

covidcheckin.de Installationsanleitung

Hinweis

Jegliche Elemente um die Webapp aufzusetzen sind in dieser Installationsanleitung vorhanden.

Voraussetzungen

Sie benötigen einen Linux Server (VPS, Root oder selbst gehostet mit Docker Unterstützung) und eine eigene Domain um die Webapp aufsetzen zu können. Als Linux Distribution empfiehlt sich Ubuntu Server 20.04, es eignet sich aber jede Distribution welche Docker unterstützt.

Kenntnisse über folgende Systeme und Dienste sollten bekannt sein:

- Linux (Ubuntu)
- Docker & docker-compose
- Domain & DNS Einträge
- Mailserver (Mailcow)
- Reverse Proxy
- SSL/TLS Zertifikate
- Git

Für eine sichere Verbindung sollten Sie ein SSL/TLS Zertifikat ausstellen können oder die Webapp durch einen Reverse Proxy leiten der dies übernimmt. In dieser Anleitung wird Traefik als Reverse Proxy mit automatischer Zertifikat Erstellung durch Let's Encrypt genutzt.

DNS Einträge

Die DNS Einträge wurden mit einem Reverse Proxy gewählt. Sollten Sie keinen einsetzen, so haben Sie die Record Type und Destinations dementsprechend anzupassen

Name	Record Type	Destination
<ihreDomain>.de	A	<ihreServerIP>
pma.<ihreDomain>.de	A	<ihreServerIP>
jenkins.<ihreDomain>.de	A	<ihreServerIP>

Falls Sie einen eigenen Mailserver hosten, gibt es noch weitere DNS Einträge zu setzen. Diese finden sich in der [Mailcow Dokumentation](#). Für diese Anleitung wird ein externer Mailserver zur Vereinfachung benutzt.

Vorinstallierte Programme

Programm	Installationsanleitung Link
Docker	docs.docker.com/engine
docker-compose	docs.docker.com/compose
git	git-scm.com
dockerized Traefik (empfohlen, Optional)	doc.traefik.io

Ordnerstruktur

Folgende Ordnerstruktur ist anzulegen für die Webapp:

- user home
 - backend
 - frontend
 - jenkins
 - jenkins_home
 - mailcow (Optional, nicht gezeigt in dieser Anleitung)

Backend

Für das Backend wird folgendes Network erstellt:

```
docker network create covidcheckin_internal
```

Falls nicht schon geschehen, für Traefik auch ein Network:

```
docker network create reverse_proxy
```

Außerdem wird folgendes Volume für die Datenbank erstellt:

```
docker volume create covidcheckin_mariadb_volume
```

Nun wird im Backend Ordner eine **docker-compose.yml** mit folgendem Inhalt erstellt:

```
version: '3'
services:
  covidcheck-mariadb:
    image: mariadb:latest
    restart: unless-stopped
    environment:
      - MYSQL_ROOT_PASSWORD=
      - MYSQL_DATABASE=covidcheckin
      - MYSQL_USER=web
      - MYSQL_PASSWORD=
    volumes:
      - covidcheck-mariadb-volume:/var/lib/mysql
    ports:
      - "3306:3306"
    networks:
      - covidcheckin_internal

  covidcheck-phpmyadmin:
    image: phpmyadmin/phpmyadmin:latest
    restart: unless-stopped
    depends_on:
      - covidcheck-mariadb
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.covidcheck-phpmyadmin.entrypoints=http"
      - "traefik.http.routers.covidcheck-phpmyadmin.rule=Host(`pma.<ihreDomain>.de`)"
      - "traefik.http.middlewares.covidcheck-phpmyadmin-https-redirect.redirectscheme.scheme=https"
      - "traefik.http.routers.covidcheck-phpmyadmin.middlewares=covidcheck-phpmyadmin-https-redirect"
      - "traefik.http.routers.covidcheck-phpmyadmin-secure.entrypoints=https"
      - "traefik.http.routers.covidcheck-phpmyadmin-secure.rule=Host(`pma.<ihreDomain>.de`)"
      - "traefik.http.routers.covidcheck-phpmyadmin-secure.tls=true"
      - "traefik.http.routers.covidcheck-phpmyadmin-secure.service=covidcheck-phpmyadmin"
      - "traefik.http.services.covidcheck-phpmyadmin.loadbalancer.server.port=80"
      - "traefik.docker.network=reverse_proxy"
      - "traefik.http.routers.covidcheck-phpmyadmin-secure.middlewares=secHeaders@file"
    networks:
      - reverse_proxy
      - covidcheckin_internal
    environment:
      - PMA_HOST=covidcheck-mariadb
      - PMA_PORT=3306

volumes:
  covidcheck-mariadb-volume:

networks:
  reverse_proxy:
    external: true
  covidcheckin_internal:
    external: true
```

Folgende Einträge sind zu Bearbeiten:

Zeile	Name	Wert
7	MYSQL_ROOT_PASSWORD	Hier sehr sicheres Passwort setzen. Dies stellt den Root Zugang zur Datenbank dar und kann verheerende Folgen haben wenn ein schwaches Passwort verwendet wird.
10	MYSQL_PASSWORD	Hier sehr sicheres Passwort für den Datenbankbenutzer setzen. Dieser greift über PHP auf die Datenbank zu.
24-35	Labels für Traefik	Die gesetzten Labels dienen zur Traefik Konfiguration. Je nach Nutzung von Traefik unterscheiden sich diese von unserem Beispiel.

Das Backend kann man anschließend mit:

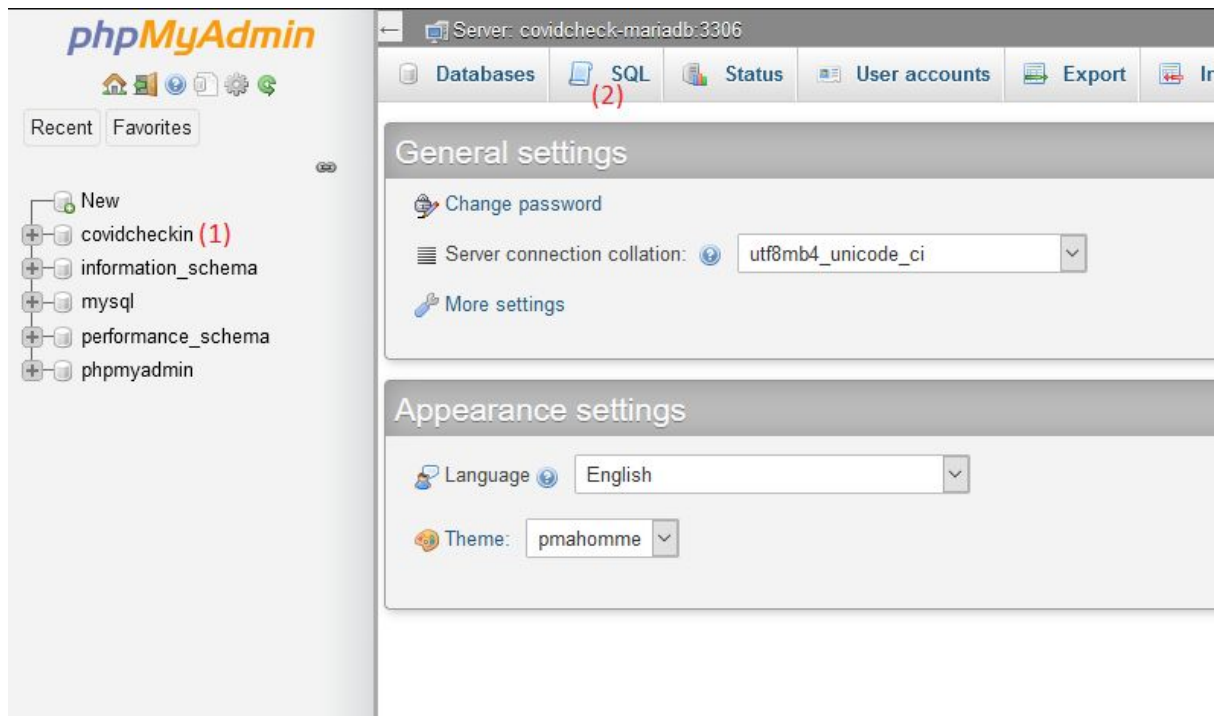
```
docker-compose up -d
```

starten. Daraufhin verbindet man sich unter der gesetzten Domain pma.<ihreDomain>.de mit dem Dienst PHPmyAdmin.



The image shows the phpMyAdmin login interface. At the top, there is a logo featuring a sailboat and the text 'phpMyAdmin'. Below the logo, it says 'Welcome to phpMyAdmin'. The interface is divided into two main sections. The first section is labeled 'Language' and contains a dropdown menu with 'English' selected. The second section is labeled 'Log in' and contains two input fields: 'Username' with the value 'root' and 'Password' which is empty. A 'Go' button is located at the bottom right of the login section.

Auf der Anmeldeseite trägt man nun das gesetzte Root-Passwort ein und betätigt 'Go'.



Hier wird erst die Datenbank covidcheckin (1) ausgewählt und dann zum SQL (2) Reiter gewechselt um hier die Datenbank zu initialisieren.

Im Textfeld wird folgendes eingefügt:

```
CREATE TABLE accounts(
  USERNAME VARCHAR(255),
  PASSWORD VARCHAR(255),
  VORNAME VARCHAR(255),
  NACHNAME VARCHAR(255),
  EMAIL VARCHAR(255),
  TOKEN VARCHAR(255),
  ACTIVE INTEGER);

CREATE TABLE verlauf (
  ID INTEGER PRIMARY KEY AUTO_INCREMENT,
  STATUS VARCHAR(255),
  TIMESTAMP VARCHAR(255),
  USERNAME VARCHAR(255));

CREATE TABLE resetRequests(
  ID INTEGER PRIMARY KEY AUTO_INCREMENT,
  CODE VARCHAR(255),
  EMAIL VARCHAR(255));

CREATE TABLE confirmEmail(
  ID INTEGER PRIMARY KEY AUTO_INCREMENT,
  CODE VARCHAR(255),
  EMAIL VARCHAR(255));
```

Daraufhin wird die Eingabe mit 'Go' bestätigt. Dieser Schritt legt die Strukturen innerhalb der Datenbank an. Nachdem dies gemacht wurde, kann die Seite geschlossen werden. Damit ist das Backend aufgesetzt.

Frontend

Falls nicht geplant ist, dass Jenkins eingesetzt wird oder die Anwendung nur getestet werden soll, kann das Frontend auch manuell gestartet werden. Hier muss jedoch darauf hingewiesen werden, dass Updates der Seite per Git sowie das neu starten der Server, manuell durchgeführt werden müssen.

Dazu wird das Covidcheckin Repository im Frontend Ordner geklont:

Für das Klonen muss man sich mit den Anmeldedaten für Bitbucket anmelden und anschließend wird das Repository in den Ordner gecloned.

Im Repository befindet sich eine **docker-compose.yml**, welche nun bearbeitet wird:

```
- "traefik.http.routers.covidcheckin-nginx.middlewares=covidcheckin-nginx-https-redirect"
- "traefik.http.routers.covidcheckin-nginx-secure.entrypoints=https"
- "traefik.http.routers.covidcheckin-nginx-secure.rule=Host(`<ihreDomain.de`)"
- "traefik.http.routers.covidcheckin-nginx-secure.tls=true"
- "traefik.http.routers.covidcheckin-nginx-secure.service=covidcheckin-nginx"
- "traefik.http.services.covidcheckin-nginx.loadbalancer.server.port=80"
- "traefik.docker.network=reverse_proxy"
- "traefik.http.routers.covidcheckin-nginx-secure.middlewares=secHeaders@file"
```

```
networks:
  covidcheckin_internal:
    external: true
  reverse_proxy:
    external: true
```

Folgende Einträge sind zu bearbeiten:

Zeile	Name	Wert
12	Volume /code	Der absolute Pfad ist wichtig für Jenkins. Falls Jenkins genutzt wird muss dieser auf Ihre Ordernamen aktualisiert werden bzw. bei nichtbenutzung von Jenkins reicht <code>-. /code:/code</code> stattdessen
26	Volume /code	Der absolute Pfad ist wichtig für Jenkins. Falls Jenkins genutzt wird muss dieser auf Ihre Ordernamen aktualisiert werden bzw. bei nichtbenutzung von Jenkins reicht <code>-. /code:/code</code> stattdessen
27	Volume /nginx.conf	Der absolute Pfad ist wichtig für Jenkins. Falls Jenkins genutzt wird muss dieser auf Ihre Ordernamen aktualisiert werden bzw. bei nichtbenutzung von Jenkins reicht <code>-. /nginx/site.conf:/code</code> stattdessen
32-46	Labels für Traefik	Die gesetzten Labels dienen zur Traefik Konfiguration. Je nach Nutzung von Traefik Unterscheiden sich diese von unserem Beispiel

Außerdem muss die Datei **/code/mysql.php** bearbeitet werden:

```
<?php
$host = "covidcheck-mariadb";
$name = "covidcheckin";
$user = "web";
$password = "gesetztes PW aus Backend";
$port = "3306";
try{
    $mysql = new PDO("mysql:host=$host;port=$port;dbname=$name",
    $user, $password);
} catch (PDOException $e){
    echo "SQL Error: ".$e->getMessage();
}
?>
```

Folgende Einträge sind zu bearbeiten:

Zeile	Name	Wert
5	\$password	Hier wird das gesetzte Datenbank-Benutzer-Passwort eingetragen, welches in der Backend docker-compose.yml gesetzt wurde.

Nachdem dies erledigt wurde kann das Frontend gestartet werden:

```
docker-compose up -d --force-recreate
```

Nachdem die Container gebaut und gestartet wurden kann nun die Webapp unter **<ihreDomain>.de** aufgerufen werden.

Die Webapp befindet sich nun in einem funktionalen Zustand, besitzt aber kein automatisches Deployment bei Änderungen im Code. Um die neuesten Änderungen zu erhalten, muss im Frontend-Ordner

```
docker-compose down
```

ausgeführt werden, um das Frontend zu stoppen. Danach muss

```
git pull
```

ausgeführt werden, um die neuesten Änderungen aus dem Repository zu erhalten. Hierbei muss man sich wieder mit den Anmeldedaten für das Bitbucket der HHN anmelden.

Anschließend wird

```
docker-compose up -d --force-recreate
```

wieder ausgeführt aus, um das Frontend neu bauen und starten zu lassen.

CI/CD Jenkins

Um automatisiert neue Updates zu erhalten, wird Jenkins als CI/CD (Continuous Integration / Continuous Deployment) Anwendung genutzt.

Installation

In den Jenkins Ordner wechseln und dort einen jenkins_home Ordner erstellen. Danach wird eine **docker-compose.yml** im jenkins Ordner erstellt, welche folgenden Inhalt hat:

```
version: '3'
services:
  jenkinsci:
    container_name: jenkinsci-docker
    build:
      context: .
      dockerfile: jenkins_lts_alpine_dood.Dockerfile
    restart: unless-stopped
    expose:
      - "8080"
    volumes:
      - ./jenkins_home:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.jenkinsci.entrypoints=http"
      - "traefik.http.routers.jenkinsci.rule=Host(`jenkins.<ihreDomain>.de`)"
      - "traefik.http.middlewares.jenkinsci-https-redirect.redirectscheme.scheme=https"
      - "traefik.http.routers.jenkinsci.middlewares=jenkinsci-https-redirect"
      - "traefik.http.routers.jenkinsci-secure.entrypoints=https"
      - "traefik.http.routers.jenkinsci-secure.rule=Host(`jenkins.<ihreDomain>.de`)"
      - "traefik.http.routers.jenkinsci-secure.tls=true"
      - "traefik.http.routers.jenkinsci-secure.service=jenkinsci"
      - "traefik.http.services.jenkinsci.loadbalancer.server.port=8080"
      - "traefik.docker.network=reverse_proxy"
      - "traefik.http.routers.jenkinsci-secure.middlewares=secHeaders@file"
    networks:
      - reverse_proxy
    environment:
      - DOCKER_HOST=unix:///var/run/docker.sock

networks:
  reverse_proxy:
    external: true
```

Folgende Einträge sind zu bearbeiten:

Zeile	Name	Wert
15-2 6	Labels für Traefik	Die gesetzten Labels dienen zur Traefik-Konfiguration. Je nach Nutzung von Traefik unterscheiden sich diese von unserem Beispiel.

Diese Jenkins-Instanz erhält Zugriff auf die docker.sock Datei, welche es Jenkins erlaubt mit der Docker-Engine zu kommunizieren und neue Container außerhalb des Jenkins-Containers zu starten.

Anschließend muss im Jenkins-Ordner auch eine **jenkins_lts_alpine_dood.Dockerfile** Datei erstellt werden und folgendes eingefügt werden:

```
FROM jenkins/jenkins:lts-alpine
ARG DOCKER_GID=999
USER root
RUN apk update --no-cache && apk add --no-cache docker-cli
docker-compose
RUN delgroup ping
RUN addgroup -g $DOCKER_GID docker
RUN addgroup jenkins docker
USER jenkins
```

Nun wird Jenkins gestartet:

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

Beim Installieren zeigt die CLI ein generiertes Passwort. Dieses wird für die erste Anmeldung bei Jenkins benötigt:

```
jenkinsci-docker |
jenkinsci-docker | *****
jenkinsci-docker | *****
jenkinsci-docker | *****
jenkinsci-docker |
jenkinsci-docker | Jenkins initial setup is required. An admin user has been created and
jenkinsci-docker | a password generated.
jenkinsci-docker | Please use the following password to proceed to installation:
jenkinsci-docker | 1234567abcd44f2a8528784321bcva
jenkinsci-docker |
jenkinsci-docker | This may also be found at:
jenkinsci-docker | /var/jenkins_home/secrets/initialAdminPassword
jenkinsci-docker |
jenkinsci-docker | *****
jenkinsci-docker | *****
jenkinsci-docker | *****
jenkinsci-docker |
```

Anschließend muss auf die angegebene Domain für Jenkins gewechselt werden und das Passwort in das Eingabefeld eingetragen werden:

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/var/jenkins_home/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

Danach muss die 'Suggested Plugins' ausgewählt werden. Nach dem Installieren muss ein Admin-User mit sicheren Zugangsdaten erstellt werden. Nachdem dies erledigt wurde, begrüßt uns das Jenkins-Dashboard:

Jenkins

search

🔔 2 👤 Towl 🚪 log out

Dashboard

New Item

People

Build History

Manage Jenkins

My Views

Lockable Resources

New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

add description

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job →

Set up a distributed build

Set up an agent →

Configure a cloud →

Learn more about distributed builds ↗

REST API

Jenkins 2.263.1

Konfiguration

Vorwort

Zur Zeit ist die Konfiguration für das Frontend (speziell die Labels des Webserver) im Repository enthalten. Dies bedeutet, dass das automatisch erzeugte Frontend auf die festgelegte Konfiguration greift und nicht auf etwaige Änderungen durch Ihr System oder Anwendungsfall.

Hierfür empfehlen wir, das Frontend entweder manuell zu starten wie auf Seite 6 gezeigt wird oder das Repository zu forken und die Volumes und Labels des Frontends (falls ein Traefik eingesetzt wird) im Repository zu ändern!

Bitbucket Server Integration

Um das Jenkins mit Bitbucket nutzen zu können, wird die Bitbucket Server Integration benötigt. Hierfür gibt es bereits eine sehr gute Anleitung von Atlassian:

[Textform](#) [Youtube Video](#)

Beim Erstellen der Bitbucket Server Instanz geben wir folgende Daten ein, wie im Tutorial beschrieben:

The screenshot shows the 'Bitbucket Server integration' configuration page. It includes a section for 'Instance details' with the following fields:

- Instance name: Bitbucket HHN
- Instance URL: https://bitbucket-student.it.hs-heilbronn.de
- Personal access token: Bitbucket HHN - Bitbucket admin token

There are buttons for 'Add', 'Test connection', and 'Delete'. A note at the bottom says 'Jenkins can connect with Bitbucket Server.'

Folgende Werte sind zu ändern:

Name	Wert
Instance Name	Bitbucket HHN
Instance URL	https://bitbucket-student.it.hs-heilbronn.de
Personal Access Token	Access Token, welches im Tutorial erstellt wurde

Docker Plugin

Daneben wird noch ein weiteres Plugin benötigt: das docker-plugin. Dieses ermöglicht Jenkins, die eingebundene Schnittstelle zu Docker zu nutzen. Dieses ist unter den verfügbaren Plugins zu finden.

Nach dem Installieren wird zu Manage Jenkins > Manage Nodes and Clouds > Configure Clouds gewechselt und somit folgende Daten einzufügen:

Docker

Name

Docker Host URI

Server credentials - none - + Add

Advanced... Test Connection

Enabled ☒

Error Duration

Expose DOCKER_HOST ☐

Container Cap

Docker Agent templates...

Delete cloud

Folgende Werte sind zu ändern:


Name	Wert
Name	docker
Docker Host URI	unix:///var/run/docker.sock
Server Credentials	none
Enabled	✓

Job Erstellung

Nachdem dies erledigt wurde, muss auf das Dashboard zurück gewechselt werden und ein neuer Job erstellt werden:


Enter an item name

» Required field




Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.




Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



Folder

Der Item Name ist covidcheckin. Die Art ist Pipeline.

Nun wird eine große Übersichtsseite angezeigt, bei der bestimmte Felder geändert werden müssen:

Spalte	Name	Wert
General	Discard old builds	✓
Build Trigger	Bitbucket Server trigger build after push	✓
Pipeline	Definition	Pipeline script from SCM

Pipeline

Definition: Pipeline script from SCM

SCM: Bitbucket Server

Credentials (for build auth): mdepta/***** [Add](#)

Bitbucket Server instance: Bitbucket HHN

Project name: LabSWPS_SE
Using 'LabSWPS_SE' at https://bitbucket-student.it.hs-heilbronn.de/projects/LABSWPSE

Repository name: LabSWP_2020_WS_Team6
Using 'LabSWP_2020_WS_Team6' at https://bitbucket-student.it.hs-heilbronn.de/projects/LABSWPSE/repos/labswp_2020_ws_team6/browse

Clone from: Primary Server

Optional SSH Credentials: - none - [Add](#)

Without credentials, Jenkins can't check out source code from non-public repositories.

Branches to build: Branch Specifier (blank for 'any'): */master [Add Branch](#)

Additional Behaviours: [Add](#)

[Test connection](#)

Script Path: Jenkinsfile

In der Pipeline ist zu ändern:

Name	Wert
SCM	Bitbucket Server
Credentials	Credential des HHN Account auswählen, welches im nächsten Bild erstellt wird
Definition	Pipeline script from SCM
Project name	LabSWPS_SE
Repository name	LabSWP_2020_WS_Team6
Clone from	Primary Server
Script Path	Jenkinsfile

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain: Global credentials (unrestricted)

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username:

Password:

ID:

Description:

Add Cancel

Name	Wert
Domain	Global credentials
Kind	username with password
Scope	Global
Username	Username des HHN Accounts
Password	Passwort des HHN Accounts
Description	Bitbucket HHN Account

Abschließend mit Save betätigen.

Pipeline starten

Nun kann auf der Seitenleiste links mit 'Build Now' das Projekt gestartet werden. Es wird ansonsten automatisch gestartet wenn ein neuer Build erscheint.

Mailserver

Mailcow ist ein komplettes Package für Mailversand. Es hat einen Ein- und Ausgangsserver, Unterstützung für alle neuesten Standards und ein Webpostfach. Aufgrund der Menge an Konfigurationen lassen wir hier den Mailserver aus und verweisen auf die [Dokumentation vom Mailcow Projekt](#).

Mailversand mit PHPMailer

Als einfache Alternative haben wir den PHPMailer mit einem GMail Account ausgestattet. In den nachfolgenden Paragraphen wird gezeigt, wie dieser Konfiguriert wird. Sollten Sie Mailcow oder einen anderen E-Mail Ausgangsserver benutzen, so können Sie hier nachlesen wie der PHPMailer zu konfigurieren ist.

Konfiguration

Damit der Mailversand stattfinden kann müssen Konfigurationen in den Dateien **sendmail.php**, **requestReset.php** und **sendregistermail.php** getroffen werden. Sollte man wie bei unserem Fall Gmail benutzen kann man diese Einstellungen für E-Mails verwenden, zu beachten ist jedoch das Apps Zugriff auf die E-Mail haben müssen.

<https://myaccount.google.com/lesssecureapps>.

```
require('phpmailer/PHPMailerAutoload.php');

$mail = new PHPMailer;
$mail->CharSet = 'UTF-8';
$mail->isSMTP();
$mail->Host = 'smtp.gmail.com'; (1)
$mail->SMTPAuth = true;
$mail->Username = 'team6checkin@gmail.com'; (2)
$mail->Password = 'ommytthetstest';(3)
$mail->SMTPSecure = 'ssl';
$mail->Port = 465;(4)

$mail->setFrom('team6checkin@gmail.com', 'team6checkin'); (5) //Absender email, Anzeige name
$mail->addAddress('graceffa@stud.hs-heilbronn.de'); (6) //Empfänger email, Anzeige name
```

sendmail.php

- (1) Es muss die SMTP des E-Mail Providers eingetragen werden.
- (2) Des Weiteren muss die E-Mail des Absenders eingetragen werden.
- (3) Danach wird das Passwort (Zur Sicherheit über App-Passwörter) eingegeben.
- (4) Zusätzlich wird der Port des E-Mail Providers eingetragen, für Gmail gilt Port 465.
- (5) Die schon eingetragene E-Mail aus Schritt (2), muss nochmals eingetragen werden.
- (6) Zum Schluss wird der Empfänger der CheckIn/CheckOut-Benachrichtigungen eingetragen, also die Coronastelle der Hochschule.


```
$mail->setFrom('team6checkin@gmail.com', 'team6checkin'); (5) /Absender email, Anzeige name  
$mail->addAddress($email); //Empfänger email, Anzeige name
```

sendregistermail.php

```
$mail->setFrom('team6checkin@gmail.com', 'team6checkin'); (5) /Absender email, Anzeige name  
$mail->addAddress($emailTo); //Empfänger email, Anzeige name  
$mail->addReplyTo('no-reply@covidcheckin.de', 'No reply'); //no-reply
```

requestReset.php

Wichtig: Bei den Dateien **sendregistermail.php** und **requestReset.php** muss Schritt (6) ignoriert werden.