Reverse-Proxy using Traefik & Docker Container- Assignment-10

What is a Reverse Proxy?

A Reverse Proxy is a server that receives incoming client requests and forwards them to the appropriate backend service (like your frontend or backend app). It hides the internal structure and helps manage multiple apps behind a single domain.

Use Cases:

- Route requests to different services (e.g., /api to backend, / to frontend)

- SSL Termination

- Load Balancing

Reverse Proxy in AWS Context

In AWS, frontend and backend are often deployed on the same EC2 instance but on different ports (e.g., 3000 and 5173). Instead of typing port numbers, we want to route using:

- http://docker.localhost → shows frontend

- http://docker.localhost/api → hits backend

To achieve this:

- Bind a domain (or local alias) to your EC2 public IP

- Use a reverse proxy like Traefik or Nginx

Step-by-Step: Set Up Docker Server on Ubuntu EC2

Create EC2 (Ubuntu + Docker):

1. Launch Ubuntu EC2

2. Choose Ubuntu 22.04 LTS

3. Open ports 22, 80, 3000, 5173

4. SSH: ssh -i your-key.pem ubuntu@<your-public-ip>

Install Docker:

- sudo su

- apt update

- apt install docker.io -y

- systemctl start docker

- systemctl enable docker

Edit /etc/hosts:

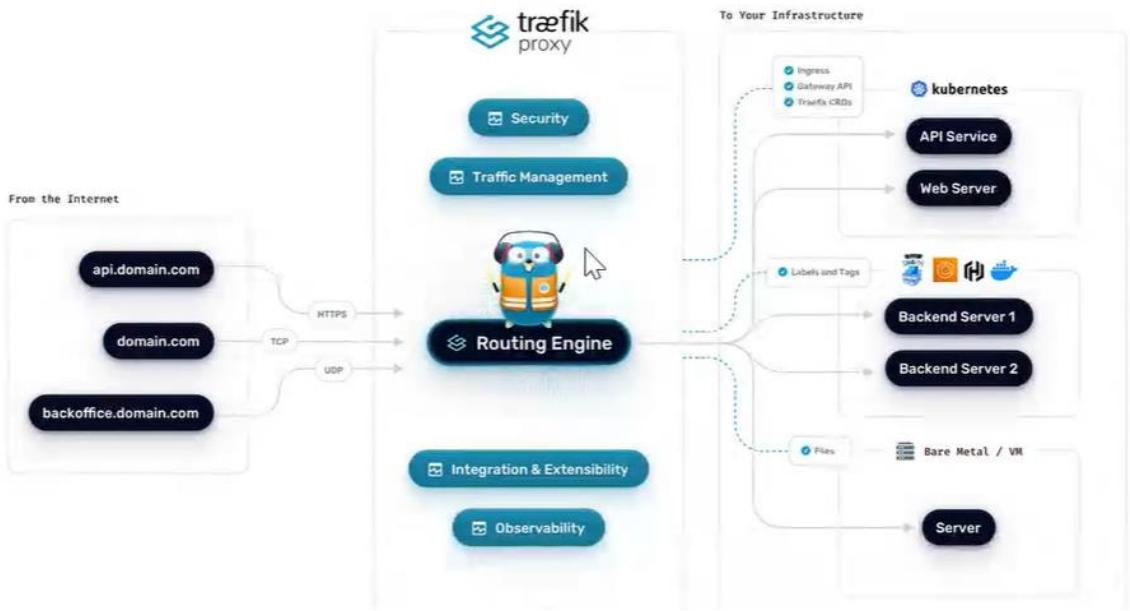
- Add '<your-ec2-ip> docker.localhost'

- Save and reboot instance

- Reconnect via SSH.

What is Traefik?

Traefik is a reverse proxy and load balancer for Docker. It detects running containers and routes traffic automatically.



Key Features:

- Dynamic Routing

- Let's Encrypt (SSL)

- Dashboard & metrics

- Easier config than Nginx in Docker

Traefik Setup

1. Create traefik.yml file:

```
providers:
  docker:
    defaultRule: "Host(`{{ trimPrefix `/` .Name }}.docker.localhost`)"
api:
  insecure: true
```

2. Run Traefik:

```
docker run -d -p 8080:8080 -p 80:80 \
  -v $PWD/traefik.yml:/etc/traefik/traefik.yml \
  -v /var/run/docker.sock:/var/run/docker.sock \
  traefik:v3
```

3. Allow port 8080 in AWS security group.

4. Open http://<your-ec2-ip>:8080 to access dashboard.

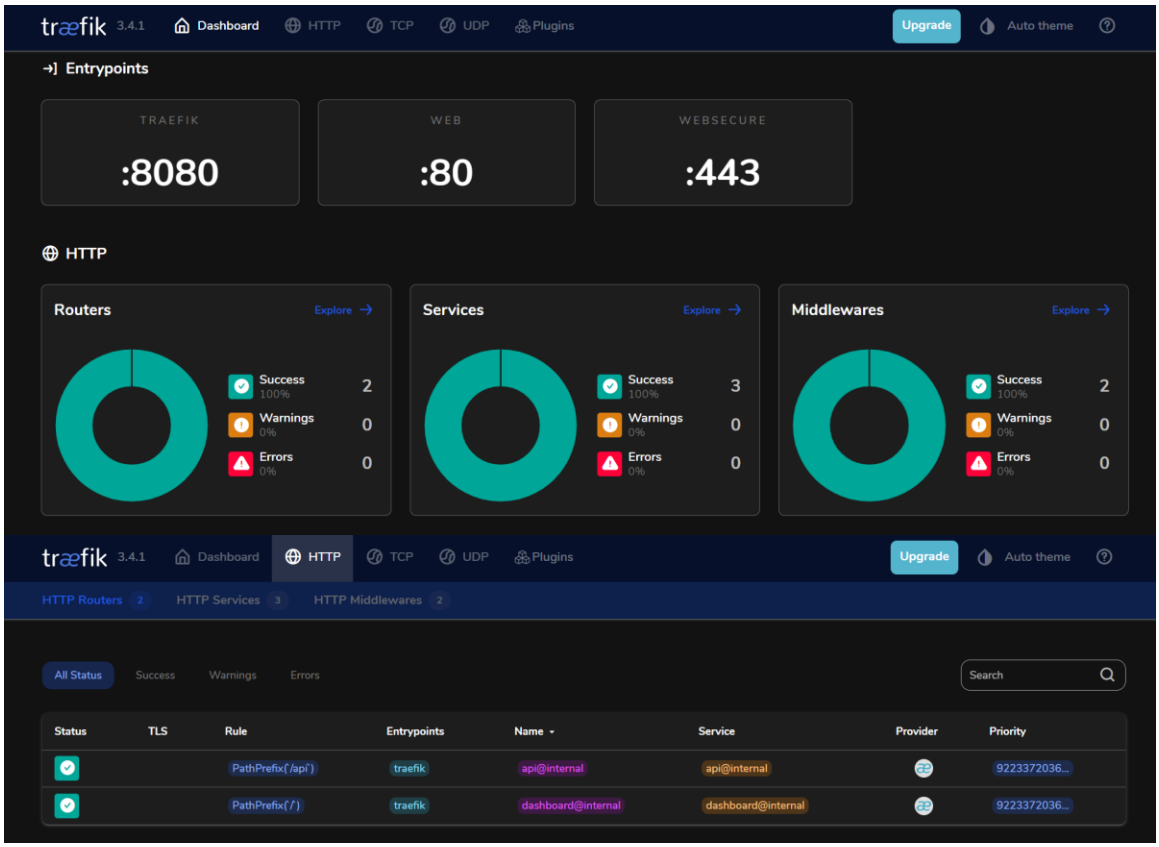
5. Run sample containers:

```
docker run -it -d --name web-server1 nginx
docker run -it -d --name web-server2 nginx
docker run -it -d --name web-server3 httpd
```

```

6. Use curl or browser to test responses.

```
127.0.0.1 localhost
13.201.128.14 docker.localhost

The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

```
# Docker configuration backend
providers:
  docker:
    defaultRule: "Host(`{{ trimPrefix / .Name }}.docker.localhost`)"

# API and dashboard configuration
api:
  insecure: true
```

```
root@ip-172-31-10-253:/home/ubuntu/traefik# docker run -d -p 8080:8080 -p 80:80 \
  -v "$PWD/traefik.yml:/etc/traefik/traefik.yml" \
  -v /var/run/docker.sock:/var/run/docker.sock \
  traefik:v3
Unable to find image 'traefik:v3' locally
v3: Pulling from library/traefik
f18232174bc9: Pull complete
22ff649176b0: Pull complete
6bb594ddbc12: Pull complete
64c9803dbe74: Pull complete
Digest: sha256:cd40ab7bc1f047731d5b22595203812343efcb6538014c4e93221cfc3a77217a
Status: Downloaded newer image for traefik:v3
2736fa268c6438dbc80155bb5103bf30c8688a7122a407eb0bb46a0439f80691
root@ip-172-31-10-253:/home/ubuntu/traefik# docker container ls
CONTAINER ID   IMAGE      COMMAND             CREATED       STATUS       PORTS                                                                                  NAM
ES
3e42d2888905   traefik:v3   "/entrypoint.sh trae…"   8 seconds ago   Up 8 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   lau
ghing_mayer
```
```



Assignment: Reverse Proxy Using NGINX

1. Run:

```

docker run -it -d --name web1 -p 8081:80 nginx

docker run -it -d --name web2 -p 8082:80 httpd

```

2. Create default.conf:

```

server {

 listen 80;

 location /web1 {

  proxy_pass http://host.docker.internal:8081;

 }
```

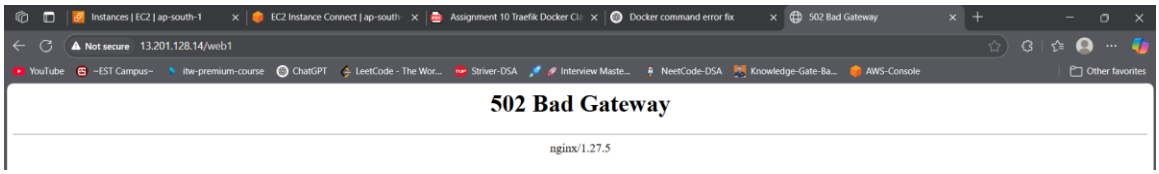
```
 location /web2 {
   proxy_pass http://host.docker.internal:8082;
 }
}
```

3. Run NGINX:

```
docker run -d -p 80:80 \
 -v $PWD/default.conf:/etc/nginx/conf.d/default.conf \
 --name nginx-proxy nginx
```

4. Test:

- http://<public-ip>/web1
- http://<public-ip>/web2



Docker Compose Setup

1. Cleanup:

docker container rm -f $(docker ps -aq)

docker rmi -f $(docker images -aq)

2. Clone repo:

git clone https://github.com/sibasish934/docker-compose-example.git

cd docker-compose-example

3. Edit frontend/index.html with EC2 IP.

4. Install Compose:

apt install docker-compose -y

5. Launch:

docker-compose up --build

6. Open:

- http://<ec2-ip>:3000

- Backend runs on port 5000