Traefik on docker - lightweight alternative to kubernetes

Stefan Ruf
Full Stack Developer @ Inxmail

Socials • steve192

Where I started

- Containers are cool
- Microservices are cool
- No experience with container orchestration
- Start-containers.sh
  - docker run
  - apache reverse proxy
  - lets encrypt manually

Where I started

```
#!/bin/sh
sudo docker stop gitlab-ce-docker
sudo docker rm gitlab-ce-docker
sudo docker run -d --name gitlab-ce-docker
--restart always \
  -p 30001:443 \
  -p 30000:80 \
  -p 5006:5005 \
  --stop-timeout 60 \
  --cpu-shares 50 \
  --memory=4096m \
  --link docker-webservice:docker-webservice \
  --link gitlab-runner:gitlab-runner \
  -v /volume2/gitlab/config:/etc/gitlab \
  -v /volume2/gitlab/data:/var/opt/gitlab \
  --health-start-period=240s \
  gitlab/gitlab-ce:latest
```

```
<VirtualHost *:443>
    ServerName app.localhost
    ProxyPreserveHost On
    ProxyRequests Off
    ProxyVia On
    ProxyPass / http://127.0.0.1:5006/
    ProxyPassReverse / http://127.0.0.1:5006/
</VirtualHost>
```

- Lots of manual setup and annoying to change
- → Ok i need something else.

Kubernetes!

Creating a cluster with kubeadm

Using `kubeadm`, you can create a minimum viable Kubernetes cluster that conforms to best practices. In fact, you can use `kubeadm` to set up a cluster that will pass the Kubernetes Conformance tests. `kubeadm` also supports other cluster lifecycle functions, such as bootstrap tokens and cluster upgrades.

The `kubeadm` tool is good if you need:

- A simple way for you to try out Kubernetes, possibly for the first time.
- A way for existing users to automate setting up a cluster and test their application.
- A building block in other ecosystem and/or installer tools with a larger scope.

You can install and use `kubeadm` on various machines: your laptop, a set of cloud servers, a Raspberry Pi, and more. Whether you're deploying into the cloud or on-premises, you can integrate `kubeadm` into provisioning systems such as Ansible or Terraform.

Professionally (!) installing kubernetes

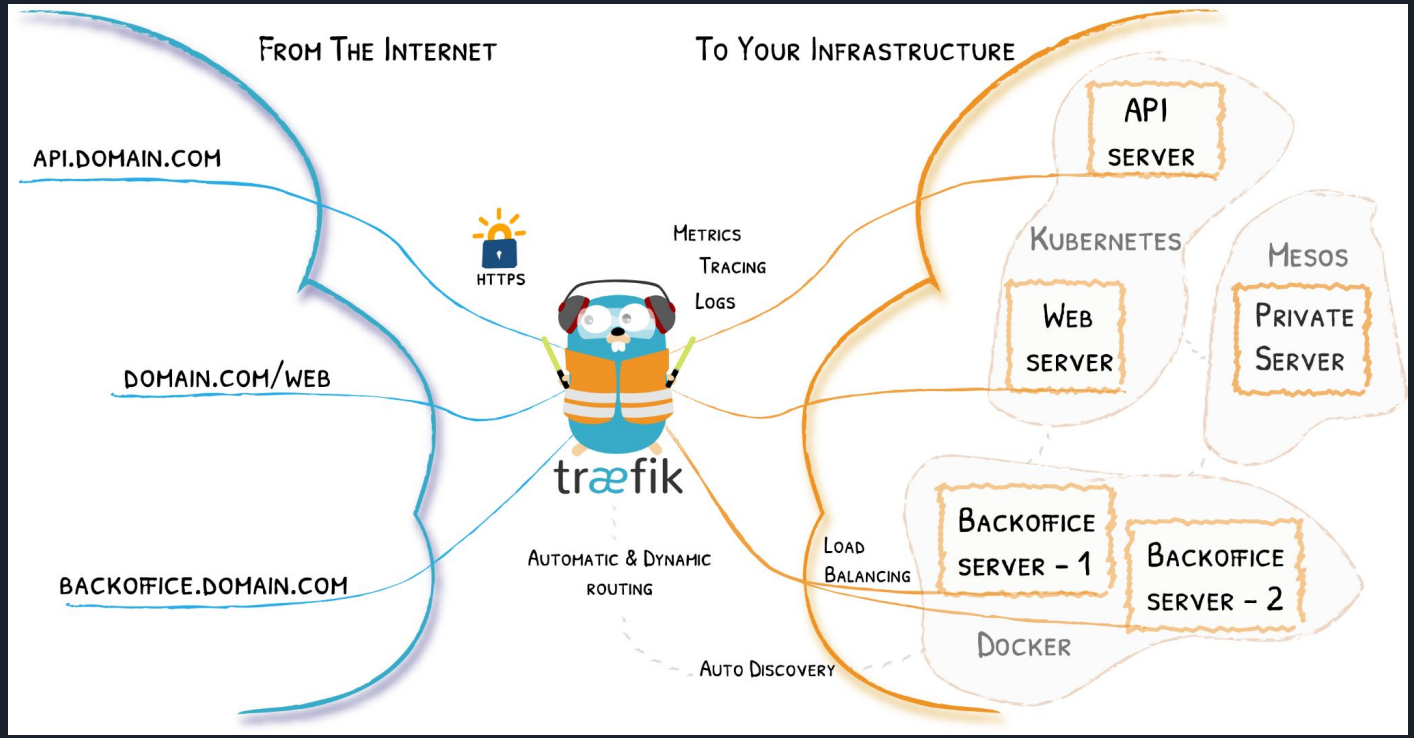
Revolutionary idea after 1 night of frustration

Maybe start slow and small?

What I actually needed

- Define and run containers
- Make containers available to the outside world as subdomains
  - Without complicated configuration and manual steps
- Password-protect some services
- SSL Certificates (without manual steps)
- Add security

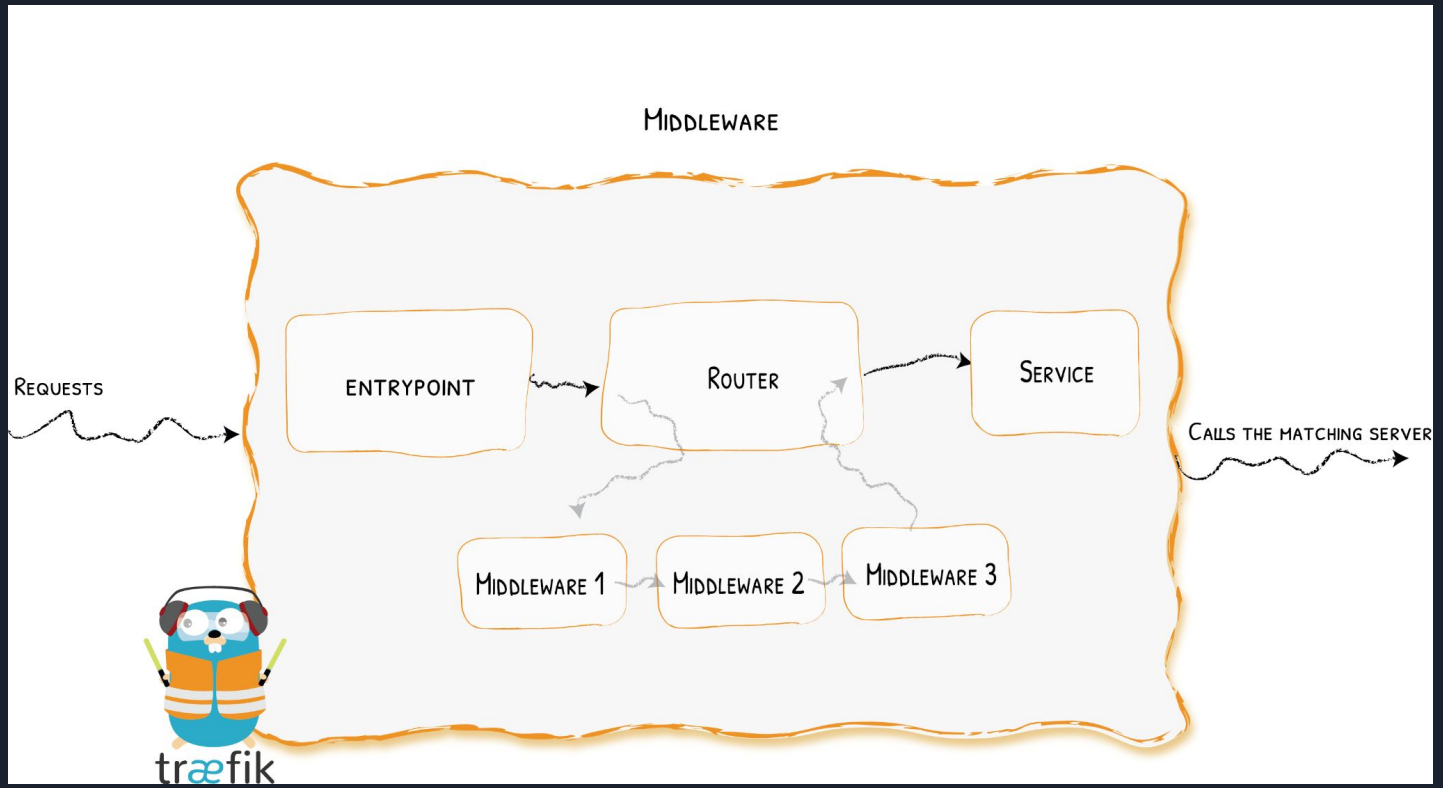
Traefik


From The Internet
To Your Infrastructure
API.DOMAIN.COM
DOMAIN.COM/WEB
BACKOFFICE.DOMAIN.COM
HTTPS
Metrics
Tracing
Logs
træfik
Automatic & Dynamic routing
Load Balancing
Auto Discovery
API server
Kubernetes
Web server
Mesos
Private Server
Backoffice server - 1
Backoffice server - 2
Docker

Install traefik by docker compose

```yaml
services:
  traefik:
    image: traefik:v2.10
    # Enables the web UI and tells Traefik to listen to docker
    command: --api.insecure=true --providers.docker
    # All services must to on the same network
    networks:
      - traefik
    ports:
      # The HTTP port
      - "80:80"
      # The Web UI (enabled by --api.insecure=true)
      - "8080:8080"
    volumes:
      # So that Traefik can listen to the Docker events
      - /var/run/docker.sock:/var/run/docker.sock

networks:
  traefik:
    name: traefik
```

Concepts of traefik



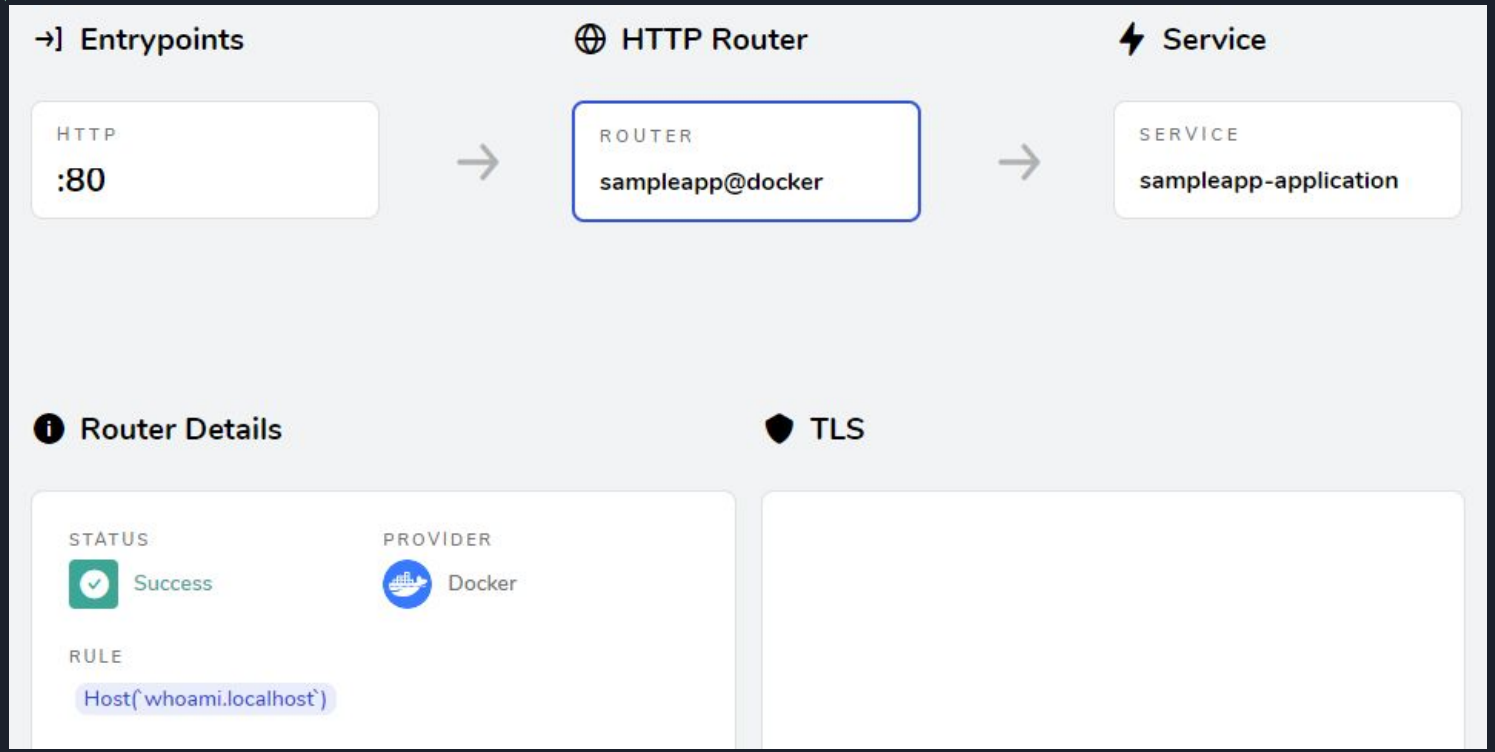
What I actually needed

- Define and run containers
- Make containers available to the outside world as subdomains
  - Without complicated configuration and manual steps
- Password-protect some services
- SSL Certificates (without manual steps)
- Add security

Setup an application with traefik

```yaml
services:
  sampleapp:
    image: dockersamples/static-site
    networks:
      - traefik
    labels:
      - "traefik.http.routers.sampleapp.rule=Host(`app.localhost`)"
#       - "traefik.http.services.sampleapp.loadbalancer.server.port=80"

networks:
  traefik:
    external: true
    name: traefik
```

What traefik interpreted



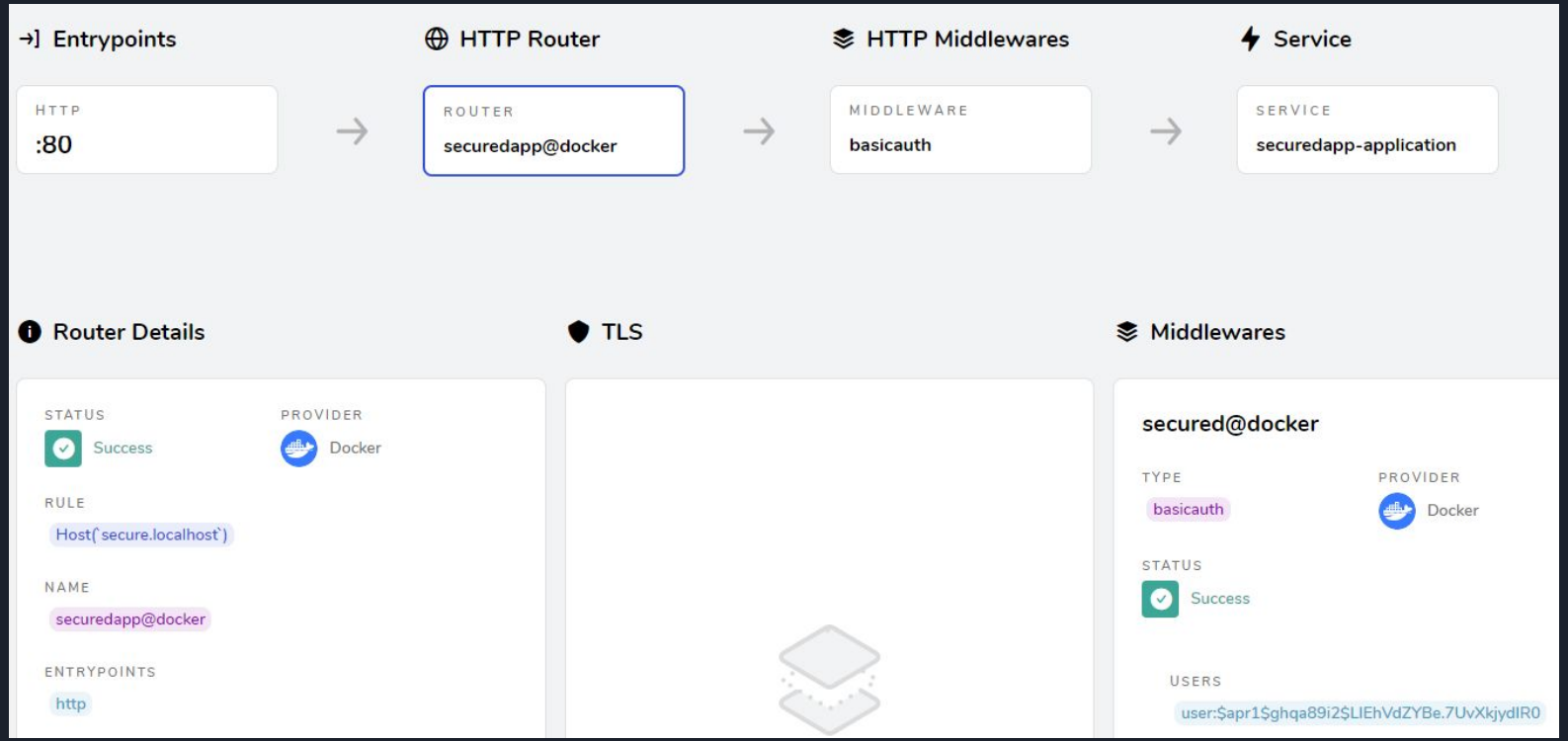
What I actually needed

- Define and run containers
- Make containers available to the outside world as subdomains
  - Without complicated configuration and manual steps
- Password-protect some services
- SSL Certificates (without manual steps)
- Add security

Use a middleware to secure a service

```
services:
    securedapp:
        image:  dockersamples/static-site
        networks:
          - traefik
        labels:
          - "traefik.http.routers.securedapp.rule=Host(`secure.localhost`)"
          - "traefik.http.routers.securedapp.middlewares=secured"
            # user:password
          - "traefik.http.middlewares.secured.basicauth.users=user:$$apr1$$ghqa89i2$$LIEhVdZYBe.7UvXkjydIR0"
```

Use a middleware to secure a service



What I actually needed

- Define and run containers
- Make containers available to the outside world as subdomains
  - Without complicated configuration and manual steps
- Password-protect some services
- SSL Certificates (without manual steps)
- Add security

Certificate resolvers

```yaml
certificatesResolvers:
  letsencrypt:
    acme:
      email: you@example.com
      storage: acme.json
      keyType: EC384
      httpChallenge:
        entryPoint: web
```

```yaml
entryPoints:
  web:
    address: :80
    http:
      redirections:
        entryPoint:
          to: websecure

  websecure:
    address: :443
    http:
      tls:
        certResolver: letsencrypt
```

What I actually needed

- Define and run containers
- Make containers available to the outside world as subdomains
  - Without complicated configuration and manual steps
- Password-protect some services
- SSL Certificates (without manual steps)
- Add security

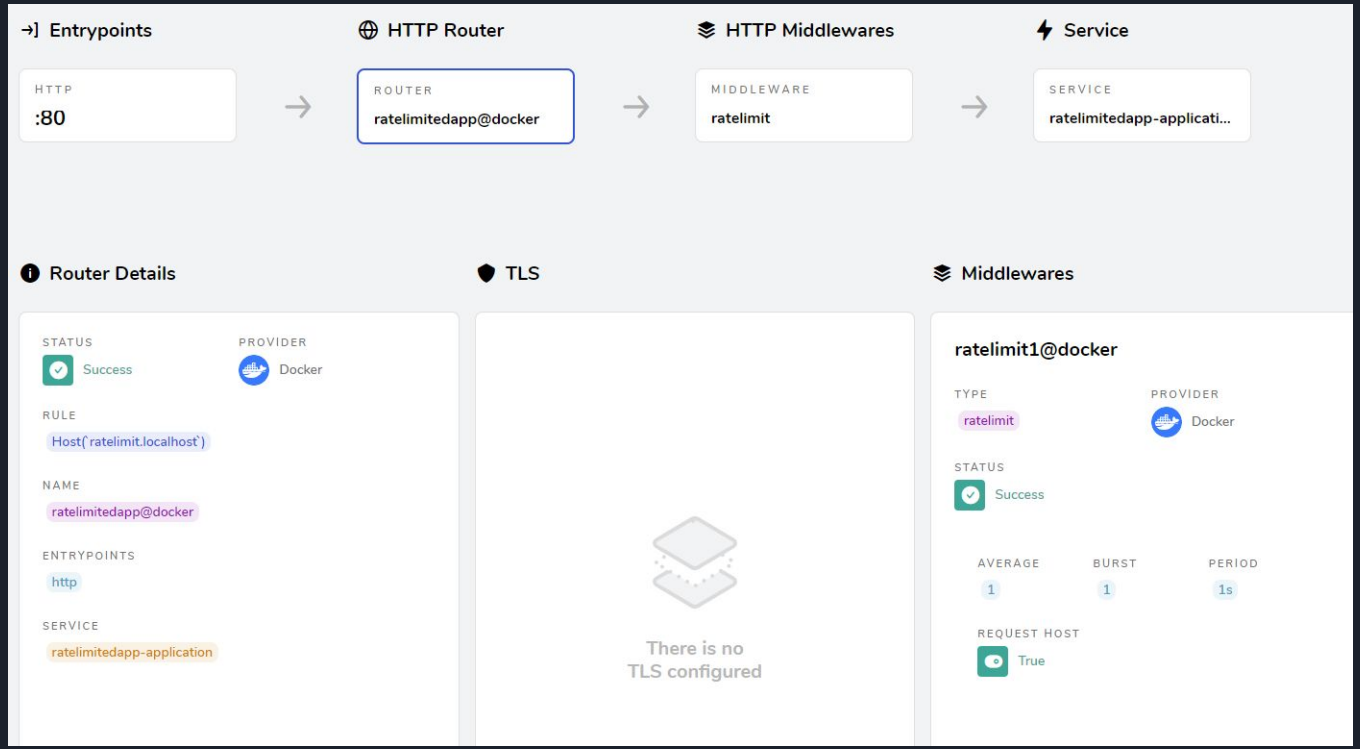
Security headers

```
labels:
      - "traefik.http.routers.securedapp.rule=Host(`secure.localhost`)"
      - "traefik.http.routers.securedapp.middlewares=secureHeaders"
        # Strict Transport Security
      - "treafik.http.middlewares.secureHeaders.forceSTSHeader=true"
      - "treafik.http.middlewares.secureHeaders.stsIncludeSubdomains=true"
        # Disallow embedding in iframe
      - "treafik.http.middlewares.secureHeaders.frameDeny=true"
        # CORS
      - "traefik.http.middlewares.secureHeaders.headers.accesscontrolallowmethods=GET,OPTIONS,PUT"
      - "traefik.http.middlewares.secureHeaders.headers.accesscontrolallowheaders=*"
      - "traefik.http.middlewares.secureHeaders.headers.accesscontrolalloworiginlist=http://secure.localhost"
      - "traefik.http.middlewares.secureHeaders.headers.accesscontrolmaxage=100"
      - "traefik.http.middlewares.secureHeaders.headers.addvaryheader=true"
```

Use a middleware to rate-limit a service

```yaml
ratelimitedapp:
  image: dockersamples/static-site
  networks:
    - traefik
  labels:
    - "traefik.http.routers.ratelimitedapp.rule=Host(`ratelimit.localhost`)"
    - "traefik.http.routers.ratelimitedapp.middlewares=ratelimit1"
    - "traefik.http.middlewares.ratelimit1.ratelimit.average=1"
    - "traefik.http.middlewares.ratelimit1.ratelimit.burst=1"
```

Use a middleware to rate-limit a service



Use a middleware to rate-limit a service

```
ratelimitedapp:
  image: dockersamples/static-site
  networks:
    - traefik
  labels:
      - "traefik.http.routers.ratelimitedapp.rule=Host(`ratelimit.localhost`)"
      - "traefik.http.routers.ratelimitedapp.priority=1"
      - "traefik.http.routers.ratelimitedapp-api.rule=Host(`ratelimit.localhost`) && Path(`/api`)"
      # Makes everything match first router and Path /api match second router with rate limit
      - "traefik.http.routers.ratelimitedapp-api.priority=2"
      - "traefik.http.routers.ratelimitedapp-api.middlewares=ratelimit1"
      - "traefik.http.middlewares.ratelimit1.ratelimit.average=1"
      - "traefik.http.middlewares.ratelimit1.ratelimit.burst=1"
```

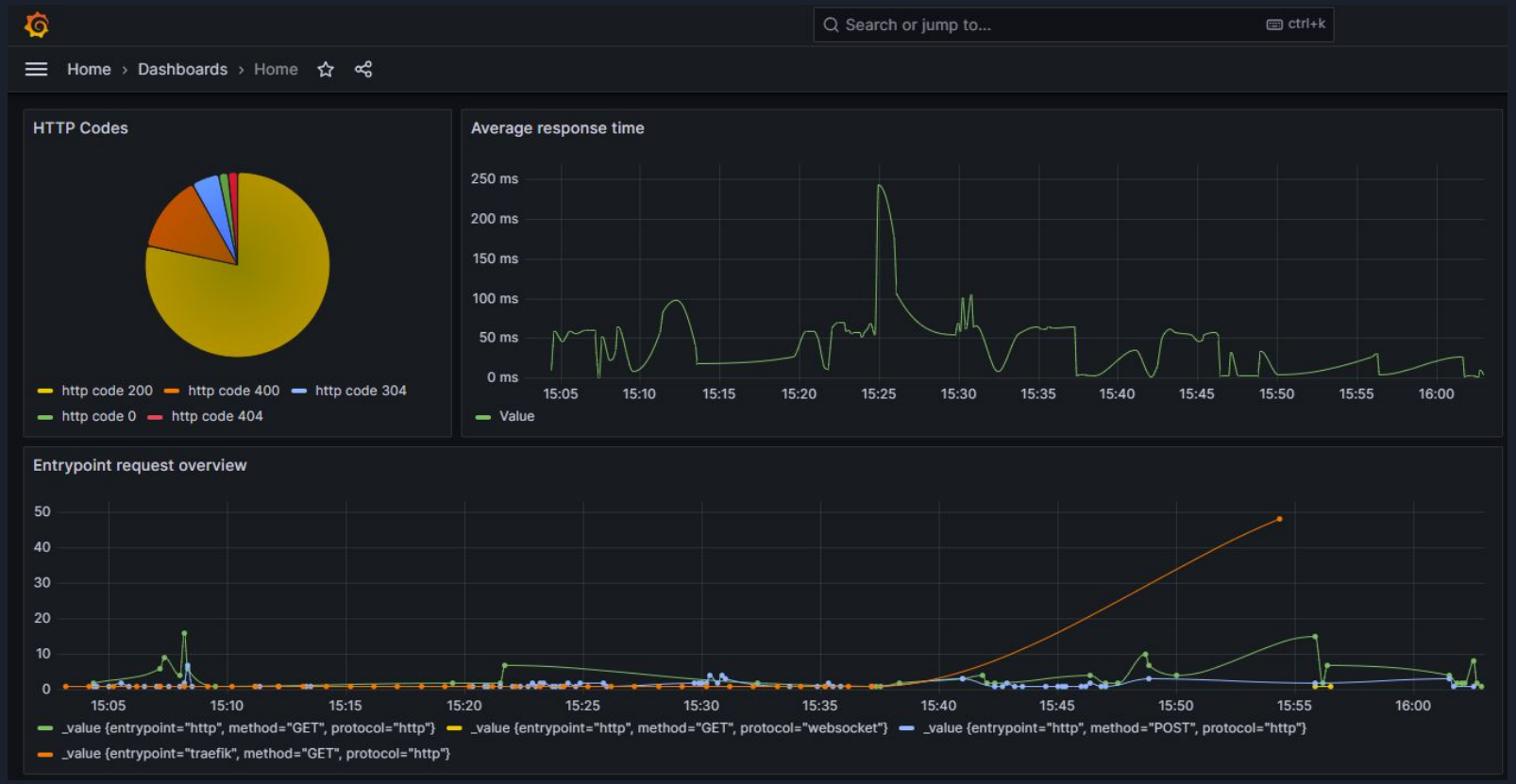
Add metrics

- Example: Use influxdb to store metrics, grafana to display

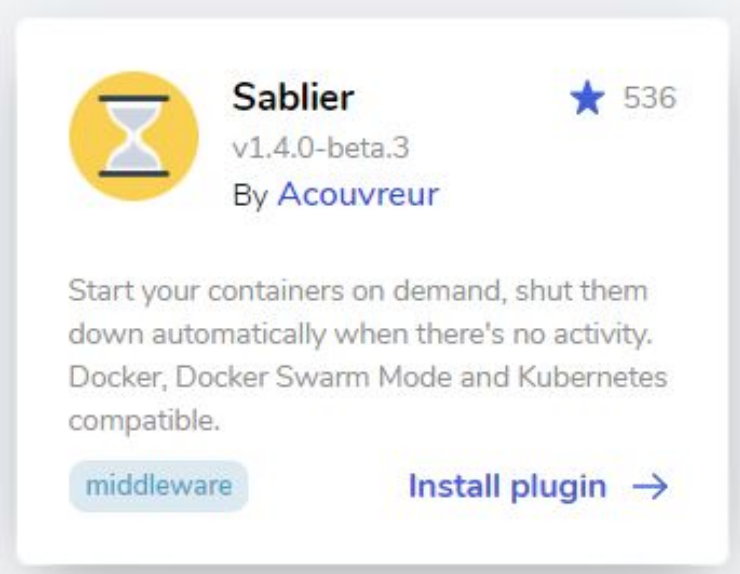
```
grafana:
    image: grafana/grafana:9.5.6
    networks:
      - traefik
    depends_on:
      - influxdb
[...]

influxdb:
    image: influxdb:2.7-alpine
    networks:
      - traefik
[...]
```

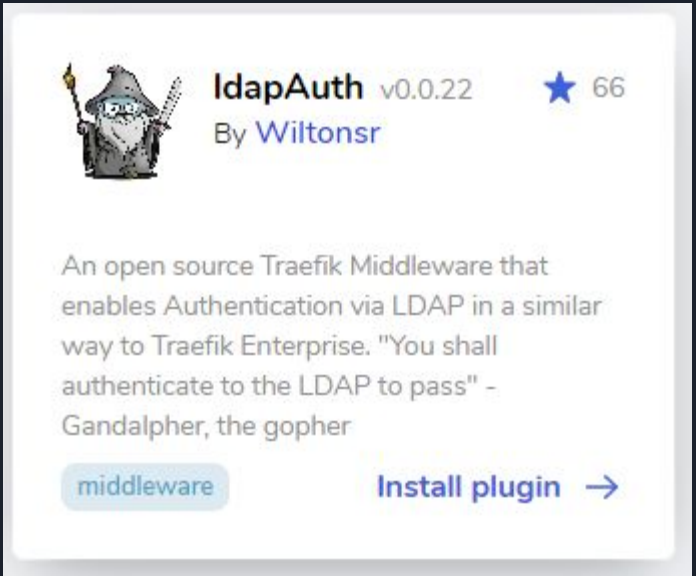
```
traefik:
    # The official v2 Traefik docker image
    image: traefik:v2.10
    command: --api.insecure=true
      --providers.docker
      --metrics.influxDB2.address=http://influxdb:8086
      --metrics.influxDB2.token=password
      --metrics.influxDB2.org=org
      --metrics.influxDB2.bucket=bucket
      --metrics.influxDB2.addEntryPointsLabels=true
      --metrics.influxDB2.addRoutersLabels=true
      --metrics.influxDB2.addServicesLabels=true
```

Add metrics


Search or jump to...
ctrl+k
Home › Dashboards › Home
HTTP Codes
http code 200
http code 400
http code 304
http code 0
http code 404
Average response time
250 ms
200 ms
150 ms
100 ms
50 ms
0 ms
15:05
15:10
15:15
15:20
15:25
15:30
15:35
15:40
15:45
15:50
15:55
16:00
Value
Entrypoint request overview
50
40
30
20
10
0
15:05
15:10
15:15
15:20
15:25
15:30
15:35
15:40
15:45
15:50
15:55
16:00
_value {entrypoint="http", method="GET", protocol="http"}
_value {entrypoint="http", method="GET", protocol="websocket"}
_value {entrypoint="http", method="POST", protocol="http"}
_value {entrypoint="traefik", method="GET", protocol="http"}

Extend traefik with plugins

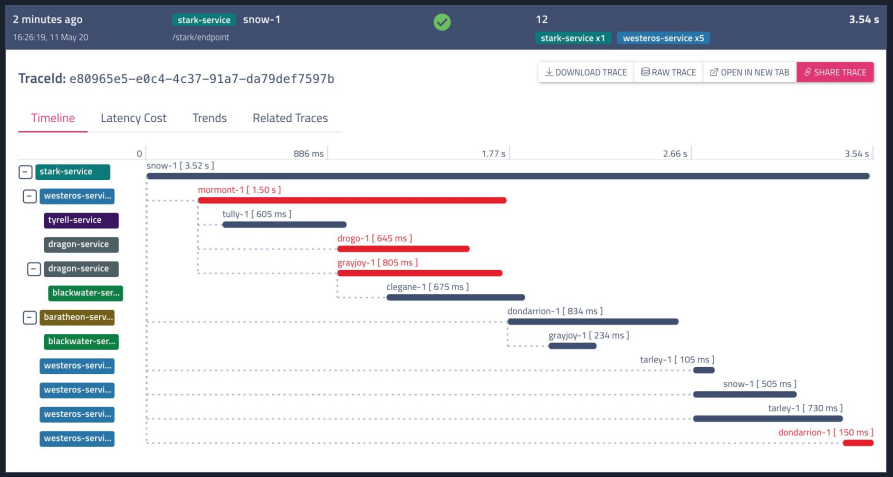




More traefik features

Tracing

- Uses open tracing to visualize call flows in infrastructure (e.g. elastic tracing)



Access logs

- Log requests including headers
- Filter based on status
- Optional JSON format

Make docker more kubernetesy

- Watchtower (https://hub.docker.com/r/containrrr/watchtower)
  - Periodically pulls new container images
- Autoheal (https://hub.docker.com/r/willfarrell/autoheal)
  - Re-creates unhealthy containers
- Try docker swarm if you need multiple nodes

Take aways

- Traefik+Docker can do what most entry users want from kubernetes
- Don’t directly jump on kubernetes without thinking what you really need
    - May be applied to other tech trends as well

Slides and code available on github

https://github.com/steve192/presentations/tree/main/traefik-on-docker