Installation

Beschreibung:

Caddy ein letsencrypt Docker Proxy/Agent der Automatisch auch Zertifikate verlängert.
Caddy besteht aus einem Containerund einer Config Datei für die Domänen

Installation

Docker installieren

```
apt install curl docker.io docker-compose
```

Beispiel Konfig für einen apache2 der auf http port 8080 lauscht und den Caddy der auf Port 80 und 443 lauscht.
Port 80 wird auf 443 umgeleitet.
Der Port 443 wieder rum an 8080.

In der Composer Datei haben wir den port bei Apache rausgenommen, weil er auf 8080 im internet Docker Netz lauscht

Verzeichnisse erstellen

```
mkdir caddy_data
mkdir caddy_config
```

Die Compose Datei

```
version: '3.8'

services:
  apache:
    image: httpd:latest
    container_name: apache-webserver
    #ports:
```

```
    #  - "8080:8080"
    volumes:
      - ./apache_html:/usr/local/apache2/htdocs/
    restart: unless-stopped

  caddy:
    image: caddy:latest
    container_name: caddy-reverse-proxy
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./Caddyfile:/etc/caddy/Caddyfile
      - ./caddy_data:/data
      - ./caddy_config:/config
    restart: unless-stopped
```

Nun noch die Caddfile anlegen

Caddyfile für die Nutzung von Lets Encrypt

```
nano Caddyfile
```

Inhalt, wenn man es erst testen möchte mit Staging einfach den Kommentar entfernen.

```
example.com {
    reverse_proxy http://<servicename_docker_name>:8080
    tls <deine_emailadresse> {
        #ca https://acme-staging-v02.api.letsencrypt.org/directory
    }
}
```

Lets Encrypt Global Einstellungen wenn gewünscht:

Man kann aber auch die Lets Encrypt defination global setzen.
Dann braucht man das nicht mehr für jede Domain.

Wenn man es erst testen möchte mit Staging einfach den Kommentar entfernen.

```
{
    #acme_ca https://acme-staging-v02.api.letsencrypt.org/directory
    email deine_emailadresse@example.com
}

example.com {
    reverse_proxy http://<servicename_docker_name>:8080
}
```

Über den Paramter tls können dann noch die Versionen angepasst werden.
Lässt man tls weg, wird automatisch das aktuellste verwendet, wie im oberen Beispiel, da haben wir kein tls mit drin.
Hier wird TLS Global definiert, sprich gillt für alle

```
{
    #acme_ca https://acme-staging-v02.api.letsencrypt.org/directory
    email admin@example.com
    tls {
        protocols tls1.2 tls1.3
    }
}

example.com {
    reverse_proxy http://localhost:8080
}

example2.com {
    reverse_proxy http://localhost:8081
}
```

Man kann aber auch in den einzelnen Domain weiterhin die Versionen angeben, macht Sinn wenn diese sich unterscheiden.
Aus irgendeinem Grund, was nicht zu empfehlen ist, braucht der Server noch TLS 1.1

```
{
    #acme_ca https://acme-staging-v02.api.letsencrypt.org/directory
    email admin@example.com
    tls {
        protocols tls1.2 tls1.3
    }
}
```

```
example.com {
	tls {
		protocols tls1.2 tls1.3
	}
	reverse_proxy http://localhost:8080
}

example2.com {
	tls {
		protocols tls1.1
	}
	reverse_proxy http://localhost:8081
}
```

Vorhandene Zertifikate nutzen:

Hier werden TLS 1.2 und TLS 1.3 erzwungen, und zusätzlich kannst du Client-Zertifikate verwenden, wenn du möchtest.

```
example.com {
    reverse_proxy http://<servicename_docker_name>:8080
    tls {
        protocols tls1.2 tls1.3
        client_auth {
            mode require_and_verify
            trusted_ca_cert_file /path/to/ca.pem
        }
    }
}
```

Caddyfile was selbst signiertes Zertifikat nutzen soll

Wir können einen Domain Namen angeben wenn wir einen eigenen DNS Server Betreiben, ansonsten Hostname oder ip Adressee angeben.

Hinweis: Ein selbst signiertes Zertifikat ist nur 12 Stunden gültig.
Es wird im Arbeitsspeicher generiert. Allerdings muss dann erneut das Zertifikat zur Ausnahme hinzugefügt werden.

Es muss unbedingt ein Domänen Namen verwendet werden!!!

Wenn kein eigener DNS Server vorhanden dann in die hosts Datei eintragen. IP Adresse geht nicht!!!!

```
example.local.lan {
    reverse_proxy http://<servicename_docker_name>:8080
    tls internal
}
```

Kurzversion in compose Datei ohne config Datei:

Man kann auch ohne config Dateien einen Caddy bauen, dann wird immer der übergebenen Domainname genommen und im parameter das redirect

Hier ist wordpress der interne name des Docker containers wordpress und der zielport auf dem Wordpress lauscht

in der .env datei kommt die PUBLIC url rein

Inhalt

```
PUBLIC_URL = "https://example.com"
```

Docker compose container

```
...
  caddy:
    image: caddy:latest
    restart: always
    ports:
      - "80:80"
      - "443:443"
    command: caddy reverse-proxy --from ${PUBLIC_URL} --to wordpress:80
    volumes:
      - ./data/caddy/data:/data
      - ./data/caddy/config:/config
```

...

Version #19
Erstellt: 14 Oktober 2024 08:11:13 von Admin
Zuletzt aktualisiert: 5 Juli 2025 18:56:53 von Admin