Installation

Wenn man nicht github gitlab etc. verwenden möchte sondern sein eigenes Git.
hier merhere Installationsmöglichkeiten.

- Gitea Docker installation mit mariadb und SSL

Gitea Docker installation mit mariadb und SSL

Beschreibung:

Ein Docker Container mit gitea. Der als Datenbank mariaDB benutzt und ein Lets Encrypt SSL Certificate holt oder ein selbstsigniertes SSL Certificate erstellt.

Docker compose mit Lets Encrypt:

Der Domainname sollte bereits auf den Server per IP a oder AAA record linken. Je nachdem ob ipv4 oder ipv6.
Genauso sollte docker compose auch installiert sein.

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

Falls apparmor auf dem system nicht installiert ist

```
# AppArmor installieren, falls es nicht installiert ist
apt-get install apparmor apparmor-utils
```

Als erstes unser Verzeichnis erstellen in root.
Darin liegen unser config Dateien für den container

```
mkdir -p /root/gitea/mariadb
mkdir -p /root/gitea/data
```

DIe .env Datei

```
# Gitea Einstellungen
USER_UID=1000
USER_GID=1000
#Registrieren link einblenen ja/nein true/false
GITEA__service__DISABLE_REGISTRATION=true
DISABLE_REGISTRATION=true
```

```
# Datenbank Einstellungen
DB_ROOT_PASS=rootpassword
DB_USER=gitea
DB_PASSWD=gitea
DB_NAME=gitea

# Port Einstellungen
GITEA_HTTP_PORT=3000
GITEA_SSH_PORT=222

# Volume directories
#Pfade mit Slash vorran. Denn die werden dann zusmmanegbaut
BASE_PATH_DIR=/root/gitea
MARIADB_VOLUME_DIR=/mariadb
DATA_VOLUME_DIR=/data
#Das ist ein Teil des certbot Pfades ,da dies eine Variable ist, da kommt noch eine weitere Variable im bestehen
Verlauf daran ${CERTBOT_VOLUME_DIR}-log
CERTBOT_VOLUME_DIR=/certbot

# Domain and email for Let's Encrypt
DOMAIN_NAME=ihredomain.de
LETSENCRYPT_EMAIL=ihre-email@beispiel.de
```

Die Composer Datei

```
version: '3.8'

services:
  server:
    image: gitea/gitea:latest
    environment:
      - USER_UID=${USER_UID}
      - USER_GID=${USER_GID}
      - DB_TYPE=mysql
      - DB_HOST=db:3306
      - DB_NAME=${DB_NAME}
      - DB_USER=${DB_USER}
      - DB_PASSWD=${DB_PASSWD}
    restart: always
```

```
  volumes:
    - ${BASE_PATH_DIR}${DATA_VOLUME_DIR}:/data

  ports:
    - "${GITEA_HTTP_PORT}:3000"
    - "${GITEA_SSH_PORT}:22"
  depends_on:
    - db
    - webserver

db:
  image: mariadb:latest
  restart: always
  environment:
    MYSQL_ROOT_PASSWORD: ${DB_ROOT_PASS}
    MYSQL_USER: ${DB_USER}
    MYSQL_PASSWORD: ${DB_PASSWD}
    MYSQL_DATABASE: ${DB_NAME}
  volumes:
    - ${BASE_PATH_DIR}${MARIADB_VOLUME_DIR}:/var/lib/mysql

certbot:
  image: certbot/certbot
  volumes:
    - ${BASE_PATH_DIR}${CERTBOT_VOLUME_DIR}-etc:/etc/letsencrypt
    - ${BASE_PATH_DIR}${CERTBOT_VOLUME_DIR}-www:/var/www/certbot
    - ${BASE_PATH_DIR}${CERTBOT_VOLUME_DIR}-log:/var/log/letsencrypt
  entrypoint: '/bin/sh -c'

webserver:
  image: nginx:alpine
  restart: unless-stopped
  volumes:
    - ./nginx.conf:/etc/nginx/nginx.conf:ro
    - ${BASE_PATH_DIR}${CERTBOT_VOLUME_DIR}-etc:/etc/letsencrypt
    - ${BASE_PATH_DIR}${CERTBOT_VOLUME_DIR}-www:/var/www/certbot
    - ${BASE_PATH_DIR}${CERTBOT_VOLUME_DIR}-log:/var/log/letsencrypt
    - ./nginx-log:/var/log/nginx
  ports:
    - "80:80"
```

```
      - "443:443"
```

Init script, erstellt die nginx config und holt erstmals das certificate.

```
#!/bin/bash

# Laden der Umgebungsvariablen aus der .env-Datei
if [ -f .env ]; then
    export $(grep -v '^#' .env | xargs)
else
    echo ".env-Datei nicht gefunden!"
    exit 1
fi

# Überprüfen, ob DOMAIN_NAME und LETSENCRYPT_EMAIL gesetzt sind
if [ -z "$DOMAIN_NAME" ] || [ -z "$LETSENCRYPT_EMAIL" ]; then
    echo "DOMAIN_NAME und/oder LETSENCRYPT_EMAIL sind in der .env-Datei nicht gesetzt!"
    exit 1
fi

# Erstellen der benötigten Verzeichnisse
mkdir -p /root/gitea/mariadb
mkdir -p /root/gitea/data
mkdir -p ./certbot-etc
mkdir -p ./certbot-www
mkdir -p ./certbot-log
mkdir -p ./nginx-log

# Erstellen der Nginx-Konfigurationsdatei für den ersten Lauf (ohne SSL)
cat << EOF > ./nginx.conf
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}
```

```
http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '\$remote_addr - \$remote_user [\$time_local] "\$request" '
                      '\$status \$body_bytes_sent "\$http_referer" '
                      '"\$http_user_agent" "\$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    server {
        listen 80;
        server_name ${DOMAIN_NAME};

        location /.well-known/acme-challenge/ {
            root /var/www/certbot;
            try_files \$uri =404;
        }
    }
}
EOF

# Starten des Webserver-Containers (erster Lauf ohne SSL)
echo "Starte Webserver-Container..."
docker-compose down
docker-compose up -d webserver

# Warten, um sicherzustellen, dass Nginx vollständig gestartet ist
echo "Warte auf Webserver..."
sleep 5

# Ausführen des Certbot für den ersten Lauf
echo "Starte Certbot..."
docker-compose run --rm --entrypoint "" certbot certbot certonly --webroot --webroot-path=/var/www/certbot \
    --email ${LETSENCRYPT_EMAIL} --agree-tos --no-eff-email \
```

```
    --domain ${DOMAIN_NAME} --rsa-key-size 4096

# Aktualisieren Sie die Nginx-Konfigurationsdatei, um SSL zu aktivieren
cat << EOF > ./nginx.conf
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '\$remote_addr - \$remote_user [\$time_local] "\$request" '
                      '\$status \$body_bytes_sent "\$http_referer" '
                      '"\$http_user_agent" "\$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    server {
        listen 80;
        server_name ${DOMAIN_NAME};

        location /.well-known/acme-challenge/ {
            root /var/www/certbot;
            try_files \$uri =404;
        }

        location / {
            return 301 https://\$host\$request_uri;
```

```
        }
    }

    server {
        listen 443 ssl;
        server_name ${DOMAIN_NAME};

        ssl_certificate /etc/letsencrypt/live/${DOMAIN_NAME}/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/${DOMAIN_NAME}/privkey.pem;

        location / {
            proxy_pass http://server:${GITEA_HTTP_PORT};
            proxy_set_header Host \$host;
            proxy_set_header X-Real-IP \$remote_addr;
            proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto \$scheme;
            client_max_body_size 100M;
        }
    }
}
EOF

# Starten Sie den Webserver-Container neu, um die aktualisierte Konfiguration zu verwenden
echo "Starte Webserver-Container neu..."
docker-compose down
docker-compose up -d webserver

echo "Setup abgeschlossen. Bitte überprüfen und anpassen Sie die nginx.conf entsprechend."
```

Lets Encrypt erneuern

Damit jede Nacht das Zertifikat erneuert wird legen wir eine Systemd Service Datei an.
Dazu

```
nano /etc/systemd/system/certbot-renew.service
```

Inhalt

```
[Unit]
Description=Certbot Renewal
```

```
[Service]
Type=oneshot
ExecStart=/usr/bin/docker-compose -f /root/gitea/docker-compose.yml run --rm --entrypoint "" certbot certbot renew --quiet
ExecStartPost=/usr/bin/docker-compose -f /root/gitea/docker-compose.yml restart  webserver
```

Und nun den passenden Timer dazu

```
nano /etc/systemd/system/certbot-renew.timer
```

Inhalt

```
[Unit]
Description=Timer for Certbot Docker Container Renewal

[Timer]
OnCalendar=daily
Persistent=true

[Install]
WantedBy=timers.target
```

Aktiviren des Timers

```
systemctl enable certbot-renew.timer
```

Starten des Timers

```
systemctl start certbot-renew.timer
```

Überpürfen des Timers

```
systemctl list-timers certbot-renew.timer
```

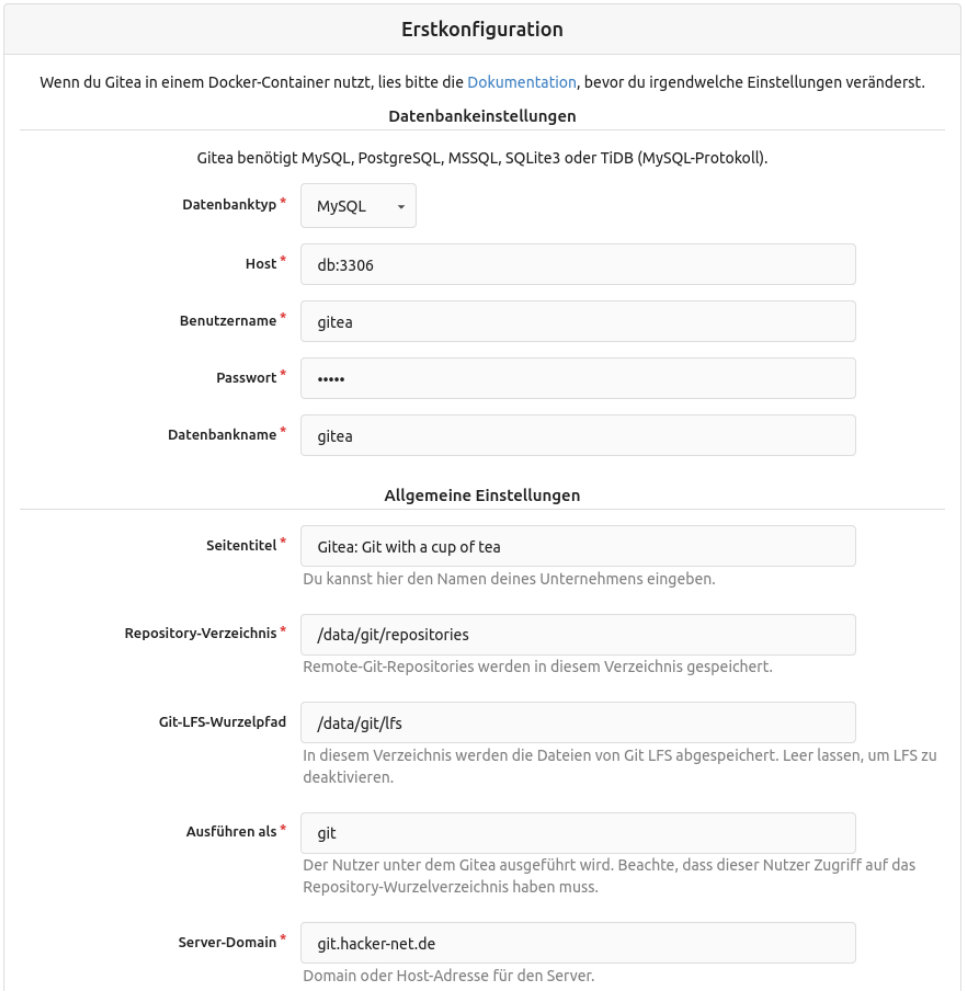
Starten:

Nun mit docker-compose up -d den kram starten.

URL Aufrufen:

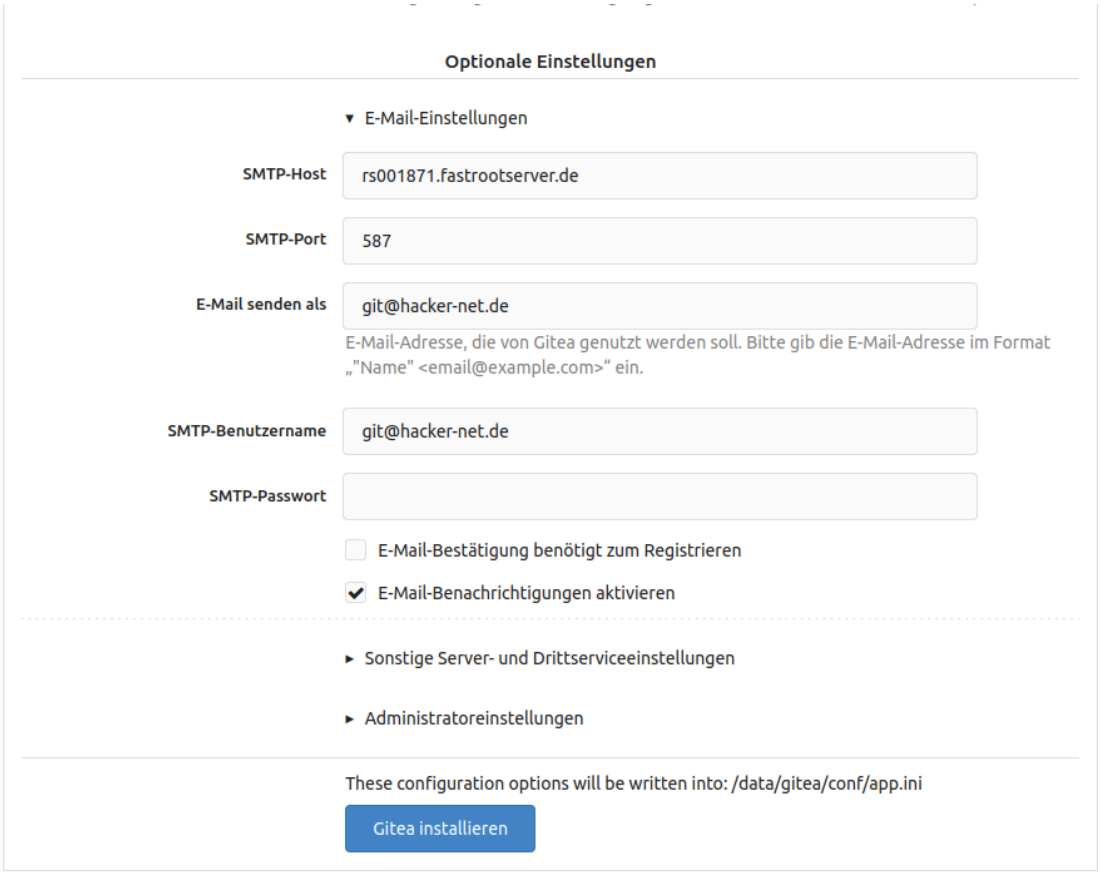
http://git.yourdomain.com

Die Einstellungen wurden aus der app.ini übernommen und aus den umgebungsvariablen, wie man sieht

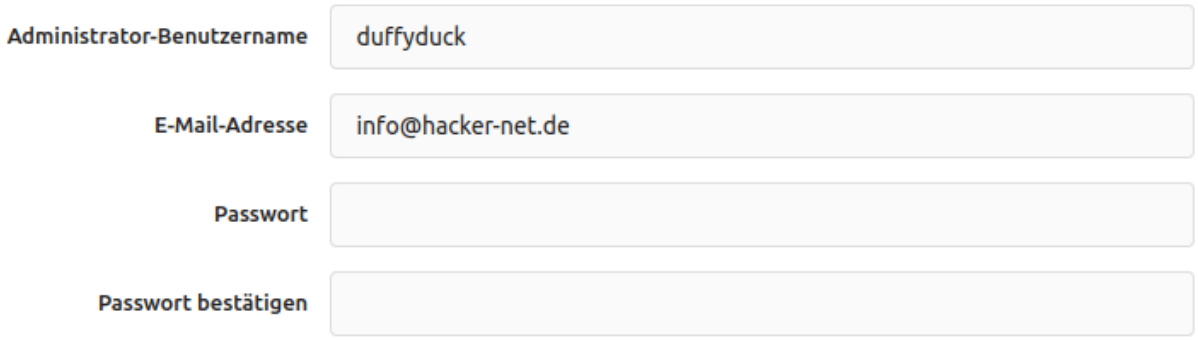


runterscrollen bis zum Pnkt Email

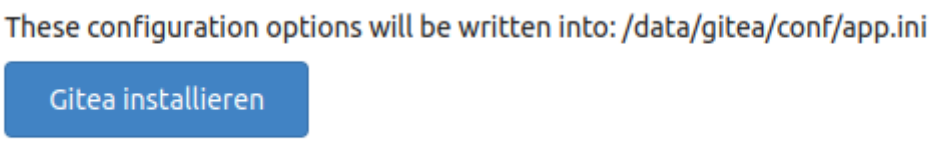
Dort Email Daten für den versand ausfüllen.



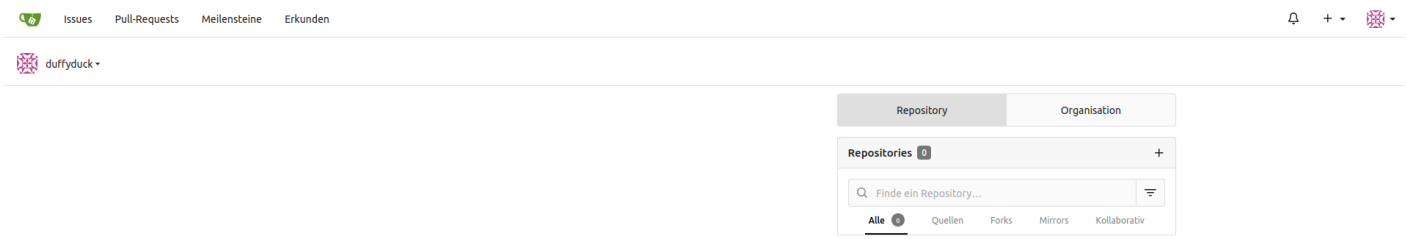
Nun noch ein Administrator Konto anlegen, Benutzername und Kennwort vergeben fert.
Der Name admin ist Intern vergeben und kann nicht genutzt werden.



Dann auf gitea installieren klicken, drücken



Danach werdet Ihr eingeloggt, fertig



Disbale/Enable Registration button nachträglich

Wenn in der der Estprovisionierung true oder fals übergeben wurde wird dies in die app.ini mit übertragen.
Möchte man den Wert in was auch immer später ändern, geht das nur in der app.ini

```
cd ~/gitea/data/gitea/conf/app.ini
```

Nun in ser service Section den Dinest auf true oder fals e stellen, jenach dem was gewünscht ist true disabled den button, false enabled den button

```
...
[service]
DISABLE_REGISTRATION = false
...
```