Container Orchestration with **Docker Swarm** And TRAEFIK

Jakub Hajek / Cometari
Technical consultant & Founder

DevOps Congress, Wrocław, October 19th, 2019

Introduction

- I am the owner and technical consultant working for Cometari
- I have been system admin since 1998.
- Cometari is an agency implementing **DevOps** culture, providing consultancy, workshops and software services.
- Our areas of expertise are DevOps, **Elastic Stack** (log analysis), **Cloud Computing.**
- We are very 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

What will be presented

- Key features of Docker Swarm working together with Traefik
- Demo of fully fledged environment running on 4 nodes working as Swarm cluster.
- Following application stacks will be deployed:
  - Traefik working as an edge router exposing services to the Internet
  - Frontend app running as Nginx
  - A simple NodeJS backend

DOCKER SWARM key features

What is Docker Swarm?

- Current version of Docker includes swarm mode natively managing a cluster of Docker Engines called Swarm

- Creating Swarm cluster is achieved by: `docker swarm init`

- Even single instance of Docker can be a cluster consisting of one node

- Adding new node to the cluster with simple command: `docker swarm join`

Decentralized design

Worker

Manager

Leader

RPC:

- HTTP/2 + PROTOBUF GOOGLE
- TWO WAY COMMUNICATION
- BI-DIRECTIONAL STREAMING
- BUILT IN AUTHENTICATION
- VERSIONED

Roles in details

LEADER:

- REPLICATION LOG
- STATE CONSISTENCY
- SCHEDULING

MANAGER ROLES:

- ORCHESTRATION
- SCHEDULING
- QUORUM

WORKER ROLES:

- JUST DOES THE WORK
- KNOWS ABOUT MANAGER
- CAN'T VIEW CLUSTER
- RETURN STATUS OF TASKS
- SCALABLE
- USES GOSSIP PROTOCOL

RAFT CONSENSUS*ALGORITHM:

- HIGH CONSISTENCY
- FAULT TOLERANCE
- LEADER ELECTION
- /VAR/LIB/DOCKER/SWARM/RAFT

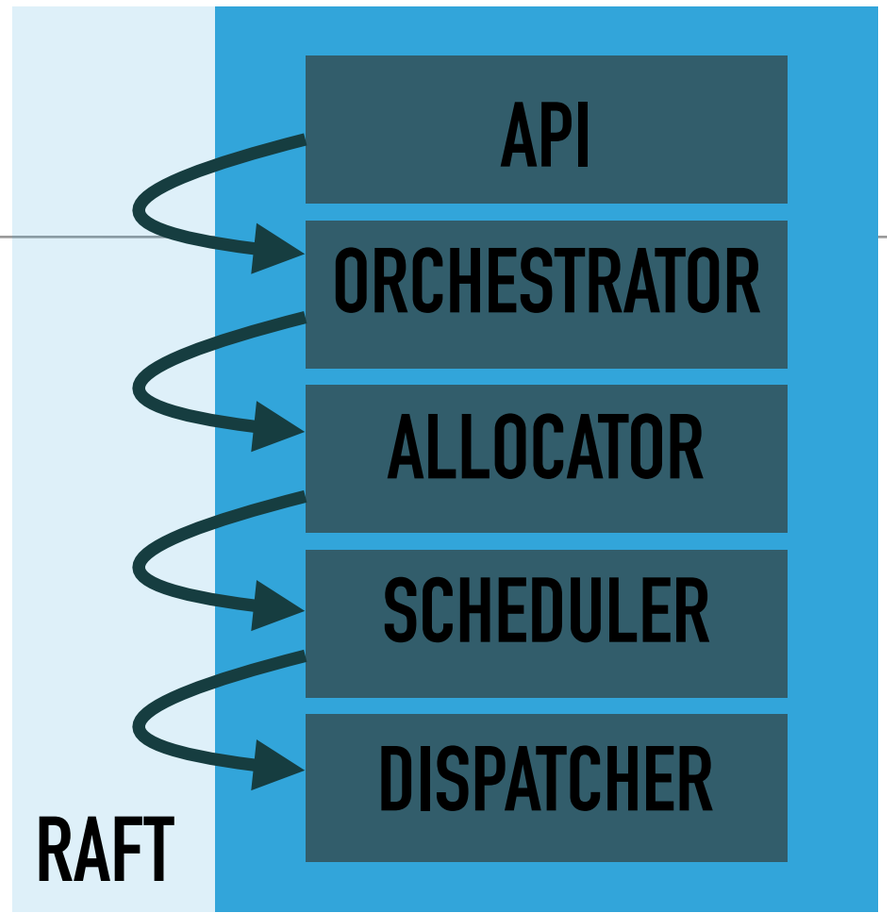
* ETCD, CONSUL,

Declarative service model

Desired state reconciliation

- The manager constantly monitor cluster state.
- In case of any difference between desired state manager tries to reconcile the expressed state.
- Service defined with 4 replicas, 1 of them crashed. Manager will try to run another missing replicas.

Docker service create



MASTER NODE

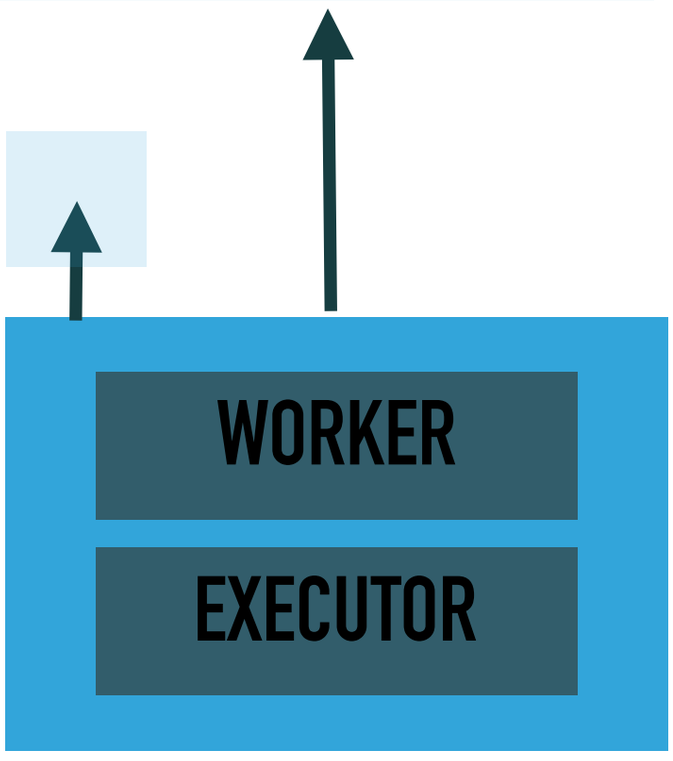
Accept commands from docker client and create service object

Reconciliation loop for service objects and create tasks

Allocates IP addresses to tasks

Assigns nodes to tasks

Checks in on workers



WORKER NODE

Connects to dispatcher to check on assigned tasks

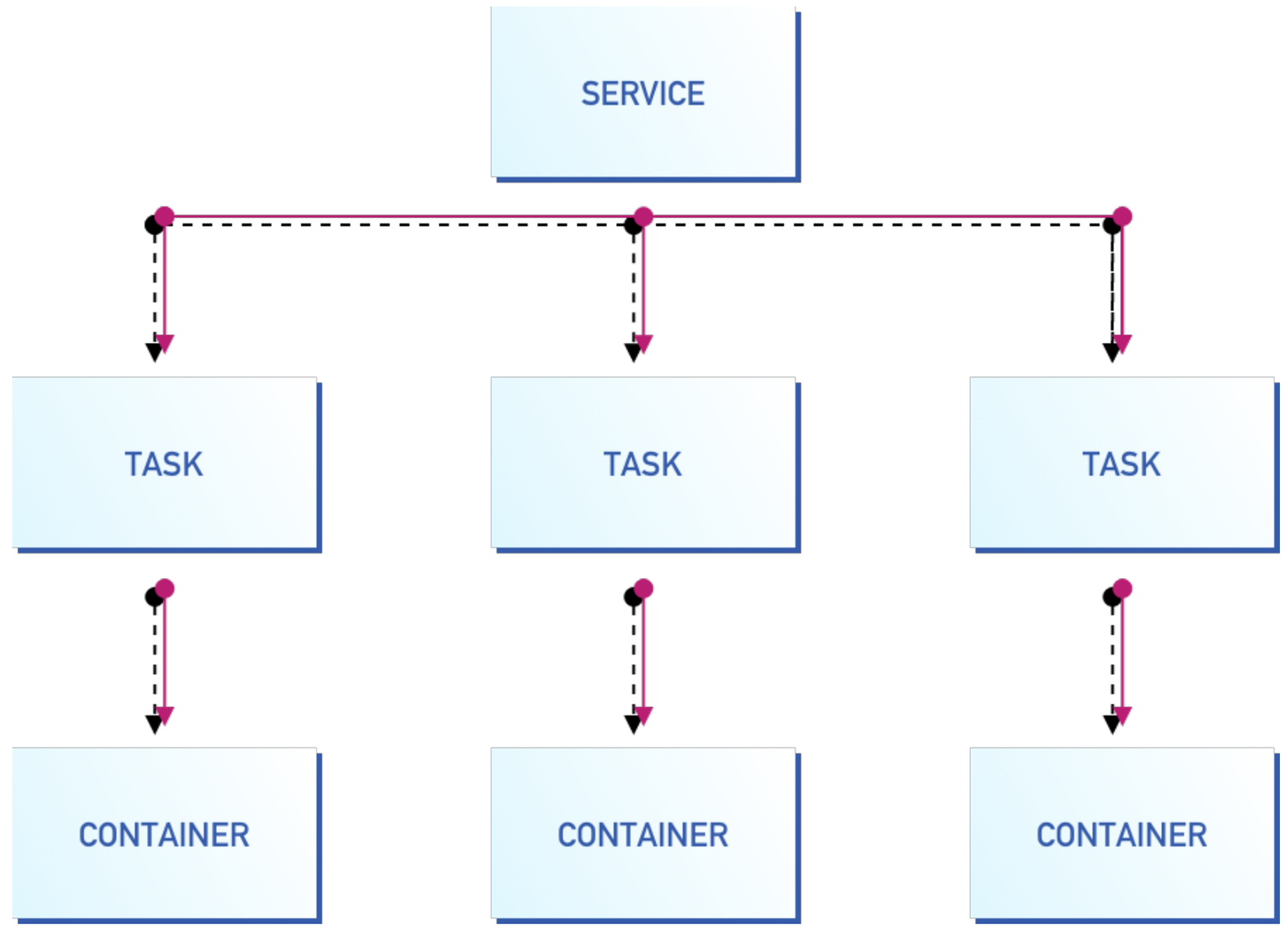
Execute the tasks assigned on worker node

Docker service create - Example usage

```
$ docker service create  \
--name web \
--replicas 3 \
--update-delay 10s \
--update-parallelism 1 \
--constraint 'node.role == manager' \
--publish 8080:80 \
--env MY_ENV=my-value \
--secret source=SSL_KEY,target=ssl-key \
nginx:1.17
```

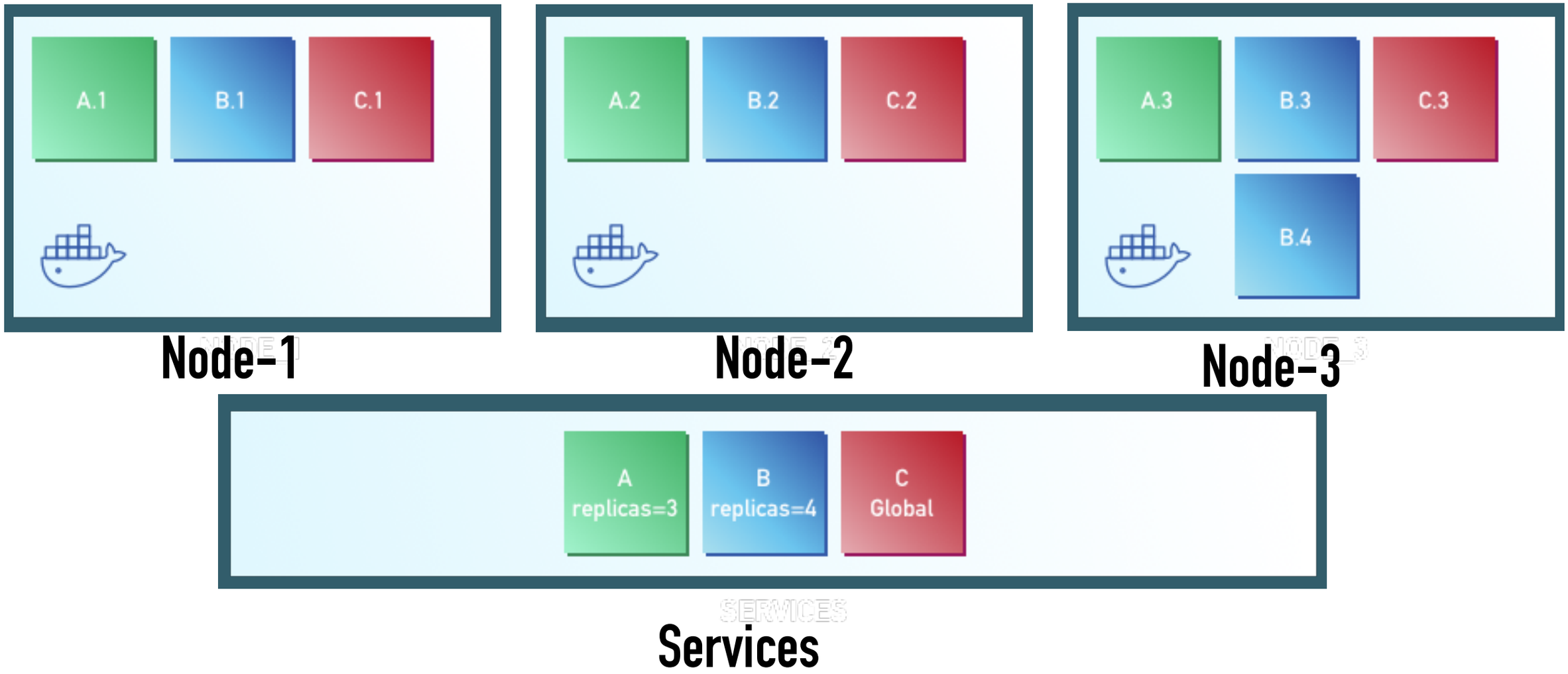
```
$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
s62wr6wohmis        web                 replicated          3/3                 nginx:1.17          *:8080->80/tcp
```

Scalability in SWARM



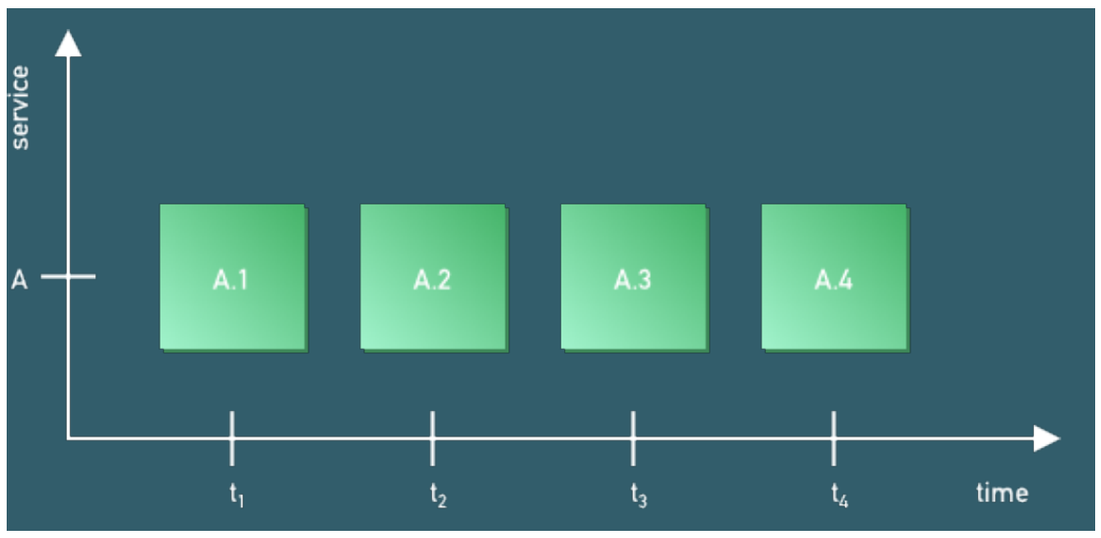
Replicas =3

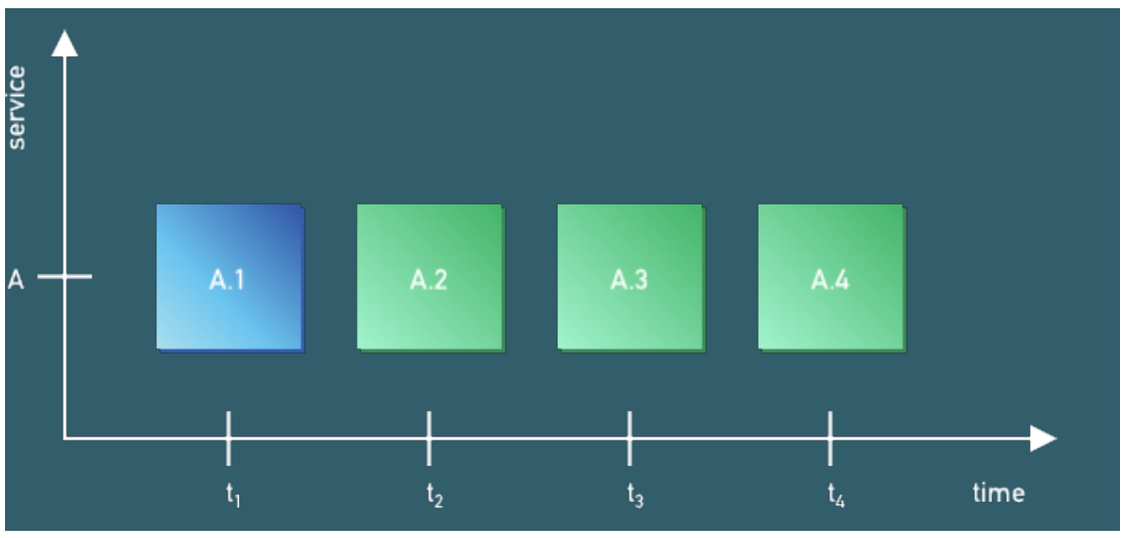
Global and replicated services



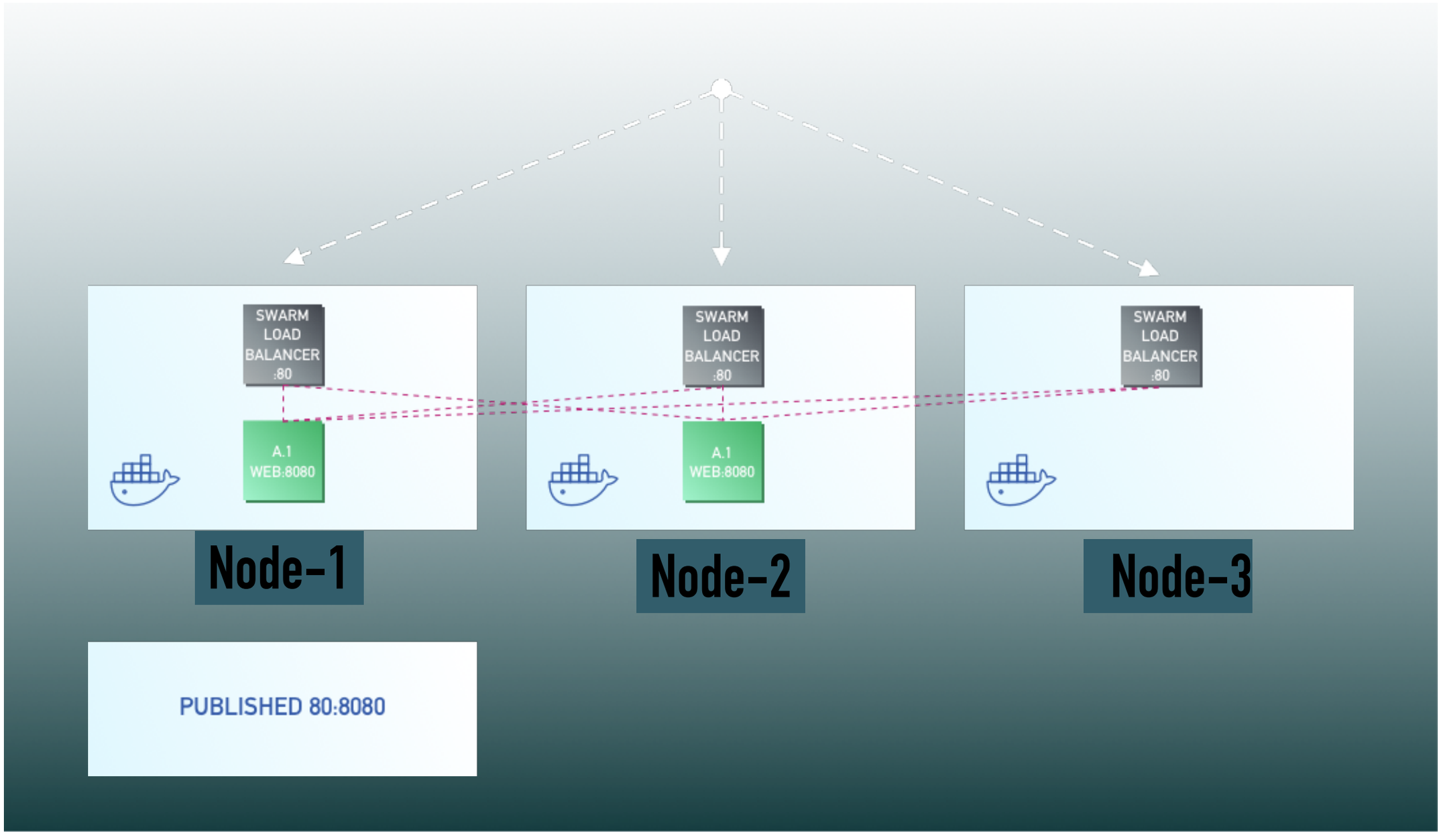
Rolling updates

- Applying update incrementally with defined delay
- When an update return state RUNNING, the schedule schedule to update another task.
- If at any time a task return FAILED, the scheduler pause the update.





Routing mesh



Docker Secrets

- Managing sensitive data as a blob of data e.g. ssh-key, API keys, SSL certificates and keys.

- Encrypted by default

- Accessible to those services which have been granted access to it and only while those tasks services are up and running.

- Inside a container a secret is available on tmpfs at /run/secrets/my_key

```
$ printf "This is a secret" | docker secret create my_secret_data -
```

Docker config

- Store non-sensitive data such as configuration files outside a container image.

- Configs operate in similar way to secret expect they are not encrypted.

- ${NGINX_CONFIG:-1}

```
$ cat nginx.conf | docker config create nginx.conf -
```

More features…

- Secure by default, TLS encryption
- Constraints
- Placement preferences
- Load balancing: DNSRR, VIP
- Service Discovery, internal DNS server
- Multi host networking, overlay network with encryption (IPSEC)
- Stacks with multiple services - compose file
- Environment variables

TRAEFIK key features

TRAEFIK key features

- Modern HTTP reverse proxy and load balancer for micro services working with Docker Swarm and Kubernetes - just need to point Traefik into your orchestrator

- Continuously update configuration, no restarts required

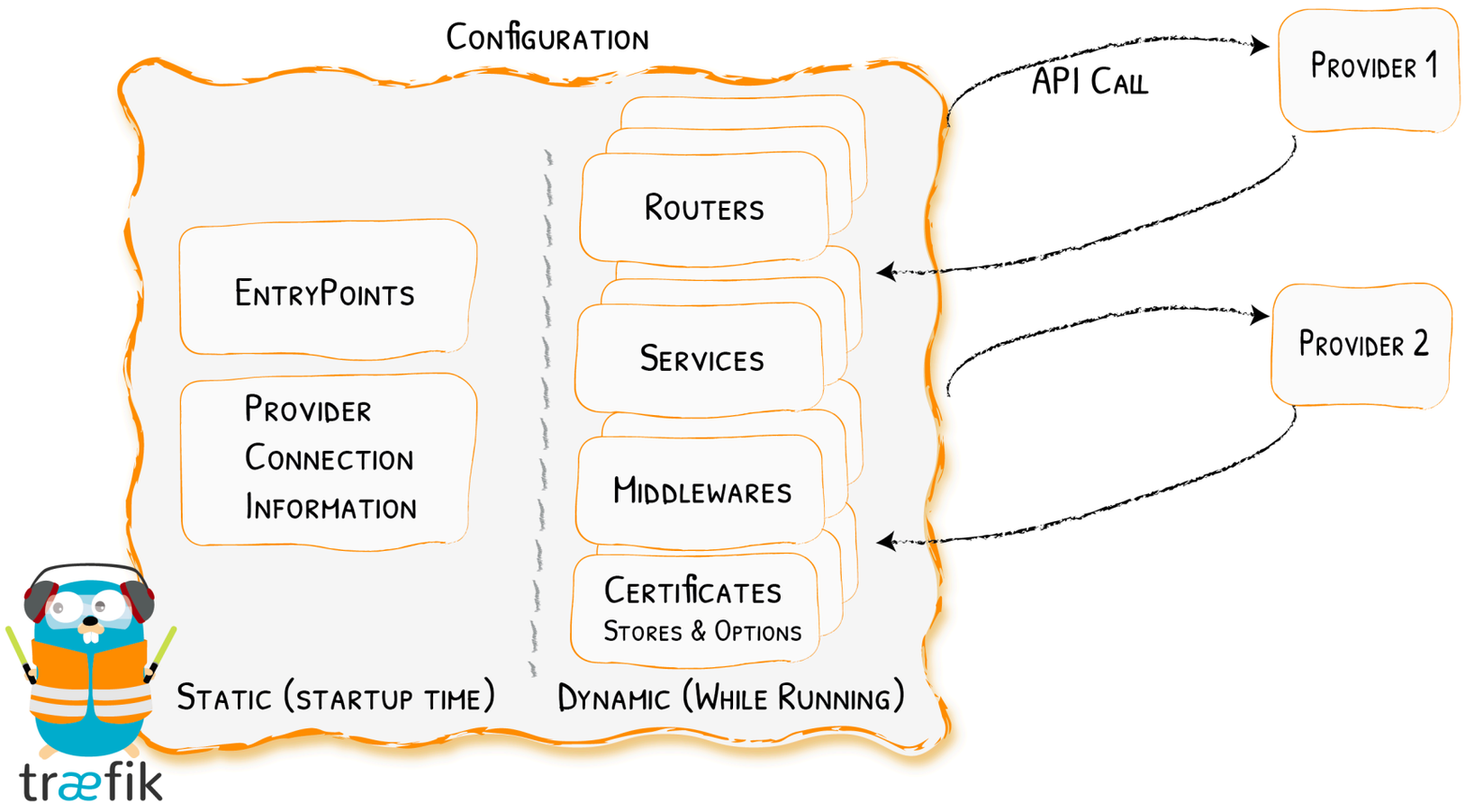
- HTTP and TCP services

- Integration with Let’s Encrypt!

- Immutable image deployed on a cluster

- Configuration done via Docker labels on a service level

Traefik configuration introduction



Source of the image: https://docs.traefik.io/getting-started/configuration-overview/

Demo environment

Demo environment in details

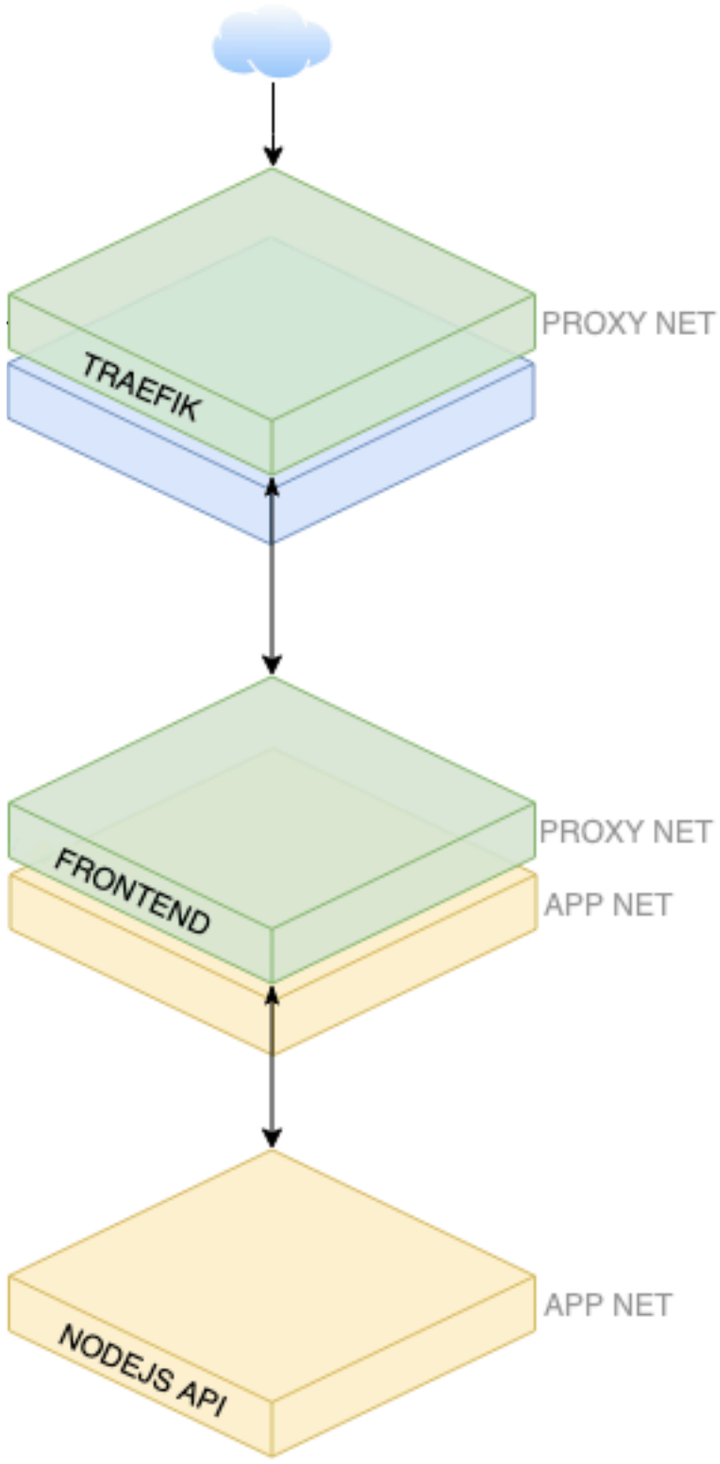
- Running on 4 virtual machines accessible from Internet,
- VM's have already installed the latest version of Docker
- Swarm cluster is already created
- The stacks files are deployed: Traefik with and App stack
- DNS round robin for each domains used in the demo
- Route 53 (AWS) manages health-checks to update DNS records in case of failure of any VMs
- Domains:
    - node-app.labs.cometari.eu
    - traefik.labs.cometari.eu
    - viz.labs.cometari.eu

Demo environment in details

- Docker overlay networks
- SSL Certificates issued by Let's Encrypt and managed via TRAEFIK 2.0
- **TRAEFIK Edge router** exposes service to the Internet
- Nginx with Frontend app connected to NodeJS backend (optional service: might be responsible for caching static content)

Diagram of demo environment

Diagram of the architecture from the service perspective



SHOW ME THE CODE!

Overview of STACK FILES!

Summary

- Fully fledged example of multi tier application stacks
- Straightforward configuration, everything is code!
- Immutable images, custom configuration managed by injecting environment variables
- Easy way to scale up / down a service
- Automatically exposed new services thanks to TRAEFIK, no need to prepare manually a config files
- Managing of SSL certs is managed in an easy way!
- Infrastructure as code!

Links

- The source code is available at https://github.com/jakubhajek/traefik-swarm
- The official Traefik documentation: https://docs.traefik.io/v2.0/
- The official Docker Swarm documentation: https://docs.docker.com/engine/swarm/
- Compose file version 3 references: https://docs.docker.com/compose/compose-file/

Thank You!

Jakub Hajek

Cometari

@_jakubhajek

kuba@cometari.com