Traefik Reverse Proxy

I use docker compose to host mysql and nodejs services. I want to use traefik as a reverse proxy / dns resolver to the nodejs applications.
This is my docker compose file:
version: "3.8"

```
services:
  db_configuration:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/configuration_data/mysql:/var/lib/mysql
      - ./databases/configuration_data/scripts:/docker-entrypoint-initdb.d/
    ports:
      - "3307:3306"
    deploy:
      mode: global

  db_sensor:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/sensor_data/mysql:/var/lib/mysql
      - ./databases/sensor_data/scripts:/docker-entrypoint-initdb.d/
    ports:
      - "3308:3306"
    deploy:
      mode: global

  db_user:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/user/mysql:/var/lib/mysql
      - ./databases/user/scripts:/docker-entrypoint-initdb.d/
    ports:
      - "3309:3306"
    deploy:
      mode: global
```

```
  node_auth:
    image: node:22-alpine3.18
    environment:
      SERVER_PORT: 80
      DB_PASSWORD_USER: /run/secrets/db_pwd_user
      ACCESS_TOKEN_SECRET: /run/secrets/node_auth_access_token
      REFRESH_TOKEN_SECRET: /run/secrets/node_auth_refresh_token
    secrets:
      - db_pwd_user
      - node_auth_access_token
      - node_auth_refresh_token
    depends_on:
      - db_user
    working_dir: /app
    command: "node app.js"
    volumes:
      - ./node/authentication/:/app
    ports:
      - "81:80"
    deploy:
      mode: global

  node_rest:
    image: node:22-alpine3.18
    environment:
      SERVER_PORT: 80
      DB_PASSWORD_CONFIGURATION: /run/secrets/db_pwd_configuration
      DB_PASSWORD_SENSOR: /run/secrets/db_pwd_sensor
    secrets:
      - db_pwd_configuration
      - db_pwd_sensor
    depends_on:
      - db_configuration
      - db_sensor
      - node_auth
    working_dir: /app
    command: "node app.js"
    volumes:
      - ./node/rest/:/app
    ports:
      - "80:80"
    deploy:
      mode: global

secrets:
  db_pwd_configuration:
    external: true
  db_pwd_sensor:
    external: true
  db_pwd_user:
```

```
  external: true
node_auth_access_token:
  external: true
node_auth_refresh_token:
  external: true
```

To use Traefik as a reverse proxy and DNS resolver for your Node.js applications in your Docker Compose setup, you'll need to add the Traefik service to your `**docker-compose.yml**` file. Traefik will handle routing traffic to your `**node_auth**` and `**node_rest**` services based on the domain names you specify.

Here's an updated version of your `**docker-compose.yml**` file that includes the Traefik service configuration:

```yaml
version: "3.8"

services:
  traefik:
    image: traefik:v2.10
    command:
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--entrypoints.web.address=:80"
    ports:
      - "8080:8080" # Traefik Dashboard
      - "80:80"     # Web traffic
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock:ro"
    deploy:
      mode: global

  db_configuration:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/configuration_data/mysql:/var/lib/mysql
      - ./databases/configuration_data/scripts:/docker-entrypoint-initdb.d/
    ports:
      - "3307:3306"
    deploy:
      mode: global

  db_sensor:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/sensor_data/mysql:/var/lib/mysql
      - ./databases/sensor_data/scripts:/docker-entrypoint-initdb.d/
    ports:
      - "3308:3306"
    deploy:
      mode: global

  db_user:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/user/mysql:/var/lib/mysql
      - ./databases/user/scripts:/docker-entrypoint-initdb.d/
```

```
    ports:
      - "3309:3306"
    deploy:
      mode: global

  node_auth:
    image: node:22-alpine3.18
    environment:
      SERVER_PORT: 80
      DB_PASSWORD_USER: /run/secrets/db_pwd_user
      ACCESS_TOKEN_SECRET: /run/secrets/node_auth_access_token
      REFRESH_TOKEN_SECRET: /run/secrets/node_auth_refresh_token
    secrets:
      - db_pwd_user
      - node_auth_access_token
      - node_auth_refresh_token
    depends_on:
      - db_user
    working_dir: /app
    command: "node app.js"
    volumes:
      - ./node/authentication/:/app
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.node_auth.rule=Host(`auth.localhost`)"
      - "traefik.http.services.node_auth.loadbalancer.server.port=80"
    deploy:
      mode: global

  node_rest:
    image: node:22-alpine3.18
    environment:
      SERVER_PORT: 80
      DB_PASSWORD_CONFIGURATION: /run/secrets/db_pwd_configuration
      DB_PASSWORD_SENSOR: /run/secrets/db_pwd_sensor
    secrets:
      - db_pwd_configuration
      - db_pwd_sensor
    depends_on:
      - db_configuration
      - db_sensor
      - node_auth
    working_dir: /app
    command: "node app.js"
    volumes:
      - ./node/rest/:/app
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.node_rest.rule=Host(`rest.localhost`)"
      - "traefik.http.services.node_rest.loadbalancer.server.port=80"
    deploy:
      mode: global

secrets:
  db_pwd_configuration:
    external: true
  db_pwd_sensor:
    external: true
  db_pwd_user:
    external: true
  node_auth_access_token:
    external: true
  node_auth_refresh_token:
    external: true
```

Explanation

1. **Traefik Service**:
    - **Image**: `` `traefik:v2.10` `` specifies the Traefik version.

- **Command**: Configures Traefik to enable the API, use Docker as a provider, and listen on port 80 for web traffic.
- **Ports**: Maps port 8080 for the Traefik dashboard and port 80 for web traffic.
- **Volumes**: Mounts the Docker socket to allow Traefik to interact with Docker.

2. **Node Services (`node_auth` and `node_rest`)**:
   - **Labels**: These labels configure Traefik to route traffic to these services.
     - `traefik.enable=true` enables Traefik for the service.
     - `traefik.http.routers.<service_name>.rule=Host(\`<hostname>`)` defines the hostname rule for routing.
     - `traefik.http.services.<service_name>.loadbalancer.server.port=80` specifies the internal port the service is listening on.
3. **MySQL Services**: No changes were necessary for the MySQL services.

With this setup, you can access `node_auth` via `http://auth.localhost` and `node_rest` via `http://rest.localhost`. Adjust the hostnames as needed to match your DNS configuration.

how do i access the 8080 from traefik

To access the Traefik dashboard on port 8080, you need to ensure that your Docker Compose file maps the Traefik dashboard port to the host machine. The current configuration in the Docker Compose file does this by mapping port 8080 of the Traefik container to port 8080 on the host machine.

Here's the relevant section of the `docker-compose.yml` file:

```yaml
services:
  traefik:
    image: traefik:v2.10
    command:
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--entrypoints.web.address=:80"
    ports:
      - "8080:8080" # Traefik Dashboard
      - "80:80"     # Web traffic
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock:ro"
    deploy:
      mode: global
```

Steps to Access the Traefik Dashboard

1. **Start Your Docker Compose Stack**: Ensure your Docker Compose stack is running by using the following command:

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

2. **Access the Dashboard**: Open your web browser and navigate to `http://localhost:8080`. This should bring up the Traefik dashboard.

Security Considerations

Using `--api.insecure=true` exposes the Traefik dashboard without any authentication, which is not recommended for production environments. For a more secure setup, you should configure authentication for the Traefik dashboard.

Here's an example of how to secure the Traefik dashboard using basic authentication:

1. **Generate a htpasswd File**: Generate a basic authentication password using a tool like `htpasswd`. If you don't have `htpasswd` installed, you can generate the password using an online tool or use Docker to run `htpasswd`:

   ```
   sh

   docker run --rm --entrypoint htpasswd httpd:2 -Bbn <username> <password> > traefik.htpasswd
   ```

2. **Update the Docker Compose File**: Mount the `traefik.htpasswd` file into the Traefik container and configure Traefik to use it for dashboard authentication.

   ```
   yaml

   services:
     traefik:
       image: traefik:v2.10
       command:
         - "--api.dashboard=true"
         - "--api.debug=true"
         - "--providers.docker=true"
         - "--entrypoints.web.address=:80"
         - "--entrypoints.websecure.address=:443"
         - "--entrypoints.websecure.http.tls=true"
         - "--entrypoints.traefik.address=:8080"
         - "--accesslog=true"
         - "--log.level=DEBUG"
       ports:
         - "80:80"     # Web traffic
         - "8080:8080" # Traefik Dashboard
       volumes:
         - "/var/run/docker.sock:/var/run/docker.sock:ro"
         - "./traefik.htpasswd:/etc/traefik/.htpasswd"
       labels:
         - "traefik.http.routers.api.rule=Host(`traefik.localhost`)"
         - "traefik.http.routers.api.service=api@internal"
         - "traefik.http.middlewares.auth.basicauth.usersfile=/etc/traefik/.htpasswd"
         - "traefik.http.routers.api.middlewares=auth"
       deploy:
         mode: global
   ```

3. **Access the Dashboard Securely**: Restart your Docker Compose stack:

   ```
   sh

   docker-compose up -d
   ```

   Access the Traefik dashboard at `http://localhost:8080` and use the username and password you set in the htpasswd file.

By following these steps, you will have a secure and functional Traefik dashboard accessible on port 8080.

the official documentation says something from a whoami service from traefik. What is that? What is it used fro?

The `whoami` service provided by Traefik is a simple web application that returns information about the incoming request. It's typically used for testing and debugging purposes to verify that the Traefik configuration is working correctly. When you access the `whoami` service through Traefik, it returns details such as the hostname, IP address, and HTTP headers of the request.

Here's a step-by-step guide on how to add and use the `whoami` service in your Docker Compose setup:

Step 1: Add the `whoami` Service to Your Docker Compose File

Extend your `docker-compose.yml` to include the `whoami` service:

```yaml
version: "3.8"

services:
  traefik:
    image: traefik:v2.10
    command:
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--entrypoints.web.address=:80"
    ports:
      - "8080:8080" # Traefik Dashboard
      - "80:80"     # Web traffic
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock:ro"
    deploy:
      mode: global

  whoami:
    image: traefik/whoami
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.whoami.rule=Host(`whoami.localhost`)"
      - "traefik.http.services.whoami.loadbalancer.server.port=80"
    deploy:
      mode: global

  db_configuration:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/configuration_data/mysql:/var/lib/mysql
      - ./databases/configuration_data/scripts:/docker-entrypoint-initdb.d/
    ports:
      - "3307:3306"
    deploy:
      mode: global

  db_sensor:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/sensor_data/mysql:/var/lib/mysql
      - ./databases/sensor_data/scripts:/docker-entrypoint-initdb.d/
    ports:
```

```
      - "3308:3306"
    deploy:
      mode: global

  db_user:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/user/mysql:/var/lib/mysql
      - ./databases/user/scripts:/docker-entrypoint-initdb.d/
    ports:
      - "3309:3306"
    deploy:
      mode: global

  node_auth:
    image: node:22-alpine3.18
    environment:
      SERVER_PORT: 80
      DB_PASSWORD_USER: /run/secrets/db_pwd_user
      ACCESS_TOKEN_SECRET: /run/secrets/node_auth_access_token
      REFRESH_TOKEN_SECRET: /run/secrets/node_auth_refresh_token
    secrets:
      - db_pwd_user
      - node_auth_access_token
      - node_auth_refresh_token
    depends_on:
      - db_user
    working_dir: /app
    command: "node app.js"
    volumes:
      - ./node/authentication/:/app
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.node_auth.rule=Host(`auth.localhost`)"
      - "traefik.http.services.node_auth.loadbalancer.server.port=80"
    deploy:
      mode: global

  node_rest:
    image: node:22-alpine3.18
    environment:
      SERVER_PORT: 80
      DB_PASSWORD_CONFIGURATION: /run/secrets/db_pwd_configuration
      DB_PASSWORD_SENSOR: /run/secrets/db_pwd_sensor
    secrets:
      - db_pwd_configuration
      - db_pwd_sensor
    depends_on:
      - db_configuration
      - db_sensor
      - node_auth
    working_dir: /app
    command: "node app.js"
    volumes:
      - ./node/rest/:/app
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.node_rest.rule=Host(`rest.localhost`)"
      - "traefik.http.services.node_rest.loadbalancer.server.port=80"
    deploy:
      mode: global

secrets:
  db_pwd_configuration:
    external: true
  db_pwd_sensor:
    external: true
  db_pwd_user:
    external: true
  node_auth_access_token:
    external: true
```

```
    node_auth_refresh_token:
      external: true
```

Step 2: Start Your Docker Compose Stack

Run the following command to start your Docker Compose stack:

```
sh

docker-compose up -d
```

Step 3: Access the `whoami` Service

Open your web browser and navigate to `http://whoami.localhost`. You should see a page displaying information about the request, such as the hostname, IP address, and HTTP headers.

Explanation of Labels for `whoami` Service

- `traefik.enable=true`: Enables Traefik for this service.
- `traefik.http.routers.whoami.rule=Host(\`whoami.localhost`)`: Defines a routing rule that directs traffic to this service based on the `whoami.localhost` hostname.
- `traefik.http.services.whoami.loadbalancer.server.port=80`: Specifies the port that Traefik should use to connect to the `whoami` service inside the container.

By following these steps, you can test and verify your Traefik configuration using the `whoami` service, ensuring that Traefik is correctly routing traffic to your services based on the defined rules.

so whoami is just for debugging?

Yes, the `whoami` service is primarily used for debugging and testing purposes in a Traefik setup. It provides a simple way to verify that your Traefik configuration is correctly routing traffic to your services. By sending requests to the `whoami` service, you can see detailed information about the incoming request, which helps in ensuring that the routing rules, load balancing, and other configurations are working as expected.

Typical Use Cases for the `whoami` Service

1. **Verify Routing Rules**: Ensure that your Traefik routing rules are correctly directing traffic to the intended service.
2. **Check Load Balancing**: If you have multiple instances of a service, you can verify that Traefik is load balancing the requests appropriately.
3. **Inspect Headers**: Examine the HTTP headers that Traefik passes to your service. This can be useful for debugging issues related to headers, such as authentication tokens or custom headers.
4. **Confirm SSL/TLS Configuration**: If you are using SSL/TLS with Traefik, you can use the `whoami` service to verify that the SSL certificates are correctly applied and that the service is accessible over HTTPS.

Example Scenario

Suppose you have set up Traefik to route traffic to multiple Node.js services based on different hostnames. You can use the `whoami` service to test each route by accessing the corresponding URLs. For example:

- Access `http://auth.localhost` to verify that traffic is routed to the `node_auth` service.
- Access `http://rest.localhost` to verify that traffic is routed to the `node_rest` service.
- Access `http://whoami.localhost` to verify that the routing rule for `whoami` is working and to inspect the request details.

Summary

The `whoami` service is a lightweight, zero-configuration tool that helps you ensure that your Traefik setup is functioning correctly. It's not meant for production use but is invaluable for development and troubleshooting. Once you have verified your configuration using `whoami`, you can confidently deploy your actual services knowing that Traefik is properly configured.

> i want to protect my node_auth and node_rest services with https. So just encrypted https traffic is allowed

To protect your `node_auth` and `node_rest` services with HTTPS using Traefik, you'll need to configure Traefik to handle HTTPS traffic and use SSL/TLS certificates. You can either use Let's Encrypt to automatically generate and manage certificates or provide your own certificates.

Step-by-Step Guide

1. **Update Traefik Configuration for HTTPS**:
    - Enable HTTPS by configuring Traefik to use Let's Encrypt for certificate generation.
    - Redirect HTTP to HTTPS to ensure all traffic is encrypted.
2. **Modify Your Docker Compose File**:
    - Add the necessary Traefik labels to your `node_auth` and `node_rest` services.
    - Ensure that Traefik listens on both HTTP and HTTPS entry points.

Here is an updated version of your `docker-compose.yml` file with these configurations:

```yaml

version: "3.8"

services:
  traefik:
    image: traefik:v2.10
    command:
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--entrypoints.web.address=:80"
      - "--entrypoints.websecure.address=:443"
      - "--certificatesresolvers.myresolver.acme.httpchallenge.entrypoint=web"
      - "--certificatesresolvers.myresolver.acme.httpchallenge=true"
      - "--certificatesresolvers.myresolver.acme.email=youremail@example.com"
      - "--certificatesresolvers.myresolver.acme.storage=/letsencrypt/acme.json"
      - "--entrypoints.web.http.redirections.entryPoint.to=websecure"
      - "--entrypoints.web.http.redirections.entryPoint.scheme=https"
    ports:
```

```
      - "80:80"     # Web traffic (HTTP)
      - "443:443"   # Web secure traffic (HTTPS)
      - "8080:8080" # Traefik Dashboard
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock:ro"
      - "./letsencrypt:/letsencrypt" # Persistent storage for Let's Encrypt certificates
    deploy:
      mode: global

  whoami:
    image: traefik/whoami
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.whoami.rule=Host(`whoami.localhost`)"
      - "traefik.http.services.whoami.loadbalancer.server.port=80"
      - "traefik.http.routers.whoami.entrypoints=web"
    deploy:
      mode: global

  db_configuration:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/configuration_data/mysql:/var/lib/mysql
      - ./databases/configuration_data/scripts:/docker-entrypoint-initdb.d/
    ports:
      - "3307:3306"
    deploy:
      mode: global

  db_sensor:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/sensor_data/mysql:/var/lib/mysql
      - ./databases/sensor_data/scripts:/docker-entrypoint-initdb.d/
    ports:
      - "3308:3306"
    deploy:
      mode: global

  db_user:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/user/mysql:/var/lib/mysql
      - ./databases/user/scripts:/docker-entrypoint-initdb.d/
    ports:
      - "3309:3306"
    deploy:
      mode: global

  node_auth:
    image: node:22-alpine3.18
    environment:
      SERVER_PORT: 80
      DB_PASSWORD_USER: /run/secrets/db_pwd_user
      ACCESS_TOKEN_SECRET: /run/secrets/node_auth_access_token
      REFRESH_TOKEN_SECRET: /run/secrets/node_auth_refresh_token
    secrets:
      - db_pwd_user
      - node_auth_access_token
      - node_auth_refresh_token
    depends_on:
      - db_user
    working_dir: /app
    command: "node app.js"
    volumes:
      - ./node/authentication/:/app
```

```
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.node_auth.rule=Host(`auth.localhost`)"
      - "traefik.http.services.node_auth.loadbalancer.server.port=80"
      - "traefik.http.routers.node_auth.entrypoints=websecure"
      - "traefik.http.routers.node_auth.tls.certresolver=myresolver"
    deploy:
      mode: global

  node_rest:
    image: node:22-alpine3.18
    environment:
      SERVER_PORT: 80
      DB_PASSWORD_CONFIGURATION: /run/secrets/db_pwd_configuration
      DB_PASSWORD_SENSOR: /run/secrets/db_pwd_sensor
    secrets:
      - db_pwd_configuration
      - db_pwd_sensor
    depends_on:
      - db_configuration
      - db_sensor
      - node_auth
    working_dir: /app
    command: "node app.js"
    volumes:
      - ./node/rest/:/app
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.node_rest.rule=Host(`rest.localhost`)"
      - "traefik.http.services.node_rest.loadbalancer.server.port=80"
      - "traefik.http.routers.node_rest.entrypoints=websecure"
      - "traefik.http.routers.node_rest.tls.certresolver=myresolver"
    deploy:
      mode: global

secrets:
  db_pwd_configuration:
    external: true
  db_pwd_sensor:
    external: true
  db_pwd_user:
    external: true
  node_auth_access_token:
    external: true
  node_auth_refresh_token:
    external: true
```

Explanation

1. **Traefik Service**:
   - **Certificates Resolver**:
     - **`--certificatesresolvers.myresolver.acme.httpchallenge.entrypoint=web`**: Uses the HTTP challenge to prove domain ownership.
     - **`--certificatesresolvers.myresolver.acme.httpchallenge=true`**: Enables the HTTP challenge.
     - **`--certificatesresolvers.myresolver.acme.email=youremail@example.com`**: Replace with your email for Let's Encrypt notifications.
     - **`--certificatesresolvers.myresolver.acme.storage=/letsencrypt/acme.json`**: Path to store the Let's Encrypt certificates.
   - **Entrypoints**:
     - **`--entrypoints.web.address=:80`**: Defines the HTTP entry point.
     - **`--entrypoints.websecure.address=:443`**: Defines the HTTPS entry point.
     - **`--entrypoints.web.http.redirections.entryPoint.to=websecure`**: Redirects HTTP traffic to HTTPS.

- `--entrypoints.web.http.redirections.entryPoint.scheme=https`: Specifies the scheme for redirection.

2. **Node Services (`node_auth` and `node_rest`)**:
   - **Labels**:
     - `traefik.http.routers.<service_name>.entrypoints=websecure`: Routes traffic through the HTTPS entry point.
     - `traefik.http.routers.<service_name>.tls.certresolver=myresolver`: Uses the Let's Encrypt resolver for HTTPS certificates.
3. **Volumes**:
   - Adds a volume for Let's Encrypt certificate storage to persist the certificates.

Starting the Services

Run the following command to start your Docker Compose stack:

```
sh

docker-compose up -d
```

Accessing Your Services

- `node_auth` should be accessible at `https://auth.localhost`.
- `node_rest` should be accessible at `https://rest.localhost`.

By following these steps, you'll ensure that your `node_auth` and `node_rest` services are protected with HTTPS, and only encrypted traffic is allowed.

i do not want to use whoami anymore. I want to use let's encrypt

Sure, you can remove the `whoami` service and configure Traefik to use Let's Encrypt for securing your `node_auth` and `node_rest` services with HTTPS. Here's an updated version of your `docker-compose.yml` file with the `whoami` service removed and Let's Encrypt configured.

Updated `docker-compose.yml`

```
yaml

version: "3.8"

services:
  traefik:
    image: traefik:v2.10
    command:
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--entrypoints.web.address=:80"
      - "--entrypoints.websecure.address=:443"
      - "--certificatesresolvers.myresolver.acme.httpchallenge.entrypoint=web"
      - "--certificatesresolvers.myresolver.acme.httpchallenge=true"
      - "--certificatesresolvers.myresolver.acme.email=youremail@example.com"
      - "--certificatesresolvers.myresolver.acme.storage=/letsencrypt/acme.json"
      - "--entrypoints.web.http.redirections.entryPoint.to=websecure"
```

```
      - "--entrypoints.web.http.redirections.entryPoint.scheme=https"
    ports:
      - "80:80"     # Web traffic (HTTP)
      - "443:443"   # Web secure traffic (HTTPS)
      - "8080:8080" # Traefik Dashboard
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock:ro"
      - "./letsencrypt:/letsencrypt" # Persistent storage for Let's Encrypt certificates
    deploy:
      mode: global

  db_configuration:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/configuration_data/mysql:/var/lib/mysql
      - ./databases/configuration_data/scripts:/docker-entrypoint-initdb.d/
    ports:
      - "3307:3306"
    deploy:
      mode: global

  db_sensor:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/sensor_data/mysql:/var/lib/mysql
      - ./databases/sensor_data/scripts:/docker-entrypoint-initdb.d/
    ports:
      - "3308:3306"
    deploy:
      mode: global

  db_user:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/user/mysql:/var/lib/mysql
      - ./databases/user/scripts:/docker-entrypoint-initdb.d/
    ports:
      - "3309:3306"
    deploy:
      mode: global

  node_auth:
    image: node:22-alpine3.18
    environment:
      SERVER_PORT: 80
      DB_PASSWORD_USER: /run/secrets/db_pwd_user
      ACCESS_TOKEN_SECRET: /run/secrets/node_auth_access_token
      REFRESH_TOKEN_SECRET: /run/secrets/node_auth_refresh_token
    secrets:
      - db_pwd_user
      - node_auth_access_token
      - node_auth_refresh_token
    depends_on:
      - db_user
    working_dir: /app
    command: "node app.js"
    volumes:
      - ./node/authentication/:/app
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.node_auth.rule=Host(`auth.localhost`)"
      - "traefik.http.services.node_auth.loadbalancer.server.port=80"
      - "traefik.http.routers.node_auth.entrypoints=websecure"
      - "traefik.http.routers.node_auth.tls.certresolver=myresolver"
    deploy:
      mode: global
```

```
  node_rest:
    image: node:22-alpine3.18
    environment:
      SERVER_PORT: 80
      DB_PASSWORD_CONFIGURATION: /run/secrets/db_pwd_configuration
      DB_PASSWORD_SENSOR: /run/secrets/db_pwd_sensor
    secrets:
      - db_pwd_configuration
      - db_pwd_sensor
    depends_on:
      - db_configuration
      - db_sensor
      - node_auth
    working_dir: /app
    command: "node app.js"
    volumes:
      - ./node/rest/:/app
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.node_rest.rule=Host(`rest.localhost`)"
      - "traefik.http.services.node_rest.loadbalancer.server.port=80"
      - "traefik.http.routers.node_rest.entrypoints=websecure"
      - "traefik.http.routers.node_rest.tls.certresolver=myresolver"
    deploy:
      mode: global

secrets:
  db_pwd_configuration:
    external: true
  db_pwd_sensor:
    external: true
  db_pwd_user:
    external: true
  node_auth_access_token:
    external: true
  node_auth_refresh_token:
    external: true
```

Explanation

1. **Traefik Configuration**:
   - **Entry Points**:
     - `--entrypoints.web.address=:80`: HTTP entry point.
     - `--entrypoints.websecure.address=:443`: HTTPS entry point.
   - **Let's Encrypt Configuration**:
     - `--certificatesresolvers.myresolver.acme.httpchallenge.entrypoint=web`: Uses the HTTP challenge to prove domain ownership.
     - `--certificatesresolvers.myresolver.acme.httpchallenge=true`: Enables the HTTP challenge.
     - `--certificatesresolvers.myresolver.acme.email=youremail@example.com`: Replace with your email for Let's Encrypt notifications.
     - `--certificatesresolvers.myresolver.acme.storage=/letsencrypt/acme.json`: Path to store Let's Encrypt certificates.
   - **HTTP to HTTPS Redirection**:
     - `--entrypoints.web.http.redirections.entryPoint.to=websecure`: Redirects HTTP traffic to HTTPS.
     - `--entrypoints.web.http.redirections.entryPoint.scheme=https`: Specifies the scheme for redirection.
2. **Node Services (`node_auth` and `node_rest`)**:
   - **Labels**:

- `traefik.enable=true`: Enables Traefik for this service.
- `traefik.http.routers.node_auth.rule=Host(`auth.localhost`)`: Routing rule based on hostname.
- `traefik.http.services.node_auth.loadbalancer.server.port=80`: Specifies the port the service listens on.
- `traefik.http.routers.node_auth.entrypoints=websecure`: Routes traffic through the HTTPS entry point.
- `traefik.http.routers.node_auth.tls.certresolver=myresolver`: Uses Let's Encrypt resolver for HTTPS certificates.

3. **Volumes**:
   - Adds a volume for Let's Encrypt certificate storage to persist the certificates.

Starting the Services

Run the following command to start your Docker Compose stack:

```
sh

docker-compose up -d
```

Accessing Your Services

- `node_auth` should be accessible at `https://auth.localhost`.
- `node_rest` should be accessible at `https://rest.localhost`.

By following these steps, your `node_auth` and `node_rest` services will be protected with HTTPS, ensuring that only encrypted traffic is allowed.

i just want to allow https traffic

If you want to only allow HTTPS traffic to your `node_auth` and `node_rest` services and disallow HTTP traffic entirely, you can configure Traefik to listen only on the HTTPS entry point (`websecure`). This will ensure that only encrypted traffic reaches your services.

Here is the updated `docker-compose.yml` file:

Updated `docker-compose.yml`

```
yaml

version: "3.8"

services:
  traefik:
    image: traefik:v2.10
    command:
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--entrypoints.websecure.address=:443"
      - "--certificatesresolvers.myresolver.acme.httpchallenge.entrypoint=websecure"
      - "--certificatesresolvers.myresolver.acme.httpchallenge=true"
      - "--certificatesresolvers.myresolver.acme.email=youremail@example.com"
```

```
      - "--certificatesresolvers.myresolver.acme.storage=/letsencrypt/acme.json"
    ports:
      - "443:443"   # Web secure traffic (HTTPS)
      - "8080:8080" # Traefik Dashboard
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock:ro"
      - "./letsencrypt:/letsencrypt" # Persistent storage for Let's Encrypt certificates
    deploy:
      mode: global

  db_configuration:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/configuration_data/mysql:/var/lib/mysql
      - ./databases/configuration_data/scripts:/docker-entrypoint-initdb.d/
    ports:
      - "3307:3306"
    deploy:
      mode: global

  db_sensor:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/sensor_data/mysql:/var/lib/mysql
      - ./databases/sensor_data/scripts:/docker-entrypoint-initdb.d/
    ports:
      - "3308:3306"
    deploy:
      mode: global

  db_user:
    image: mysql:8.0.18
    environment:
      MYSQL_ROOT_PASSWORD: 123
      MYSQL_ONETIME_PASSWORD: "yes"
    volumes:
      - ./databases/user/mysql:/var/lib/mysql
      - ./databases/user/scripts:/docker-entrypoint-initdb.d/
    ports:
      - "3309:3306"
    deploy:
      mode: global

  node_auth:
    image: node:22-alpine3.18
    environment:
      SERVER_PORT: 80
      DB_PASSWORD_USER: /run/secrets/db_pwd_user
      ACCESS_TOKEN_SECRET: /run/secrets/node_auth_access_token
      REFRESH_TOKEN_SECRET: /run/secrets/node_auth_refresh_token
    secrets:
      - db_pwd_user
      - node_auth_access_token
      - node_auth_refresh_token
    depends_on:
      - db_user
    working_dir: /app
    command: "node app.js"
    volumes:
      - ./node/authentication/:/app
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.node_auth.rule=Host(`auth.localhost`)"
      - "traefik.http.services.node_auth.loadbalancer.server.port=80"
      - "traefik.http.routers.node_auth.entrypoints=websecure"
      - "traefik.http.routers.node_auth.tls.certresolver=myresolver"
    deploy:
      mode: global
```

```
  node_rest:
    image: node:22-alpine3.18
    environment:
      SERVER_PORT: 80
      DB_PASSWORD_CONFIGURATION: /run/secrets/db_pwd_configuration
      DB_PASSWORD_SENSOR: /run/secrets/db_pwd_sensor
    secrets:
      - db_pwd_configuration
      - db_pwd_sensor
    depends_on:
      - db_configuration
      - db_sensor
      - node_auth
    working_dir: /app
    command: "node app.js"
    volumes:
      - ./node/rest/:/app
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.node_rest.rule=Host(`rest.localhost`)"
      - "traefik.http.services.node_rest.loadbalancer.server.port=80"
      - "traefik.http.routers.node_rest.entrypoints=websecure"
      - "traefik.http.routers.node_rest.tls.certresolver=myresolver"
    deploy:
      mode: global

secrets:
  db_pwd_configuration:
    external: true
  db_pwd_sensor:
    external: true
  db_pwd_user:
    external: true
  node_auth_access_token:
    external: true
  node_auth_refresh_token:
    external: true
```

Explanation

1. **Traefik Configuration**:
   - **Entry Points**:
     - **`--entrypoints.websecure.address=:443`**: Only defines the HTTPS entry point.
   - **Let's Encrypt Configuration**:
     - **`--certificatesresolvers.myresolver.acme.httpchallenge.entrypoint=websecure`**: Uses the HTTPS entry point for the HTTP challenge (since we're only allowing HTTPS).
     - **`--certificatesresolvers.myresolver.acme.httpchallenge=true`**: Enables the HTTP challenge.
     - **`--certificatesresolvers.myresolver.acme.email=youremail@example.com`**: Replace with your email for Let's Encrypt notifications.
     - **`--certificatesresolvers.myresolver.acme.storage=/letsencrypt/acme.json`**: Path to store Let's Encrypt certificates.
2. **Node Services (`node_auth` and `node_rest`)**:
   - **Labels**:
     - **`traefik.enable=true`**: Enables Traefik for this service.
     - **`traefik.http.routers.node_auth.rule=Host(`**auth.localhost**`)`**: Routing rule based on hostname.
     - **`traefik.http.services.node_auth.loadbalancer.server.port=80`**: Specifies the port the service listens on.
     - **`traefik.http.routers.node_auth.entrypoints=websecure`**: Routes traffic through the HTTPS entry point.

- `traefik.http.routers.node_auth.tls.certresolver=myresolver`: Uses Let's Encrypt resolver for HTTPS certificates.

3. **Volumes**:
   - Adds a volume for Let's Encrypt certificate storage to persist the certificates.

Starting the Services

Run the following command to start your Docker Compose stack:

```
sh

docker-compose up -d
```

Accessing Your Services

- `node_auth` should be accessible at `https://auth.localhost`.
- `node_rest` should be accessible at `https://rest.localhost`.

By following these steps, you'll ensure that only HTTPS traffic is allowed to reach your `node_auth` and `node_rest` services.