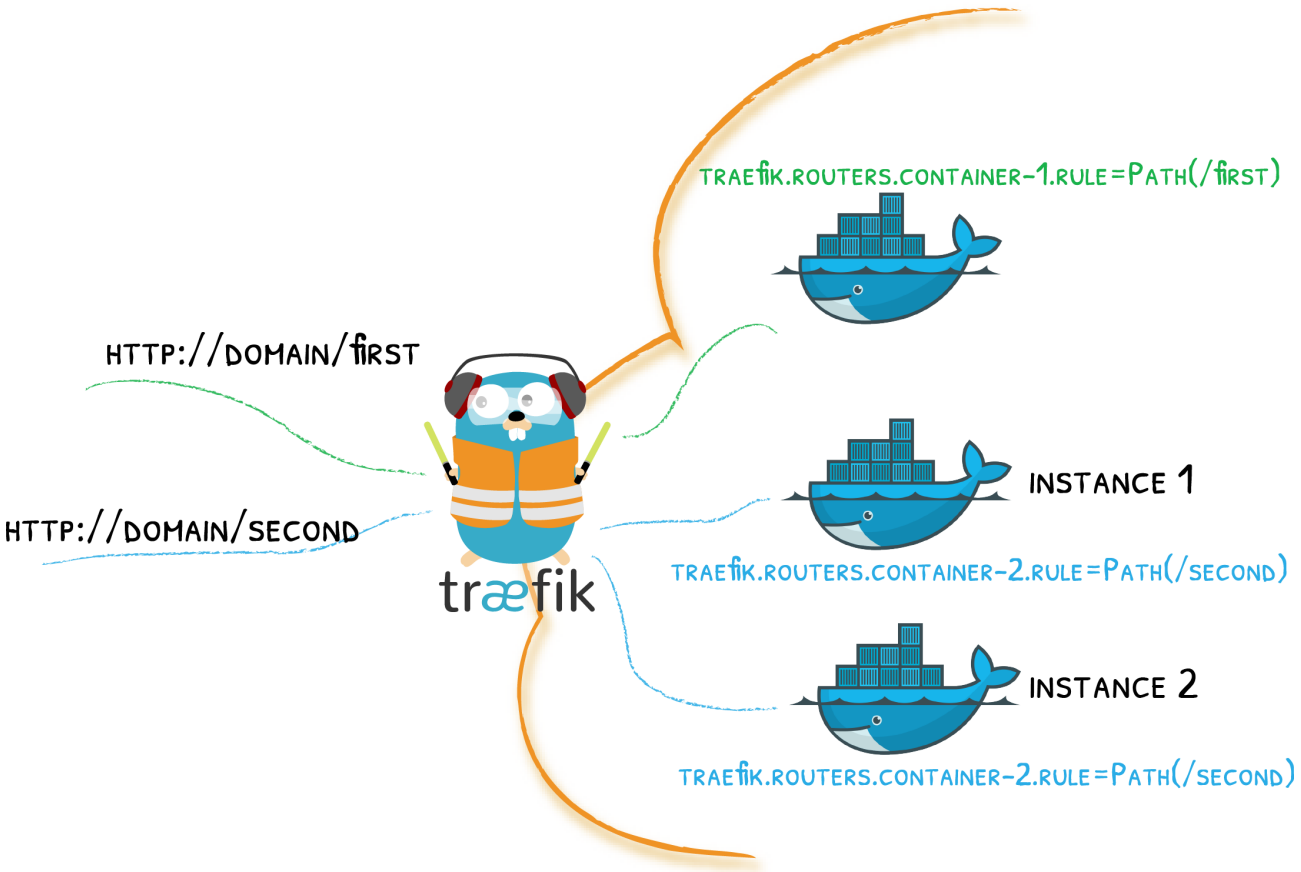
Beginner-Guide: Traefik & Docker Standalone Engine



Step 1: Install Docker

Before setting up Traefik, make sure Docker is installed on your system. You can follow My Dockerengine Guide for instructions on installing Docker Engine.

Step 2: Setting Up Traefik

Create a `docker-compose.yml` file to define the Traefik service and its basic configuration.
Example `docker-compose.yml`:

```
version: "3.7"

services:
  traefik:
    image: traefik:v3.0
    container_name: traefik
    command:
      - "--api.insecure=true"  # Enable Traefik dashboard (insecure mode, disable in production)
      - "--providers.docker=true"  # Enable Docker as a provider
      - "--entrypoints.web.address=:80"  # Define HTTP EntryPoint
      - "--entrypoints.websecure.address=:443"  # Define HTTPS EntryPoint
      - "--certificatesresolvers.myresolver.acme.tlschallenge=true"  # Enable Let's Encrypt TLS challenge
      - "--certificatesresolvers.myresolver.acme.email=your-email@example.com"  # Email for Let's Encrypt
      - "--certificatesresolvers.myresolver.acme.storage=/letsencrypt/acme.json"  # Storage for certificates
    ports:
      - "80:80"       # Expose Traefik on port 80 (HTTP)
      - "443:443"     # Expose Traefik on port 443 (HTTPS)
      - "8080:8080"   # Traefik Dashboard
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock:ro"  # Allow Traefik to access Docker API
      - "./letsencrypt:/letsencrypt"  # Volume to store Let's Encrypt certificates
    networks:
      - traefik-net

networks:
  traefik-net:
    driver: bridge
```

Explanation:

- **Docker Provider (`--providers.docker=true`)**: Enables Traefik to watch Docker containers for routing configuration.
- **EntryPoints**: Define which ports Traefik listens to. We use port 80 for HTTP and port 443 for HTTPS traffic.
- **Let’s Encrypt Configuration**: Automatically issues SSL certificates using the Let’s Encrypt **ACME** protocol.
- **Docker Socket (`/var/run/docker.sock`)**: Traefik uses the Docker socket to monitor container changes and apply routing rules dynamically.
- **Dashboard**: Exposes Traefik’s dashboard on port `8080` for monitoring purposes.

Step 3: Run Traefik

To start Traefik, run the following command in the directory containing the `docker-compose.yml` file:

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

> This command will start Traefik in detached mode. You can check if Traefik is running by visiting `http://localhost:8080`. You should see the Traefik dashboard.

Step 4: Exposing a Service via Traefik

Now that Traefik is up and running, let's deploy a simple service, such as an Nginx container, and configure it to route traffic through Traefik.

Create a new `docker-compose.yml` for the Nginx service:

```
version: "3.7"

services:
  nginx:
    image: nginx
    container_name: nginx
    labels:
      - "traefik.enable=true"  # Enable Traefik routing for this container
      - "traefik.http.routers.nginx.rule=Host(`nginx.local`)"  # Route traffic based on hostname
      - "traefik.http.services.nginx.loadbalancer.server.port=80"  # Nginx runs on port 80
    networks:
      - traefik-net

networks:
  traefik-net:
    external: true
```

Explanation:

- **Labels**: Labels are key to telling Traefik how to route traffic.
  - `traefik.enable=true`: Enables Traefik for this service.
  - `traefik.http.routers.nginx.rule=Host('nginx.local')`: Configures a routing rule where requests with the `Host` header `nginx.local` will be routed to this service.
  - `traefik.http.services.nginx.loadbalancer.server.port=80`: Defines the port where Nginx listens for incoming traffic.

- **Shared Network**: The `nginx` service needs to be in the same Docker network as Traefik to allow communication.

Run the following command to deploy the Nginx service:

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

At this point, Traefik should automatically detect the Nginx service and route traffic based on the hostname `nginx.local`. To test this setup locally, you can add the following line to your `/etc/hosts` file:

```
127.0.0.1 nginx.local
```

> Now, visiting `http://nginx.local` should display the Nginx default welcome page.

Step 5: Routing Configuration with Labels

In Traefik, **labels** define how traffic is routed to services. These labels are attached to the Docker containers and can configure anything from simple routing rules to advanced load balancing configurations.

Here are some useful labels you can apply to your containers:

- **Basic Routing**:
  - `traefik.enable=true`: Enable routing for the container.
  - `traefik.http.routers.<router_name>.rule=Host('example.com')`: Routes requests to `example.com` to the container.
- **Load Balancing**:
  - `traefik.http.services.<service_name>.loadbalancer.server.port=<port>`: Defines the internal port where the container listens.
  - `traefik.http.services.<service_name>.loadbalancer.sticky=true`: Enable sticky sessions to ensure a user connects to the same container during a session.
- **Middleware (for modifying requests/responses)**:
  - `traefik.http.middlewares.<middleware_name>.addPrefix.prefix=/api`: Adds `/api` to the beginning of every request.
  - `traefik.http.routers.<router_name>.middlewares=<middleware_name>`: Applies middleware to the router.
- **SSL/TLS**:
  - `traefik.http.routers.<router_name>.tls=true`: Enables TLS for a router (required for HTTPS).

- `traefik.http.routers.<router_name>.tls.certresolver=myresolver`: Uses the Let's Encrypt resolver to obtain SSL certificates.

Port Detection

By default, Traefik automatically detects which port to use based on the ports exposed by the Docker container:

- **Single Port**: If the container exposes only one port, Traefik will use it.
- **Multiple Ports**: If multiple ports are exposed, Traefik will select the lowest numbered port. For example, if ports `80` and `8080` are exposed, Traefik will select port `80`.

If Traefik cannot determine the correct port, you can manually define the port using the label:

```
labels:
  - "traefik.http.services.my-service.loadbalancer.server.port=8080"
```

Security Considerations

When running Traefik with Docker, there are a few important security considerations:

1. **Docker API Access**: Traefik requires access to the Docker socket (`/var/run/docker.sock`) to monitor container events and retrieve routing configuration. This can expose your Docker environment to potential security risks. Ensure only trusted services have access to the Docker API.
2. **TLS/SSL**: Enable SSL certificates for your services by configuring Let's Encrypt or using your own certificate. Traefik's **ACME** integration with Let's Encrypt allows for automatic certificate management.
3. **Secure the Dashboard**: By default, the Traefik dashboard is exposed on port `8080` in insecure mode. In production environments, disable the insecure API or secure the dashboard with authentication.

Conclusion

Traefik makes it incredibly easy to manage traffic and load balancing for your Docker containers. With its dynamic service discovery, routing rules based on labels, and automatic SSL management, Traefik can simplify your Docker environment and reduce the complexity of traditional reverse proxy setups.

By following this guide, you should have a basic understanding of how to set up Traefik with Docker, configure routing with labels, and expose services dynamically. As you gain more experience, you can explore advanced features such as middlewares, sticky sessions, and custom SSL configurations.

Traefik is a powerful tool for anyone looking to manage traffic in a Docker environment, whether you're running a few containers or managing a large-scale microservices architecture.

Revision #4
Created 12 September 2024 14:31:27 by aeoneros
Updated 12 January 2025 11:26:08 by aeoneros