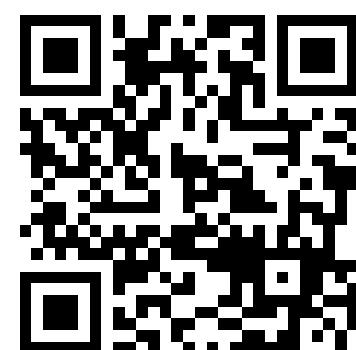


Le Seigneur Des Conteneurs

Un atelier de migration vers Kubernetes et Traefik



<https://containous.github.io/slides/devoxx-fr-2019>

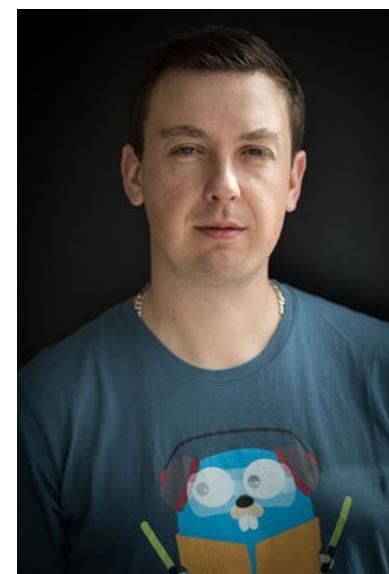
How To Use These Slides?

- **Browse the slides:** Use the arrows
 - Change chapter: Left/Right arrows
 - Next or previous slide: Top and bottom arrows
- **Overview of the slides:** keyboard's shortcut "o"
- **Speaker mode (and notes):** keyboard's shortcut "s"

Whoami 1/2

Nicolas Mengin

- Developer, Part Time DevOps @ Containous
- Maintainer of [Træfik](#)
- [!\[\]\(746d018fdf6ab02bf5fb7681133e8b29_img.jpg\) @nicomengin](#)
- [!\[\]\(5daa6eee1904cb6b9d765700250de764_img.jpg\) nmengin](#)



Whoami 2/2

- Damien DUPORTAL:
 - Træfik's Developer  Advocate @ Containous
-  @DamienDuportal
-  dduportal



Containous

<https://containo.us>

- We Believe in Open Source
- We Deliver Traefik
- Commercial Support for Traefik
- 20 people, 90% tech



Why Traefik?



Why, Mr Anderson?

Evolution Of Software Design



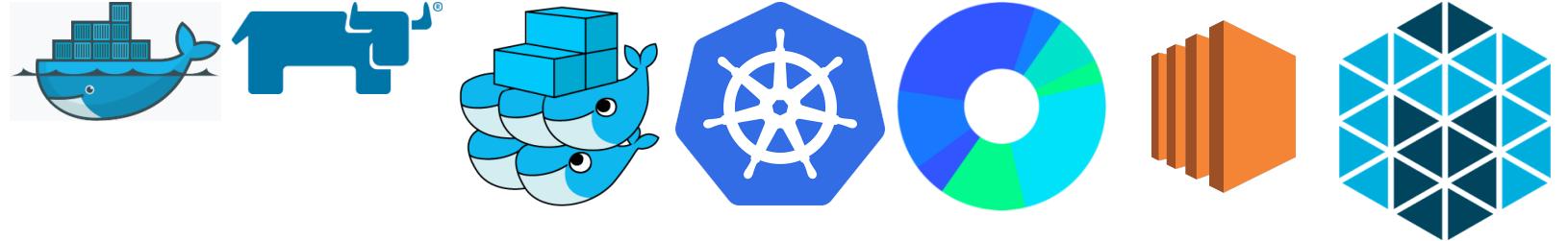
The Premise Of Microservices...



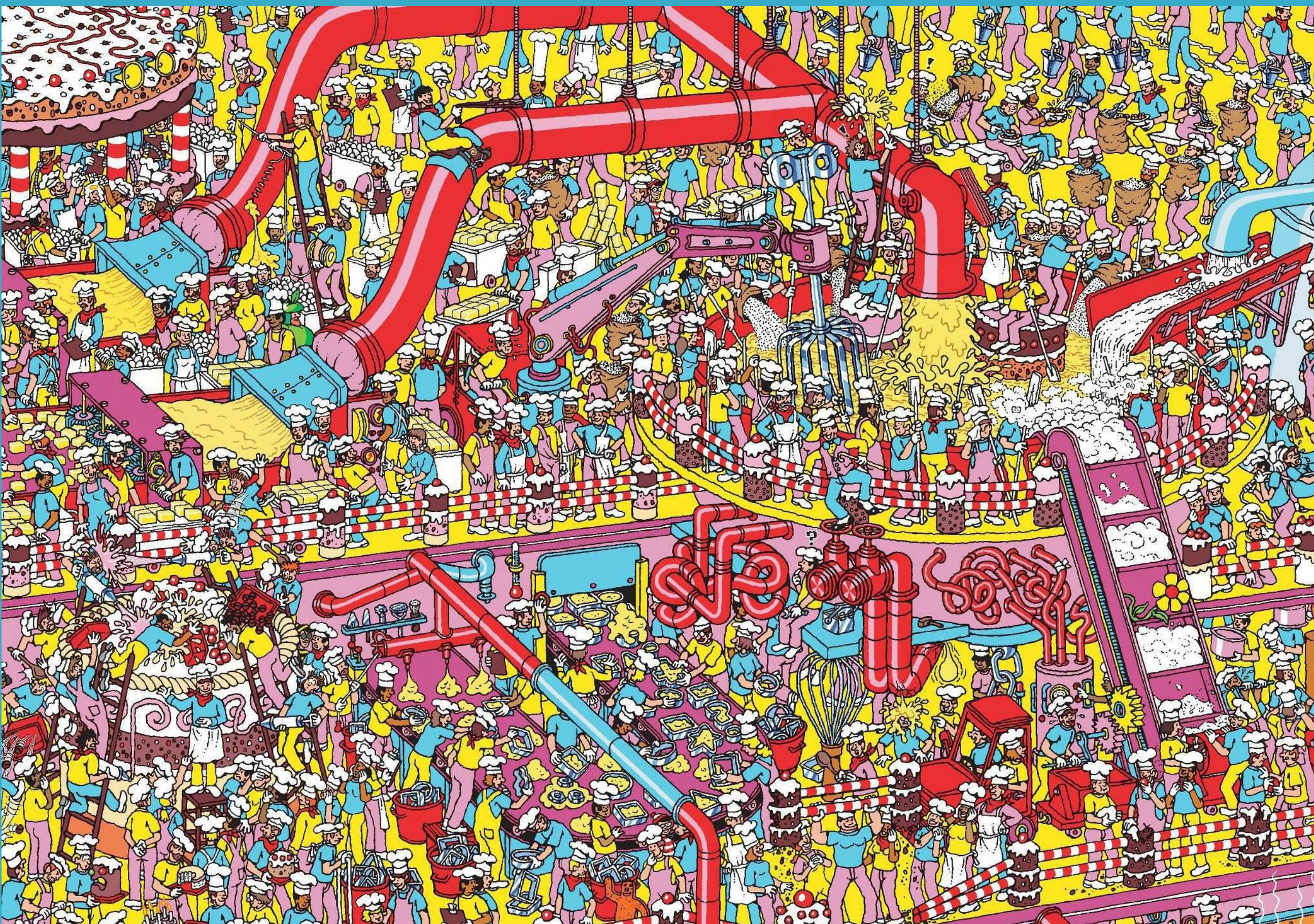
...And What Happens

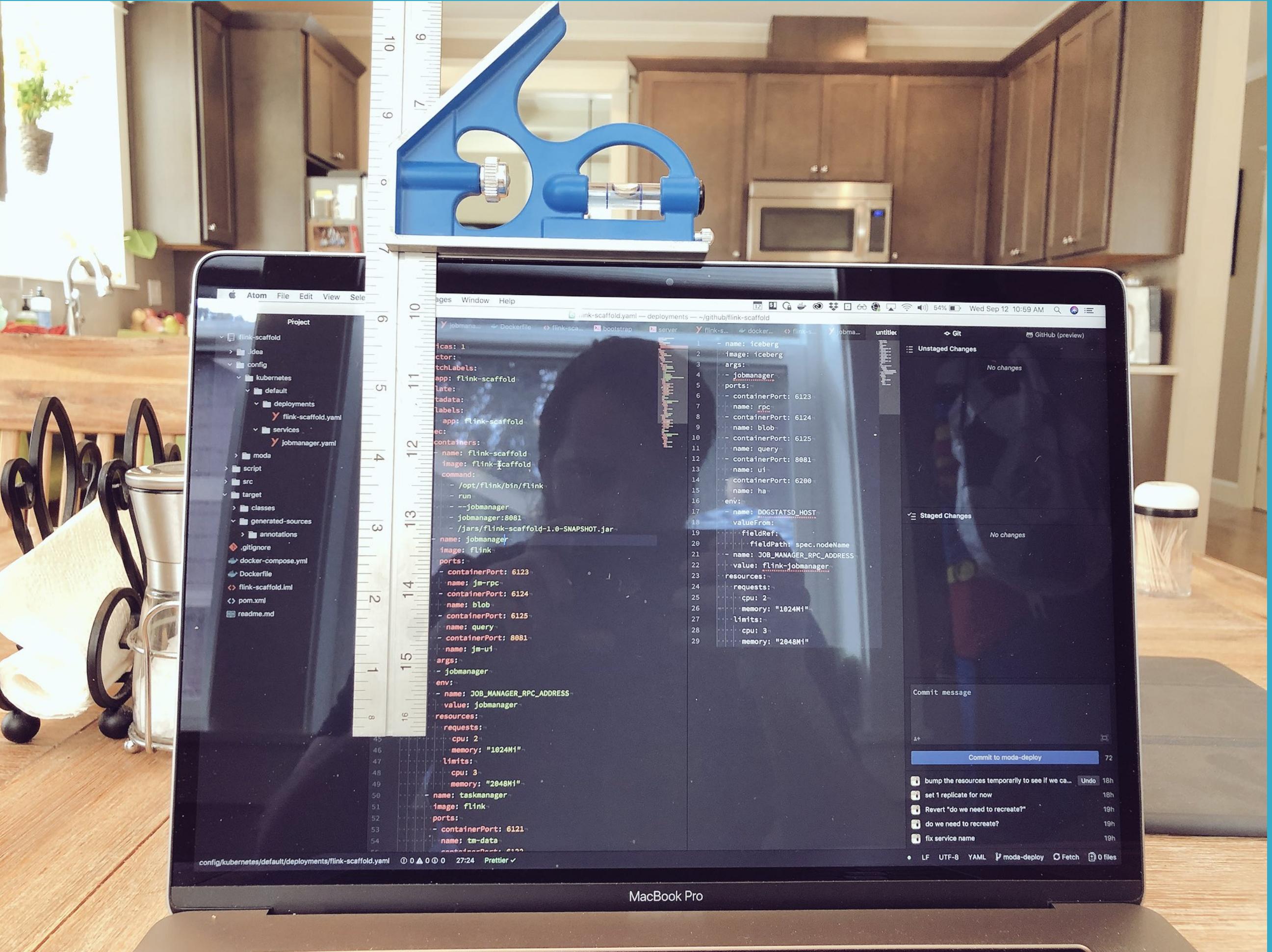


Tools Of The Trade



Where's My Service?





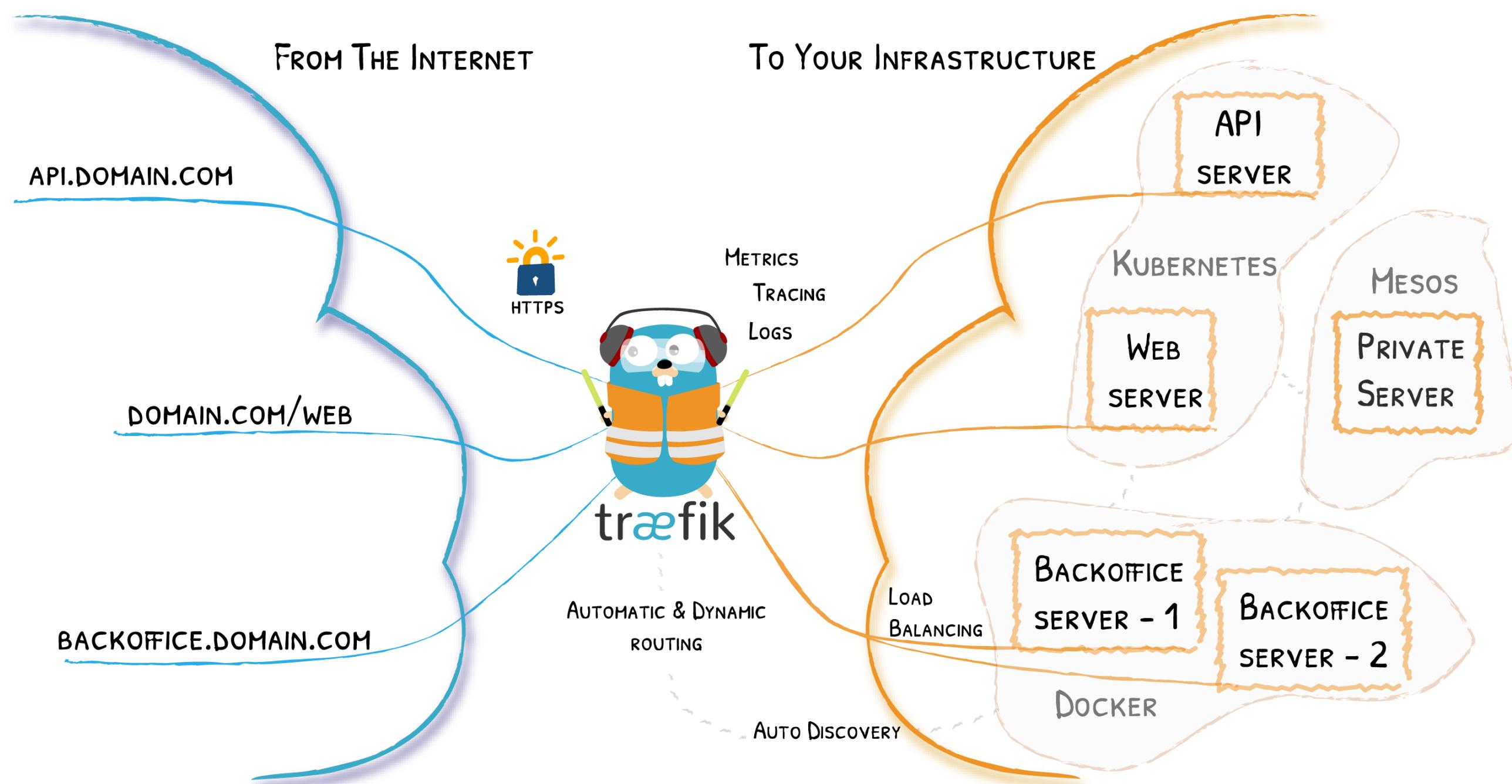
Source: <https://twitter.com/Caged/status/1039937162769096704>

What If I Told You?



That You Don't Have to Write This Configuration File...?

Here Comes Traefik!

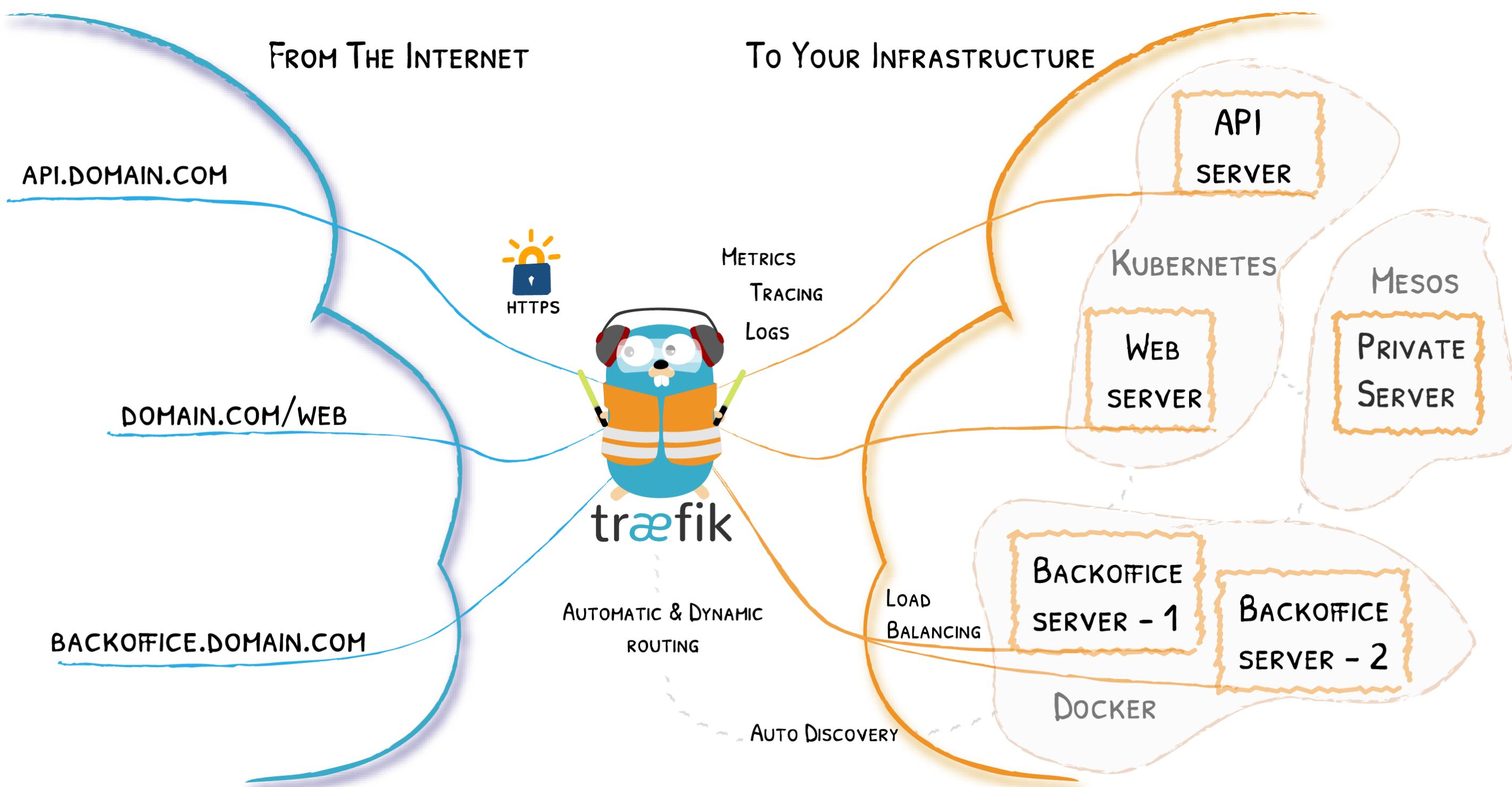


Traefik Project

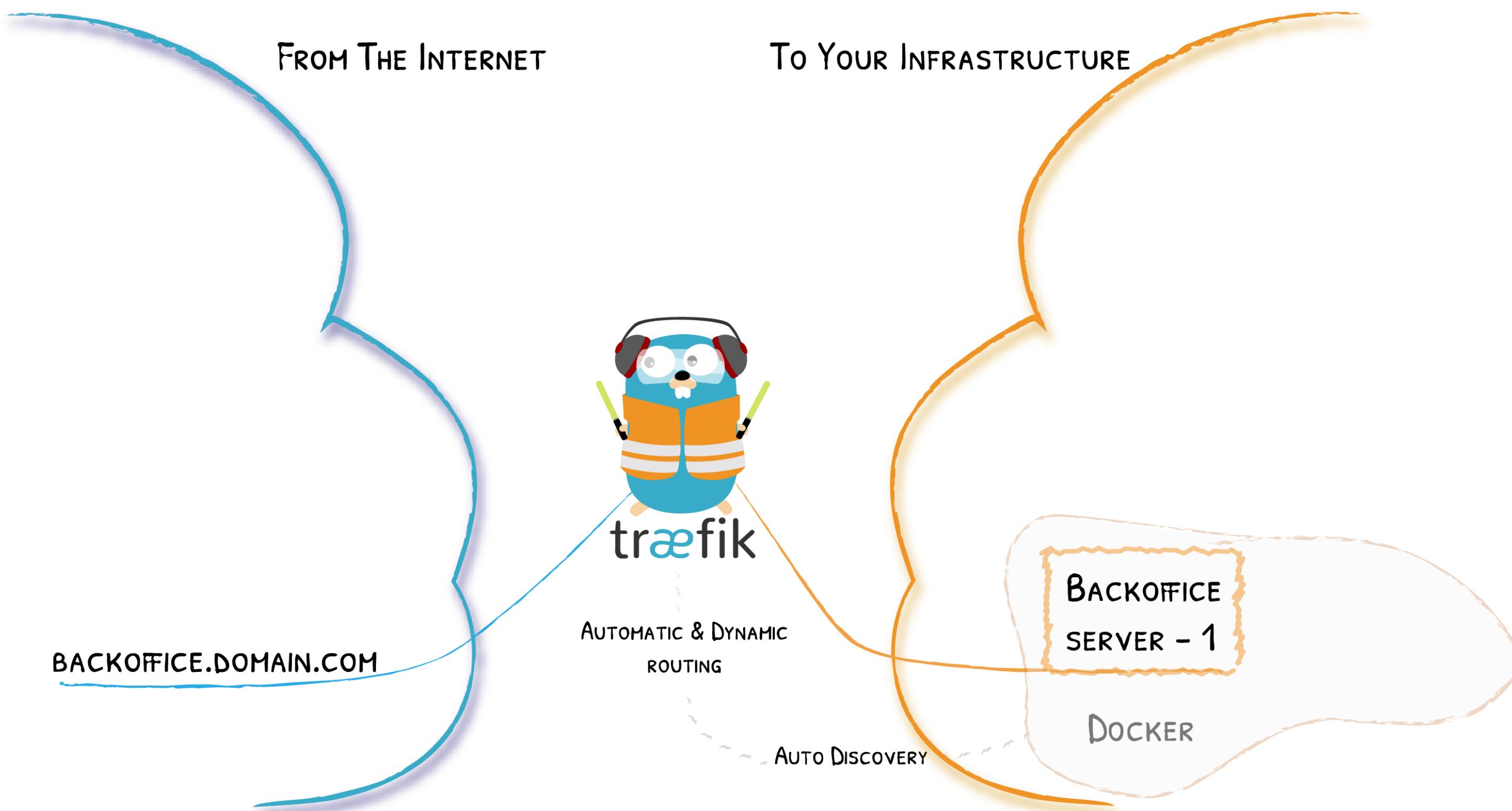
-  <https://github.com/containous/traefik>
- MIT License
- Written in Go
- 21,000+ 
- 600M+ 
- 350+ 

Lab 1 Traefik & Docker

Remember The Diagram?



Let's Simplify



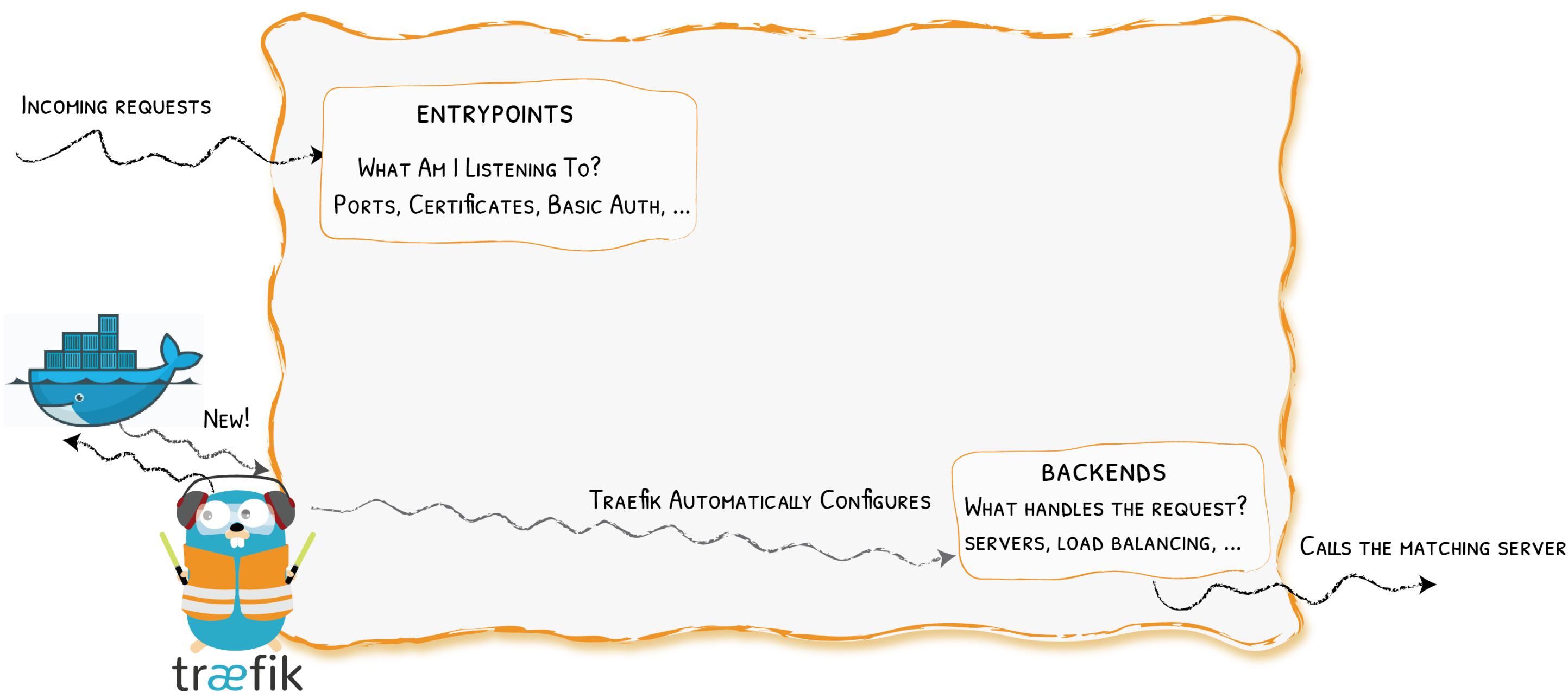
Providers



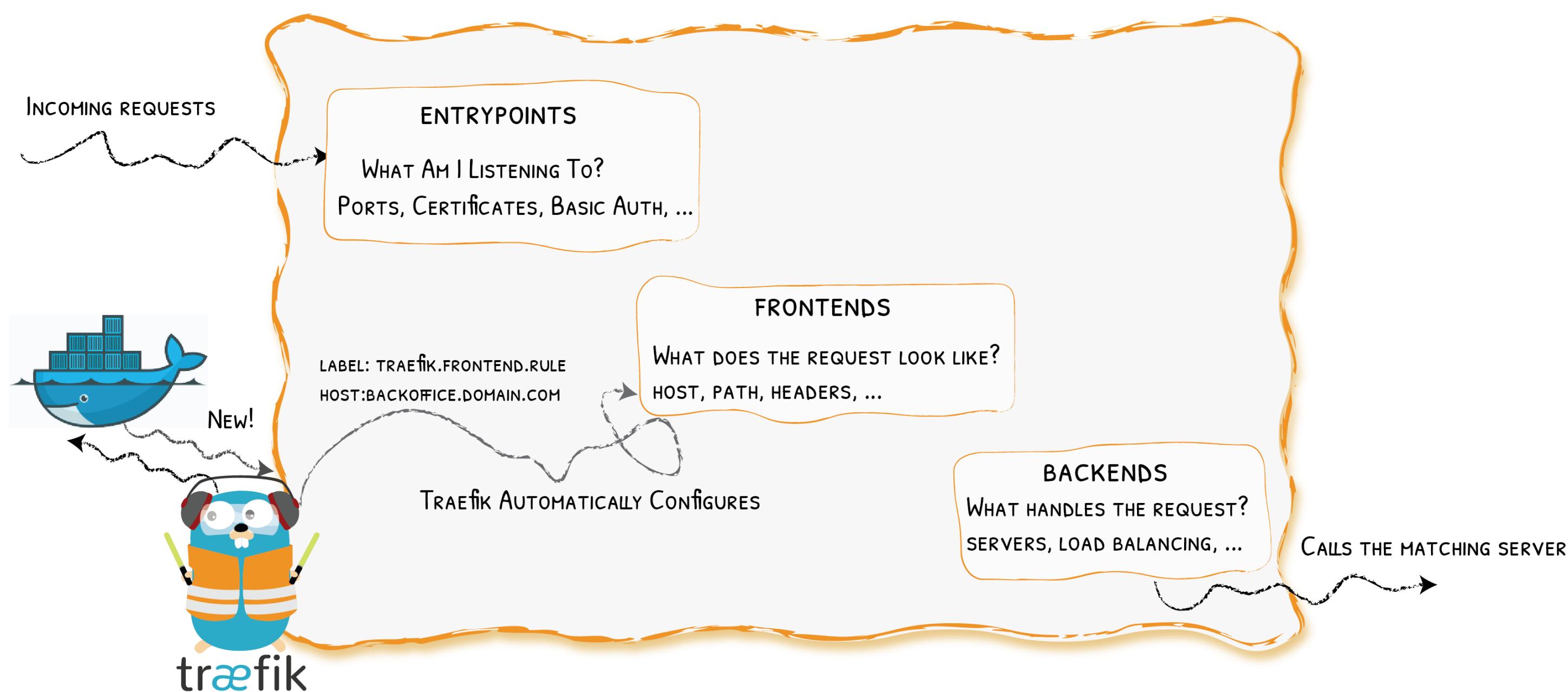
Entrypoints



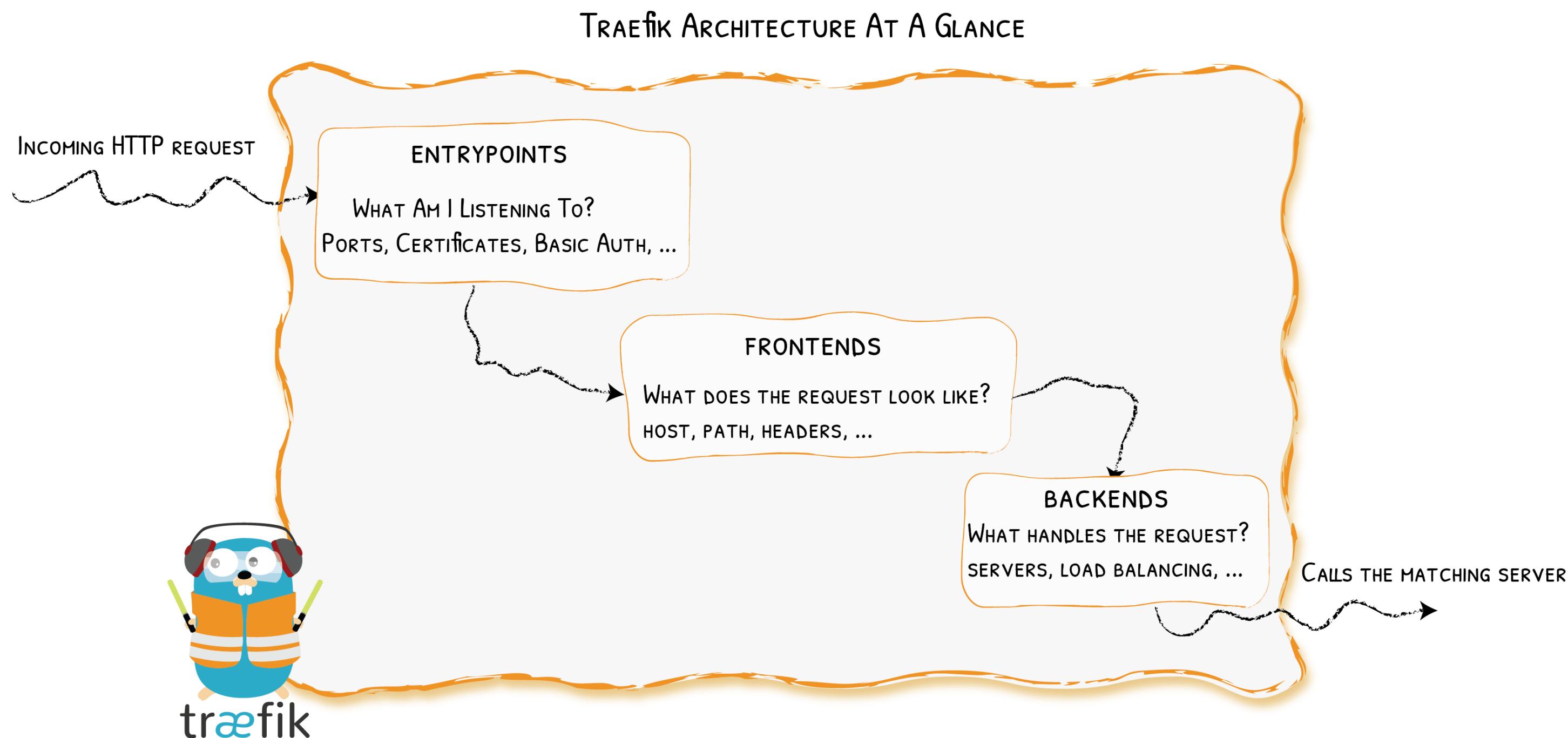
Backends



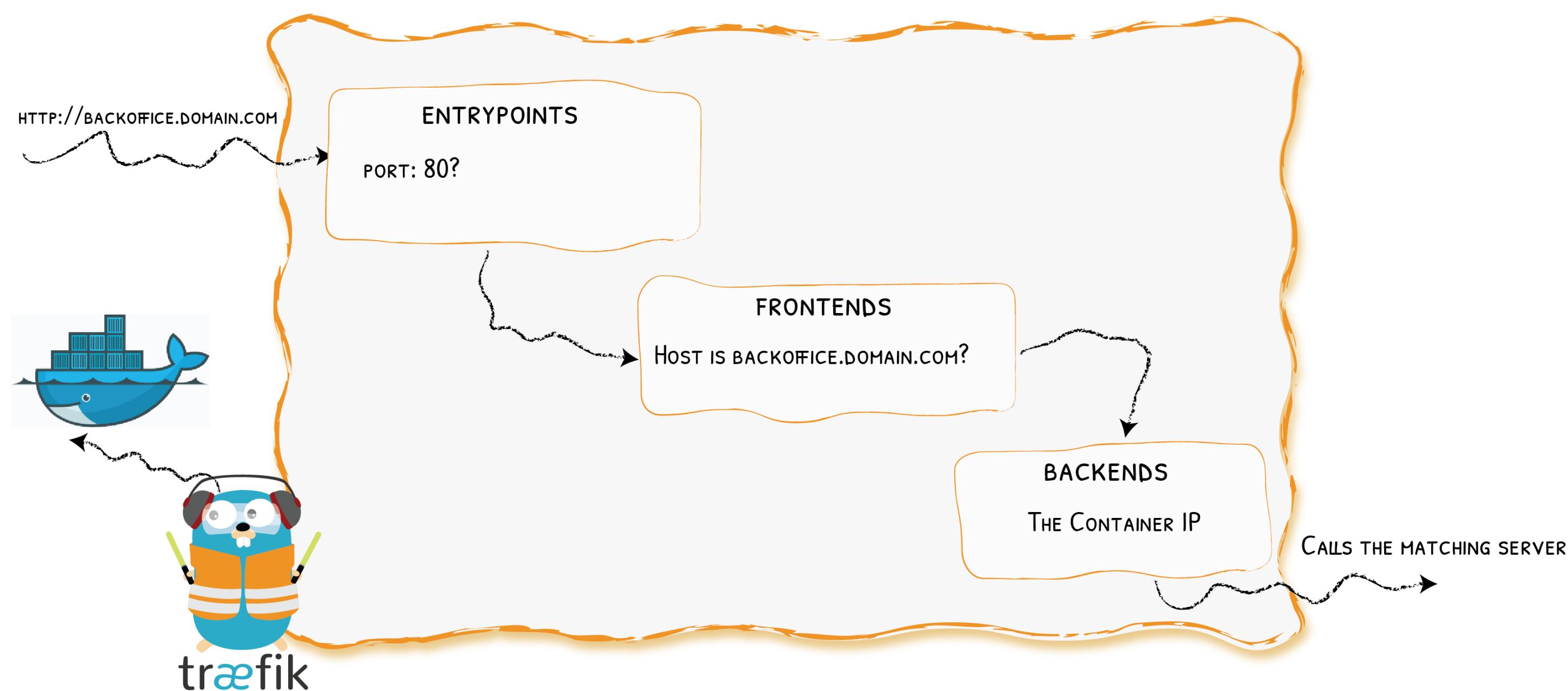
Frontends



At A Glance



In Practice



Let's Go

The Shack



The Shack

We want a server in Google Cloud:

- to host our own SCM Server,
- and our own Continuous Integration,
- and a static web site,
- and a "web" command line.

Infrastructure Setup

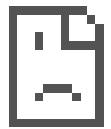
- A Google Cloud VM with a public IP 35.178.178.237 and SSH access
- A domain name lab01.demo.containous.cloud pointing to 35.178.178.237

```
$ dig +short lab01.demo.containous.cloud  
35.178.178.237
```

- Docker and docker-compose for services

Reality Check

<http://lab01.demo.containous.cloud/>



This site can't be reached

traefik-demo.containous.cloud refused to connect.

Search Google for [traefik demo containous cloud](#)

ERR_CONNECTION_REFUSED

Traefik

Traefik Setup

- Step 1: Compose file in /home/ubuntu/lab-docker-k8s/01-docker/docker-compose.yml:

```
version: '2.4'

services:
  edge:
    image: traefik:1.7.10
    command:
      - "--docker.domain=lab01.demo.containous.cloud"
    ports:
      - "80:80"
      - "443:443"
    volumes:
      # To communicate with the Docker Engine
      - /var/run/docker.sock:/var/run/docker.sock
```

- Step 2: Start the stack

```
$ docker-compose up -d
```

Reality Check

<http://lab01.demo.containous.cloud/>



It's good: we have an HTTP answer!

Simple Web Server

Goal

- We want to host a static webserver behind Traefik

Problem

- How to tell Traefik to route requests to the web server?

```
http://lab01.demo.containous.cloud/index.html
  -> Traefik
      -> http://<Webserver Private IP>/index.html
```

The Web Server Setup

- Step 1: web server in Compose. Check the labels:

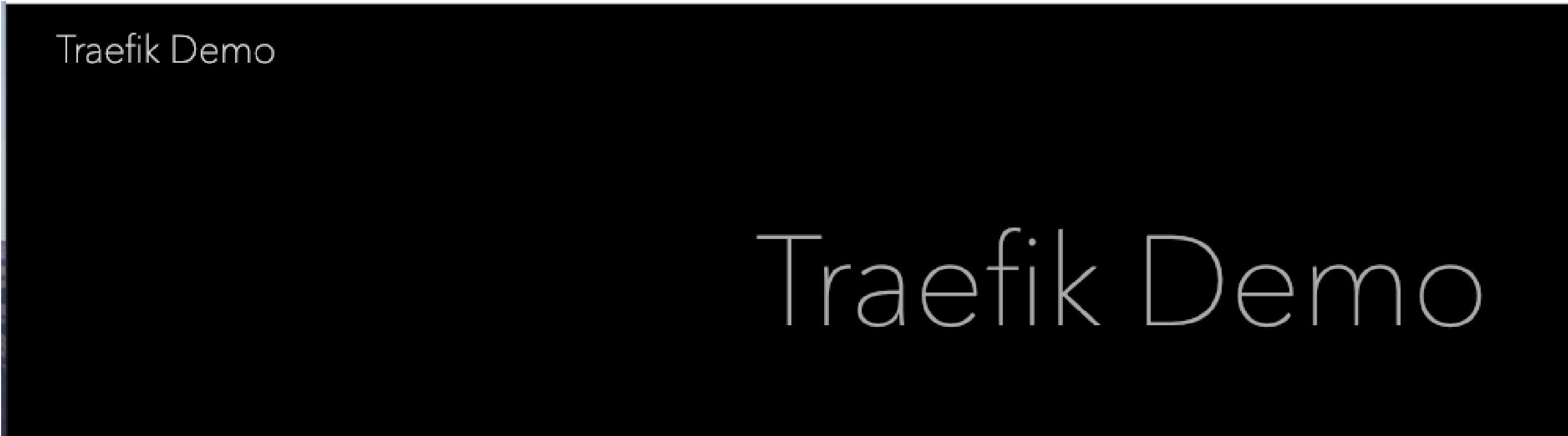
```
web:  
  build: ./web/  
  labels:  
    - "traefik.frontend.rule=Path:/"
```

- Step 2: Start the Web Server:

```
docker-compose up -d web
```

Reality Check

<http://lab01.demo.containous.cloud/>



It's good: we have a web page!

Adding A CI Server

Goal

- We want to host our own automation system for Continuous Integration
 - Let's use Jenkins

Challenge 1/3

- Problem:
 - Jenkins exposes 2 ports: 8080 and 50000. How to let Traefik know to only use 8080?
- Solution:
 - Select the port with the label `traefik.port`:

```
- "traefik.port=8080"
```

Challenge 2/3

- Problem:
 - How to let Traefik know when to send requests to the Jenkins backend instead of the webserver?
- Solution:
 - Change the frontend rule to use PathPrefix:

```
- "traefik.frontend.rule=PathPrefix:/jenkins"
```

```
http://demo.containous.cloud/jenkins/configuration
  -> Traefik
  -> http://<Jenkins Private IP>:8080/jenkins/configuration
```

Challenge 3/3

- **Problem:**
 - How to tell Jenkins to accept requests under /jenkins?
- **Solution:**
 - Use the Jenkins flag --prefix=/jenkins with the variable JENKINS_OPTS:

```
environment:  
  - JENKINS_OPTS=--prefix=/jenkins
```

Jenkins Setup

- Step 1: Edit Compose file:

```
jenkins:  
  image: jenkins/jenkins:2.150.3-alpine  
  expose:  
    - 8080  
    - 50000  
  environment:  
    - JENKINS_OPTS=--prefix=/jenkins  
  labels:  
    - "traefik.port=8080"  
    - "traefik.frontend.rule=PathPrefix:/jenkins"
```

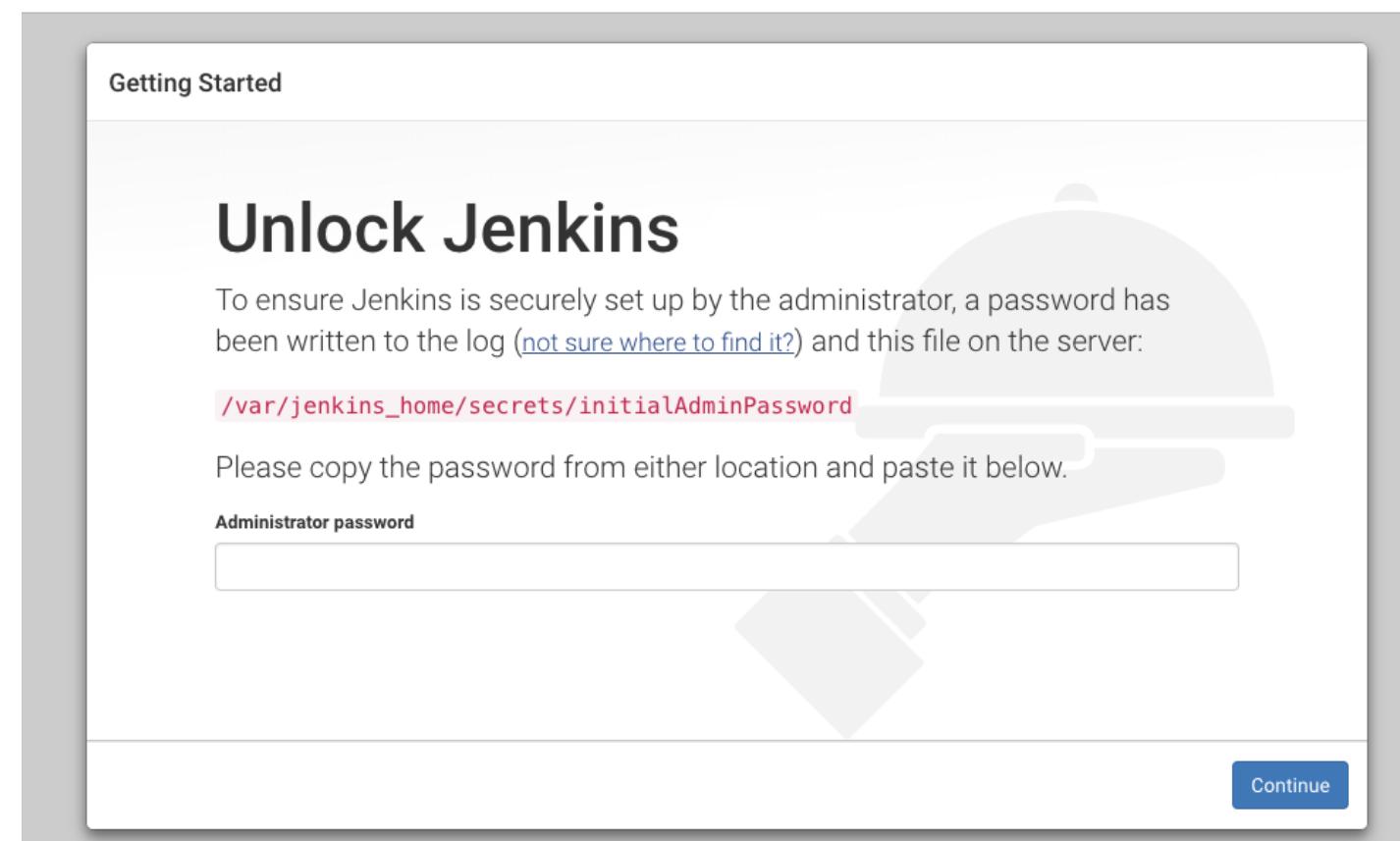
- Step 2: update the service

```
docker-compose up -d jenkins
```

Reality Check

Click on the "Jenkins" link:

<http://lab01.demo.containous.cloud/jenkins>



It's good: we can setup Jenkins!

SCM: A Gitea Git Server

Goal

- We want to host our own git server
 - Let's use Gitea, A painless self-hosted Git service.

Challenge

- Problem:
 - Gitea only serves requests under /: How to remove the prefix /gitserver?

```
http://{{lab-domain}}/gitserver/index.html
  -> Traefik
      -> http://<Gitea private IP>:3000/index.html
```

- Solution:
 - Use the Traefik's Frontend Rule PathPrefixStrip

```
- "traefik.frontend.rule=PathPrefixStrip:/gitserver"
```

Gitea Setup

- Step 1: Edit Compose file:

```
gitserver:  
  image: gitea/gitea:latest  
  expose:  
    - "3000"  
    - "22"  
  environment:  
    - ROOT_URL=/gitserver  
  labels:  
    - "traefik.port=3000"  
    - "traefik.frontend.rule=PathPrefixStrip:/gitserver"
```

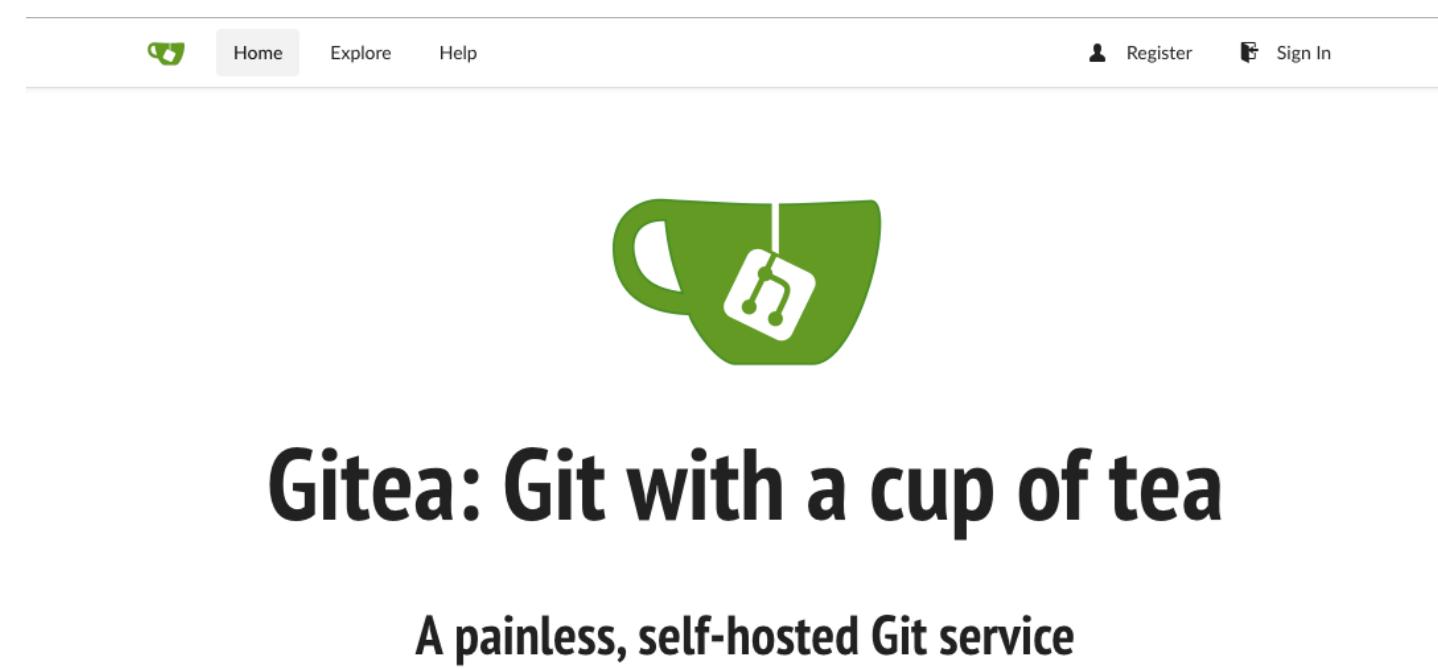
- Step 2: create the service

```
docker-compose up -d gitserver
```

Reality Check

Try the "Gitea Git server" link:

<http://lab01.demo.containous.cloud/gitserver>



It's good: we can setup Gitea!

A Web CLI

Goal

- We want to host our own Web Command Line
 - Let's use TTYD, Share your terminal over the web.

Challenge

- **Problem:** TTYD requires Websockets
- **Solution:**
 - It's not even a problem with Traefik!

Easy Peasy!

- Step 1: Edit Compose file:

```
ttyd:  
  image: ts10922/ttyd  
  labels:  
    - "traefik.frontend.rule=PathPrefixStrip:/ttyd"
```

- Step 2: create the service

```
docker-compose up -d ttyd
```

Reality Check

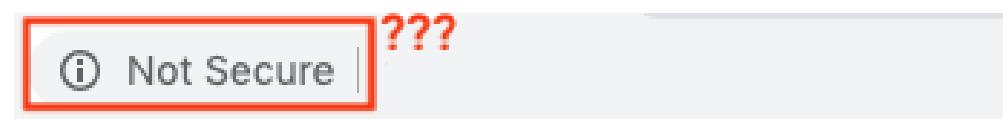
Try the "TTYD Web Command Line" link:

<http://lab01.demo.containous.cloud/ttyd>



It's good: we have our own "Dev Box" in a web browser!

HTTPS For Everyone



Goals

- Use HTTPS instead of HTTP
- Do NOT care about certificates and renewal
- Redirect any incoming HTTP request to HTTPS:

```
http://lab01.demo.containous.cloud/  
-> https://lab01.demo.containous.cloud/
```



Let's Encrypt is a free, automated, and open Certificate Authority.

It uses the "ACME" protocol to verify that you control a given domain name and to issue you a certificate.

Problem 1/4

- **Problem:**
 - How to tell Traefik to listen on port 443 for HTTPS requests?
- **Solution:**
 - Create a new entrypoint with the flag `--entrypoints`,
 - Set it as a default entrypoint with `--defaultEntryPoints`:

```
- "--entryPoints=Name:https Address::443 TLS"
- "--defaultEntryPoints=https,http"
```

Problem 2/4

- **Problem:**
 - How to tell Traefik to redirect request from http to https?
- **Solution:**
 - Configure the new entrypoint http with the flag --entrypoints:
 - `--entryPoints=Name:http Address::80" # Redirect.EntryPoint:https"`

```
--entryPoints=Name:http Address::80" # Redirect.EntryPoint:https"
```

Problem 3/4

- Problem:
 - How to tell Traefik to use Let's Encrypt for HTTPS?
- Solution:
 - Configure the ACME/Let's Encrypt provider with the flags –
–acme.*:

```
- "--acme.entryPoint=https"          # Uses LE certificates on the entrypoint "https" .  
- "--acme.email=damien@containo.us" # Specifies the contact email for the certificates.  
- "--acme.storage=acme.json"        # Stores certificates in the file "acme.storage".  
- "--acme.tlsChallenge=true"       # Uses TLS Challenge.  
- "--acme.onHostRule=true"         # Get cert's domain names from frontend rules.
```

Problem 4/4

- **Problem:**
 - Traefik detects itself as a docker container with a port, and tries to request a 2nd certificate for edge.lab01.demo.containous.cloud
- **Solution:**
 - Exclude Traefik's container with the label traefik.enable=false

Traefik Setup

- Step 1: Adapt Compose file:

```
command:  
  - "--entryPoints=Name:http Address::80" # Redirect.EntryPoint:https"  
  - "--entryPoints=Name:https Address::443 TLS"  
  - "--defaultEntryPoints=https,http"  
  - "--acme.entryPoint=https"          # Uses LE certificates on the entrypoint "https" .  
  - "--acme.email=damien@containo.us"  # Specifies the contact email for the certificates.  
  - "--acme.storage=acme.json"         # Stores certificates in the file "acme.storage".  
  - "--acme.tlsChallenge=true"        # Uses TLS Challenge.  
  - "--acme.onHostRule=true"          # Get cert's domain names from frontend rules.  
  - "--docker.domain=lab01.demo.containous.cloud"  
labels:  
  - "traefik.enable=false"
```

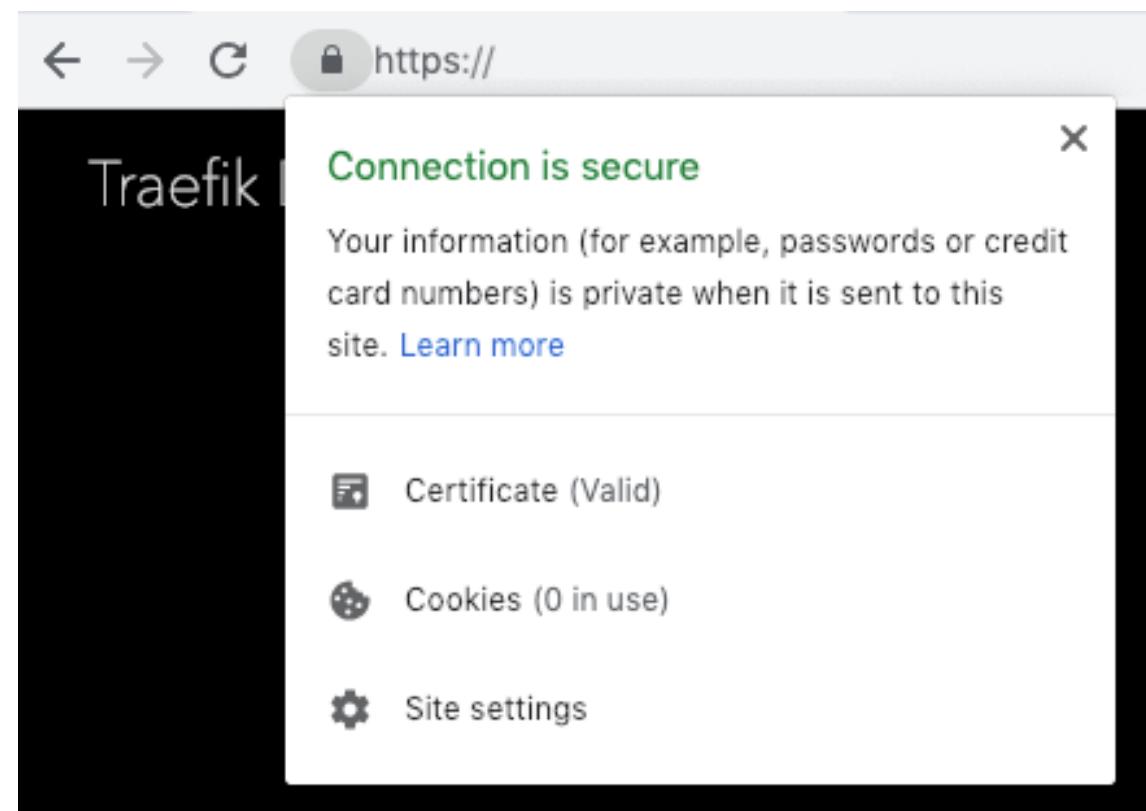
- Step 2: Update the edge service

```
docker-compose up -d edge
```

Reality Check

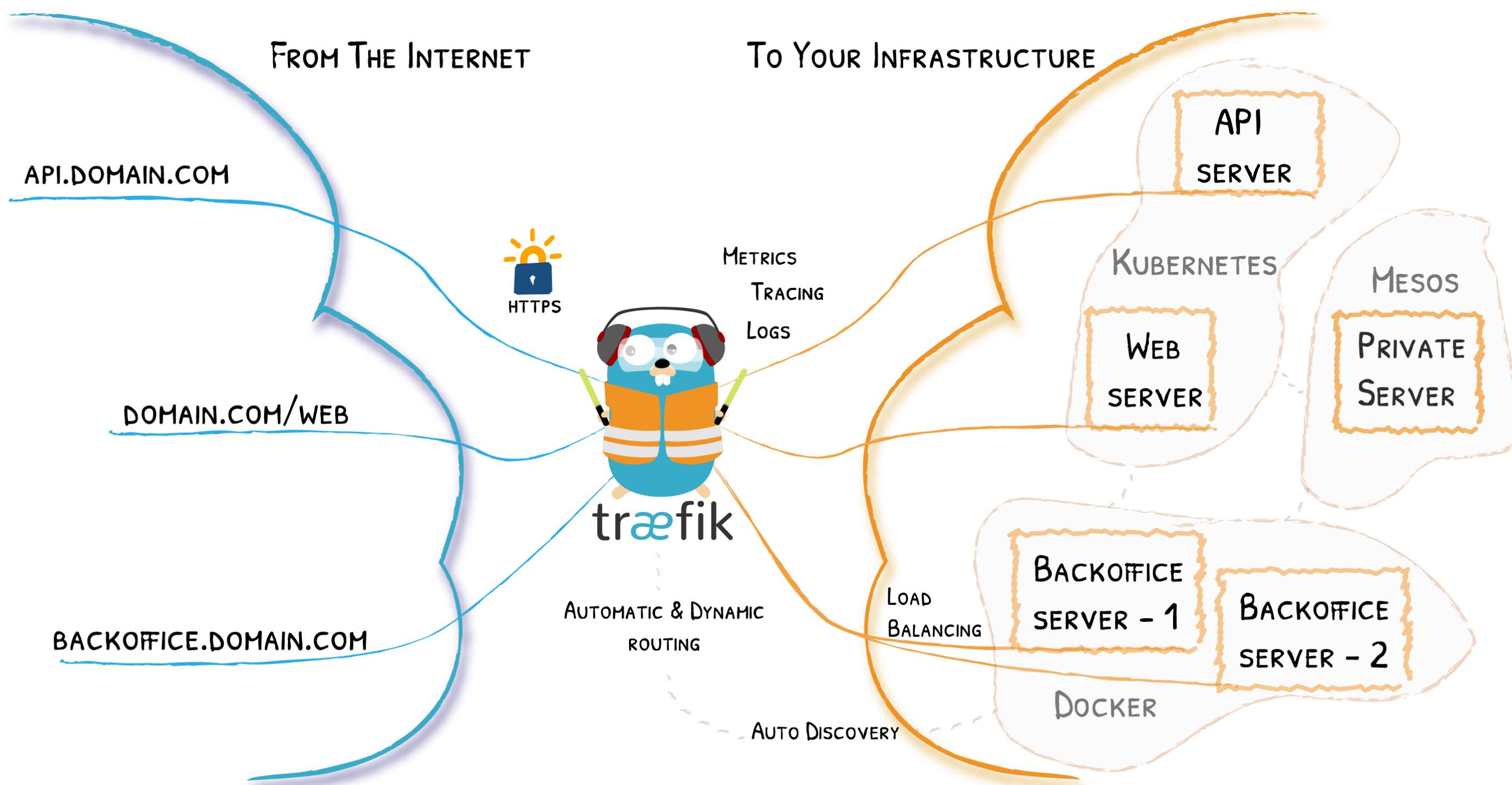
- Wait a few seconds (time to get the certificate from Let's Encrypt) and reload the main page:

<https://lab01.demo.containous.cloud>



Lab 2 Traefik & Kubernetes

Remember The Diagram?



In Kubernetes

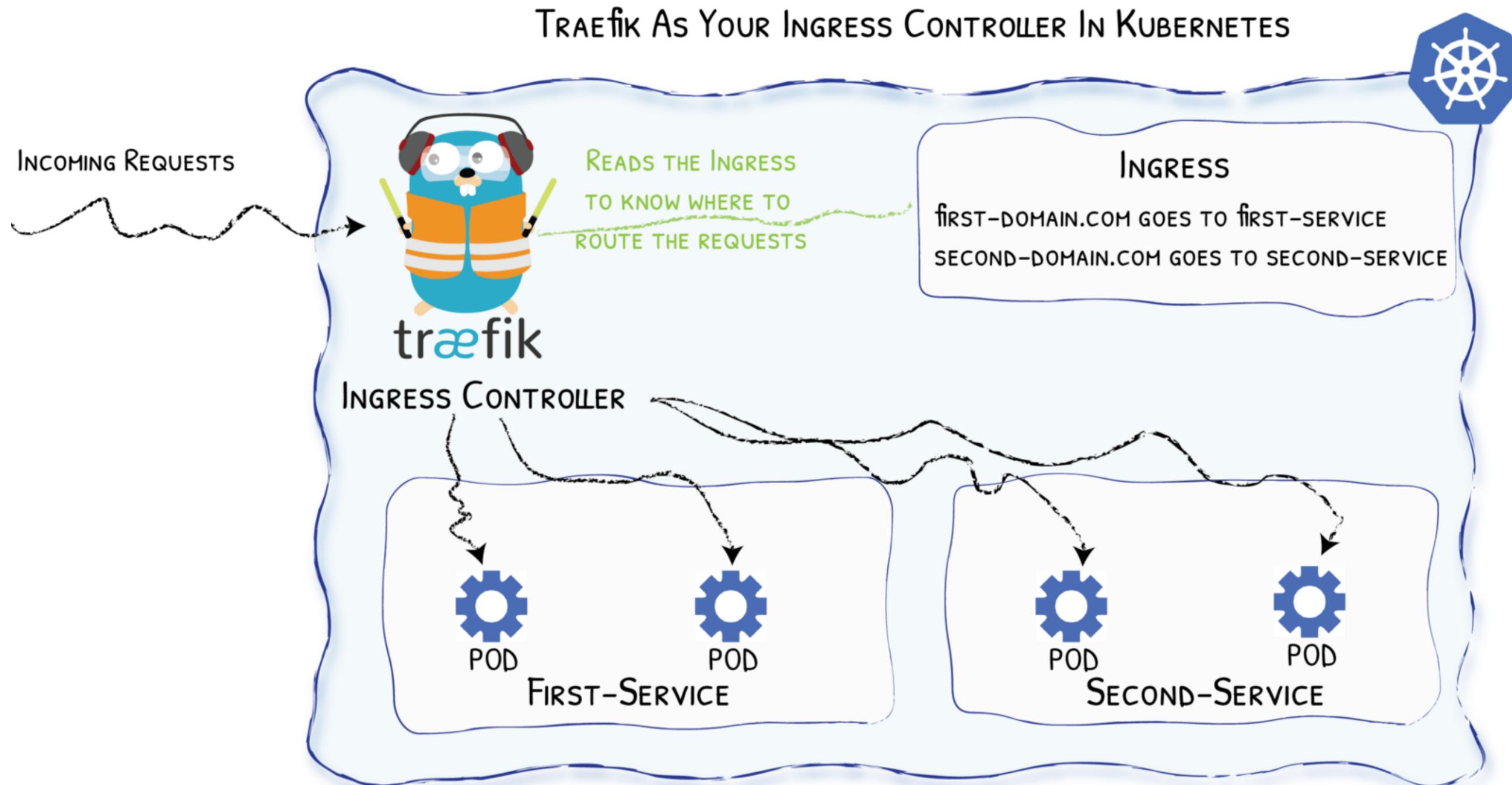


Diagram from <https://medium.com/@geraldcroes>

Let's Go

The House



The House

We want to begin the migration of our services from the Google Cloud VM to a Kubernetes cluster in Google Cloud:

- keep the Docker services
- migrate Traefik to Kubernetes
- access to the Docker services through Traefik in Kubernetes

Infrastructure Setup

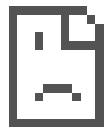
- A Google Cloud GKE server with a public IP 35.178.178.237 and SSH access
- A domain name lab01.demo.containous.cloud pointing to 35.178.178.237

```
$ dig +short lab01.demo.containous.cloud  
35.178.178.237
```

- Kubernetes and Helm to deploy the infrastructure

Reality Check

<http://lab01.demo.containous.cloud/>



This site can't be reached

traefik-demo.containous.cloud refused to connect.

Search Google for [traefik demo containous cloud](#)

ERR_CONNECTION_REFUSED

Traefik

Traefik Setup

- Step 1: Install Helm

```
curl https://raw.githubusercontent.com/kubernetes/helm/master/scripts/get | bash
# add a service account within a namespace to segregate tiller
kubectl --namespace kube-system create sa tiller
# create a cluster role binding for tiller
kubectl create clusterrolebinding tiller \
    --clusterrole cluster-admin \
    --serviceaccount=kube-system:tiller
# initialized helm within the tiller service account
helm init --service-account tiller
# updates the repos for Helm repo integration
helm repo update
```

- Step 2: Deploy Traefik

```
helm install stable/traefik \
    --name traefik-devoxx \
    --namespace kube-system
helm ls
```

Reality Check

<http://lab01.demo.containous.cloud/>



It's good: we have an HTTP answer!

Simple Web Server

Goal

- We want to host a static webserver behind Traefik

Problem

- How to tell Traefik to route requests to the web server?

```
http://lab01.demo.containous.cloud/index.html
  -> Traefik
      -> http://<Webserver Private IP>/index.html
```

Traefik Configured To Listen To Kubernetes API

- Step 1: Configure the Helm Chart values.yml file

```
# Allow creating the needed Role and Service Account
rbac:
  enabled: true
```

- Step 2: Install Traefik in a specific namespace

```
helm delete --purge traefik-devoxx
helm install stable/traefik --name traefik-devoxx --namespace devoxx -f ./values.yml
```

The Web Server Setup

- Step 1: Declare a Headless Service

```
---  
apiVersion: v1  
kind: Service  
metadata:  
  name: web-service  
  namespace: devoxx  
  labels:  
    guilde: web  
spec:  
  ports:  
  - port: 80  
    name: fake-http  
  type: ExternalName  
  externalName: redirect.containous.cloud
```

- Step 2: Declare the Ingress Rule

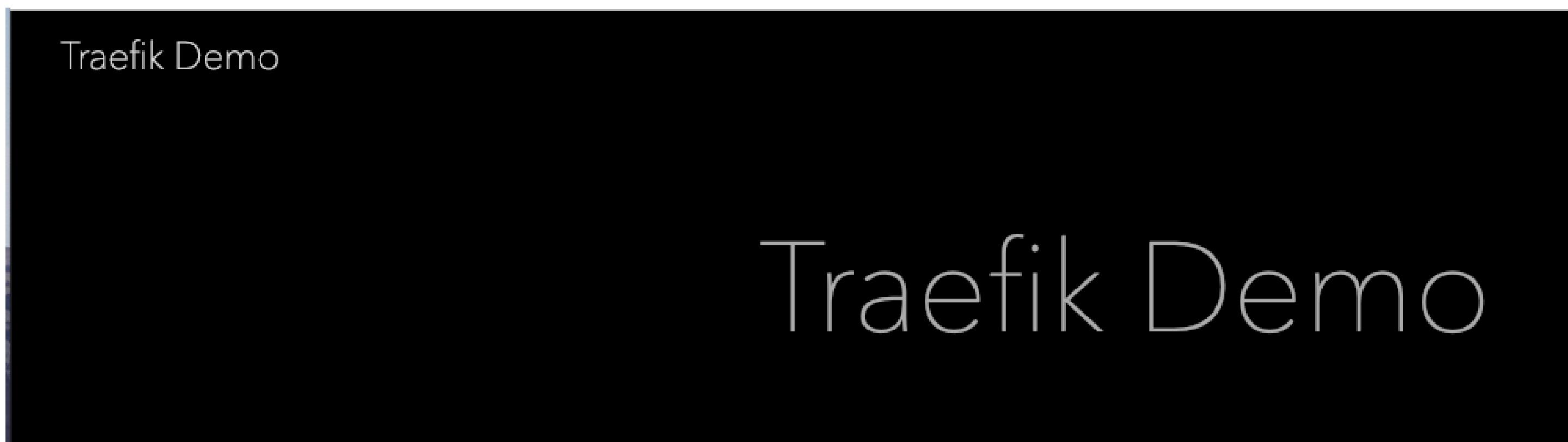
```
---  
apiVersion: extensions/v1beta1  
kind: Ingress  
metadata:  
  name: web-ingress  
  namespace: devoxx  
  labels:  
    guilde: web  
  annotations:  
    kubernetes.io/ingress.class: 'traefik'  
    traefik.frontend.passHostHeader: "false"  
    traefik.frontend.rule.type: PathPrefix  
spec:  
  rules:  
  - host: lab.nicolas.containous.cloud  
    http:  
      paths:  
      - path: /foobar  
        backend:  
          serviceName: web-service  
          servicePort: fake-http
```

- Step 3: Apply the configuration

```
kubectl apply -f ./web/svc-ingress.yml
```

Reality Check

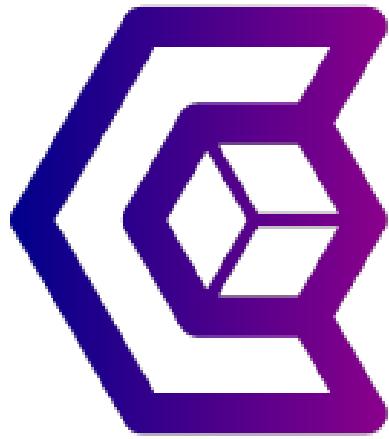
<http://lab01.demo.containous.cloud/>



It's good: we have a web page!



We Are Hiring!



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

Thank You!

 @nicomengin  nmengin

 @DamienDuportal  dduportal



- Slides (HTML): <https://containous.github.io/slides/devoxx-fr-2019>
- Slides (PDF): <https://containous.github.io/slides/devoxx-fr-2019/slides.pdf>
- Source on : <https://github.com/containous/slides/tree/devoxx-fr-2019>