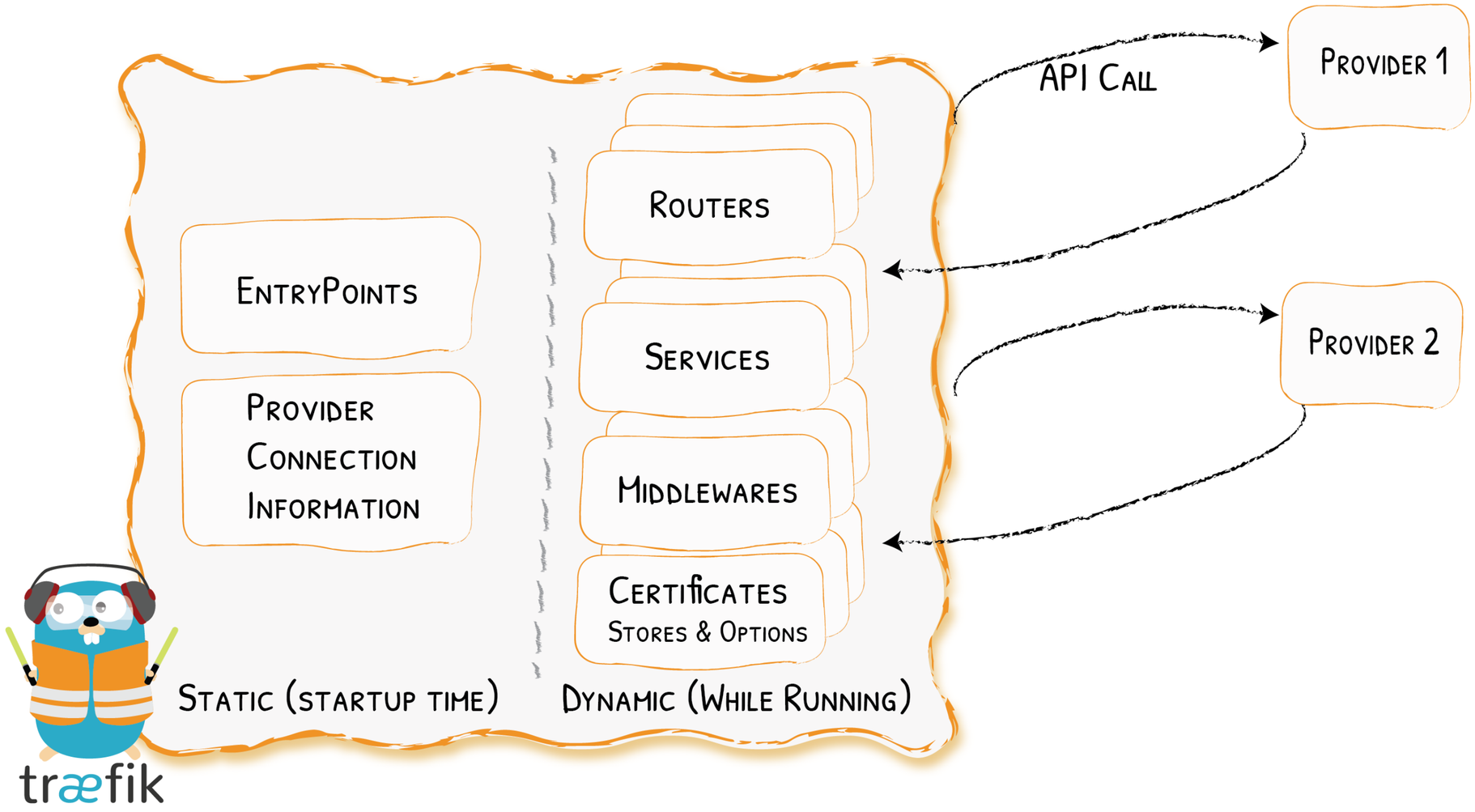
Services


Requests
EntryPoints
Routers
Service 1
LoadBalancer
Stickiness
Service 2
PassHostHeader
Server 1
Server 2
Server
træfik

Architecture (Again) At A Glance

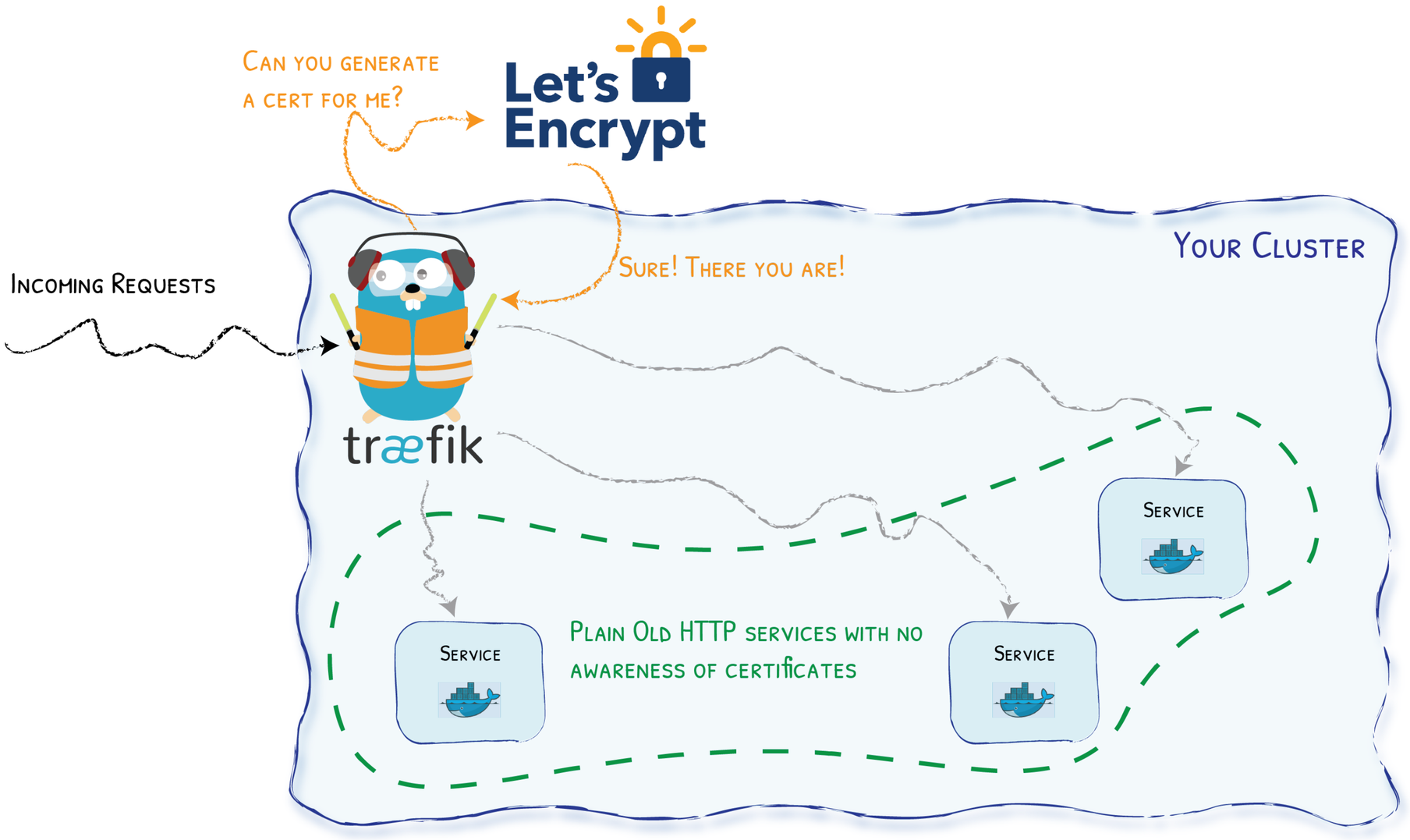


Static & Dynamic Configuration



Traefik And Let’s Encrypt

HTTPS & Let's Encrypt



Traefik With ⎈

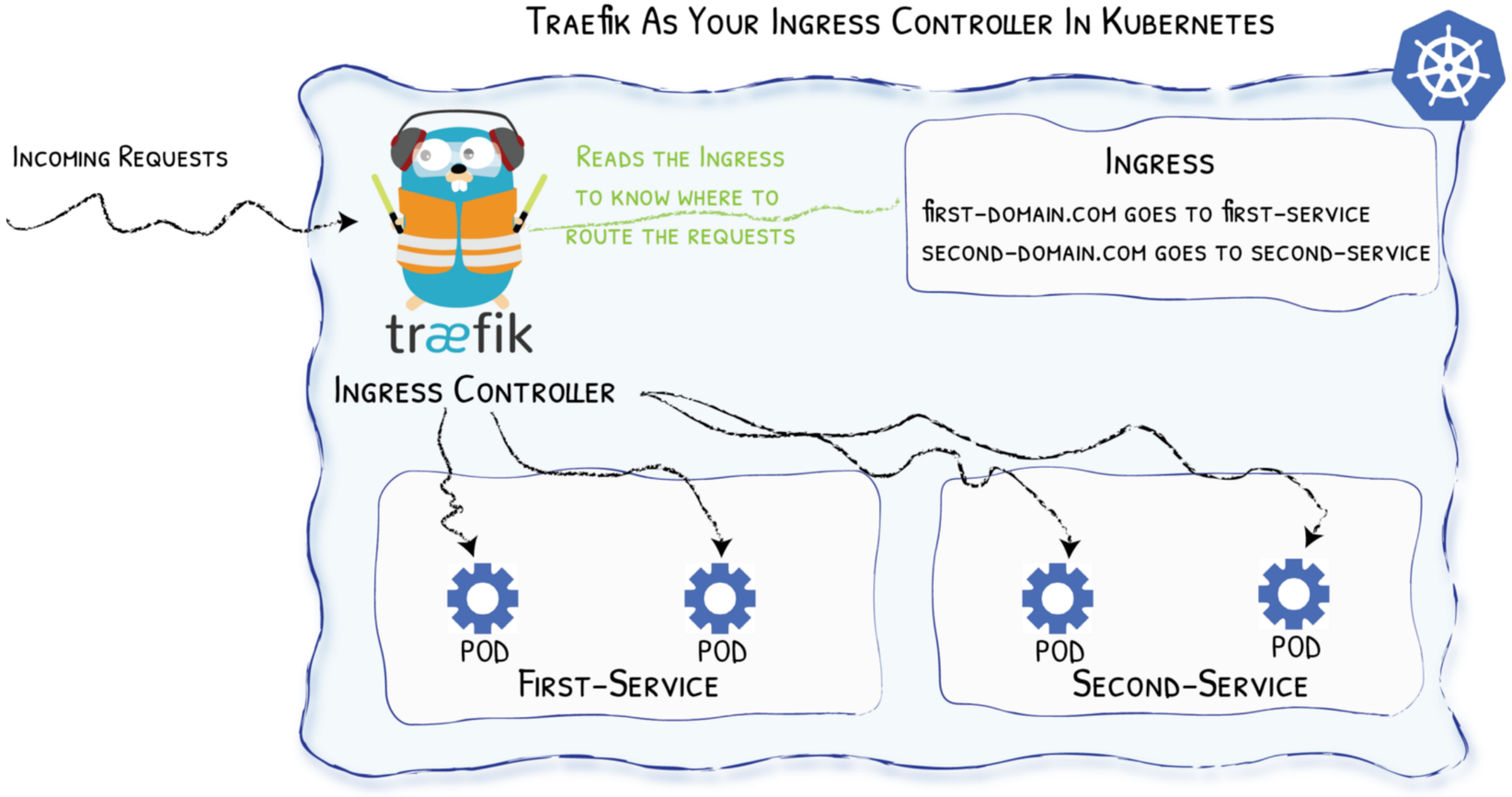


Diagram from https://medium.com/@geraldcroes

Ingress Example With ☸

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: corporate-webapp
  annotations:
    kubernetes.io/ingress.class: 'traefik'
spec:
  rules:
  - host: localhost
    http:
      paths:
      - backend:
          serviceName: corporate-webapp
          servicePort: 80
```

☸ CRD - Custom Resources Definition

```
# File "webapp.yaml"
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRoute
metadata:
  name: simpleingressroute
spec:
  entryPoints:
    - web
  routes:
  - match: Host(`localhost`) && PathPrefix(`/whoami`)
    kind: Rule
    services:
    - name: webapp
      port: 80
```

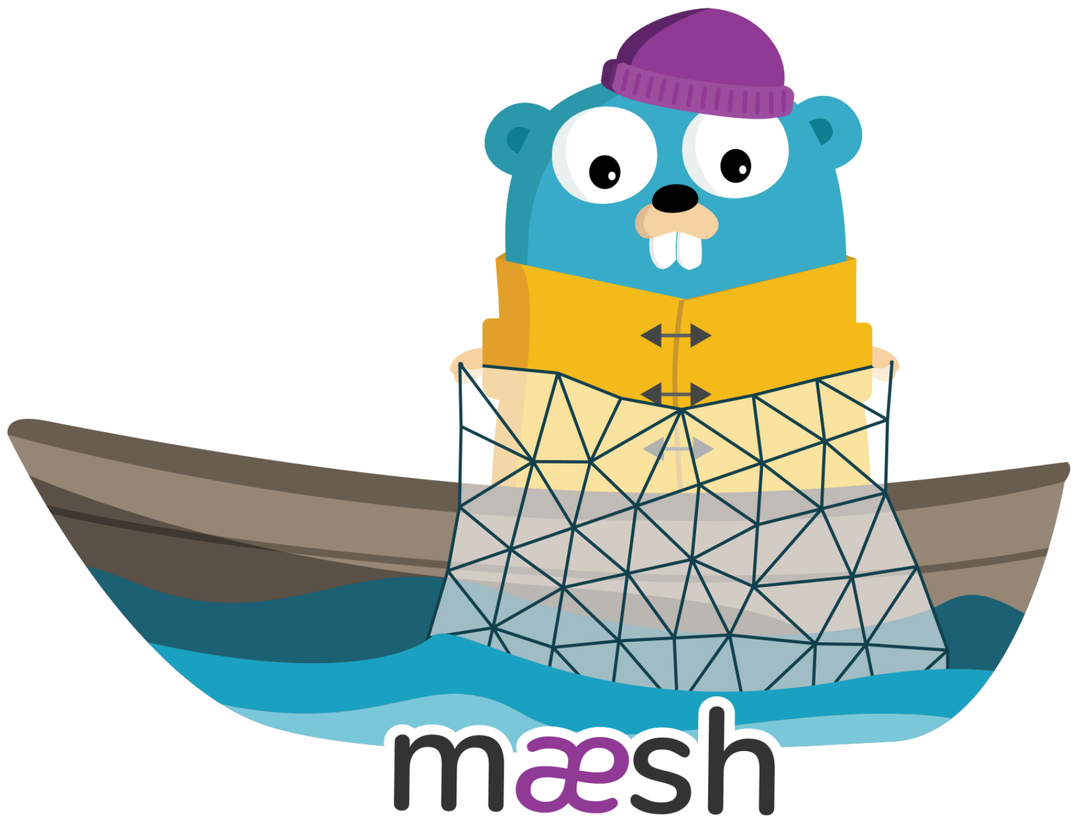
```
$ kubectl apply -f webapp.yaml
$ kubectl get ingressroute
```

☸ & TCP (With CRD)

```
apiVersion: traefik.containo.us/v1alpha1
kind: IngressRouteTCP
metadata:
  name: ingressroutetcpmongo.crd
spec:
  entryPoints:
    - mongotcp
  routes:
  - match: HostSNI(`mongo-prod`)
    services:
    - name: mongo-prod
      port: 27017
```

East / West Traefik

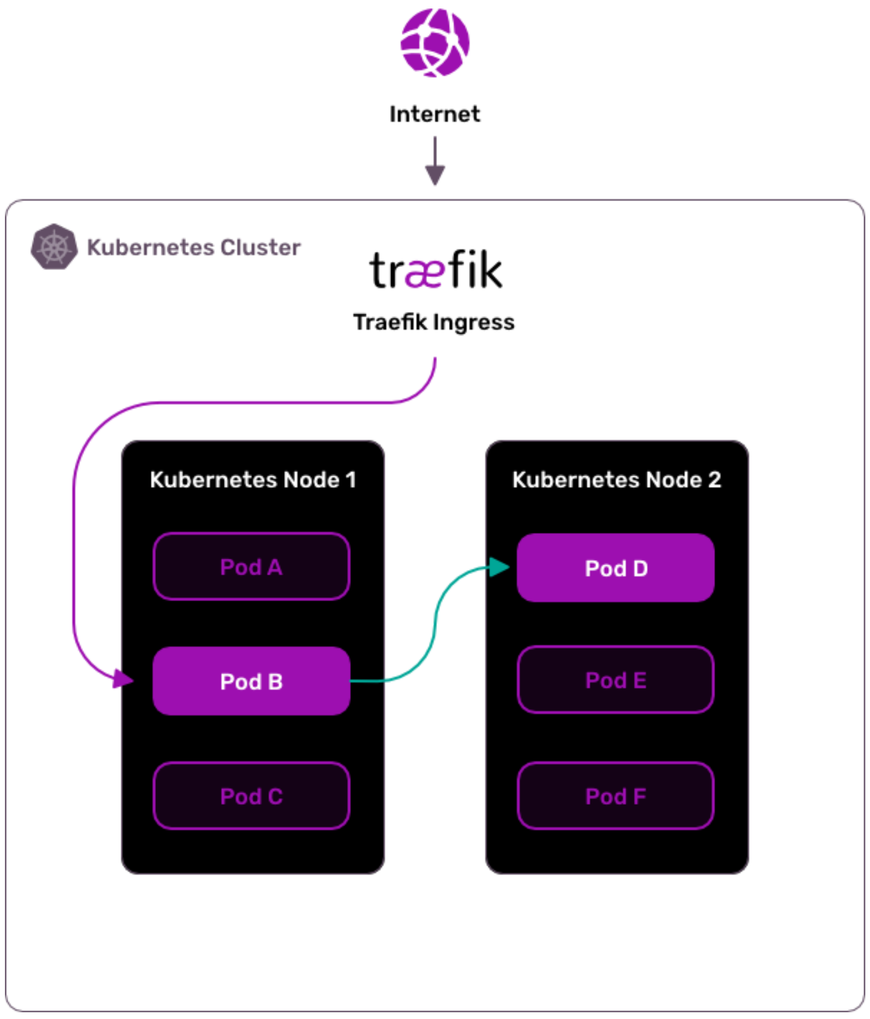
Say Hello To Maesh

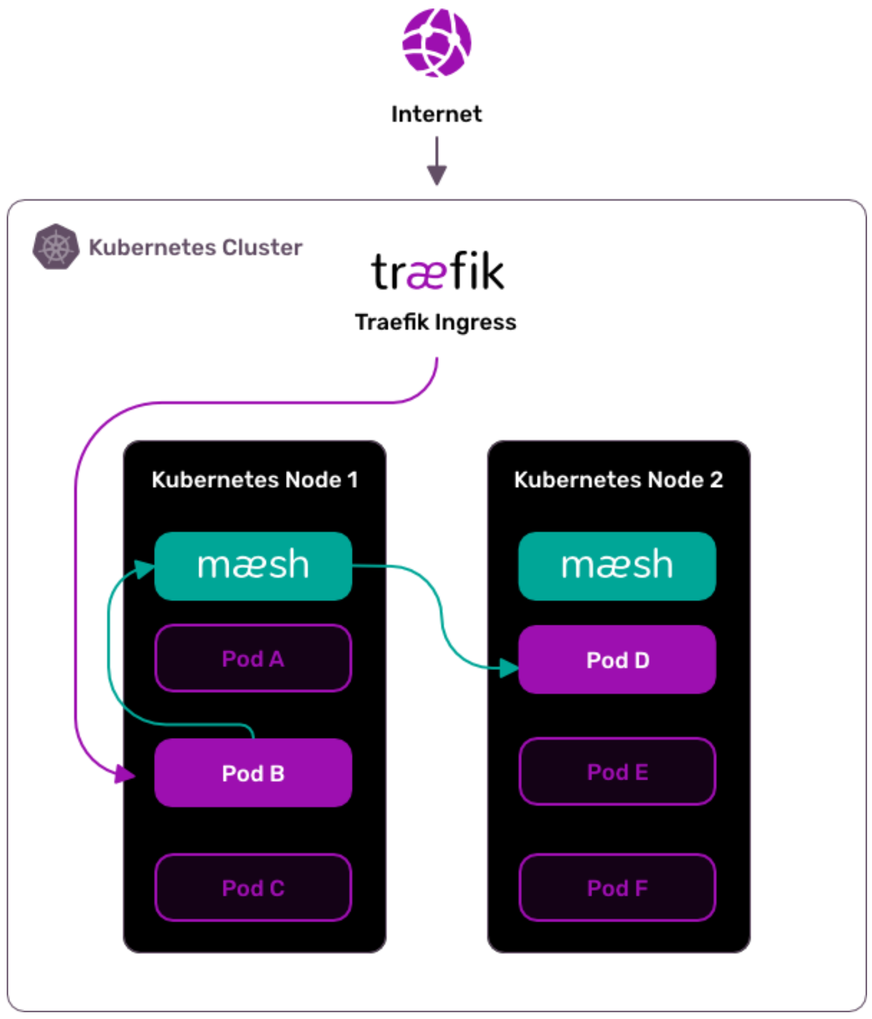


What Is Maesh?

Maesh is a lightweight, easy to configure, and non-invasive service mesh that allows visibility and management of the traffic flows inside any Kubernetes cluster.

Maesh Architecture





More On Maesh

- Built on top of Traefik,
- SMI (Service Mesh Interface specification) compliant,
- Opt-in by default.

Maesh Website

Show Me The Code!

- Install Maesh (Helm Chart):

```
helm repo add maesh https://containous.github.io/maesh/charts
helm repo update
helm install --name=maesh --namespace=maesh maesh/maesh --values=./maesh/values.yaml
```

- Deploy Applications:

```
kubectl apply -f apps/0-namespace.yaml
kubectl apply -f apps/1-svc-accounts.yaml
kubectl apply -f apps/2-apps-client.yaml
kubectl apply -f apps/3-apps-servers.yaml
kubectl apply -f apps/4-ingressroutes.yaml
```

- Deploy SMI Objects to allow traffic in the mesh:

```
kubectl apply -f apps/5-smi-http-route-groups.yaml
kubectl apply -f apps/6-smi-traffic-targets.yaml
```

A Closer Look To SMI Objects

```yaml
apiVersion: specs.smi-spec.io/v1alpha1
kind: HTTPRouteGroup
metadata:
  name: app-routes
  namespace: apps
matches:
- name: all
  pathRegex: "/"
  methods: ["*"]
---
apiVersion: access.smi-spec.io/v1alpha1
kind: TrafficTarget
metadata:
  name: client-apps
  namespace: apps
destination:
  kind: ServiceAccount
  name: apps-server
  namespace: apps
specs:
- kind: HTTPRouteGroup
  name: app-routes
  matches:
  - all
sources:
- kind: ServiceAccount
  name: apps-client
  namespace: apps
```

That’s All Folks!

We Have Stickers!

We Are Hiring!

CONTAINOUS

```
docker run -it containous/jobs
```

Thank You!

- @mZapfDE
- SantoDE

- Slides (HTML): https://containous.github.io/slides/london-devops-meetup-2020
- Slides (PDF): https://containous.github.io/slides/london-devops-meetup-2020/slides.pdf
- Source on : https://github.com/containous/slides/tree/london-devops-meetup-2020