JANUARY 15TH, 2020

JAKUB HAJEK

CONTAINER ORCHESTRATION WITH TRAEFIK 2.X

Cometari

INTRODUCTION

- I am the owner and technical consultant working for Cometari
- I've been system admin since 1998
- Cometari is a solutions company implementing DevOps culture and providing consultancy, workshops and software services.
- Our expertise are DevOps, Elastic Stack - log analysis, Cloud
- We are deeply involved in the travel tech industry
- However our solutions go much further than just integrating travel API's.

"I strongly believe that implementing **DevOps** culture, across the entire organisation, should provide measurable value and solve the real issue rather than generate a new one."

Jakub Hajek, Cometari

The goal of this presentation is to show you how we work with distributed systems and how Traefik makes our daily work easier.

CONTAINERS

IMMUTABLE CONTAINERS

- Mutable vs Immutable
- No incremental changes to the image
- No more drifting configuration
- (No) imperative updates
- Base image + source code = An artefact / immutable image
- The artefact is scaling unit in distributed systems
- Canary and mirror deployments
- Rollback if an error occurs

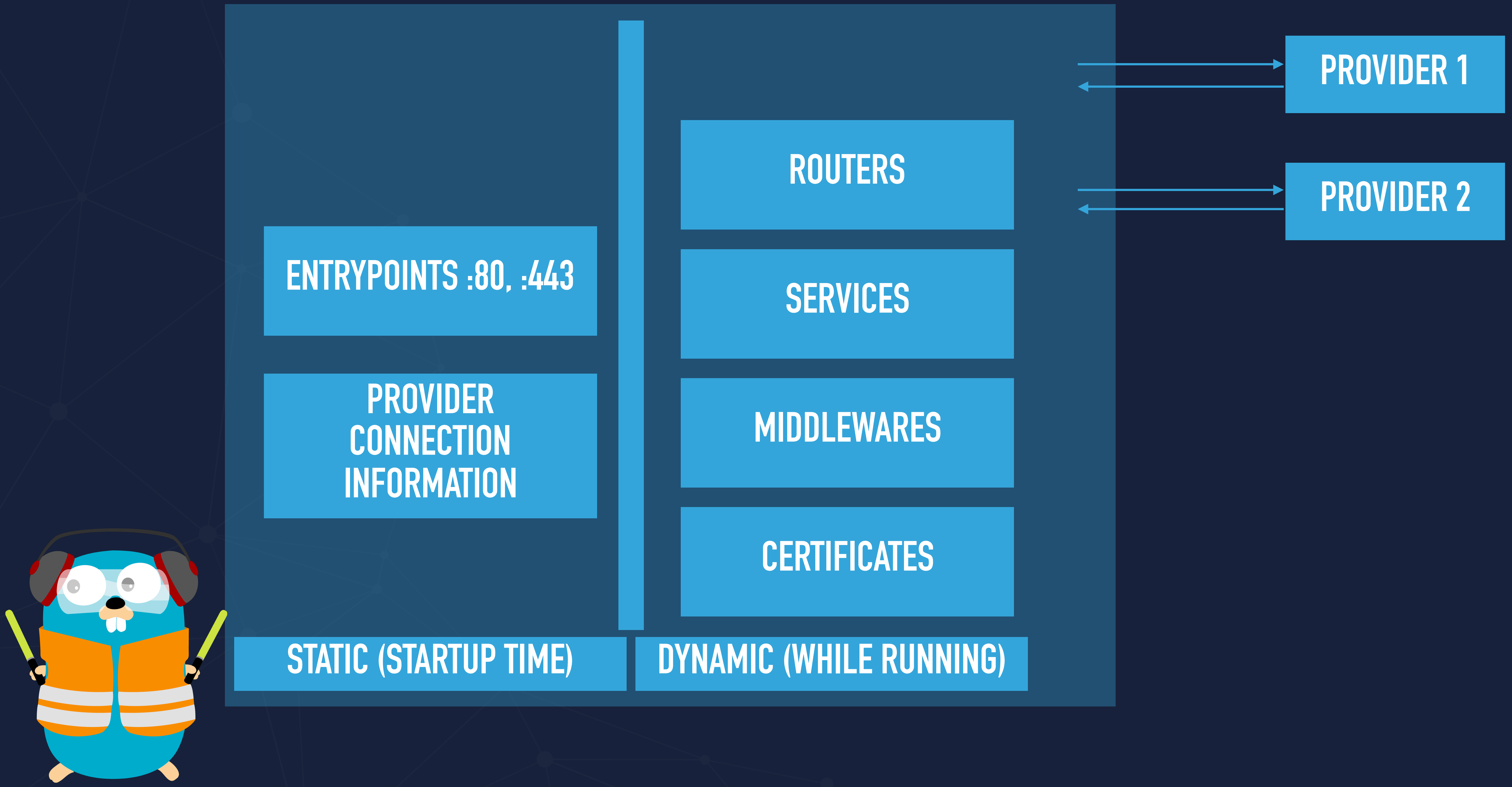
Immutable containers are at the core of any distributed systems

TRAEFIK 2.X

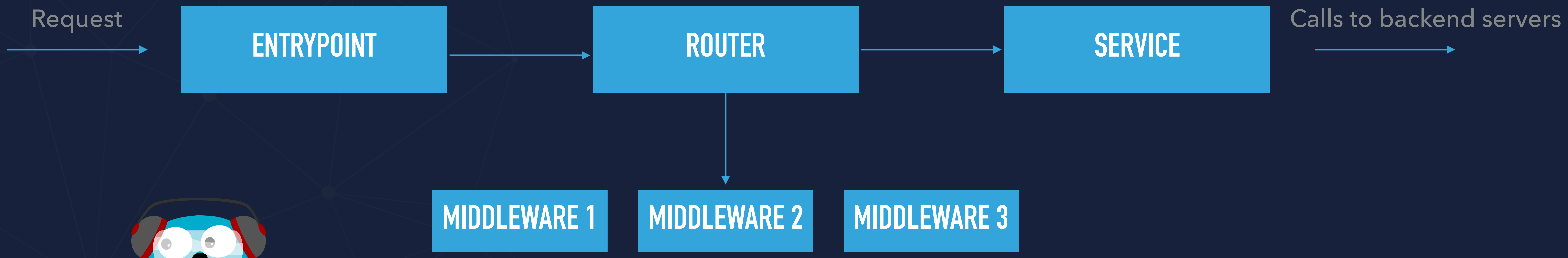
TRAEFIK 2.X KEY FEATURES

- TCP support
- **ROUTER**= frontend, **SERVICE**=backend, **MIDDLEWARES**=rules
- Fully customisable routes via middleware, which can be reused on many routers
- YAML, TOML is still good
- A new dashboard with web UI
- **Canary** deployment with Service Load balancer
- Network traffic **Mirroring** with Service Load balancer
- Consul catalog

Traefik configuration introduction



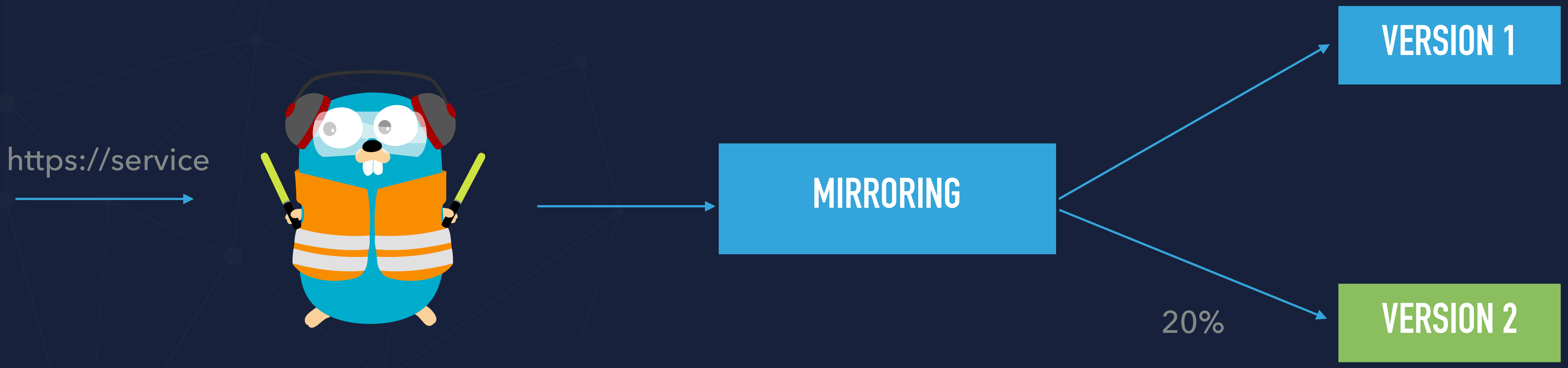
Tweaking request before / after the arrives to their destination



MIRRORING OR LIVE TRAFFIC SHADOW

- Understand difference between **Deployment** vs **Release**
- **Deployment** brings new code to the production, **no production traffic yet!**
- Run smoke, integration tests to make sure that new deployment has no impact to your users
- **Release** brings live traffi c to a deployment.
  - We can shadow live traffic to the new deployment and reduce the risk of failure.

Mirroring with Service Load Balancer

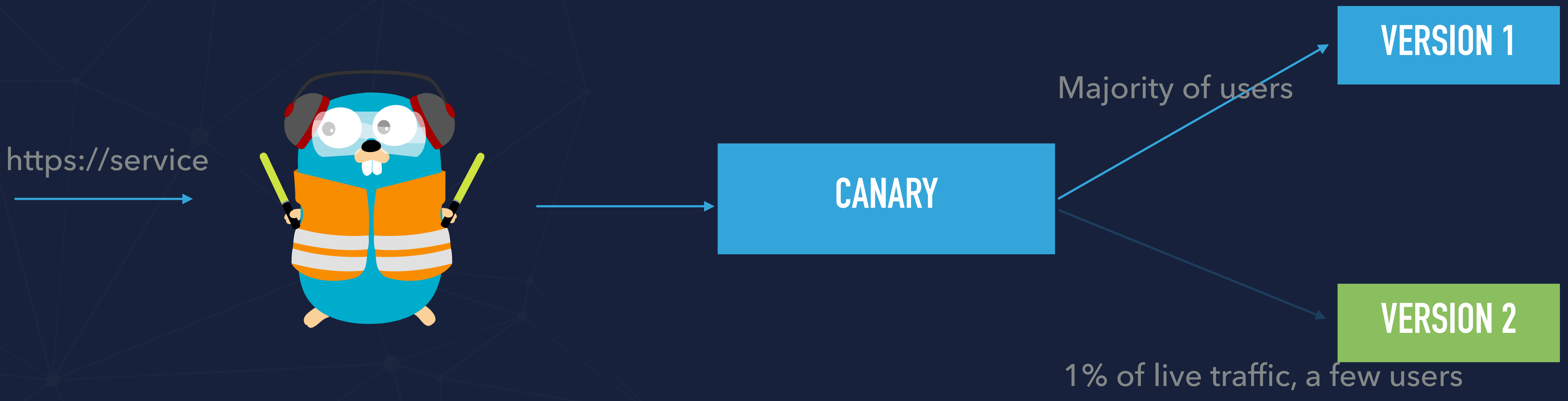


https://github.com/containous/traefik/issues/2989

CANARY DEPLOYMENT

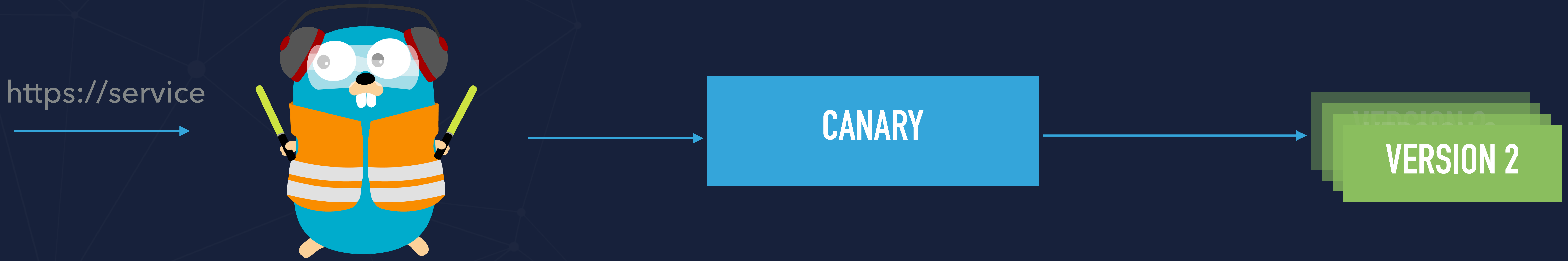
- **Deployment** vs **Release**
- Instead of switching to new version in one step, we use a phased approach
- We **deploy** a new app in a small part of the production infrastructure
- Only **a few users (1%)** are routed to the newest version (**Release**)
- With no errors reported, the new version can be released to the rest of the infrastructure.

Canary deployment with Service Load balancers



https://github.com/containous/traefik/issues/1164

Canary deployment with Service Load balancers



IMMUTABLE CONTAINER WITH TRAEFIK

- Custom image with Traefik with added SSL certificate into image
- Configuration files added directly to the image
- Works perfectly if you bought SSL cert and don’t use dynamically updated Let’s Encrypt
- Horizontal scalability is simple, no need to care about the persistence for Let’s Encrypt certificates

OBSERVABILITY

- Enable Prometheus or any other backend
- Use Grafana to visualise metrics
- Use existing dashboards to visualise data (or develop your own)

LOGGING AND VISUALISING ACCESS LOGS

- Traefik logs are in JSON including startup and errors events
- Access logs are written to STDOUT in JSON format.
- Treat logs as an event and transfer them to external system (Elastic Stack + Fluentd)
- Use Kibana and Logs tab to have live data streaming
- Develop dashboard with a map and place GEO points of IP addresses

CONFIGURATION TIPS

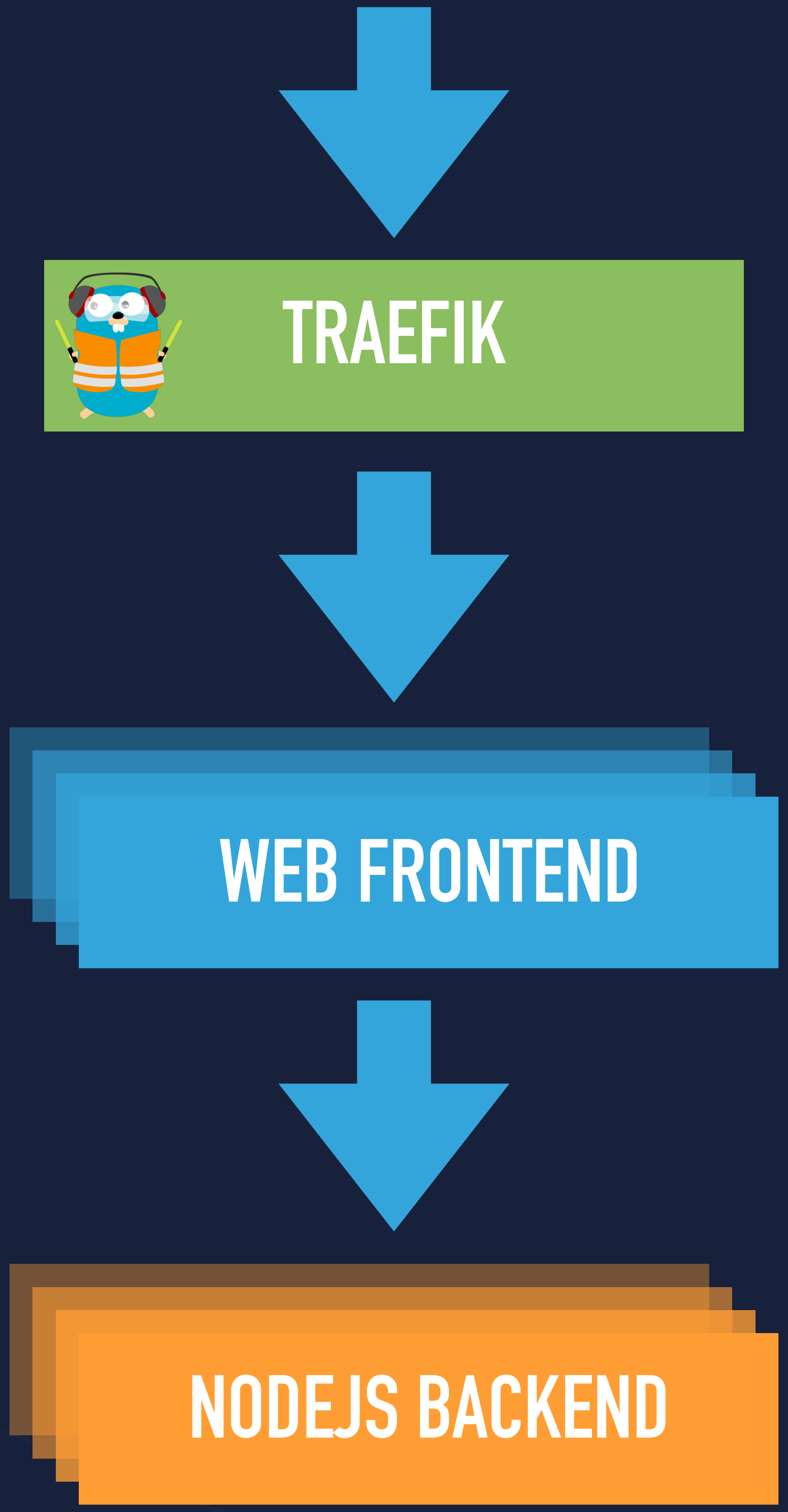
- Don't mix **static** configuration vs **dynamic** configuration
- **CLI** command can be used for static config or if you prefer you can define config file as well
- **Labels** can be used to define dynamic configuration or config files
  - directory with WATCH flag enabled as well
- More **advanced** rules configuration via middleware are dynamically defined
- The most flexible is to run **Traefik** as container, instead of binary directly from host
- **Healthcheck** for your services are crucial

DEMO ENVIRONMENT

DEMO ENVIRONMENT IN DETAILS

- Docker Swarm cluster* consisting of 4 nodes
- SSL certs issued by Lets Encrypt
- FQDN domains:
  - https://traefik.labs.cometari.eu
  - https://node-app.labs.cometari.eu
  - https://canary.labs.cometari.eu
- DNS Round Robin: Route 53 with its implemented health checks
- Prometheus and Grafana
- Elastic stack, Fluentd to store and visualise data logs
- Web Server and NodeJS backend
- Stacks: Traefik, App stack, Consul for consul catalog

Diagram of demo environment



Overview of configuration files!

DEMO TIME

DEMO SCENARIOS

- Web UI to see how services are deployed
- Scaling services and generating some network traffic via Slapper
- Example of **Canary** deployment
- Metrics from Traefik in open metrics format

SUMMARY

- Traefik provides flexible way to expose services, auto discovery
- It can be configured in multiple way, there are no ready to use config - just refer to configuration tips
- Fully customise routes via middlewares
- Easily integrates with every major cluster technology
- Lets Encrypt integrated, managing SSL certs is easy
- Metrics, Tracing, Logs
- Rolling out releases thanks to Canary deployments
- Mirroring - duplicating incoming request and send them to different services.

@_JAKUBHAJEK

JAKUB.HAJEK@COMETARI.COM

THANK YOU

Cometari