# TraeSentinel Deployment Guide

## 1. Introduction

This guide provides comprehensive, step-by-step instructions for deploying the TraeSentinel application stack. The deployment process utilizes Docker and Docker Compose to containerize the application services and Traefik as a reverse proxy to manage network traffic and automate SSL certificate issuance.

Following these instructions will result in a fully operational and securely configured instance of the application.

---

## 2. Prerequisites

Before you begin the deployment process, ensure you have the following prerequisites in place:

- **A Valid Domain Name:** You must own a domain name (e.g., your-domain.com) that you can manage DNS records for.
- **Cloud DNS Provider API Credentials:** API credentials for a supported DNS provider like Cloudflare or Namecheap are required for Traefik to automate the SSL certificate process (DNS-01 challenge).
- **Linux Server:** A Linux server or virtual machine. This guide has been tested on **Ubuntu Server 24.04**.
- **Docker and Docker Compose:** Ensure that Docker and Docker Compose are installed and running on your server.

---

## 3. Initial Setup

Follow these steps to prepare the application environment on your server.

1. Clone the Repository
   Clone the TraeSentinel source code from its GitHub repository.
   Bash
   ```
   git clone https://github.com/Anganba/TraeSentinel.git
   ```

2. Navigate to the Project Directory
   Change into the newly created project directory.
   Bash
   ```
   cd TraeSentinel
   ```

3. Create Docker Networks
   The application's docker-compose.yml file is hardcoded to use specific Docker networks. Create them manually with the following commands:
   Bash
   ```
   sudo docker network create frontend
   sudo docker network create monitoring
   ```

4. Set Permissions for Certificate Storage
   Traefik requires specific permissions for the file where it stores SSL certificate information. Set the correct permissions to avoid issues during certificate generation.
   Bash
   ```
   sudo chmod 600 Traefik/data/*
   ```

   *Note: The original instructions mentioned certs/*.json, but the standard file for Let's Encrypt certificates with Traefik is acme.json.*

---

# 4. Custom Domain Configuration

The project uses Traefik to automatically route traffic based on the domain name. To use your own domain, you must find and replace the default domain (anganba.me) in the project's configuration files.

## How it Works

Traefik discovers services using Docker labels within the docker-compose.yml file. A specific label, traefik.http.routers.<service-name>.rule, defines which domain or subdomain directs traffic to a particular service.

For example, the label for one service might look like this:
"traefik.http.routers.monitoring.rule=Host(\mon.anganba.me`)"`
This rule tells Traefik to send any traffic for mon.anganba.me to the "monitoring" service. You will need to change this to your own domain.

## Steps to Configure Your Domain

1. **Locate Configuration Files:** Open the docker-compose.yml file and any other relevant configuration files (e.g., .env files) in the TraeSentinel directory.
2. **Search and Replace:** Search for all occurrences of anganba.me and replace them with your custom domain.
   - For a root domain, replace anganba.me with your-domain.com.
   - For subdomains, replace mon.anganba.me with a corresponding subdomain like monitoring.your-domain.com.
3. Update DNS Records:
   After updating the configuration files, you must configure your domain's DNS records to point to your server's public IP address.
   - Log in to your domain registrar (e.g., Cloudflare, Namecheap).
   - Create an **A Record** for each domain and subdomain you configured (e.g., your-domain.com, monitoring.your-domain.com) and point it to your server's public IP address.

This step is critical. Without it, traffic will not reach your server, and Traefik will be unable to issue a valid SSL certificate.

---

## 5. Deploying the Application

Once the initial setup and domain configuration are complete, you can deploy the application stack.

1. **Make the Deployment Script Executable**
   ```Bash
   sudo chmod +x deploy.sh
   ```

2. Start All Services
   Run the deployment script with the up command to build and start all the application containers in detached mode.
   ```Bash
   sudo./deploy.sh up
   ```

The initial startup may take a few minutes as Docker downloads the necessary images.

---

## 6. Managing the Application

The deploy.sh script provides simple commands for managing the application lifecycle.

- **Start the Application:**
  ```Bash
  sudo./deploy.sh up
  ```

- Stop and Remove the Application:
  This command stops and removes all containers, networks, and volumes.
  ```Bash
  sudo./deploy.sh down
  ```

- **Restart the Application:**
  ```Bash
  sudo./deploy.sh restart
  ```

- Check Application Status:
  This shows the running containers and their health status.
  ```Bash
  sudo./deploy.sh status
  ```