Modul 239 Dokumentation

Michalis Apollon Chatzimichalis
Modul 239 – Internetservices anbieten
Marcello Callisto

Impressum

Dokumenteigenschaften

Titel: Modul 239 Dokumentation

Autoren/innen: Michalis Apollon Chatzimichalis

Firma: Vinylo

Verfasser: Michalis Apollon Chatzimichalis

Erstelldatum: 01.02.2021

Veröffentlichungsdatum: 12.04.2021

Seitenanzahl: xx

Auftraggeber

Institution: Technische Berufsschule Zürich

Fachrichtung: Systemtechnik

Klasse: ST18a

Fachlehrer: Marcello Callisto

Änderungsverzeichnis

Version	Status	Datum	Autor	Beschreibung
0.1	Erledigt	01.02.2021	Michalis Chatzimichalis	Doku und Inhaltsverzeichnis mit Überschriften erstellt
0.2	Erledigt	22.02.2021	Michalis Chatzimichalis	Kompetenzen I2 und P erledigt
0.21	Erledigt	22.02.2021	Michalis Chatzimichalis	Kompetenz I1 ausgefüllt
0.22	Laufend	25.02.2021	Michalis Chatzimichalis	Kap. Informieren ausgefüllt
0.3	Laufend	01.02.2021	Michalis Chatzimichalis	Kap. Planen mit GANTT erledigt (Lernjournal wird laufend geführt)
0.4	Erledigt	28.02.2021	Michalis Chatzimichalis	Konzepte (Namens, Sicherheit und Betrieb) erläutert
0.41	In Bearbeitung	06.03.2021	Michalis Chatzimichalis	Entscheidungsmatrizes erledigt
0.5	Erledigt	20.03.2021	Michalis Chatzimichalis	Webserver mit NGINX und Docker (mit Domäne einrichten)
0.51	Erledigt	31.03.2021	Michalis Chatzimichalis	Wordpress mit Docker einrichten
0.6	Erledigt	30.03.2021	Michalis Chatzimichalis	Mailserver mit Docker eingerichtet
0.7	Nicht bearbeitet	30.03.2021	Michalis Chatzimichalis	Funktionelle Tests
0.8	Nicht bearbeitet	01.04.2021	Michalis Chatzimichalis	Kap. Auswerten ausgefüllt
0.9	Nicht bearbeitet	08.04.2021	Michalis Chatzimichalis	Review Check 1 von 2
0.91	Nicht bearbeitet	10.04.2021	Michalis Chatzimichalis	Präsentation bearbeitet und geübt
1.0	Nicht bearbeitet	10.04.2021	Michalis Chatzimichalis	Review Check 2 von 2

Tabelle 1: Änderungsverzeichnis

Inhaltsverzeichnis

1 Vorwort 7
1.1 Management Summary 7
1.2 Projekthintergrund 7
1.3 Darstellung und Ablauf 7
1.4 Modulidentifikation 8
2 Ausgangslage (I1) 9
2.1 Firma vinylo 9
2.2 IST-Zustand 9
2.3 SOLL-Zustand 10
2.4 Dimensionierung 11
2.5 Lastprofil 11
2.6 Skalierbarkeit 12
2.7 Art des Hostings 12
3 Informieren 13
3.1 Kompetenzraster (I2) 13
3.1.1 Virtual Hosts 13
3.1.2 DNS 13
3.1.3 DNS Tools 14
3.1.4 Proxy 14
3.1.5 DMZ 15
3.2 Protokolle kennen 15
3.2.1 HTTP (P.1) 15
3.2.2 E-Mail (P.2) 21
3.2.3 Secure Shell (P.3) 22
3.3 Hosting-Provider 23
3.3.1 Domain Konfiguration 24
3.3.2 DNS-Propagation (Ausbreitung) 24
3.3.3 WHOIS Überblick 24
3.3.4 DNS-Server Provider 24
3.4 Webserver 25
3.4.1 ACME Protokoll 26

3.4.2 NGINX 26
3.5 Mailserver 26
3.6 Content Management System 27
3.7 Dateitransfer 27
3.8 Datenbanken 27
3.8.1 Redis 28
3.9 Docker 28
3.9.1 Docker Hub 29
3.9.2 Die Vorteile von Docker 29
3.9.3 Docker Orchestration 29
3.10 API 29
3.10.1 REST 29
3.10.2 GraphQL 29
3.11 Postman 30
3.12 Prometheus 30
3.12.1 Wie funktioniert Prometheus 31
3.12.2 Prometheus im Vergleich 32
3.13 Grafana 33
3.14 Splunk 33
3.15 Für Inhouse-Entwickler 33
4 Planen 34
4.1 GANTT/Zeitplan 34
4.2 Lernjournal (L) 34
5 Entscheiden 36
5.1 Namenskonzept (V) 36
5.1.1 Gerätetypen 36
5.1.2 Interne Mitarbeiter 36
5.1.3 Externe Kunden 36
5.1.4 Externe Lieferanten 36
5.2 Sicherheitskonzept (V) 37
5.2.1 Benutzerrechte/Kundenrechte (B1) 37
5.2.2 Massnahmen 38

5.3 Betriebskonzept (V) 38
5.3.1 Anforderungen an den Betrieb 38
5.3.2 Organisation 38
5.3.3 Systemtechnik 39
5.3.4 Systemüberwachung 40
5.3.5 Incident Management 40
5.3.6 Problem Management 40
5.3.7 Change-Management 40
5.3.8 Beschreibung der Aspekte der Sicherheit 41
5.3.9 Anforderungsabdeckung 41
5.4 Risikoanalyse (V) 41
5.4.1 Erklärung 41
5.4.2 Vorgehensweise 41
5.4.3 Risikoanalysetabelle 42
5.4.4 Risikomatrix 42
6 Realisieren 43
6.1 Umgebung 43
6.1.1 Physische mit Diensten 43
6.1.2 Virtuelle Arbeitsfläche 43
6.1.3 SSH Alias erstellen 44
6.2 Installation (SW 1) 44
6.2.1 Ubuntu auf einem Raspberry Pi einrichten 44
6.2.2 Docker und Docker-Compose einrichten 44
6.2.3 Domäne kaufen 44
6.2.4 Domain-Name Konfiguration 45
6.2.5 Portainer (Container Manager) 47
6.2.6 WordPress und DB einrichten (Fehlerhaft) 49
6.2.7 NGINX (Fehlerhaft) 54
6.2.8 Fehlerhaftes Redirect (Fehlerhaft) 58
6.2.9 TLS Zertifikat erstellen (Fehlerhaft) 59
6.2.10 Vinylo über Hosting Provider 64
6.2.11 WordPress einrichten 64

6.2.12 TLS Zertifikat 66
6.2.13 Mailcow....68
6.2.14 Postfix....75
6.2.15 FTP 79
6.2.16 Postman 84
6.3 Troubleshooting (Docker) 86
6.3.1 Installation WordPress, NGINX (Fehlerhaft DigitalOcean) 86
6.3.2 Container Hinzufügen mit Docker-Compose 89
6.3.3 WordPress im Portainer einrichten (Fehlerhaft)....96
6.3.4 WordPress über CLI (Fehlerhaft, keine DB-Anbindung als eigenständiges Container) 97
6.3.5 Obsolet NGINX RP Manager via Portainer....99
6.3.6 Lokale Installation WordPress, Apache....102
6.3.7 Mailcow über Pi (Troubleshooting) 110
6.4 Protokollierung (SW3)....112
6.4.1 Ablageort Logfiles....112
7 Kontrollieren 113
7.1 Funktions- und Lasttests 113
7.1.1 Webserver 113
7.1.2 WordPress (CMS) 113
7.1.3 Mailcow....113
7.1.4 FTP 114
8 Auswerten 114
8.1 Reflexion 115
8.2 Zielerreichnung 115
8.3 Ausblick....115
9 Glossar 116
10 Verzeichnisse 124
10.1 Quellenverzeichnis (L) 124
10.2 Tabellenverzeichnis 124
10.3 Abbildungsverzeichnis....125

1 Vorwort

1.1 Management Summary

Dieses Dokument umfasst meine Projekte und die dazu erworbene Theorie zur eine Menge von technischen Themen rund um das Ausnutzen von Schwachstellen, sei es in einer Web-Applikation oder Computer.

1.2 Projekthintergrund

Dieses Projekt basiert auf das aufgebaute Wissen in der LB1 und der Lektion sowie Eigeninteresse aus dem geschäftlichen sowie privaten Umfeld.

1.3 Darstellung und Ablauf

Als Rechtschreibehilfe wurde die integrierte Überprüfungsfunktion von Word verwendet. Ausserdem wurde die Dokumentation von verschieden Personen auf die Rechtschreibung überprüft.

Es wird unter verschiedenen Textsorten unterschieden. Dafür wurde die Formatierung selbst definiert:

Zitierte Zitierte Texte werden kursiv geschrieben

Text Textausschnitte, welche besonders zu beachten sind, werden fett hervorgehen

URL Links werden hellblau markiert und unterstrichen

Querverweis Querverweise zu Kapiteln oder Inhalte im Dokument werden dunkelblau geschrieben

Glossar-Befehl Werden pink markiert

Abbildungen

Abbildung 1: Beispiel einer Abbildung

Tabellen

Spalte 1	Spalte 2

Tabelle 2: Beispiel einer Tabelle

1.4 Modulidentifikation

Modulnummer	239
Titel	Internetserver in Betrieb nehmen
Kompetenz	Internetserver mit verschiedenen Diensten konfigurieren und in Betrieb nehmen und dabei Sicherheitsvorgaben und betriebliche Anforderungen beachten.
Handlungsziele	1. Anforderungen (Sicherheit, Lastprofil, Datenvolumen, Verfügbarkeit, zu integrierende Applikationen) an einen Internetserver aufnehmen und dokumentieren. 2. Bestehende Infrastruktur (Server, Netzwerk, Dienste) mit den Anforderungen abgleichen und notwendige Anpassungen bzw. Erweiterungen vorschlagen. 3. Erforderliche Einstellungen gemäss Sicherheits- und Betriebskonzept festlegen. 4. Software installieren, konfigurieren und Dienste einrichten. 5. Zugriffsberechtigungen vergeben, sichere Kommunikation und Log-Services einrichten. 6. Internetserver testen (Last-, Sicherheits- und Crashtest).

Tabelle 3: Modulidentifikation

2 Ausgangslage (I1)

2.1 Firma vinylo

Mein fiktives Unternehmen wird die Dienstleistung des Schallplattenverkaufs sowie die Beratung bei einem Kauf und Realisation ein guten Hi-Fi-Setups zu Hause, im Betrieb oder auch wo sonst.

Kontaktdaten der Firma

CEO	Michalis Apollon Chatzimichalis
Mitarbeiter	Bob Baby (Berater) Karl Ice (Berater) Max Mann (Senior Fullstack Engineer) Till Vosul (ITIL Spezialist)
Lieferant	TBD
Post	Die schweizerische Post
Webseite	https://www.vinylo.shop/
E-Mail	info@vinylo.ch

Tabelle 4: Kontaktdaten der Firma

Der Standort dieser Firma befindet sich zurzeit beim Wohnungsblock des CEOs Michalis Chatzimichalis, der tagtäglich einen Firmensitz in der Stadt Zürich sucht.

2.2 IST-Zustand

Der IST-Zustand der Firma ist so weit beschrieben.

Logo

Abbildung 2: vinylo Logo

Team

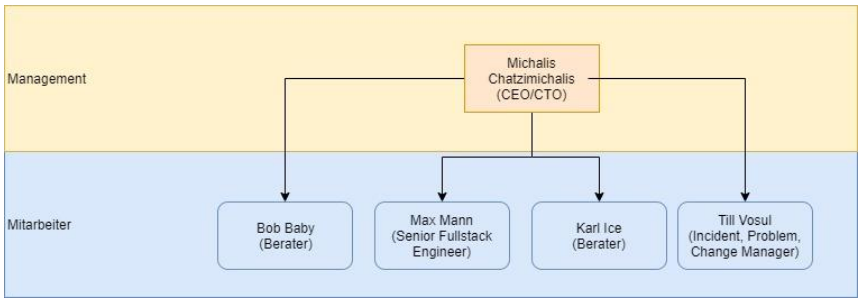


Abbildung 3: Team von vinylo

Infrastruktur

Zurzeit verfügen wir über einen einfachen Raspberry Pi 4, welcher mit 8GB RAM ausgestattet ist und einen 32GB gemountet USB-Laufwerk für die Speicherung aller Daten. Der Datendurchsatz beträgt bei maximaler Leistung 1GB/s

[Link zum Produkt – Digitec]

Quelle 1: Raspberry Pi auf Digitec

Leitung

Die bestehende Leitung ist eine 500/50 Kupferleitung und dient für unsere interne Zwecke, also die Dienste zur Verfügung stellen, ausreichend.

2.3 SOLL-Zustand

Der gewünschter SOLL-Zustand wird nachfolgend beschrieben

Das interne Netzwerk und die Dienste, welcher unsere Firma zur Verfügung erstellt und dessen Zugriffe, werden im untenstehenden Bild detaillierter abgebildet.

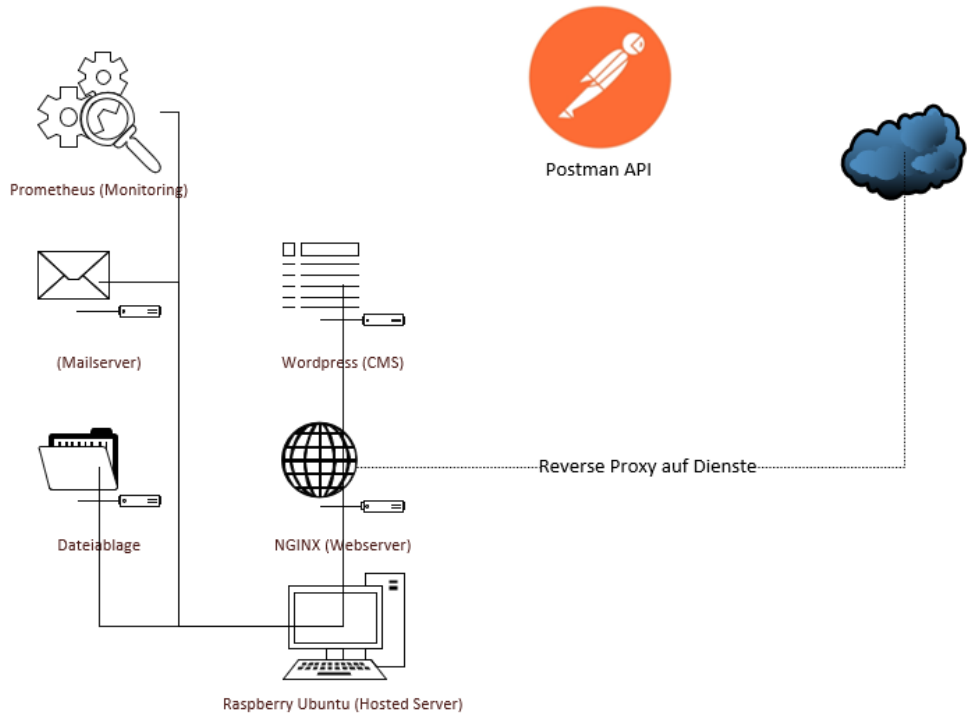


Abbildung 4: SOLL-Zustand der Dienste

Das Logo, Team und die Leitung werden bis auf Weiteres dasselbe sein und wenn wir auf in unseren Analyse-Tools merken, dass die Webseite am «Boomen» ist, werden wir die Leitung auf einiges aufstocken sowie ergibt sich das Potenzial einige neue Gesichter im kleinen Team zu begrüssen.

2.4 Dimensionierung

Die zukünftigen Dimensionen der allgemeinen Firma und physischen Produkte werden im folgenden Abschnitt erläutert. Als Erstes soll die Firma vinylo mit 1 Standort, der über 1 Stock eröffnen. Bei gutem Gelingen und Bewertungen wird ein weiterer, inländischer Standort angekündet.

Die Betriebsunterhalt der Webseite wird weg vom Pi zu einem mächtigeren, physischen und einen virtuellen Server in der Cloud. Der Letztere wird für die Produktentwicklung und Inhouse-Aufrechterhaltung der Webseite.

Die technischen Anpassungen einer zukünftigen Entwicklung wären wie folgt abgebildet

Komponente	Jetzt	Später
Prozessoren	1.5GHz 64-bit quad-Core	2.8GHz
Speicherplatz	30GB	10TB
RAM	8GB	32GB
Datendurchsatz	500/50 Mbit/s Kupfer	1000/100Mbit/S Glasfaser

Tabelle 5: Dimensionierung

2.5 Lastprofil

Dieses Lastprofil dient als technische Abbildung der zahlreichen Kunden. Das Lastprofil habe ich wie folgt unterteilt; Durchschnittliche Benutzer unter der Woche und Wochenende und dies zu den abgebildeten Uhrzeiten. Am Anfang rechnen wir mit wenig Traffic, jedoch nach unsere erste Werbungskampagne auf zahlreiche Social-Media Plattformen u.a auch LinkedIn, wird hoffentlich den Traffic auf das Abgebildete steigern. Durch unseres Top-Angebot an Schallplattenspieler und der grossen Auswahl an Schallplatten sowie die Flexibilität der internen Berater, wird die Website zukünftig etwa die doppelte Anzahl an Aufrufe verfügen.

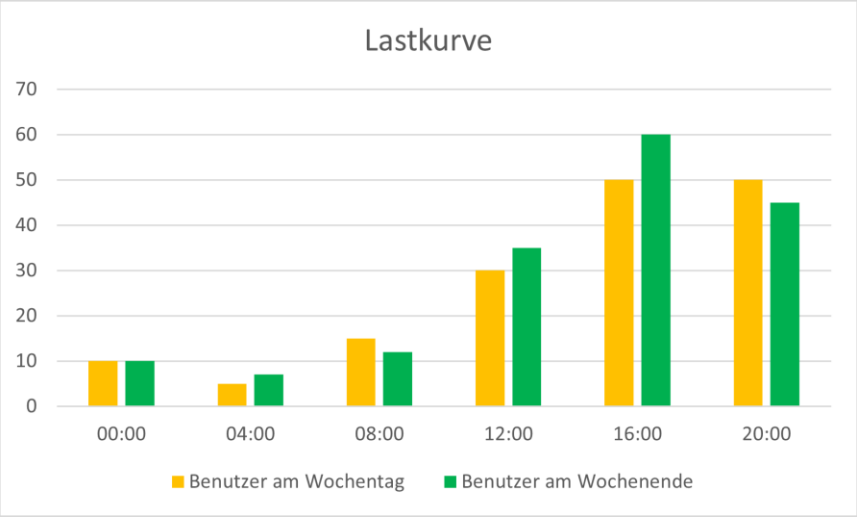


Abbildung 5: Lastprofil der Webseite

2.6 Skalierbarkeit

Zukünftig wäre geplant bei einem riesigen Wachstum an Kunden, mit einfachen, integrierten Dienste, die heutzutage diese frühere, zeitintensive Grossarbeit mit Docker um Vielfaches vereinfacht wurde.

2.7 Art des Hostings

Da wir über das Raspberry Pi verfügen und an keiner Firma angewiesen sind, betreffend das Hosten jeglicher Server ist die definitive Art; **Inhouse**.

3 Informieren

Der erste Unterkapitel dieses Kapitel befassen sich mit den Fragen des Kompetenzrasters und weiterhin mit der grundsätzlichen Informierung über die möglich, einsetzbare Mitteln für mein fiktives Unternehmen.

3.1 Kompetenzraster (I2)

Nachfolgend werden Fragen vom Kompetenzraster abgearbeitet.

3.1.1 Virtual Hosts

Wie kann derselbe Webserver unterschiedliche Websites hosten (Virtual Hosts)?

Webserver können mehrere Websites (Unterwebsites) anbieten, in dem für jede, unterschiedliche Webseite ein sogenannten Virtual Hosts (vHost) erstellen. Der Begriff vHost ist in dem Apache2 Webserver enthalten und zugleich in dem konkurrierenden nginx, jedoch ist bei nginx die Benamung solcher Virtual Hosts, Server Blocks.

3.1.2 DNS

Wie funktioniert DNS?

Der DNS (Domain Name Services) ist ein Hilfstool um Domain-Namen (eine Webseite) in Nummern (IP-Adressen des Seites) zu verwirklichen. Für uns Menschen ist es jedoch sehr schwierig eine unzählige von IP-Adressen zu merken. Dafür dient einen DNS-Server, welcher alle IP-Adressen in unvergessliche Domain-Namen wie srf.ch oder bbc.co.uk, übersetzt. Menschen verwenden die Domain-Namen und Computer die zugehörigen IP-Adressen.

Welche Ressource-Record-Typen gibt es?

DNS gibt eine Datenbank gewisse **Informationselementen** an **Netzwerkressourcen** an. Diese Informationselementen werden mit einer Liste von DNS-Datensatztypen, den Ressource-Records (RRs), kategorisiert und organisiert. **Jeder Datensatz** hat einen Typ (Name und Nummer), eine Verfallszeit (TTL), eine Klasse (CLASS) und typspezifische Daten.

Die wichtigsten und meistverwendeten sind folgende;

RR Typ	Beschreibung	Funktion
A	Address Record	Ordnet eine IPv4 Adresse zu einem Hostname zu
AAAA	IPv6 Address Record	Ordnet eine IPv6 Adresse zu einem Hostname zu
CNAME	Canonical Name Record	Alias eines Namens zu einem anderen: Die DNS-Lookup wird fortgesetzt, indem die Lookup mit dem neuen Namen erneut versucht wird.
MX	Mail Exchange Record	Ordnet einen Domainnamen einer Liste von Nachrichtentransferagenten für diese Domain zu
NS	Name Server Record	Delegiert eine DNS-Zone zur Verwendung der angegebenen autoritativen Nameserver

PTR	PTR Resource Record	
SOA	Start of Authority	Gibt massgebliche Informationen zu einer DNS-Zone an, einschließlich des **primären Namensservers**, der **E-Mail des Domain-Administrators**, der **Domain-Seriennummer** und **mehrerer Zeitgeber** für die Aktualisierung der Zone.
SRV	Service locator	Verallgemeinerter Service Location Record, der für neuere Protokolle verwendet wird, anstatt protokollspezifische Records wie MX zu erstellen.

Tabelle 6: DNS-Ressource-Records

3.1.3 DNS Tools

Wie können beliebige Angaben über einen fremden Server

Beliebige Angaben können über «DNS-Befehle» angeschaut werden, und zwar lautet der Windows-Befehl **nslookup** und auf Unix-basierte Maschinen **dig**. Mit jeglichen Tools (**nmap**) können wir Infos über den Server selbst und allfällige offene Ports erfahren. Der Verkehr zwischen den Client und Server könnten wir sogar mit Webproxy-Tools wie BURPsuite oder ZAP überwachen.

3.1.4 Proxy

Was ist ein Proxy? Welchen Funktionen übernimmt er?

Wenn man von einem Proxy spricht, meint man meistens den **Forward Proxy**. Ein Forward-Proxy ist nämlich ein Mittel, um der Datenverkehr zwischen einen Server (Webserver) und der Anfrager (Client) zu regulieren und mitzuverfolgen.

Der Prozess verläuft folgendermassen;

Der Einsatz eines Forward-Proxy können wir in folgenden Situationen

Ein **Reverse-Proxy** ist das akribische Gegenteil eines Forward-Proxys, und zwar nimmt er alle Anfragen vom Server entgegen und entscheidet anhand vordefinierten Regelns wie und an welchem Endserver/Endanwendung er den Paketen leiten soll. Der Vorteil einer Reverse Proxy ist die Definition der Dienste, den man erstellen kann. Also anstelle von 4 Dienste, welcher über die gleiche Domäne aber via unterschiedliche Ports verfügbar sind, kann man eine «Maske» für den 4 Diensten einrichten, wie folgt;

Plain Text

```
# Without Reverse Proxy
# Domain Name: http://mydomain.com
# Mysql wordpress: http://mydomain.com:10088
# Angular app: http://mydomain.com:7787
# Backend: https://mydomain:9876
```

Plain Text

```
# With Reverse Proxy
# Domain Name: http://mydomain.com
# Mysql wordpress: http://mydomain.com/db
# Angular app: http://mydomain.com/ang
# Backend: https://mydomain/wp
```

Einen **Web-Proxy** ist ein lokales HTTP/S-Proxy, welches in den eigenen Browsereinstellungen gesetzt werden kann, um mit einem Sniffing Tool (ZAP/BURPsuite) den Datenverkehr zu verfolgen und mitzuhören.

3.1.5 DMZ

Was ist eine DMZ und wozu dient sie?

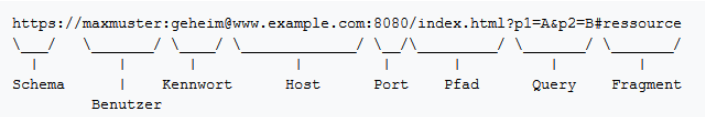
Die DMZ (demilitarisierte Zone) ist ein Netzwerksegment im internen Netzwerk, welcher Dienste für den externen Zugriff herausstellt. Sie ist vom internen Netzwerk getrennt, sodass kein bösartiger Angriff beim der DMZ ablaufen können. In der DMZ stehen meistens Webserver für Kunden, Datenbanken für das Herauslesen der Daten von einem API und weiteres.

3.2 Protokolle kennen

3.2.1 HTTP (P.1)

HTTP (Hypertext Transfer Protocol) wird für das Surfen vom Internet verwendet. Ein User ruft eine Webseite auf (HTTP Request) mit der GET-Methode. Der Webserver antwortet mit der Webseite und dessen Inhalte zurück (HTTP Response) mit einem Statuscode, Metadaten des Dokumentes (Webseite) und schlussendlich den Inhalt des HTML-Dokumentes, also die eigentliche Webseite.

URL (Uniform Resource Locator) ist die eingegebene Adresse in dem Suchfeld des Browsers. Eine URL zusammen mit einer **URN** (Uniform Resource Name) sind die beiden Bestandteile der **URI** (Uniform Resource Identifier). In einer URL werden die Felder wie folgt belegt;



Bestimmte Zeichen sind fürs HTTP reserviert (Link zur Artikel)

- Query-String wird mit dem Fragezeichen **(?)** in der URL eingeleitet
- Das Gleichheitszeichen **(=)** steht zwischen dem Namen eines Parameters und seinem Wert
- Das Et-Zeichen **(&)** steht als Trennzeichen zwischen Parametern im Query-String,
- Zuletzt folgt das Doppelkreuz **(#)** (der Name eines Dokumentenankers).

Quelle 2: Wikipedia URL-Encoding

Die **Methoden**, wie bspw. GET, POST, PUT usw. sind normale Englisch-Verben. POST wird für das Einsenden von Nutzereingebebene Daten verwendet. Die Daten werden bspw. in einem Formularfeld oder einen Kommentarbalken reingeschrieben und verlaufen in einen ähnlichen Prozess wie bei der GET-Methode. Mehr findet man hier.

Quelle 3: Mozilla Liste aller HTTP-Methoden

Die Liste hat uns Hr. Calisto auf dem BSCW zur Verfügung gestellt.

Methoden	Bedeutung	seit Version
GET	Dokument anfordern	0.9
POST	Umfangreiche Formulardaten senden	1.0
HEAD	Nur den Header anfordern	1.0
PUT	Dokument auf dem Server ablegen	1.0
DELETE	Dokument vom Server löschen	1.0
LINK	Verknüpfung erstellen	1.0
UNLINK	Verknüpfung löschen	1.0
TRACE	Proxies im Header zeigen	1.1
CONNECT	Proxy-Zugriff auf einem gesicherten Server	1.1
OPTIONS	Liste von Optionen anfordern	1.1

Tabelle 7: HTTP Methoden mit der Version

GET hat die Begrenzung von **2048 Zeichen** (Pfadlänge wird nicht mitgezählt), jedoch mit POST steht **keine Limitierung** an.

Als letzte Info für die HTTP-Header der Request werden die **Versionen** ernannt. Die weiteren, modernen Versionen (2.0/3.0) werden hier beschrieben.

Version	Vorteile
0.9	• GET [Pfad zur Ressource] > [HTML-Dokument]
1.0	• Version-Info wird nach der Methode hinzugefügt (GET [Pfad] [HTTP/1.0] • Bei der Response wurde ein Status-Code hinzugefügt • HTTP Headers wurden mit Metadaten umfangreicher und nutzvoller • Aufgrund des Content-Types konnten HTML-Dok. hinzugefügt
1.1	• Persistente Verbindungsaubau, um nachfolgend Inhalte schnellstmöglich herunterzuladen • Pipelining, um die Möglichkeit einer zweiten Anfrage zu gestalten, bevor die erste fertig war • Cache Controlling-Mechanismen • Sprache, Kodierung und Typ wurden zu den Metadaten ergänzt • Aufgrund des Host-Headers hat der Server die Möglichkeit, verschiedene Domains unter der gleichen IP-Adresse zu hosten

Tabelle 8: HTTP Versionen

Die **Statuscodes** werden im Header des HTTP-Paketes hinterlegt und nachfolgend sind die 5 verschiedene Kategorien aufgelistet. Sind im Kapitel 10 des RFC 2616 zu finden. Eine aktualisierte Version findet man hier.

Quelle 4: Mozilla Liste aller HTTP Status-Codes

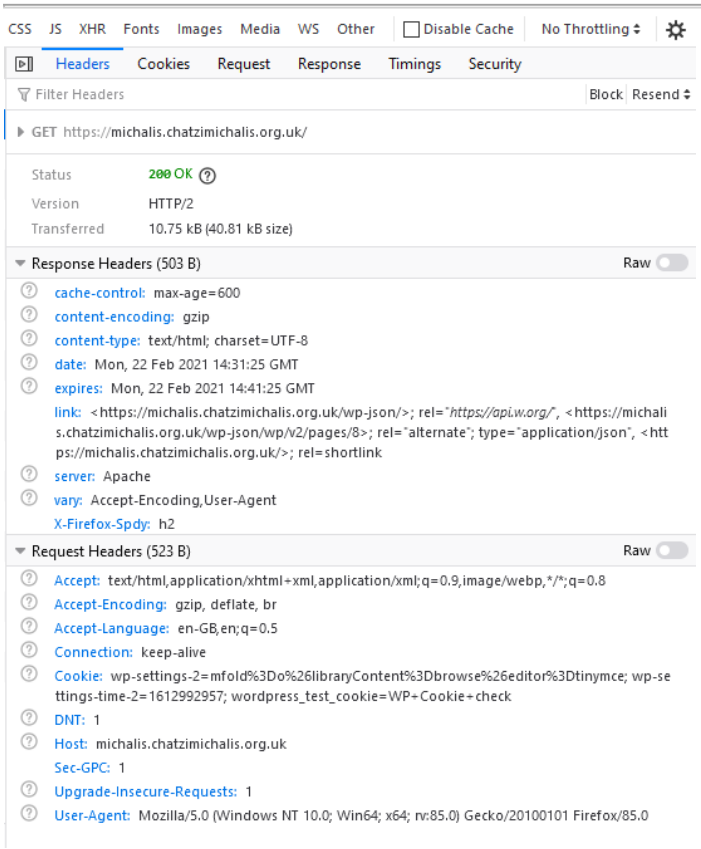
Code-Bereich	Bezeichnung	Beispiele
100-199	Informational Respones	100 - Continue 101 – Switching Protocol 102 – Processing (WebDAV)
200-299	Successful Responses	200 – Succesful (abhängig von der HTTP-Methode) 202 - Accepted 204 – No Content
300-399	Redirects (Umleitungen)	301 – Moved Permanently 302 - Found 308 – Perm. Redirect
400-499	Client errors (Client-seitige Fehler)	400 – Bad Request 401 - Unauthorized 404 - Not Found
500-599	Server errors (Server-seitige Fehler)	500 – Internal Server Error 502 – Bad Gateway 503 – Service Unavailable

Tabelle 9: HTTP Codes

Der gänzliche Aufbau einer Response HTTP-Headers sieht für die Seite michalis.chatzimichalis.org.uk wie folgt aus.

```
C:\Users\apoll>curl -I https://michalis.chatzimichalis.org.uk
HTTP/1.1 200 OK
Date: Mon, 22 Feb 2021 14:31:41 GMT
Server: Apache
Link: <https://michalis.chatzimichalis.org.uk/wp-json/>; rel="https://api.w.org/", <https://michalis.chatzimichalis.org.
uk/wp-json/wp/v2/pages/8>; rel="alternate"; type="application/json", <https://michalis.chatzimichalis.org.uk/>; rel=shor
tlink
Upgrade: h2
Connection: Upgrade
Cache-Control: max-age=600
Expires: Mon, 22 Feb 2021 14:41:41 GMT
Vary: User-Agent
Content-Type: text/html; charset=UTF-8
```

Im Browser sieht die Anfrage und Antwort wie folgt aus. Um auf diese Anzeige zu kommen, muss man mit F12 Inspect Element öffnen und zur Netzwerk Tab wechseln. Darin wählt man den allerersten Eintrag aus.



Der Anhang hinter dem Fragezeichen wird als Query-String bezeichnet. Die einzelnen Formularfelder haben die Form Name=Wert und werden durch & -Zeichen oder Semikola (;) voneinander getrennt. Zusätzlich müssen die eigentlichen Formulardaten URL-codiert werden, weil zahlreiche Zeichen in URLs nicht zulässig sind: Leerzeichen werden durch + ersetzt; fast alle Zeichen, die keine Buchstaben oder Ziffern sind, werden durch ein %-Zeichen und ihren hexadezimalen ASCII-beziehungsweise ANSI-Code ersetzt

Häufig verwendete Befehle für das Aufrufen, Herunterladen von URLs in der Kommandozeile sind cURL und wget. Die beiden Befehle unterschieden sich im Wesentlich durch folgendes;

- *Die große Stärke von **wget** im Vergleich zu **curl** ist die Fähigkeit, **rekursiv** herunterzuladen.*
- ***wget** ist ein reines Kommandozeilenprogramm. Es gibt keine Lib oder ähnliches, aber die Funktionen von **curl** werden von libcurl unterstützt.*
- ***curl** unterstützt FTP, FTPS, HTTP, HTTPS, SCP, SFTP, TFTP, TELNET, DICT, LDAP, LDAPS, FILE, POP3, IMAP, SMTP, RTMP und RTSP. **wget** unterstützt HTTP, HTTPS und FTP.*
- ***curl** lässt sich auf mehr Plattformen als wget aufbauen und ausführen.*
- ***wget** ist unter einer freien Software-Copyleft-Lizenz (der GNU GPL) veröffentlicht. curl ist unter einer freien Software-Permissiv-Lizenz (einem MIT-Derivat) veröffentlicht.*
- ***curl** bietet Upload- und Sendefunktionen. wget bietet nur einfache HTTP-POST-Unterstützung.*

Link: Difference wget vs curl

Quelle 5: Unterschied wget und curl

LB1 HTTP Antworten (P)

Teil A – Multiple Choice

Welcher GET-Request wird vom Server erfolgreich beantwortet?

Inhalt weniger als 1024 Zeichen

Welche HTTP-Methode wurde mit der Version HTTP 1.0 eingeführt und dient insbesondere dem Versand von umfangreichen Formulardaten?

POST

Welche Methode ergibt folgenden Output? (Hands-on http Übung)

HEAD

Dieser Status-Code tritt auf, wenn ein moderner Webbrowser, der die Daten noch im Cache hat, eine Anfrage an den Server sendet. Stellt der Server fest, dass die sich die Daten seit dem letzten Ladezeitpunkt nicht geändert haben, antwortet er mit folgendem Code?

304 Not Modified

Welches Zeichen in einer GET-Anfrage leitet den Querystring ein?

?

Was bezweckt der HTTP-Header Parameter «Cache-Control»?

Definiert, wie lange eine Seite im Browser des Besuchers gespeichert wird

Wohin gehört der Eintrag Connection: close?

In den Header der Server-Antwort

Welche der folgenden Aussagen stimmt?

Bei der GET Methode können nur ASCII-Charakter übermittelt werden

Teil B – Offene Fragen

Was verstehen Sie unter dem Begriff QueryString im Zusammenhang mit einem http Request mit der Methode GET?

Unter dem QueryString versteht man den gesuchten Wert. Dieser QueryString wird mit einem Frageziechen hinter der eigentlicher URL dargestellt.

Wie arbeiten TCP und HTTP zusammen?

Wenn man eine Webseite aufruft, wird zuerst im lokalen DNS-Cache gesucht, ob sich die Adresse dort befindet. Wenn sie nicht gefunden wird, versucht der Router die Adresse im Netz zu finden. Wenn der Router die Webseite gefunden hat, versucht er den 3-Way

Handshake aufzubauen. Der Client schickt ein SYN-Paket am Webserver und der Webserver schickt einen SYN-ACK Paket zurück am Client. Der Client bestätigt den erfolgreichen Verbindungsaufbau mit einem ACK-Paket.

Auf welchen Ebenen des TCP/IP-Stacks (Layer) befinden sich diese Protokolle?

TCP befindet sich auf Layer 3 – Transport Layer und HTTP auf Layer 4 – Application Layer im TCP/IP Stack

Frage 11

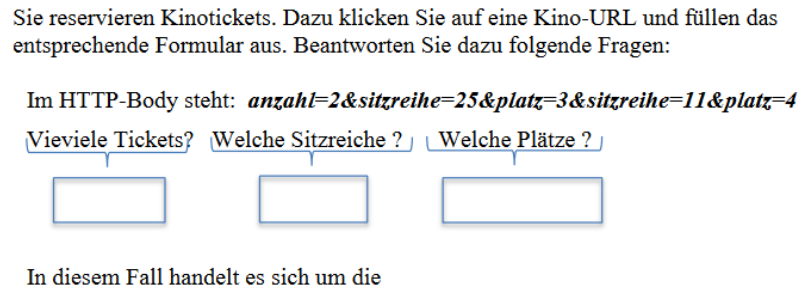


Abbildung 6: Frage 11 des Tests

Tickets: 2, Sitzreihe; 25 und 11, Plätze: 3 und 4. Hier handelt es sich um die POST-Methode.

Erklären sie folgende Statuszeile: HTTP/1.1 301 Moved Permanently

Dieser Statuscode zeigt eine erfolgreiche Umstellung der Website-Location.

Weshalb benutzen Webadmins diesen Code? Bemerkt der Benutzer beim Browsen etwas?

Webadmins nutzen diesen Code, um die Webserver mitzuteilen, dass die suchende Website umgeleitet wurde. Die Benutzer Nein, da

Nennen Sie drei wesentliche Unterschiede zwischen der GET- und POST-Methode?

- GET ist im Gegensatz zur POST-Methode sicher
- Bei POST wird auch beim fehlerhaften Request ein Body mitgesendet (HTML-Code)
- GET ist im Vergleich zur POST idempotent. Mehr zur Idempotent im Glossar

Was bedeuten die Status-Codes aus den Gruppen 3xx, 4xx, 5xx. Nennen Sie je ein Beispiel.

3xx: Umleitungen; 307 – Temporary Redirect

4xx: Clientseitige Fehlermeldungen; 401 - Unauthorized

5xx: Serverseitige Fehlermeldungen; 502 – Service Unavailable

Nennen Sie die wichtigste Neuerung von HTTP/1.1 gegenüber HTTP/1.0.

Die wichtigste Neuerung war die persistente Verbindungsaufbau.

3.2.2 E-Mail (P.2)

POP3 (Post Office Protocol 3) ist ein mittlerweile veraltetes Mail-Protokoll, welcher für das **Empfangen** von E-Mails dient. Mit POP3 werden auf dem Server gespeicherten Mails beim Abrufen vollständig auf dem Computer übertragen, sodass bei einem Unterbruch an der Internetverbindung, die Mails gelesen werden könnten. Dies erfordert jedoch eine Menge Speicherplatz, je nach Grösse der Mails. In der modernen, überall verfügbaren Inhalt-Welt ist POP3 nicht geeignet. Für eine detaillierte Erklärung der technischen Seite vom POP3 und die Nachrichten an sich, findet Ihr hier.

Quelle 6: POP3 im Detail

IMAP (Internet Message Access Protocol) dient zu dem Empfangen vom E-Mail und hat das Vorteil gegenüber POP3, dass die E-Mails von mehreren Computern zugegriffen werden können. Bei IMAP bleiben die Emails auf dem Server (es sei denn man löscht sie), es werden Ordner unterstützt, in die man die Emails verschieben kann. Außerdem werden zum Beispiel Flags auf dem Server gespeichert, wenn eine E-Mail gelesen wurde. Für eine detaillierte Erklärung der technischen Seite vom IMAP und die Nachrichten an sich, findet Ihr hier.

Quelle 7: IMAP im Detail

SMTP (Simple Messaging Transfer Protocol) dient zu dem Versenden von E-Mails. Die E-Mails werden vom Computer an den Mailserver geschickt, wo sie dann bearbeitet werden. SMTP wird auch für das Hin- und Herzuschicken zwischen Mailservern zuständig. Der Statuscode (250 OK) wird für das erfolgreiche Versenden verwendet. Für eine detaillierte Erklärung der technischen Seite vom SMTP und die Nachrichten an sich, findet Ihr hier.

Quelle 8: SMTP im Detail

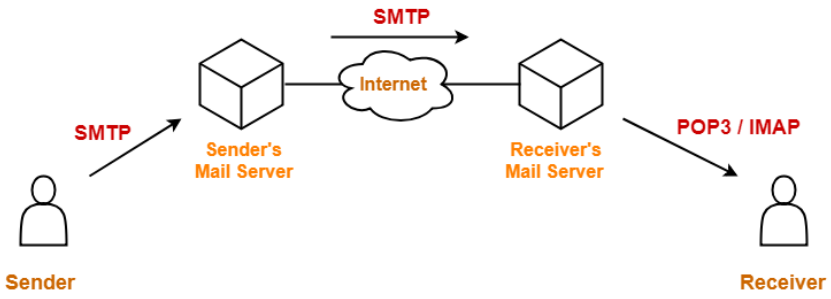


Abbildung 7: E-Mail-Verkehr

Wie ein Mail Server funktioniert.

Wenn ein User auf seinem E-Mail-Client (GMAIL, Thunderbird usw.) auf sein Postfach zugreift und einen E-Mail verschickt wird dies durch den sogenannten **MUA** (Mail User Agent) geregelt. Der baut mit dem Mailserver eine Verbindung auf, in dem er nach seinem MX Record sucht und die IP-Adresse ausfindig macht. Nachdem er die Nachricht über SMTP am Mailserver erlangt hat, schaut der Mailserver für das MX-Record des

Zielservers und übergibt dann die Aufgabe an dessen Server. Dies nennt man **MTA** (Mail Transfer Agent). Dieser Server lagert die Nachricht und wartet auf eine Postfachverbindung vom Empfänger, damit er über sein MUA die Nachricht empfangen kann.

3.2.3 Secure Shell (P.3)

SSH, was Secure Shell ausgeschrieben bedeutet, wird für die sichere Fernverbindung universell verwendet. SSH bietet ein Kennwort oder öffentlichem Schlüssel basierende Authentifizierung und verschlüsselt Verbindungen zwischen zwei Netzwerkendpunkten. Es ist eine sichere Alternative zu herkömmlichen Anmeldeprotokollen (wie telnet, rlogin) und unsicheren Dateiübertragungsmethoden (wie FTP).

SSH bietet nicht nur eine starke Verschlüsselung, sondern wird auch häufig von Netzwerkadministratoren verwendet, um Systeme und Anwendungen aus der Ferne zu verwalten, Software-Patches bereitzustellen oder Befehle auszuführen und Dateien zu verschieben.

Weiteren Einsatzzwecken wären **SCP** (Secure Copy) oder **SFTP** (Secure FTP).

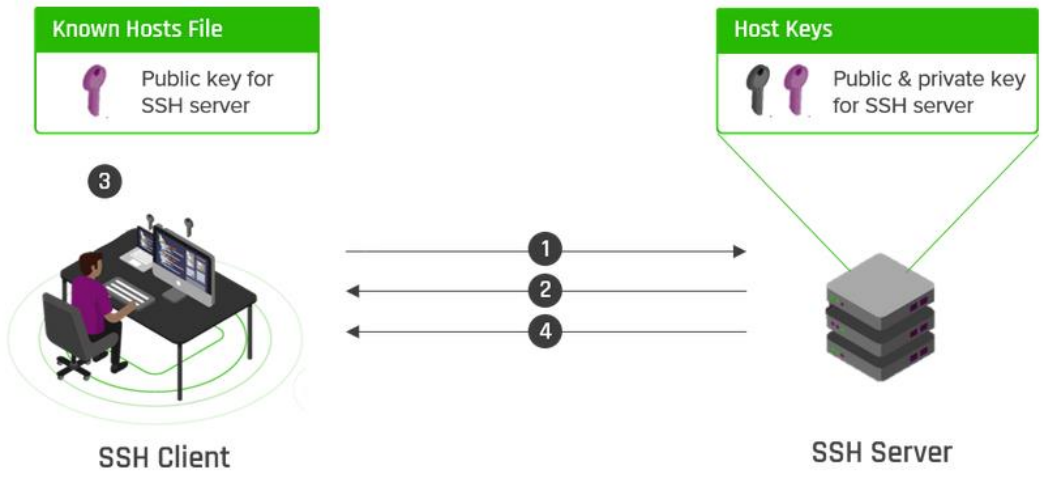


Abbildung 8: SSH Verbindungsaufbau

Die Schritte lauten wie folgt

1. Client initiiert Verbindung zum SSH-Server.
2. Der Server sendet seinen öffentlichen Schlüssel an den Client.
3. Der öffentliche Schlüssel des Servers wird in der Datei der bekannten Hosts des Clients gespeichert.
4. Der Client und der Server handeln die Verbindungsparameter aus und stellen die Verbindung her.

Es gibt zwei verschiedene Wege wie die PKI (Public Key Infrastruktur) verlaufen kann. Entweder über symmetrische oder asymmetrische Verschlüsselung. Die zwei Vorgänge werde ich im folgenden Abschnitt ein bisschen näher bringen

Symmetrische Verschlüsselung

Bei der symmetrischen Verschlüsselung, auch bekannt als Secret Key-Verschlüsselung, wird ein einziger Schlüssel zum Ver- und Entschlüsseln von Daten verwendet. Sie müssen diesen Schlüssel mit dem Empfänger teilen. Nehmen wir an, Sie möchten schreiben: „Ich liebe dich, Mama“, dann würden Sie Ihre E-Mail schreiben und dann einen geheimen Schlüssel zum Verschlüsseln festlegen. Wenn Mama dann die Nachricht erhält, würde sie den geheimen Schlüssel eingeben, um die E-Mail zu entschlüsseln.

Asymmetrische Verschlüsselung

Erstens muss ein öffentlicher Schlüssel veröffentlicht werden, um die Daten zu verschlüsseln. Zweitens wird ein privater Schlüssel zur Entschlüsselung der Daten verwendet.

Der öffentliche Schlüssel und der private Schlüssel sind nicht dasselbe, aber sie sind miteinander verknüpft. Sie erstellen Ihre Nachricht und verschlüsseln sie dann mit dem öffentlichen Schlüssel des Empfängers. Danach müsste der Empfänger, wenn er Ihre Nachricht entschlüsseln möchte, dies mit seinem privaten Schlüssel tun. Halten Sie den (privaten) Schlüssel immer privat, am besten wäre es, dies lokal zu speichern.

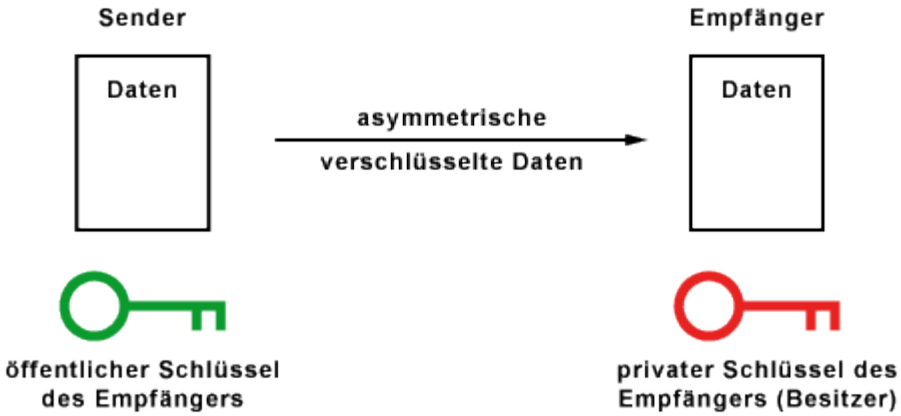


Abbildung 9: Asymmetrischer Schlüsselvorgang

3.3 Hosting-Provider

Um den eigentlichen Betriebsunterhalt meiner Onlinepräsenz zu gewährleisten, muss ich mich für einen Hosting-Provider entscheiden. Folgende Hosting-Provider habe ich einerseits aus dem privaten Umfeld wie auch vom geschäftlich gefunden; **DreamHost**

Ich habe mich für ein Shared-Hosting, da dies bei uns im privaten Umfeld, die einzige Option ist. Bei DreamHost kann man nach einem Domänenkauf entscheiden, wie

Die Schritte, die man für eine erfolgreiche Anlegung und Bereitstellung einer Domäne bei einem Hosting Provider braucht, sind folgende:

1. Domäne registieren
2. Eine Hosting-Option für die Domäne auswählen

3.3.1 Domain Konfiguration

Fully Hosted – Wenn eine Domain "Fully Hosted" ist, wird sie auf einem Webserver eingerichtet und DNS-Einträge zugewiesen. Nach der DNS-Zuweisung können wir dann die Website im Internet besuchen. Dazu sind ein paar Schritte erforderlich:

Redirect – Leitet auf eine Fully Hosted-Website um. Eine Anfrage auf diese Webseite/IP-Adresse wird auf die konfigurierte Weiterleitungsadresse weitergeleitet. Die Umleitung ändert nur, wo die Website gehostet wird. Ihre MX-Einträge (Mail) werden in keiner Weise geändert.

Mirrored – Spiegelt eine Fully Hosted-Website. Ein Mirror ermöglicht es Ihnen, Ihre Inhalte für eine Site (Site A) auf Ihren DreamHost-Server hochzuladen, aber die URL einer anderen Site (Site B) auf demselben Konto zu verwenden, um die ursprüngliche Site (Site A) anzuzeigen. Technisch gesehen ist eine Mirror-Domain ein Server-Alias auf dem DreamHost-Server, der es Ihnen ermöglicht, auf denselben DreamHost-Site-Inhalt unter mehr als einer Adresse zuzugreifen. So können Sie z.B. sowohl eine .com- als auch eine .net-Site haben, die sich den gleichen Site-Inhalt teilen, oder Sie können auf Ihre Site live zugreifen, bevor Ihre Domain eingerichtet ist. Eine Mirror-Domain kann nur eine bestehende Domain bei DreamHost "spiegeln", da es sich lediglich um einen Server-Alias handelt. Eine Mirror-Domain ist ein "Spiegel" nur in Bezug auf das Adress-Aliasing; es ist kein physischer Spiegel und es gibt keine Inhaltsduplizierung, noch ist es eine "Cloaking"- oder "Domain-Greif"-Einrichtung.

DNS Only – Erstellt für die Domäne einen DNS-Eintrag mit dem A-Record der öffentlichen Adresse des Routers, sodass die Namenserver von DreamHost wissen, wohin sie den Traffic auf der Domäne verleiten sollen. Da ich die Webseite mit dem Raspberry Pi aufsetzen werde, muss ich diesen Eintrag so angeben.

Zuletzt gibt es die Option **Parked Domain** zu haben, was nichts anderes ist, als eine reservierte Domäne und dient als Placeholder, bis der Inhaber der Domäne sich entscheidet, in welchem Modus die Domäne konfiguriert werden soll.

3.3.2 DNS-Propagation (Ausbreitung)

Info https://help.dreamhost.com/hc/en-us/articles/215840248-DNS-propagation-overview

3.3.3 WHOIS Überblick

WHOIS ist das Tool, mit dem Sie Details zur Registrierung von Domänennamen nachschlagen können. Diese Details enthalten Informationen über das Datum, an dem die Domain registriert wurde, ihr Ablaufdatum, Informationen über den Registranten, Nameserver und den Domain-Registrar. Wenn eine Domain registriert wird, verlangt die ICANN, dass diese Informationen in einer öffentlichen Datenbank aufgeführt werden, die von jedem eingesehen werden kann, der das WHOIS-Protokoll verwendet. Mehr dazu hier.

3.3.4 DNS-Server Provider

Cloudflare bspw. bietet folgendes zusammen mit der Hosting Provider

- **Benutzererfahrung**: Eine reaktionsschnelle Website erhöht die Benutzerzufriedenheit und ermutigt zu wiederholten Besuchen.
- **SEO steigern**: Eine höhere Website-Geschwindigkeit ist ein Faktor zur Verbesserung Ihres Suchmaschinen-Rankings sowie zur Verbesserung der E-Commerce-Konversionsraten.
- **Kostenloses Universal SSL**: Hilft, Ihre Website sicherer zu machen und Ihre SEO zu verbessern. Sehen Sie sich den Blog-Beitrag von Cloudflare dazu an. Bitte beachten Sie, dass es sich hierbei nicht um ein voll funktionsfähiges SSL-Zertifikat handelt.
- **Hoher Traffic**: Die richtige Verwendung eines CDN kann die Auswirkungen eines starken Anstiegs des Datenverkehrs auf Ihrer Website abmildern oder reduzieren.
- **Website Protection**: Cloudflare schützt vor bösartigem Traffic und Internetangriffen.
- **Verringerung der Serverlast**: Das Zwischenspeichern Ihrer Inhalte in einem CDN reduziert die Last auf Ihrem Server und die verwendete Bandbreite. DreamHost bietet unbegrenzte Bandbreite für alle Hosting-Pläne, so dass dies nicht so sehr ein Problem darstellt, wie die Reduzierung der Server-Ressourcen.

3.4 Webserver

Für den Frontend-Betrieb habe ich die folgende Webserver gefunden und eine kleine Übersichtstabelle erstellt

Webserver	Funktionen	Vorteile	Nachteile
Apache	• Laden von dynamischen Modulen • Load Balancer • Auto-Indexing • verbessertes API • IPv6	+ etablierter Dienst	
NGINX	• Reverse Proxy • Load Balancer • HTML WebSockets • IPv6	+ einfachere Handhabung + für statische Seiten geeignet	- too much
Caddy	• Reverse proxy • Load balancer • API gateway • Ingress controller • Prozessüberwachung • Taskplaner		

Tabelle 10: Webserver Vergleich

Kompatibilität mit dem Hosting-Provider ist mit DreamHost rüstet alle ihrer Webseiten mit dem Apache Webserver aus, da es über. Der Wechsel auf NGINX (Engine X) ist jedoch möglich. Dafür ist eine Einstellungsanpassung notwendig

3.4.1 ACME Protokoll

Bei der Zertifizierung der Webseite mit HTTP wird dies bei meisten privaten Anwender über die Organisation Let's Encrypt erledigt. Let's Encrypt verwendet für die Ausstellung und Verwaltung von Zertifikaten das standardisierte Protokoll namens ACME

3.4.2 NGINX

NGINX handelt mit Blocks, welche in zwei Gruppen unterteilt werden, **Server** und **Location** Blocks.

Ein **Serverblock** ist eine Teilmenge der Nginx-Konfiguration, die einen virtuellen Server definiert, der für die Bearbeitung von Anfragen eines bestimmten Typs verwendet wird. Administratoren konfigurieren oft mehrere Serverblöcke und entscheiden, welcher Block welche Verbindung basierend auf dem angeforderten Domain-Namen, Port und der IP-Adresse behandeln soll.

Ein **Locationblock** lebt innerhalb eines Serverblocks und wird verwendet, um zu definieren, wie Nginx Anfragen für verschiedene Ressourcen und URIs für den übergeordneten Server behandeln soll. Mehr dazu hier (auf Englisch).

Quelle 9: Wie ist die NGINX Konfiguration aufgebaut

3.5 Mailserver

Ein Mailserver ist der Orchestrator der ganzen Mailthematik und verwaltet sowohl die Ablage der Mails auf einer eingerichteten Datenbank als auch der verschlüsselten Datenaustausch der E-Mailverkehr.

Für den Mailserver-Betrieb habe ich die folgende Mailserver gefunden und eine kleine Übersichtstabelle erstellt.

Mailserver	Funktionen	Vorteile	Nachteile
Axigen	• Skalierbar und konfigurierbar • SMTP, POP3, IMAP, WebMail (Desktop- und Mobile-Versionen) • enthält Listenserver, Logging, Reporting und FTP-Backup • WebAdmin-Schnittstelle	+ kompatibel mit Linux-Distributionen, FreeBSD, Solaris und Windows + Modulares Aufbau der Services	- Kostenpflichtig ab bestimmte User # - arbeitet auf x64-Architekturen
Postfix	• Container Unterstützung • Saubere Junk-Mail-Verwaltung • Unterstützt sehr viele Protokolle • MySQL, PostgreSQL, SQLite Unterstützung	+ Open Source + low Overhead	
Mailcow	• Black- und Whitelists pro Domain und pro Benutzer • Mail-Tags dem Betreff voranstellen oder Mail in Unterordner verschieben (pro Benutzer)	+ Docker Image	- viel Overhead

	• Fail2ban-ähnliche Integration • Quarantäne-System • Antivirus-Scanning inkl. Makro-Scanning in Office-Dokumenten • Integriertes Monitoring		

Tabelle 11: Mailserver Vergleich

3.6 Content Management System

Ein CMS (Content Management System) wird für die Darstellung und Verwaltung von anzuzeigenden Inhalten. Diese Inhalte können von einfachen Bildern hinzu voll entwickelte Webseiten. Hinter den meisten CMS-Programme steht ein Portal, auf den der Kunde sich anmelden kann und seine Inhalte anrichten

CMS	Funktionen	Vorteile	Nachteile
WordPress	• Benutzermanagement • Medienmanagement • SEO out-of-the-Box • Einfachheit bei der Veröffentlichung von Seiten/Artikeln	+ Sehr breite Community + für Anfänger geeignet + Grosse Bibliothek an Plugins	-Plugins mit Sicherheitslücken -für
Joomla	• Benutzerverwaltung • Medienverwaltung • Bannerverwaltung • Web Services		
Drupal	• Mehrsprachig OOBE • Web Services		

Tabelle 12: CMS Vergleich

3.7 Dateitransfer

Um eine erfolgreiche Datentransfer zwischen interne Mitarbeiter sowohl die Herausstellung von Dateien für Kunden, müssen wir über eine Dateitransfer Software verfügen, da FTP-Servern heutzutage als sehr unsicher gelten.

Dateitransfer	Funktionen	Vorteile	Nachteile
Dropbox			
NextCloud			
Filecoin			

Tabelle 13: Dateitransfer Vergleich

Das Entscheidungsmatrix findet unter Kap hier.

3.8 Datenbanken

Für die Aufrechterhaltung und Deponierung aller E-Mails verwendet der Mail-Server eine mySQL-Datenbank

Für Monitoring mit Prometheus/Grafana werde ich, wie es bei denen Services vorgesehen ist, eine time-series Datenbankart einsetzen, welcher von Prometheus selbst.

3.8.1 Redis

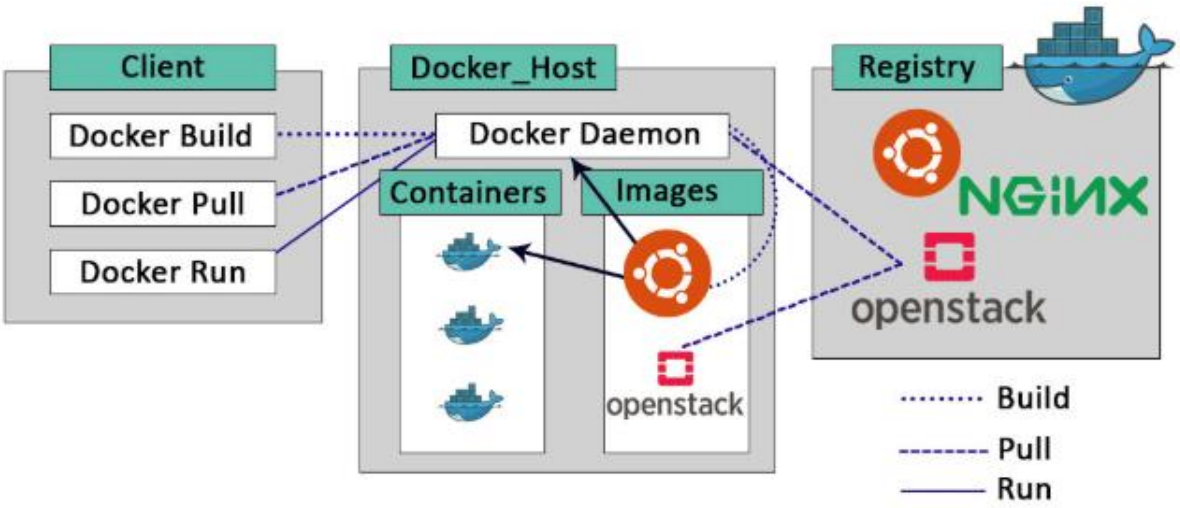
Der DB-Typ welcher für Time-Series Applikationen wie Prometheus geeignet ist.

3.9 Docker

Um die Skalierbarkeit und die Entwicklung allen angebotenen Services zu vereinfachen werde ich mit Docker Container arbeiten. Im Grunde genommen basiert Docker auf einer Client-Server Architektur, und zwar mit folgenden Komponenten:

1. Docker-Daemon: Der Daemon (dockerd) ist ein Prozess, der im Hintergrund weiterläuft und auf Befehle vom Client wartet. Der Daemon ist in der Lage, verschiedene Docker-Objekte zu verwalten.
2. Docker-Client: Der Client (docker) ist ein Programm mit einer Kommandozeilenschnittstelle, das hauptsächlich für den Transport der von den Benutzern erteilten Befehle zuständig ist.
3. REST-API: Die REST-API fungiert als Brücke zwischen dem Daemon und dem Client. Jeder Befehl, der über den Client ausgegeben wird, durchläuft die API, um schließlich den Daemon zu erreichen.

Sie als Benutzer werden in der Regel Befehle über die Client-Komponente ausführen. Der Client verwendet dann die REST-API, um den lange laufenden Daemon zu erreichen und Ihre Arbeit zu erledigen. Zusätzlich gibt es den Docker Hub, welche allen offiziellen Images in einer für allen zugänglich, Repository namens Registry, zur Verfügung stellt.



Container sind isolierte Prozesse, die auf einem gemeinsamen Betriebssystem laufen. Container sind Abstraktionen der Anwendungsschicht, die den Code und die Abhängigkeiten in einem Paket bündeln. Darüber hinaus ist mit Compose die Möglichkeit gewährleistet mehrere Container - z. B. eine Anwendung und eine Datenbank - koordinieren, um miteinander zu kommunizieren.

Dockerfile

Images werden via das sogennante Dockerfile erstellt

Docker-Compose

Docker-Compose wird für das Zusammenführen von mehrere Containers in einem File. In dem YAML-File werden die unterschiedliche Containers als Services eingerichtet mit all Ihren Abhängigkeiten (wie Ports, Laufwerke, Environment variables u.v.m)

3.9.1 Docker Hub

Das Docker Hub ist die Suchseite für alle Docker-Repositories, die jemals online publiziert worden sind. Um Docker Repos auf die lokale Maschine herunterzuladen brauchen wir keinen Account, jedoch für das «Pushen» logischerweise schon. Nachdem ich erfolgreich einen Account hinterlegt habe, kann ich z

3.9.2 Die Vorteile von Docker

Nachstehend sind einige Vorteile von Docker und Effizienzsteigerungen, welches die E

- ROI – Return on Investment
- Standardisierung & Produktivität
- CI-Effizienz
- Kompatibilität & Wartbarkeit
- Einfachheit & schnellere Konfigurationen
- Schnelles Deployment
- Kontinuierliches Deployment & Testen

3.9.3 Docker Orchestration

Mit Docker Swarm oder externe Orchestrator (Kubernetes, Portainer, Openshift, usw.)

3.10 API

Was ist ein API?

Ein API (Application Programming Interface) ist eine Schnittstelle, bei dem das Frontend mit dem Backend (Datenbanken) kommunizieren kann. Also im Client-Server Architektur ist .

Die zwei bekanntesten APIs sind **REST** (Representational State Transfer) und **SOAP** (Simple Object Access Protocol).

3.10.1 REST

Grundsätzlich kann man zwischen das Empfangen von json-Paketen und XML unterscheiden, die von der Datenbank und Webserver verschickt werden

Postman als Verwalter von APIs. Link

Quelle 10: Postman Webseite

3.10.2 GraphQL

Eine alternative Methode, um mit APIs zu kommunizieren und Anfragen zu gestalten, kann mit dem Service GraphQL gemacht werden. GraphQL bietet einige Vorteile im Gegensatz zur REST an. Einerseits können wir beim GraphQL nur die effektiv gewollten Daten angezeigt bekommen anstelle des ganzen Datensatzes. Dies wird durch die Vorbereitung von Schemen im Backend angestellt. Ein weiterer Vorteil ist die Kompatibilität, die GraphQL mit allen low-/high-level Programmiersprachen anbietet.

GraphQL ist eine Abfragesprache für Ihre API und eine serverseitige Laufzeitumgebung für die Ausführung von Abfragen unter Verwendung eines Typsystems, das Sie für Ihre Daten definieren. GraphQL ist nicht an eine bestimmte Datenbank oder Speicher-Engine gebunden und wird stattdessen von Ihrem bestehenden Code und Ihren Daten unterstützt.

Wie funktionieren GraphQL Abfragen und Schemas

Ein GraphQL-Dienst wird erstellt, indem man Typen und Felder auf diesen Typen definiert und dann Funktionen für jedes Feld auf jedem Typ bereitstellt. Ein GraphQL-Dienst, der uns mitteilt, wer der angemeldete Benutzer ist (ich) und wie der Name dieses Benutzers lautet, könnte zum Beispiel so aussehen:

3.11 Postman

Um APIs zu testen und die

Ein gutes Tutorial, um Postman näher kennenzulernen und eines, welches ich selbst angeschaut und durchgearbeitet habe ist folgendes (von FreeCodeCamp) Link

3.12 Prometheus

Um die Verfügbarkeit und andere Metriken der Docker Container anzuschauen und zu überwachen werde ich Prometheus einsetzen. *Prometheus ist ein Open-Source-Toolkit zur Systemüberwachung und Alarmierung.*

Die Komponenten welchen Prometheus verwendet sind folgende;

- der **Prometheus-Hauptserver**, der Zeitreihendaten sammelt und speichert
- **Client-Bibliotheken** zur Instrumentierung von Anwendungscode
- ein **Push-Gateway** zur Unterstützung **kurzlebiger Jobs**
- spezielle **Exporters für Dienste** wie HAProxy, StatsD, Graphite, etc.
- einen **Alert-Manager** zur Verwaltung von Alarmen
- verschiedene Support-Tools

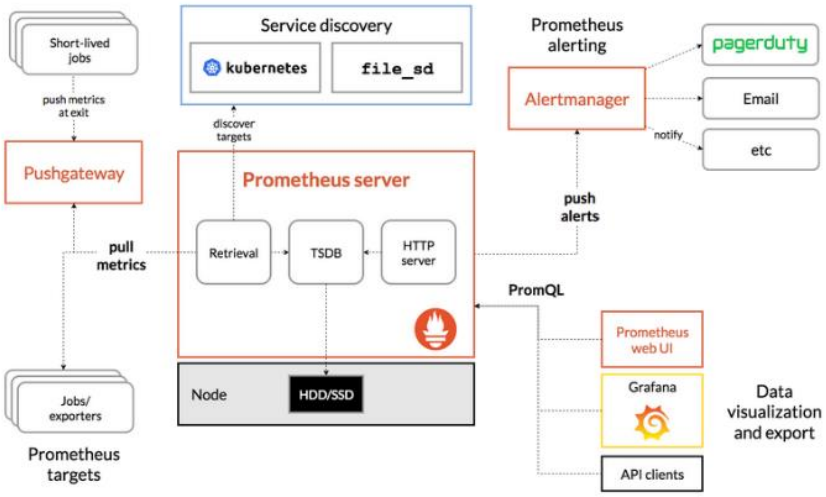


Abbildung 10: Darstellung der Prometheus Komponente

Kompatible Einsatzzwecke für Prometheus

Prometheus eignet sich gut für folgende Sachen;

- die Aufzeichnung beliebiger rein numerischer Zeitreihen.
- maschinenzentrierte Überwachung
- Überwachung von hochdynamischen serviceorientierten Architekturen.

Prometheus ist auf Zuverlässigkeit ausgelegt, um das System zu sein, zu dem Sie während eines Ausfalls gehen, damit Sie Probleme schnell diagnostizieren können.

Inkompatible Einsatzzwecke für Prometheus

Für folgende Aspekte eignet sich Prometheus eher weniger;

- *Abrechnung pro Anfrage*
- *detaillierter und vollständige Daten*
- *100%ige Genauigkeit*

In einem solchen Fall verwenden Sie am besten ein anderes System, um die Daten für die Abrechnung zu sammeln und zu analysieren, und Prometheus für den Rest der Überwachung.

Gem. Prometheus Overview

Quelle 11: Prometheus Überblick

3.12.1 Wie funktioniert Prometheus

Prometheus speichert alle Daten als **Zeitreihen**: Ströme von zeitgestempelten Werten, die zur gleichen Metrik und zum gleichen Satz von beschrifteten Dimensionen gehören. Was Zeitreihe (time series) ist findet Ihr im Glossar.

Metrik

Jede Zeitreihe wird durch ihren **Metriknamen** und optionale Schlüssel-Wert-Paare, sogenannte **Labels**, eindeutig identifiziert. Der Name dieser Metriken gibt das **allgemeine Merkmal eines Systems an**, das gemessen wird (z. B. http_requests_total - die Gesamtzahl der empfangenen HTTP-Anfragen). Er kann ASCII-Buchstaben und Ziffern sowie Unterstriche und Doppelpunkte enthalten.

Labels

Labels ermöglichen das **dimensionale Datenmodell** von Prometheus: Jede gegebene Kombination von Labels für denselben Metriknamen identifiziert eine bestimmte dimensionale Instanziierung dieser Metrik (zum Beispiel: alle HTTP-Anfragen, die die Methode POST an den /api/tracks-Handler verwendet haben). Die Abfragesprache ermöglicht das **Filtern** und **Aggregieren** basierend auf diesen Dimensionen. Das **Ändern eines beliebigen Beschriftungswertes**, einschließlich des Hinzufügens oder Entfernens einer Beschriftung, **erzeugt eine neue Zeitreihe**.

Anderes

Sammelt Daten via unterschiedliche HTTP Endpoints (Glossar). Ein Exporter wird jegliche Zeitreihendaten von Hosts (Computer) und Services (Dienste) abholen

Prometheus wird in YAML-Sprache geschrieben, also ist die Konfiguration sehr simplifiziert.

3.12.2 Prometheus im Vergleich

Tabellenvergleich mit den wichtigsten Punkten gem. Comparison

	Graphite	InfluxDB	OpenTSDB	Nagios	Sensu
Bereich					
Data models / Storage					
Architektur					
Summary					

Tabelle 14: Prometheus im Vergleich zu anderen time-series DB Tools

Die unterstützten DB-Aktionen sind folgende GEBRAUCHT??

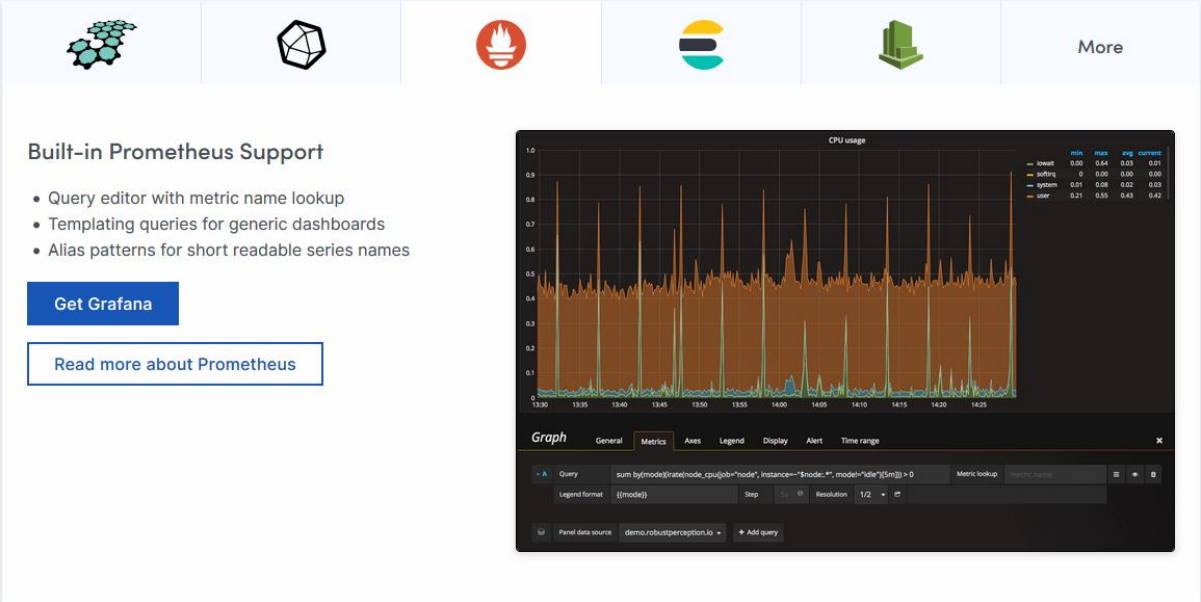
- Checkpoints
- Compression
- Key-Value Data Model

- Indexierung
- Logging
- Query Execution
- REST und GraphQL als Query Interface

3.13 Grafana

Grafana ermöglicht es Einem, Ihre Metriken abzufragen, zu visualisieren, zu alarmieren und zu verstehen, unabhängig davon, wo sie gespeichert sind. Grafana greift auf sogenannten Data-Sources

Grafana ist mit einer Prometheus-Instanz sehr gut kombinierbar (siehe Abbildung:)



3.14 Splunk

Via Luis und den breiten Internet u.a LinkedIn habe ich Splunk kennengelernt. Splunk dient zur grafischen Analyse und Monitoring von jeglichen Log-Files.

3.15 Für Inhouse-Entwickler

Wie im Kap. 3.11 Container beschrieben, habe ich für die Entwickler und Bereitsteller (am Anfang nur ich) herausgedacht die Aufsetzung der Applikation so unkompliziert wie möglich zu halten, damit die Weiterentwicklung der Dienste sehr einfach gehandelt werden kann. Das Arbeiten mit Docker-Container enthaltende mehrere Services hat im Gegensatz zu einer normalen, monolithischen Applikation.

4 Planen

4.1 GANTT/Zeitplan

Zeitplan erstellen

Gemäss Änderungsverzeichnis

4.2 Lernjournal (L)

Nachfolgend befinden sich meine getätigten Arbeiten an jenen Tagen

Tag 1 (01.02.2021)

Heute haben wir den Moduleinstige gehabt und bekamen unseren Auftrag, ein fiktives Unternehmen herauszudenken, mit einer angebotenen, externen Dienstleistung. Ein Webauftritt ist dafür notwendig.

Ich habe für die Dokumentation einen «Entwurfsgrundgerüst» gem. IPERKA erstellt mit all den abgefragten Kompetenzen, welche ich mit Klammern, neben den (Unter-) Kapiteln gekennzeichnet habe.

Tag 1.2/1.3 (18/20.02.2021)

Folgendes habe ich bearbeitet;

- Kapitel 3.2.1, 3.9-3.11.2
- Kapitel 6.1 und 6.2-6.2.6
- Kapitel 8.1 und 8.3
- Domäne gekauft (vinylo.shop)
- Kapitel

Tag 2 (22.02.2021)

Am zweiten Tag hatten wir eine Info über das HTTP-Protokoll

- Kapitel 2, 3.2 erledigt
- Kapitel 5.6 überarbeitet
- Kompetenzen I1, I2 und P abgegeben

Tag 2.1 (28.02.2021)

Folgendes habe ich erledigt;

- Kompetenz P erweitert
- IST-Zustand, Dimensionierung, Art des Hostings
- Kompetenz V angefangen

Tag 3 (01.03.2021)

Am dritten

- Namens-, Sicherheits- und Betriebskonzept erledigt und abgegeben
- Risikoanalyse aufgestellt

Tag 4 (08.03.2021)

Folgendes habe ich bearbeitet;

- Foxtrail-Aufgabe

Tag 4.1 (10.03.2021)

Folgendes habe ich erledigt;

- Docker Container Webserver, Datenbank, Wordpress und Certbot initialisiert und grob dokumentiert

Tag 5 (22.03.2021)

- WP und DB auf Pi eingerichtet
- über vinylo.shop (HTTP) erreichbar

Tag 6 (29.03.2021)

Heute habe ich einiges abgegeben

- Logfiles + (L1/L2)
-

Tag 6.1 (01.04.2021)

- Über Hosting-Provider WordPress eingerichtet
- Mailcow eingerichtet
- Casts für Sicherheitskonzept (V), Mailcow (SW 1), HTTPS (B2), Testing (T) und SSH (P) + Lernprozess

5 Entscheiden

5.1 Namenskonzept (V)

5.1.1 Gerätetypen

Die Geräte sind wie folgt benamset

Abkürzung	Bezeichnung
rt	Router
fw	Firewall
rpi	Raspberry Pi
dkr	Docker Container
wbs	Web Server
msx	Mail Server
db	Datenbank
hsg	Hosting-Provider

Tabelle 15: Namenskonzept

5.1.2 Interne Mitarbeiter

Grundsätzlich werden die Benutzernamen der Mitarbeiter wie folgt zusammengestellt; Der Mitarbeiter Max Mann oder Karl Ice werden die folgenden Benutzernamen erhalten, **mmann** und **kice**. Wenn ein neuer Mitarbeiter namens Martin Mann angestellt wird, werden die nächstweiteren Buchstaben im Vornamen zum Benutzernamen ergänzt, ergo mamann.

5.1.3 Externe Kunden

Unsere Kundschaft sollen über ein nahtloses, hochgesichertes Anmeldungsverfahren verfügen. Die Kunden haben einerseits die Option als Gast zu bestellen, wo sie die Festlegung der Lieferadresse und Zahlungsmethode als Auswahlmöglichkeiten hatten oder anderseits ein Konto anzulegen, womit die getätigten Bestellungen das Kauf- und UX-Experience verbessern würden. Die Kunden sind in der Auswahl eines Benutzernamens frei (Gross-, Kleinzeichen und Ziffern). Symbolen sind nicht erlaubt, da dies nicht der Normen entspricht. Die Kontodaten werden zur gleichen Zeit in einer internen Datenbank abgespeichert.

5.1.4 Externe Lieferanten

Unsere externen Lieferanten, welche für die Herausstellung unserer Schallplatten verantwortlich sind, erhalten auch einen Kundenlogin, jedoch hätten im Gegensatz zur Kundschaft, Zugriff auf die internen Dienste, welche für Ihnen relevant seien (Bestellungsübersicht, Kontaktdaten des zuzustellenden Pakets).

Bsp. Coop Genossenschaft: coopgmbh als Benutzernamen und E-Mail coop.gmbh@coop.ch

5.2 Sicherheitskonzept (V)

5.2.1 Benutzerrechte/Kundenrechte (B1)

Der Berechtigungsmatrix für die interne Mitarbeiter sowie der den Kunden finden wir nachstehend;

Benutzer	Benutzernamen	Gruppe
Michalis Chatzimichalis	mchatzimichalis	GL
Max Mann	mmann	IT
Till Vosul	tvosul	IT
Bob Baby	bbaby	BRT
Karl Ice	kice	BRT
Externe Lieferanten	-	-
Externe Kundschaft	-	-

Tabelle 16: Benutzerrechte

Gruppennennung

Folgende Gruppen werden intern angewendet

Gruppe	Gruppekürzel
Geschäftsleitung	GL
IT	IT
Berater	BRT

Tabelle 17: Gruppen

Berechtigungsmatrix

Da noch die verschiedenen Berechtigungsstufen für die unterschiedlichen Dienste

	GL	IT	Berater
Mailcow	L	L,S	L
WordPress	-	L	L,S
FTP	L,S	L,S	L,S

Tabelle 18: Berechtigungsmatrix

5.2.2 Massnahmen

Folgende technische Massnahmen habe ich mir für die Firma erachtet:

- Die Server müssen an einer USV verbunden werden.
- Saubere und nachvollziehbare Berechtigungsvergabe
- Zugriff mit Monitoring und Alerting-Tools überwachen. Die Logs natürlich nachführen
- Alle Infrastruktur relevanten Geräte (Servern) verfügen über ein sicheres Admin Passwort.
- Tägliche (inkrementelle) und wöchentliche (Full-) Backups müssen vom Backuptool gemacht werden.

Folgende nicht-technische Massnahmen habe ich ebenfalls erachtet

- Interne Berater und Engineer auf die Infrastruktur schulen und Auswirkungen/Folgen erklären
- Keine fremden Geräte (Switches, USBs von Kunden usw.) in der Firma erlauben
- 3rd Level Support für wichtige Dienste sowie Systeme müssen genau dokumentiert werden.

5.3 Betriebskonzept (V)

Gem. dem Bundesrat und seinen Tool HERMES ist ein Betriebskonzept so zu verstehen. Link

Quelle 12: HERMES Bundesrat

5.3.1 Anforderungen an den Betrieb

Die **Kundschaft** haben die Anforderung einen kompetenten und vertrauenswürdiger Schallplattenverkäufer zu kennen.

Die **Investoren** haben die Anforderung, dass vinylo sich auf dem Markt als etablierter Verkäufer antritt und fortlaufend wächst.

5.3.2 Organisation

Aufbauprozess

Für den Betrieb des IT-Systems relevante Organisation (Organigramm, Stellen, Funktionen)

Organigramm

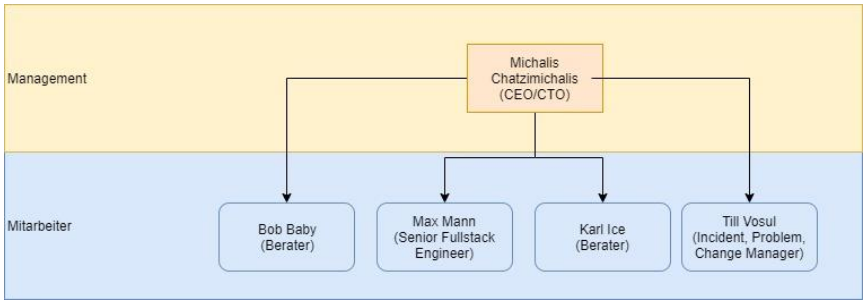


Abbildung 11: Organigramm

Stellen und Funktion

Namen	Stelle	Funktion
Michalis Chatzimichalis	CEO/CTO	Day-to-Day Operations, Letzte Ansprechstelle bzgl. Finanzen und Technologische Fortschritte.
Till Vosul	ITIL Spezialist (Incident, Problem und Change Manager)	Handelt alle Probleme und Fälle in den Incident, Problem und Change-Management Bereichen
Max Mann	Senior Fullstack Engineer	Bereitstellung und Aufrechterhaltung des Webshops, Entwicklung neuer Funktionen
Bob Baby	Berater	Beratungshilfe an neugierige Kunden
Karl Ice	Berater	Beratungshilfe an neugierige Kunden

Tabelle 19: Stellen und Funktionen

5.3.3 Systemtechnik
IT-Infrastrukturkonzept

Die Infrastruktur des jetzigen Webshops und der Firma besteht aus einen einzigen Pi, einen Heimrouter und einen USB-Stick für die Datenabspeicherung (Datenbanken, Logs).

Systeme, eingesetzte Komponenten, Versionen

Den verwendeten Systemen wären zurzeit Ubuntu auf das Pi und dazu für all den angebotenen Diensten Docker. Die Versionen von deren Diensten (Web-, Mailserver, Docker-Compose, Docker Engine usw.) sind allen auf den bisherigen, aktuellen Stand.

Netze

Da Docker-Compose ein eigenes Netzwerk bei der Bereitstellung und Ausführung der Container erstellt, wird dieses verwendet. Die einzelnen Container verfügen über eine ansprechbare IP im internen Netz und können miteinander kommunizieren (Abfragen). Da der Webserver über der Pi, welches sich im internen Heimnetzwerk lauft, wird, wie im Kap. **Error! Reference source not found.** beschrieben, eine Weiterleitungsregel am R outer erstellt, sodass all die Abfragen an dem Webserver zum richtigen Ziel führen.

Datensicherung

Die Backupregel beruht beim Unternehmen, trotz der kleine Grösse mit dem 3-2-1 Prinzip. Das 3-2-2 Prinzip bezeichnet 3 alternierende Disks zu einem bestimmten Rhythmus, mit 2 verschiedenen Medien und an 2 Orten (Physisches Ort und Cloud).

5.3.4 Systemüberwachung

Prometheus mit Grafana für das Überwachen der Ressourcen und Traffic, welches vom Webshop verwendet wird. Für die Qualitätssicherung und Totalität der Logdateien wird Splunk eingesetzt.

Um den Datenschutz der Kunden zu gewährleisten und die Integrität zu kontrollieren wird eine externe ISO-Zertifizierungsfirma für einen Audit eingestellt.

5.3.5 Incident Management

Das Incident Management kümmert sich Dienste so schnell wie möglich wiederherzustellen, oft durch die Anwendung von Übergangslösungen. Folgende Schritte sind zu tun:

1. Incident (Problem) wird gemeldet
2. Incident wird angeschaut und vom 1st Level Support akzeptiert
3. Wenn es am 1st Level Support nicht gelingt, den Incident zu lösen, wird es zum 2nd Level eskaliert
4. Incident wird erfolgreich gelöst, wenn nicht gibt es den 3rd Level Support (an Vendors)
5. Ticket geht zurück ins 1st Level Support und wird abgeschlossen

5.3.6 Problem Management

Das Problem Management befasst sich mit dem Analysieren eins eingetroffenen Problems und die Erkennung der Ursache. Folgende Schritte sind zu tun:

1. Proaktive Erkennung des Problems gem. Kundenrückmeldung
2. Kategorisierung und Einstufung des Problems
3. Diagnostik vom Problem – evtl. temporäre Lösungsansatz
4. Bei permanenter Lösungsansatz kann das Problem abgeschlossen und eine endgültige Evaluation gemacht werden
5. Bericht des Problems mit den notwendigen Details

Die laufende Überwachung aller gemeldeten Problemen ist stets im Ticket-Tool vorhanden.

5.3.7 Change-Management

Bei einer angekündigten Änderung an der Webseite (neue Features, aktualisierter Code) werden Changes im Ticketsystem angemeldet und dies an dem CEO/CTO (Michalis Chatzimichalis

Es ist unter diesen 3 Changes zu unterscheiden;

- **Standardänderungen**: Vorautorisierte, risikoarme Changes, die einem bekannten Verfahren folgen.
- **Emergency-Changes**: Changes, die sofort implementiert werden müssen, z. B., um einen Major Incident zu beheben.

- **Normale Changes**: Alle anderen Changes, die keine Standard Changes oder Emergency Changes sind.

Die Normale Changes werden zusätzlich in Major/Signifikant oder Minor eingeschätzt. **Major Changes** und **Signifikante Changes** müssen vom Change Manager genehmigt werden.

Wenn ein nicht standardmässiger Change erforderlich ist, wird einen Request-for-Change (RFC) beim Change-Management eingereicht. Das Change-Management wird dann den Change erfassen, analysieren und genehmigen (oder ablehnen).

Emergency Changes werden vom CEO (Michalis Chatzimichalis) bewertet und genehmigt, die kurzfristig für Notfälle zur Verfügung steht.

5.3.8 Beschreibung der Aspekte der Sicherheit

siehe Kap. Sicherheitskonzept (V)

5.3.9 Anforderungsabdeckung

ID	Anforderung	Zuordnung zu Subsystem	Beurteilung der Anforderungsabdeckung
A1	Support	Supportzeiten von 08:00 – 12:00 / 13:00 – 18:00	100%
A2	Sicherheit	Verschlüsselte Kommunikation und Ablage in der Datenbank. Sicheren Anmeldungsverfahren	100%

Tabelle 20: Auflistung der Anforderungen mit Zuordnungen und Abdeckung

5.4 Risikoanalyse (V)

5.4.1 Erklärung

Bei der Risikoanalyse handelt es sich um eine vorausschauende Diagnose, um mögliche Probleme zu erkennen, einzudämmen und zu minimieren.

Gründe für eine Risikoanalyse sind die Prävention für eventuell auftauchende Probleme, die vorausschauende Planung des Projektes und die Garantie eines reibungslosen Ablaufs.

5.4.2 Vorgehensweise

1. **Ziele SMART beschreiben**
    a. M, R und T sind Vorgaben der Risikoanalyse
2. **Risikobereich identifizieren**
    a. Suchen von möglichen Risiken, dabei alle Projektdimensionen beachten (Qualität, Ressourcen, Zeit). Dabei ist es wichtig, die Ursachen der Risiken zu benennen – nicht die Symptome (progressiv abstrahieren).
3. **Symptome benennen**

   a. Symptome sind Erkennungsmerkmale für Risiken, die anzeigen, ob ein Problem bereits eingetreten ist oder einzutreten droht.

4. **Risiken bewerten und gewichten mittels Risikomatrix**
   a. Jedem Risiko die Kriterien «Wahrscheinlichkeit des Eintreffens» und Tragweite zuordnen.
5. **Vorbeugende Massnahmen umsetzen mittels Risikoanalysetabelle**
   a. Verbindliche Umsetzung von Gegenmassnahmen, die entweder das Problem verhindern oder seine Auswirkung begrenzen.
6. **Evtl. Massnahmen planen (Alternativplan, Katastrophenplan) mittels Risikoanalysetabelle**
   a. Bei besonderen kritischen Problembereichen sollen bereits in der Planungsphase alternativen Vorgehensweisen vorgesehen werden.

5.4.3 Risikoanalysetabelle

Nr.	Risiko	Symptome	W-keit	Tragweite	Gegenmassnahmen
1	Lieferungsverzug	Anrufe, E-Mails werden wegen keiner Lieferung einkommen	niedrig	mittel	Eines zweiten Lieferanten als Backup haben
2	Ständige Überlastung des Webshops	Absteigende Performanz Langsame Bestellabwicklung	mittel	hoch	Performantere Infrastruktur aufbauen
3	Berater werden überlastet		hoch	mittel	Beratung bei einer kompetenten und stabilen Firma outsourcen

Tabelle 21: Risikoanalysetabelle

5.4.4 Risikomatrix

Tragweite Wahrscheinlichkeit	Niedrig	Mittel	Hoch
Niedrig	1		
Mittel			2
Hoch		3	

Tabelle 22: Risikomatrix

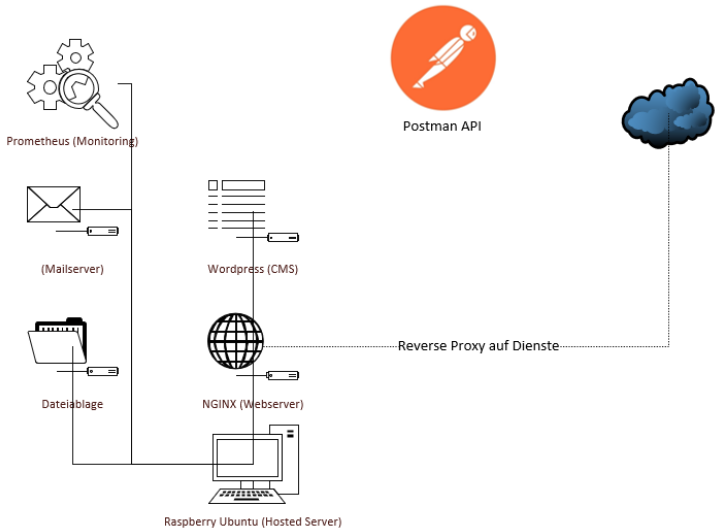
6 Realisieren

6.1 Umgebung

6.1.1 Physische mit Diensten

Firewall wird benötigt, verwende deshalb OPNsense. Mit Wordpress verbinden

Folgendes Visio habe ich für die Visualisierung erstellt (TESTBILD) Wird mit draw.io überarbeitet !



6.1.2 Virtuelle Arbeitsfläche

Für das Zugreifen und Bearbeiten aller Konfigurationsfiles auf Ubuntu habe ich im Visual Studio Code ein Add-On namens Remote Development, welcher von Microsoft herausgestellt wird, heruntergeladen. Dieses Add-On ermöglicht mir das professionelle Bearbeiten der Konfigurationsfiles mit allen nötigen und verfügbaren Plugins seitens VS Code sowie das Ausführen und Bewegung in der Linux CLI via das Terminal

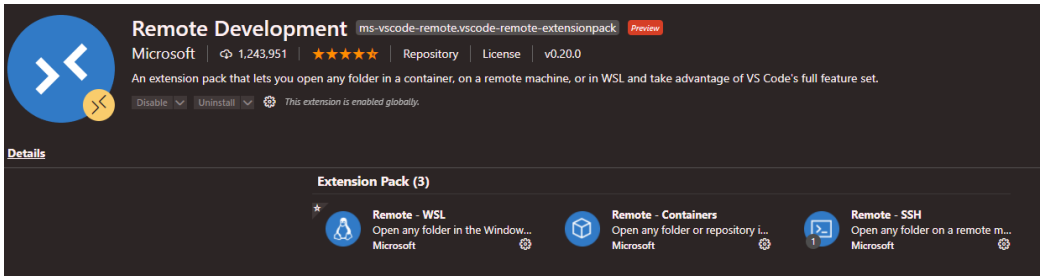


Abbildung 12: VS Code Remote Dev Add-On

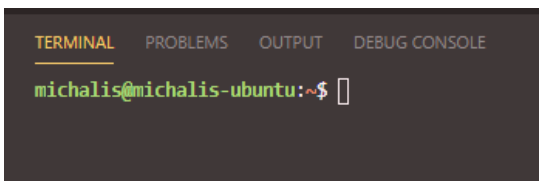


Abbildung 13: VS Code eingebautes Terminal

Der Stammverzeichnis habe ich auf meinem Pi wie folgt benannt Docker_Projects\modul_239

Die Extensions für die Kompatibilität von Docker (und Dockerfiles) lade ich auch herunter, resp. wird beim Erstöffnen eines Dockerfile zum Herunterladen vorgeschlagen.

6.1.3 SSH Alias erstellen

Ein SSH Alias dient für die schnellere Verbindung zu einem SSH-Host und ist lediglich eine Abkürzung. Um ein Alias für meinen Pi zu erstellen, öffne ich mein lokales CMD und wechsle zum Ordner .ssh. Dort öffne und bearbeite ich die config Datei

cd .ssh und **notepad ./config**

```
Host pi
  HostName 192.168.1.117
  User michalis
  IdentityFile ~/.ssh/id_rsa
```

Nachdem gebe ich nun ssh pi ein und schon habe ich eine erfolgreiche Verbindung zu meinem Pi via dieser Alias aufgebaut.

6.2 Installation (SW 1)

Die Installation der Services und alles rundum der Einrichtung der Domäne wird im folgenden Kapitel beschrieben.

Hinweis: Ab diesem Kapitel wird der beschriebene Vorgang in der Ich-Person erläutert, da es wesentlich einfach für den Leser ist zu sehen, was erledigt wurde.

6.2.1 Ubuntu auf einem Raspberry Pi einrichten

Das allererste Schritt, welches ich für dieses Projekt gemacht habe, ist Ubuntu 20.10 auf einer SD-Karte mit dem Raspberry Pi Imager Tool (Download-Link) eingerichtet. Nachdem dieser Schritt fertig war, habe ich die SD-Karte in dem laufenden Pi gesteckt und den Kleincomputer neugestartet.

Quelle 13: Raspberry Pi Imager Tool

Nachdem habe ich das System erstmals aufgesetzt und für den SSH-Zugriff vorbereitet.

6.2.2 Docker und Docker-Compose einrichten

Um Docker auf Ubuntu ein Als Erstes müssen wir die docker Pakete installieren. Docker gibt's als Desktop Programm auf Windows/Mac, jedoch nur als CLI auf Linux, und zwar namens Docker Engine. Um Docker Engine zu installieren, öffnen wir die Konsole und geben den folgenden Befehl ein.

Link https://docs.docker.com/engine/install/ubuntu/

6.2.3 Domäne kaufen

Die gesuchte Domäne war leider bei meinem privaten Hosting-Provider nicht. Jedoch bietet DreamHost eine nahtlose, unkomplizierte Migration der Domäne von einem anderen Hosting-Provider an. Die Domäne kostet mir 2.-/Jahr und werde sie, bis dato, weiterführen und danach sie annullieren.

Der Kauf der Domäne hat mein Vater getätigt, da ich über die benötigten Rechte in meinem DreamHost-Account nicht verfüge.

Da klicke ich auf den ersten Eintrag und gebe meine Domäne ein. Mit Next: Hosting gehe ich zu der Auswahl des Hosting-Artes. Die Domäne, die ich im vorherigen Schritt eingekauft habe, hat der Status Parked. Mehr zu den verschiedenen Status, welcher DreamHost an eingekauften Domänen gibt, findet Ihr hier.

Quelle 14: DreamHost Domain-Status

Diese wähle ich an und da ich mit WordPress einige Erfahrung wegen meiner privaten Seite gesammelt habe, enthake ich die zwei fakultativen Optionen. Der Domänennamen ist jetzt mit dem DNS-Eintrag des Routers verknüpft.

Maildomäne

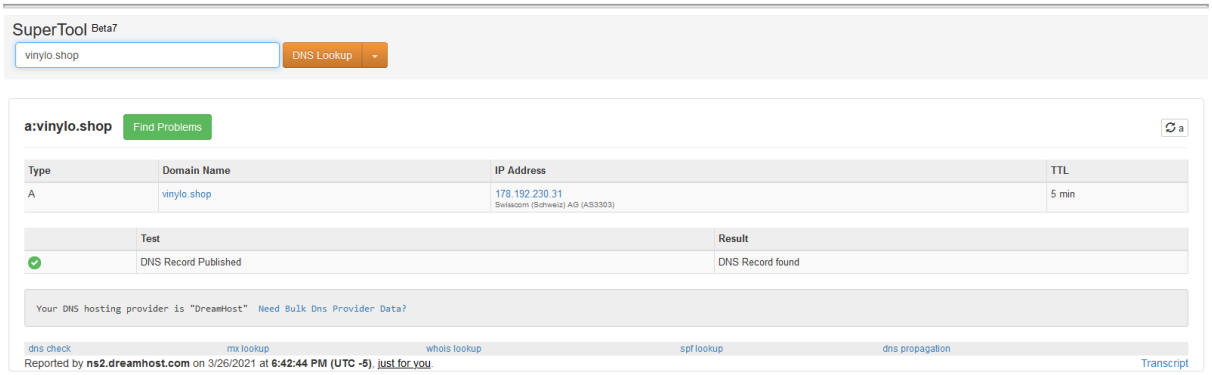
Um den Mailserver von aussen über einen Domänennamen verfügbar zu machen, erstelle ich zusätzlich eine Subdomäne für dieser. Im DreamHost Adminportal gehe ich auf Domains > auf vinylo.shop «DNS» > Add Record

Hinweis: @ heisst auf alle/root?

DNS-Eintrag	Name	Wert
A	mail	178.192.230.21 (öffentliche IP des Routers)
CNAME	autoconfig	mail.vinylo.shop
CNAME	autodiscover	mail.vinylo.shop
MX	@	mail.vinylo.shop (Prio: 10)
TXT (SPF)	@	
TXT (DMARC)	_dmarc	"v=DMARC1; p=reject; rua=mailto:mailauth-reports@example.org"

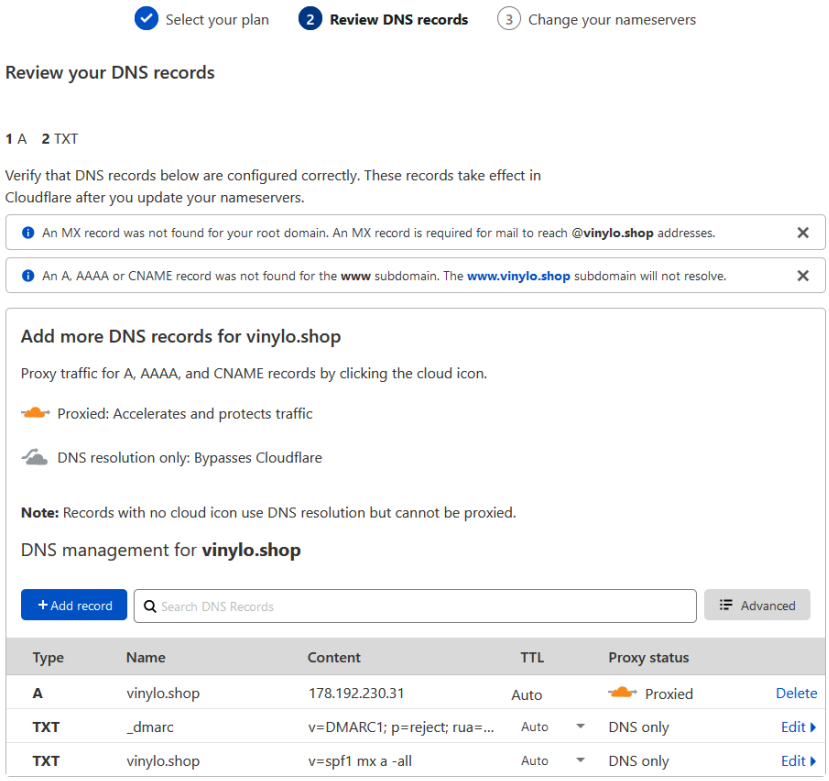
6.2.4 Domain-Name Konfiguration

Um die DNS-Records anzuschauen, gehe ich auf folgende Seite, um die zu überprüfen

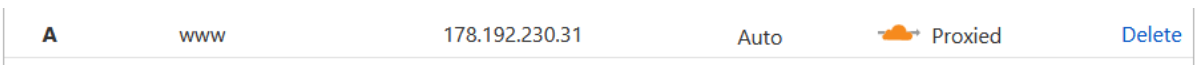


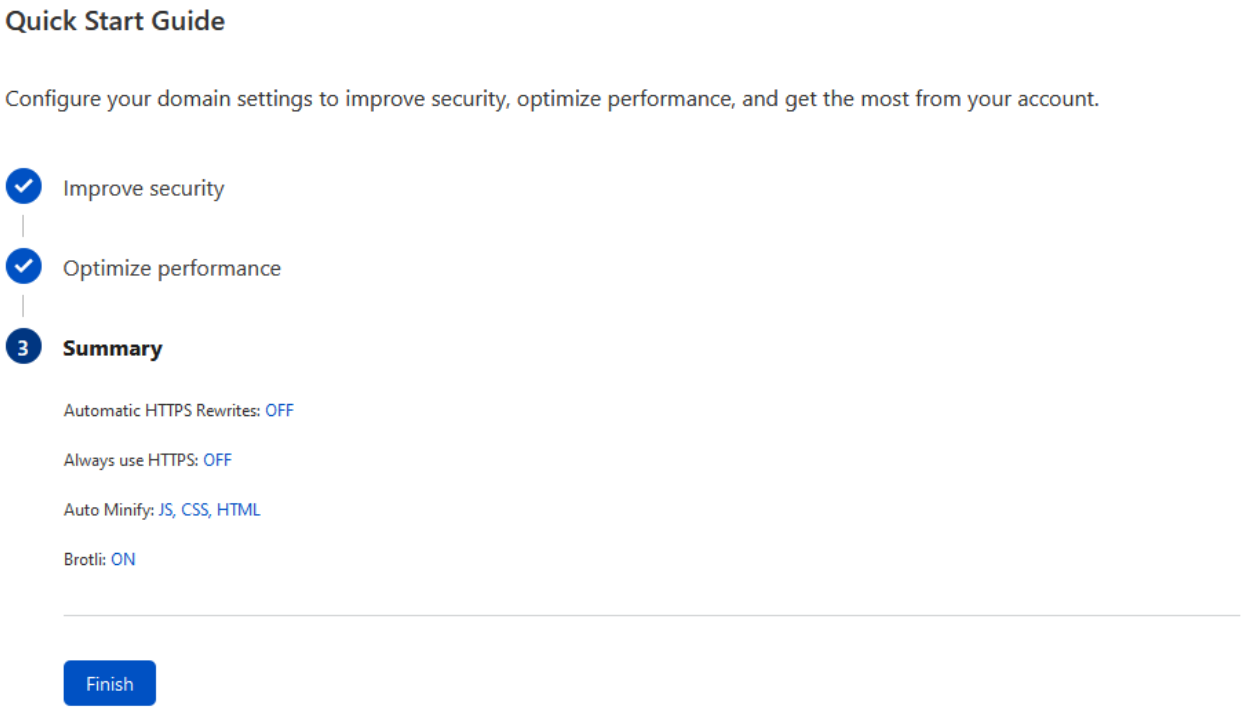
Cloudfare wird jetzt eingesetzt > cloudfare.com > Sign Up > Konto erstellen und Domäne bei Aufruf eingeben > Free plan entscheiden.

Jetzt scannt Cloudfare für bestehende DNS-Einträge für meine Domäne. Da ich schon einige eingerichtet habe



Die zwei Meldungen werde ich beachten und einen A-Record für das die Subdomain www einrichten



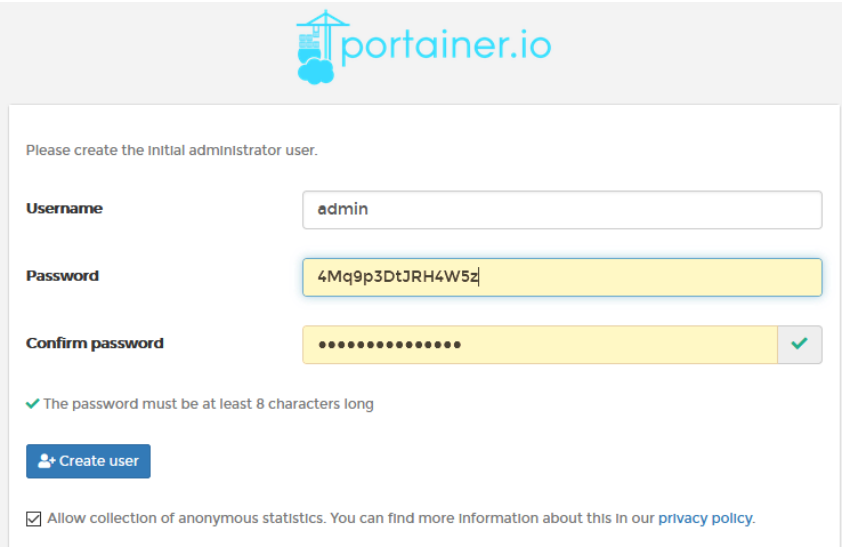


6.2.5 Portainer (Container Manager)

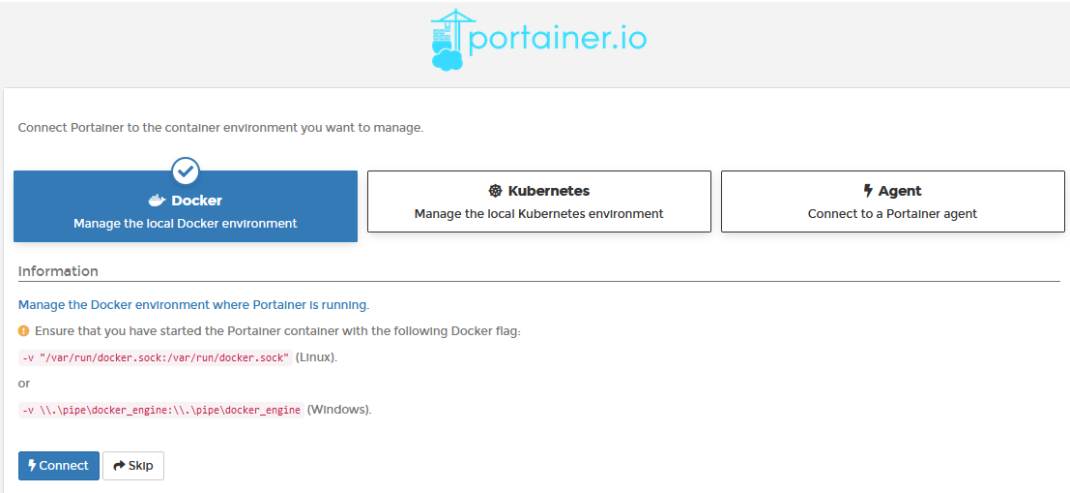
Nachdem ich schon nginx und Wordpress manuell mit Docker-Compose aufgesetzt und eingerichtet habe, stosse ich auf das Tool Portainer, welcher für das Verwalten von mehreren Docker Container, Kubernetes usw. anhand eines übersichtliches und intuitives GUI, zuständig ist. Für die Installation gebe ich folgendes ein

sudo docker run -d \ --name="portainer" \ --restart on-failure \ -p 9000:9000 \ -v /var/run/docker.sock:/var/run/docker.sock \ -v portainer_data:/data \ portainer/portainer-ce

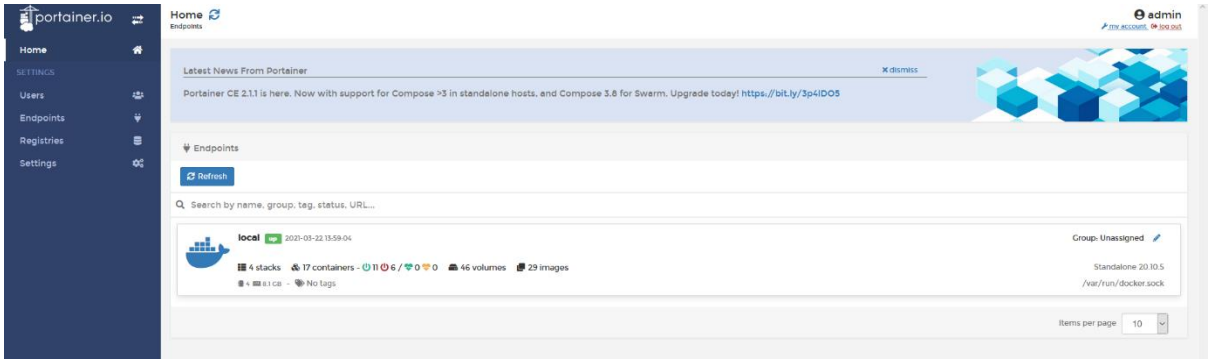
Das Docker Container für Portainer wird jetzt eingerichtet und kann nach erfolgreicher Inbetriebnahme via **[IP-Adresse]:9000** erreichbar.



Da wird Docker ausgewählt Docker.

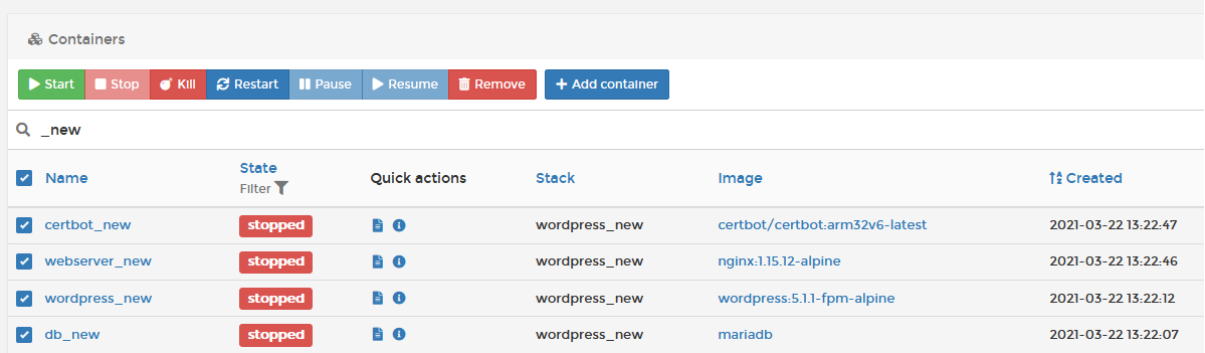


Der Portainer Home GUI sieht wie folgt aus. In der Mitte sind alle Nodes (Knoten), welche

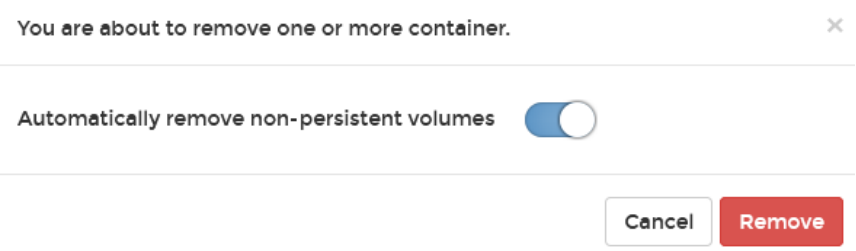


Container löschen

Um neu erstellte Fehlversuche zu löschen, gehe ich auf Container und gebe der gewünschte Namen des Containers ein. Nachdem ich alle ausgewählt und heruntergefahren habe, lösche ich sie mit dem Button Remove.



Bei der Frage, ob ich die gegebenen Volumes löschen will, wähle ich ja



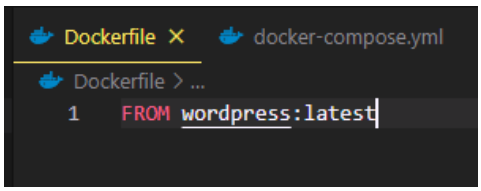
6.2.6 WordPress und DB einrichten (Fehlerhaft)

HINWEIS: Ab da + bis und mit Kapitel 6.2.9 habe ich versucht die ganzen Dienste über Docker und self-Hosted Domäne hochzuziehen, jedoch nicht ganz ans Ende gekommen. Darum habe ich mich ab Kap. 6.2.10 entschieden direkt auf dem Hosting-Provider mit einem Hosting-Plan weiterzufahren und die Aufgaben erfolgreich abzuschliessen.

Als Erstes erstelle ich über das CLI einen neuen Ordner.

- cd Docker_Projects/modul_239/
- sudo mkdir website && sudo chown michalis website
- cd website
- touch Dockerfile && touch docker-compose.yml

Im VS Code wird das Dockerfile mit der Anweisung gefüllt, er solle das aktuelle WordPress-Image verwenden

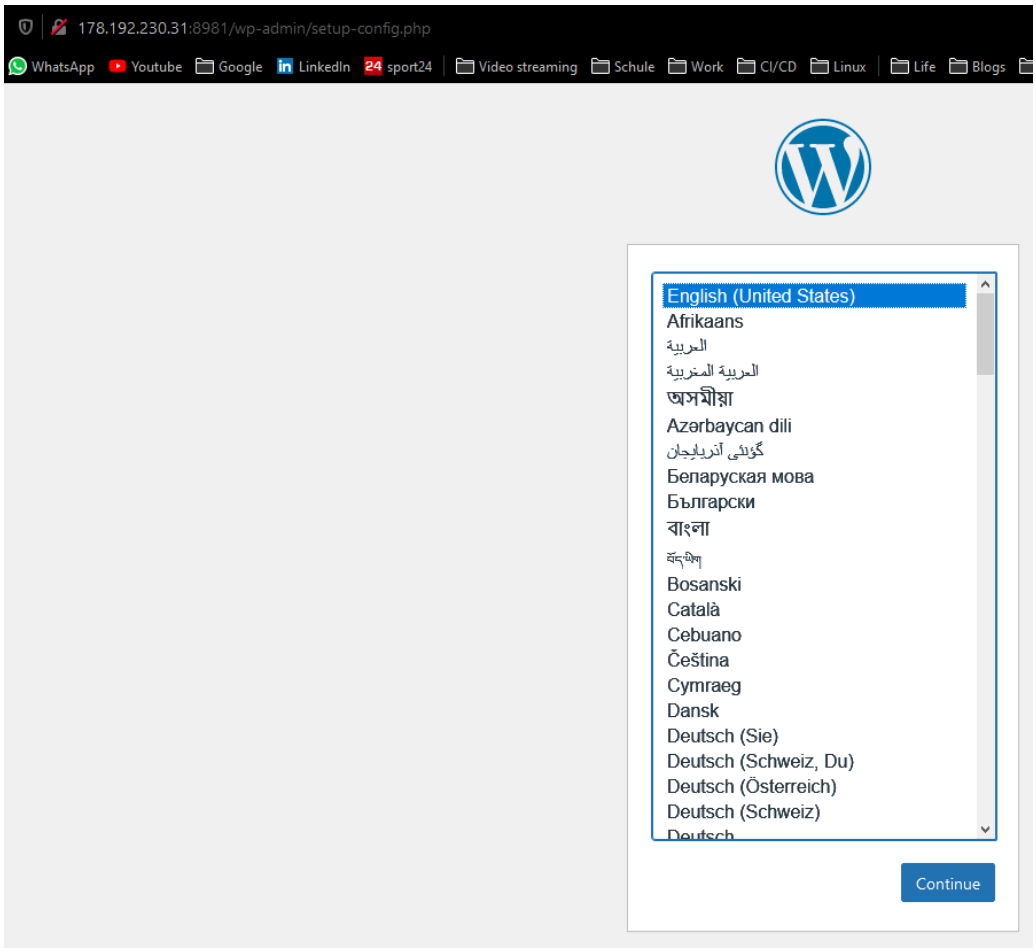


Im Docker-Compose-File wird nun der WordPress-Service eingerichtet, mit der Anweisung er solle vom jetzigen Ordner das aktuelle Image aufgrund des konfigurierten Dockerfiles holen

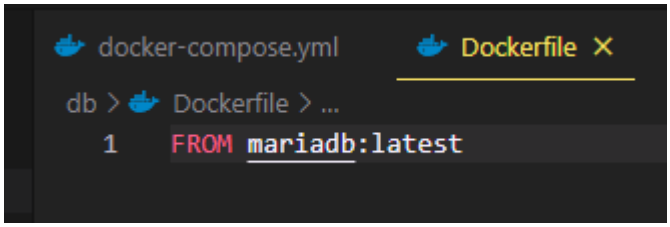
```yaml
# docker-compose.yml
version: '3.7'

services:
    wordpress:
      build: .
      container_name: wordpress_final
      ports:
        - "8982:80"

```

Die Port-Forwarding-Regel auf dem Router einrichten

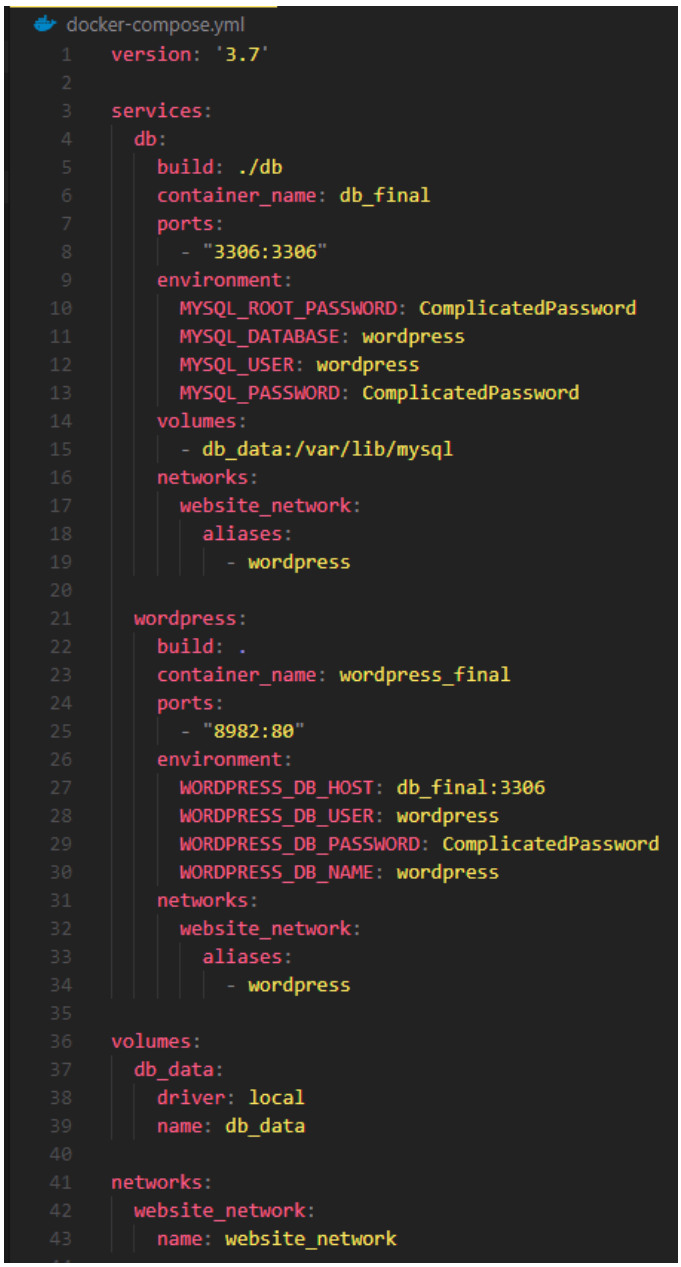
Ich verbinde mein PC nun mit meinem Handy-Hotspot. Die erfolgreiche Verbindung von aussen;



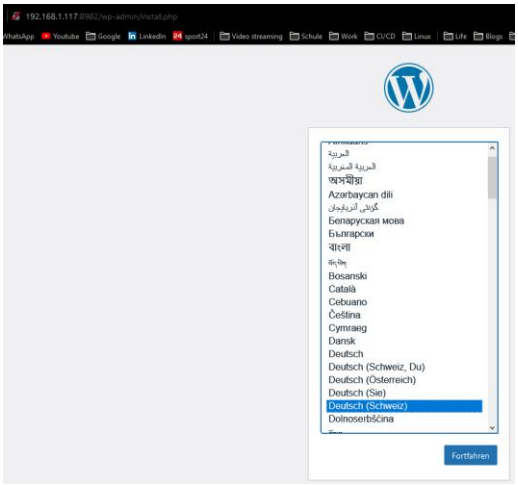
Für die Datenbank erstelle ich im website Ordner einen Unterordner, um ein weiteres Dockerfile anzugeben. Dies sieht dann wie folgt aus. Das Image ist ein für das Raspberry Pi speziell konfiguriertes SQL-Fork.



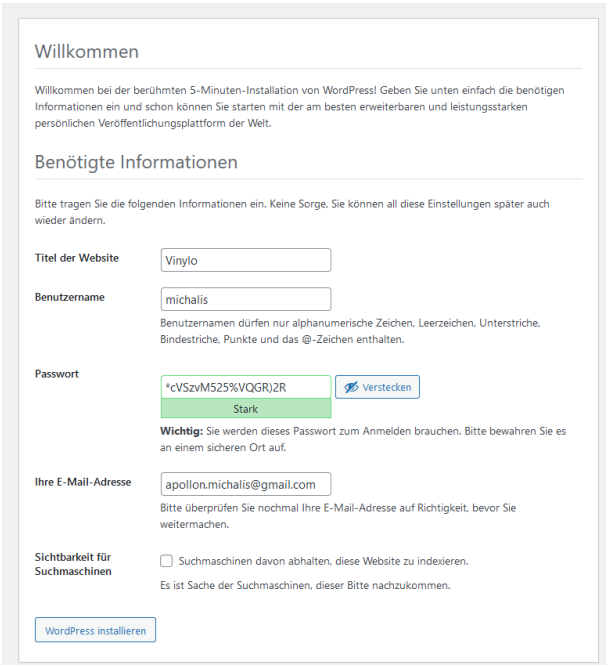
Das Docker-Compose wird wie folgt mit dem DB-Service (Konfig. + Volumes + Networks) sieht wie folgt aus



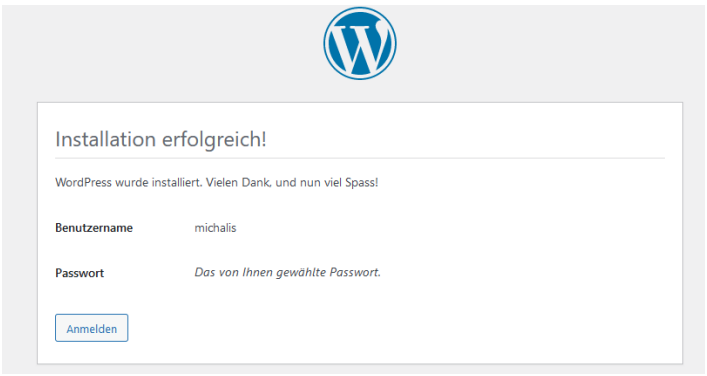
Die WordPress Loginseite kommt erneut und da wähle ich Deutsch (Schweiz) > Fortfahren



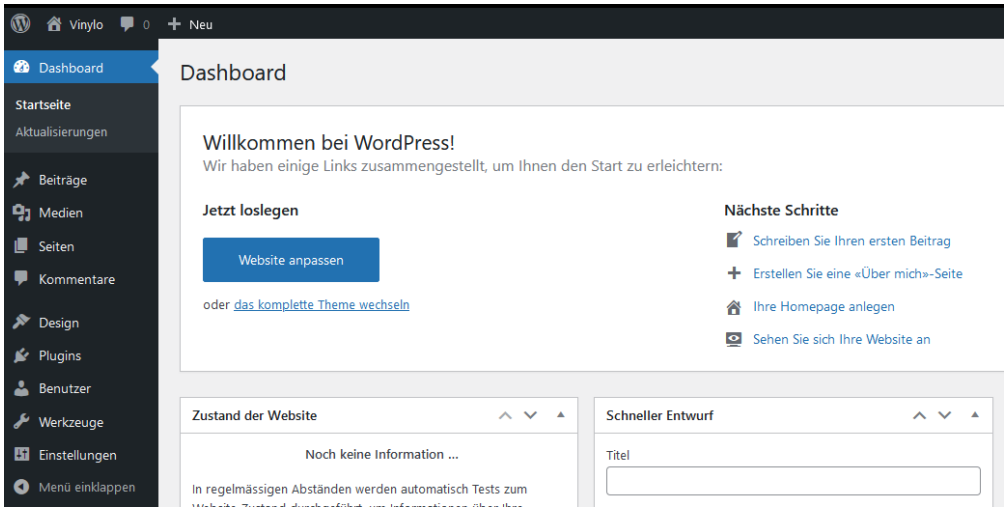
Danach fülle ich alle Felder aus > WordPress installieren



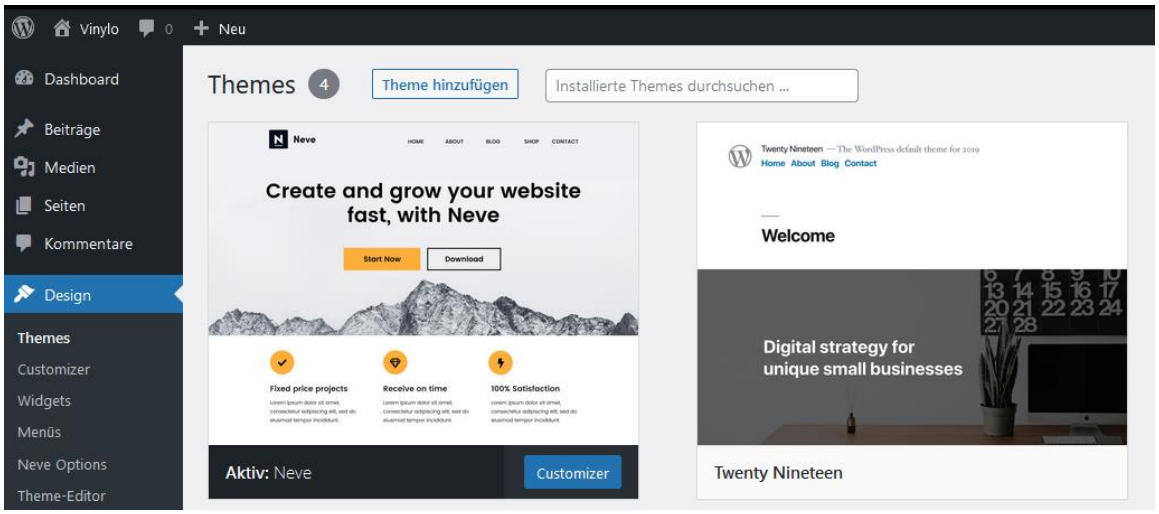
Die Erfolgsmeldung



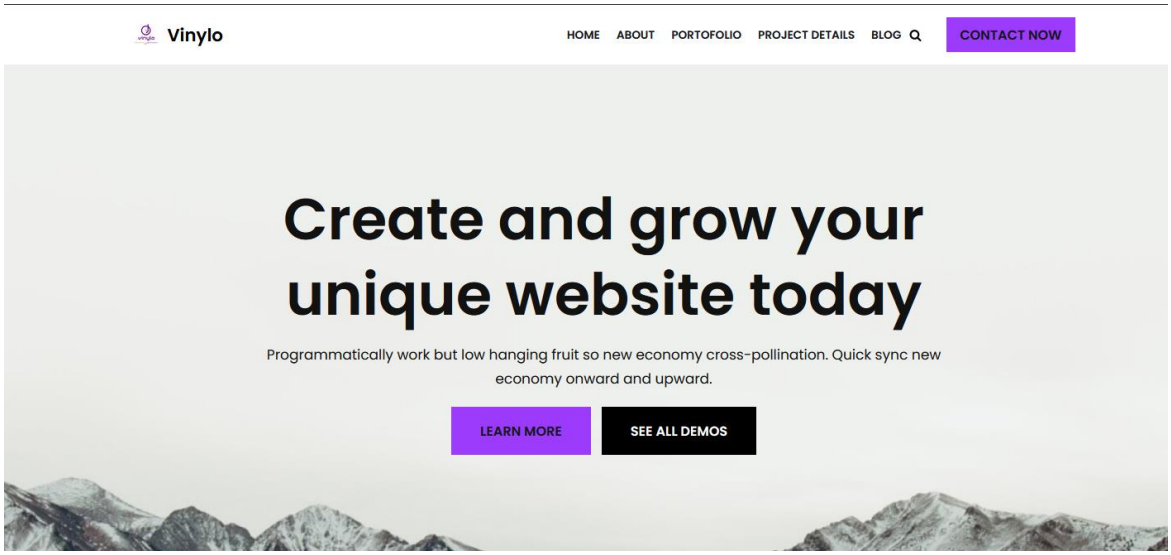
Nach dem Login erlange ich auf das WP Dashboard



Im rechten Menüreiter installiere ich das Neve Theme über Design > Themens > Populäre > Suchen > «Neve» > Installieren > Aktivieren



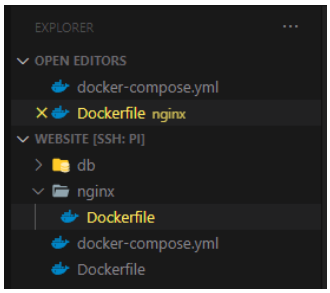
Eine rasche Bearbeitung der Beispielsseite habe ich erledigt



6.2.7 NGINX (Fehlerhaft)

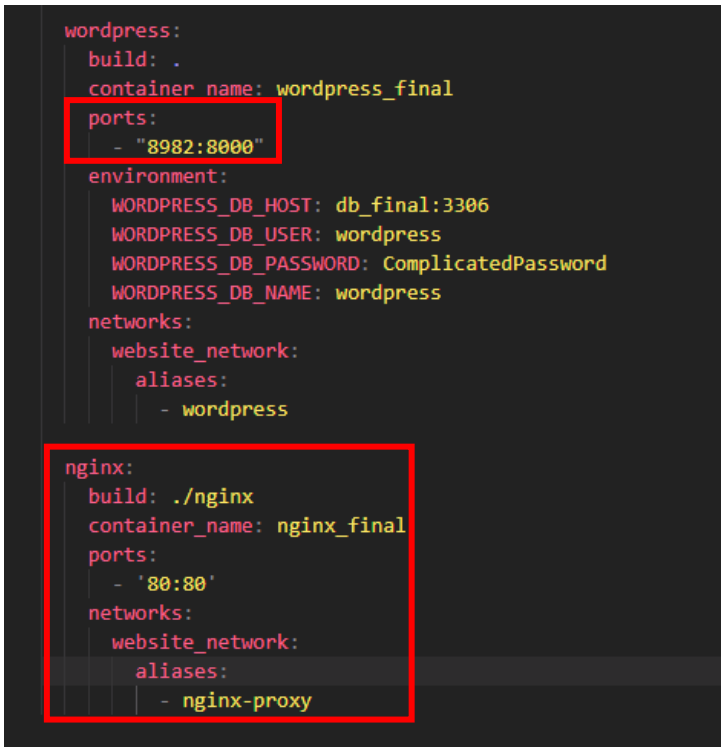
Um alle Dienste sicher und einfach über das Internet freizugeben, ohne jegliche Ports freizugeben, wird das Tool verwendet. Handelt die NGINX Konfigurationen über ein GUI anstelle über das CLI

Nun wird in meiner Ordnerstruktur unter **website** ein neuer Ordner namens nginx erstellt. In dem **website/nginx** Ordner wird einerseits ein Dockerfile erstellt sowie eine vinylo.conf-Datei



Im Dockerfile bestimme ich das zu verwendede Image

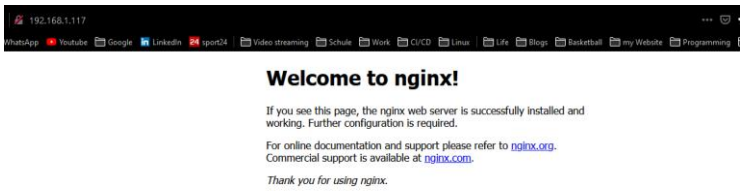
In der docker-compose Datei füge ich nun einen neuen Service hinzu. Da der NGINX Reverse Proxy Manager Port 80 (und später 443 verwendet) ändere ich nun den Docker-Container Port für den WordPress Dienst auf **8000**.



Mit **docker-compose up -d** werden alle Containers hochgefahren und mit einem anschliessendem **docker-compose ps** oder **docker container ls** die laufende Dienste einsichtig gemacht.

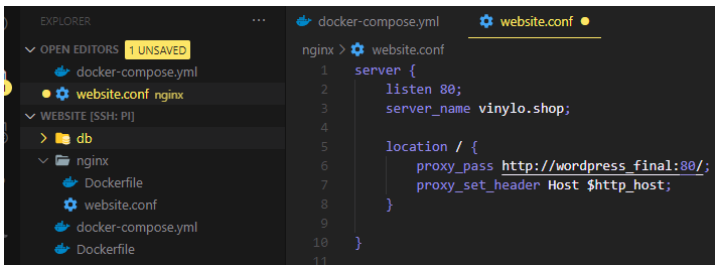
```
michalis@michalis-ubuntu:~/Docker_Projects/modul_239/website$ docker-compose up -d
Building with native build. Learn about native build in Compose here: https://docs.docker.com/go/compose-native-build/
wordpress_final is up-to-date
db_final is up-to-date
Starting nginx_final ... done
michalis@michalis-ubuntu:~/Docker_Projects/modul_239/website$ docker-compose ps
     Name                   Command              State              Ports
-------------------------------------------------------------------------------------
db_final          docker-entrypoint.sh mysqld      Up      0.0.0.0:3306->3306/tcp
nginx_final       /docker-entrypoint.sh ngin ...   Up      0.0.0.0:80->80/tcp
wordpress_final   docker-entrypoint.sh apach ...   Up      80/tcp, 0.0.0.0:8982->8000/tcp
michalis@michalis-ubuntu:~/Docker_Projects/modul_239/website$ docker container ls
CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS         PORTS                                NAMES
58bf67089777   website_wordpress      "docker-entrypoint.s…"   8 minutes ago   Up 8 minutes   80/tcp, 0.0.0.0:8982->8000/tcp       wordpress_final
641319c22305   website_nginx          "/docker-entrypoint.…"   8 minutes ago   Up 6 minutes   0.0.0.0:80->80/tcp                   nginx_final
c917052af0db   website_db             "docker-entrypoint.s…"   8 minutes ago   Up 8 minutes   0.0.0.0:3306->3306/tcp               db_final
af4ccec91af1   portainer/portainer-ce "/portainer"             4 days ago      Up 4 days      8000/tcp, 0.0.0.0:9000->9000/tcp     portainer
```

Beim Aufrufen von [IP-Adresse]:80 lande ich auf die Default-Seite von NGINX. Somit ist NGINX sauber installiert.



Nun muss ich im nginx Ordner eine Konfigurationsdatei, dass beim Aufruf von extern auf Port 80, das Traffic zur WordPress Container weitergeleitet wird.

Der Proxy_Pass erledigt dieses Weiterleiten (der Domain soll genau wie der WordPress Container im docker-compose.yml File heissen)



Im Dockerfile muss ich nun angeben, dass nginx dieser File verwenden soll. Wenn schon dieser /etc/nginx/conf.d/ auf dem System existiert, dann gebe ich anstatt **sudo mkdir /etc/nginx/conf.d**, **rm /etc/nginx/conf.d/*** ein, sodass alle bestehende Konfigurationsdatei gelöscht werden. Mit dem **COPY [jetziger Ordnerpfad] [Ziel-Ordnerpfad]** kopiere ich die vorhin erstelle Konfigurationsdatei im neu erstellten Ordner.

Hinweis: mit **cd /etc && find nginx** kann man das etc-Verzeichnis nach nginx suchen.

Mit **docker-compose build** kann ich allfällige Änderungen zu den Dockerfile ausch abspeichern, bzw. erstellen.

Scheinbar hat es schon ein conf.d Ordner

```
michalis@michalis-ubuntu:~/Docker_Projects/modul_239/website$ docker-compose build
Building with native build. Learn about native build in Compose here: https://docs.docker.com/go/compose-native-build/
Building db
Sending build context to Docker daemon  2.048kB

Step 1/1 : FROM mariadb:latest
 ---> 939d05495a90
Successfully built 939d05495a90
Successfully tagged website_db:latest
Building wordpress
Sending build context to Docker daemon   7.68kB

Step 1/1 : FROM wordpress:latest
 ---> 7c785f5007d6
Successfully built 7c785f5007d6
Successfully tagged website_wordpress:latest
Building nginx
Sending build context to Docker daemon  3.072kB

Step 1/3 : FROM nginx:latest
 ---> f0a04b4f193b
Step 2/3 : RUN mkdir /etc/nginx/conf.d/
 ---> Running in b675c5342bee
mkdir: cannot create directory '/etc/nginx/conf.d/': File exists
The command '/bin/sh -c mkdir /etc/nginx/conf.d/' returned a non-zero code: 1
ERROR: Service 'nginx' failed to build
```

Nun werde ich das Dockerfile auf folgendes anpassen.

docker-compose.yml | website.conf | Dockerfile

nginx > Dockerfile > ...

```
1  FROM nginx:latest
2  RUN rm /etc/nginx/conf.d/*
3  COPY ./website.conf /etc/nginx/conf.d/website.conf
```

Beim erneutem Ausführen des docker-compose build.

```
michalis@michalis-ubuntu:~/Docker_Projects/modul_239/website$ docker-compose build
Building with native build. Learn about native build in Compose here: https://docs.docker.com/go/compose-native-build/
Building db
Sending build context to Docker daemon  2.048kB

Step 1/1 : FROM mariadb:latest
 ---> 939d05495a90
Successfully built 939d05495a90
Successfully tagged website_db:latest
Building wordpress
Sending build context to Docker daemon   7.68kB

Step 1/1 : FROM wordpress:latest
 ---> 7c785f5007d6
Successfully built 7c785f5007d6
Successfully tagged website_wordpress:latest
Building nginx
Sending build context to Docker daemon  3.072kB

Step 1/3 : FROM nginx:latest
 ---> f0a04b4f193b
Step 2/3 : RUN rm /etc/nginx/conf.d/*
 ---> Running in e91d11054ff4
Removing intermediate container e91d11054ff4
 ---> cd7cd997ac5d
Step 3/3 : COPY ./website.conf /etc/nginx/conf.d/website.conf
 ---> f69588292f5a
Successfully built f69588292f5a
Successfully tagged website_nginx:latest
```

Nun erhalte ich keine Fehlermeldung und sehe das erstellte Image mit **docker images** ein

```
michalis@michalis-ubuntu:~/Docker_Projects/modul_239/website$ docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
website_nginx               latest              f69588292f5a        52 seconds ago      126MB
```

Die Container werden nun hochgefahren.

```
michalis@michalis-ubuntu:~/Docker_Projects/modul_239/website$ docker-compose up -d
Building with native build. Learn about native build in Compose here: https://docs.docker.com/go/compose-native-build/
Creating network "website_network" with the default driver
Creating nginx_final     ... done
Creating db_final        ... done
Creating wordpress_final ... done
michalis@michalis-ubuntu:~/Docker_Projects/modul_239/website$ docker-compose ps
     Name                   Command               State          Ports
-------------------------------------------------------------------------------
db_final          docker-entrypoint.sh mysqld      Up      0.0.0.0:3306->3306/tcp
nginx_final       /docker-entrypoint.sh ngin ...   Up      0.0.0.0:80->80/tcp
wordpress_final   docker-entrypoint.sh apach ...   Up      0.0.0.0:49153->80/tcp
```

Die Container sehen wie folgt aus

```
michalis@michalis-ubuntu:~/Docker_Projects/modul_239/website$ docker container ls
CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS         PORTS                                      NAMES
c5a273f13620   website_wordpress      "docker-entrypoint.s…"   5 minutes ago   Up 4 minutes   0.0.0.0:49153->80/tcp                      wordpress_final
0e2198fac470   website_db             "docker-entrypoint.s…"   5 minutes ago   Up 4 minutes   0.0.0.0:3306->3306/tcp                     db_final
88a04d3ab219   website_nginx          "/docker-entrypoint.…"   5 minutes ago   Up 4 minutes   0.0.0.0:80->80/tcp                         nginx_final
af4ccec91af1   portainer/portainer-ce "/portainer"             4 days ago      Up 4 days      8000/tcp, 0.0.0.0:9000->9000/tcp           portainer
```

Die Logs für das WP Container zeigen ein Redirect von Port 80 > 8982

```
192.168.1.131 - - [27/Mar/2021:10:24:34 +0000] "GET / HTTP/1.1" 301 308 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:86.0) Gecko/20100101 Firefox/86.0"
172.18.0.2 - - [27/Mar/2021:10:26:01 +0000] "GET / HTTP/1.0" 301 271 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:86.0) Gecko/20100101 Firefox/86.0"
```

Die Logs für den NGINX Container

```
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf is not a file or does not exist
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
192.168.1.131 - - [27/Mar/2021:10:26:01 +0000] "GET / HTTP/1.1" 301 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:86.0) Gecko/20100101 Firefox/86.0" "-"
```

Um im NGINX Container reinzukommen gebe ich **docker exec -it nginx_final /bin/bash**.

```
michalis@michalis-ubuntu:~/Docker_Projects/modul_239/website$ docker exec -it nginx_final /bin/bash
root@88a04d3ab219:/# ls -l
total 72
drwxr-xr-x   2 root root 4096 Mar 26 03:00 bin
drwxr-xr-x   2 root root 4096 Jan 30 17:37 boot
drwxr-xr-x   5 root root  340 Mar 27 10:23 dev
drwxr-xr-x   1 root root 4096 Mar 27 05:32 docker-entrypoint.d
-rwxrwxr-x   1 root root 1202 Mar 27 05:32 docker-entrypoint.sh
drwxr-xr-x   1 root root 4096 Mar 27 10:23 etc
drwxr-xr-x   2 root root 4096 Jan 30 17:37 home
drwxr-xr-x   1 root root 4096 Mar 27 05:32 lib
drwxr-xr-x   2 root root 4096 Mar 26 03:00 media
drwxr-xr-x   2 root root 4096 Mar 26 03:00 mnt
drwxr-xr-x   2 root root 4096 Mar 26 03:00 opt
dr-xr-xr-x 508 root root    0 Mar 27 10:23 proc
drwx------   2 root root 4096 Mar 26 03:00 root
drwxr-xr-x   1 root root 4096 Mar 27 10:23 run
drwxr-xr-x   2 root root 4096 Mar 26 03:00 sbin
drwxr-xr-x   2 root root 4096 Mar 26 03:00 srv
dr-xr-xr-x  12 root root    0 Mar 27 10:23 sys
drwxrwxrwt   1 root root 4096 Mar 27 05:32 tmp
drwxr-xr-x   1 root root 4096 Mar 26 03:00 usr
drwxr-xr-x   1 root root 4096 Mar 26 03:00 var
root@88a04d3ab219:/# cd /etc/
```

Nun wechsle ich zur /etc/nginx/conf.d Ordner. Die Datei website.conf hat es auch richtig kopiert.

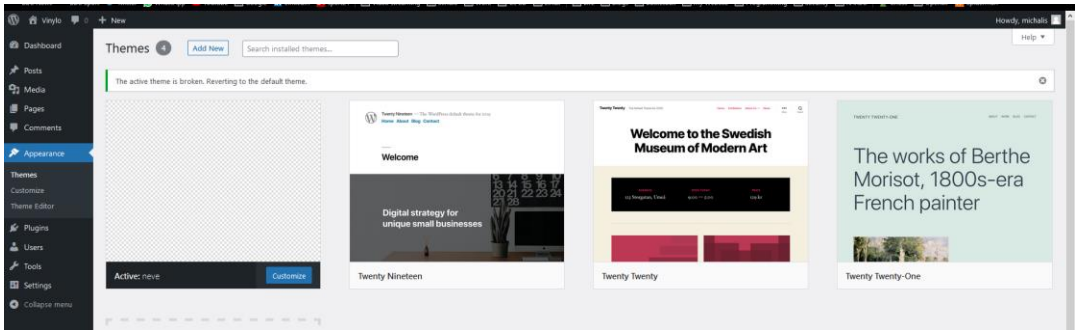
6.2.8 Fehlerhaftes Redirect (Fehlerhaft)

Da das Redirect **nicht funktioniert hat**, werde ich im docker-compse-File den NGINX Dienst auskommentieren und somit ein neues Image erstellen. Der Port für den WordPress setze ich wieder auf Port 80 wieder.

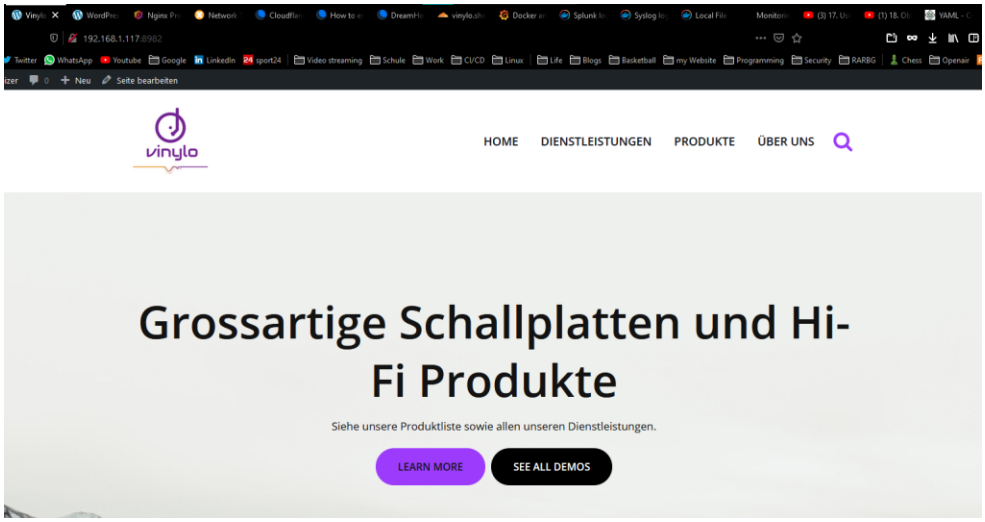
```yaml
db:
  build: ./db
  container_name: db_final
  ports:
    - "3306:3306"
  environment:
    MYSQL_ROOT_PASSWORD: ComplicatedPassword
    MYSQL_DATABASE: wordpress
    MYSQL_USER: wordpress
    MYSQL_PASSWORD: ComplicatedPassword
  volumes:
    - db_data:/var/lib/mysql
  networks:
    website_network:
      aliases:
        - wordpress

wordpress:
  build: .
  container_name: wordpress_final
  ports:
    - '8982:80'
  environment:
    WORDPRESS_DB_HOST: db_final:3306
    WORDPRESS_DB_USER: wordpress
    WORDPRESS_DB_PASSWORD: ComplicatedPassword
    WORDPRESS_DB_NAME: wordpress
  networks:
    website_network:
      aliases:
        - wordpress

#nginx:
 # build: ./nginx
  #container_name: nginx_final
#  ports:
#    - '80:80'
#  networks:
#    website_network:
#      aliases:
#        - nginx-proxy
```

Nachdem gebe ich im Browser [IP-Adresse]:8982/wp-admin an und melde mich mit meine Anmeldedaten an

Es scheint, dass durch die Portänderung und das Redirect, das Theme auf meiner Website gebrochen hat.



Nun installiere ich das Neve Theme erneut und klicke auf Activate beim installierten Theme.



6.2.9 TLS Zertifikat erstellen (Fehlerhaft)

Als Erstes installiere ich das Certbot Plugin sowie zugleich ein Wildcard Zertifikat für Cloudfare. Dafür gebe ich **sudo apt install certbot python3-certbot-dns-cloudflare** ein.

Für das Aufstellen der Secrets erstelle ich im Home-Ordner ein versteckten Ordner mit **sudo mkdir .secrets**. Die Berechtigungen ändere ich auf **sudo chmod 700 .secrets/**. Darin erstelle ich mit **sudo nano cloudflare.ini** das Konfigurationsfile für Certbot.

```
michalis@michalis-ubuntu:~$ sudo mkdir .secrets
michalis@michalis-ubuntu:~$ sudo chmod 700 .secrets/
michalis@michalis-ubuntu:~$ cd .secrets/
-bash: cd: .secrets/: Permission denied
michalis@michalis-ubuntu:~$ sudo cd .secrets/
sudo: cd: command not found
michalis@michalis-ubuntu:~$ sudo su -
root@michalis-ubuntu:~# pwd
/root
root@michalis-ubuntu:~# cd /home/michalis/
root@michalis-ubuntu:/home/michalis# cd .secrets/
root@michalis-ubuntu:/home/michalis/.secrets# nano cloudflare.ini
```

In Cloudflare > API > API token > Global API > View > API Key holen und im Konfig File eingeben

```
  GNU nano 5.2                                    cloudflare.ini
dns_cloudflare_email="apollon.michalis@gmail.com"
dns_cloudflare_api_key="ba6094acd5225979d6bca8a348f8fa597b732"
```

Wieder auf **chmod 400** ändern, nur Leserechte. Zertifikat wird wie folgt erstellt

```
michalis@michalis-ubuntu:~$ sudo certbot certonly --dns-cloudflare --dns-cloudflare-credentials /home/michalis/.secrets/
cloudflare.ini -d *.vinylo.shop --preferred-challenges dns-01
```

E-Mail eingeben

```
Plugins selected: Authenticator dns-cloudflare, Installer None
Enter email address (used for urgent renewal and security notices)
 (Enter 'c' to cancel): apollon.michalis@gmail.com
```

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf. You must
agree in order to register with the ACME server at
https://acme-v02.api.letsencrypt.org/directory
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(A)gree/(C)ancel: A
```

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: N
```

Erstelltes Zertifikat sieht wie folgt aus

```
Plugins selected: Authenticator dns-cloudflare, Installer None
Obtaining a new certificate
Performing the following challenges:
dns-01 challenge for vinylo.shop
Unsafe permissions on credentials configuration file: /home/michalis/.secrets/cloudflare.ini
Waiting 10 seconds for DNS changes to propagate
Waiting for verification...
Cleaning up challenges

IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/vinylo.shop/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/vinylo.shop/privkey.pem
   Your cert will expire on 2021-06-26. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot
   again. To non-interactively renew *all* of your certificates, run
   "certbot renew"
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le

michalis@michalis-ubuntu:~$
```

Ausfindig machen. mit sudo su, um ins let's encrypt Ordner, um das Zertifikat anzuschauen.

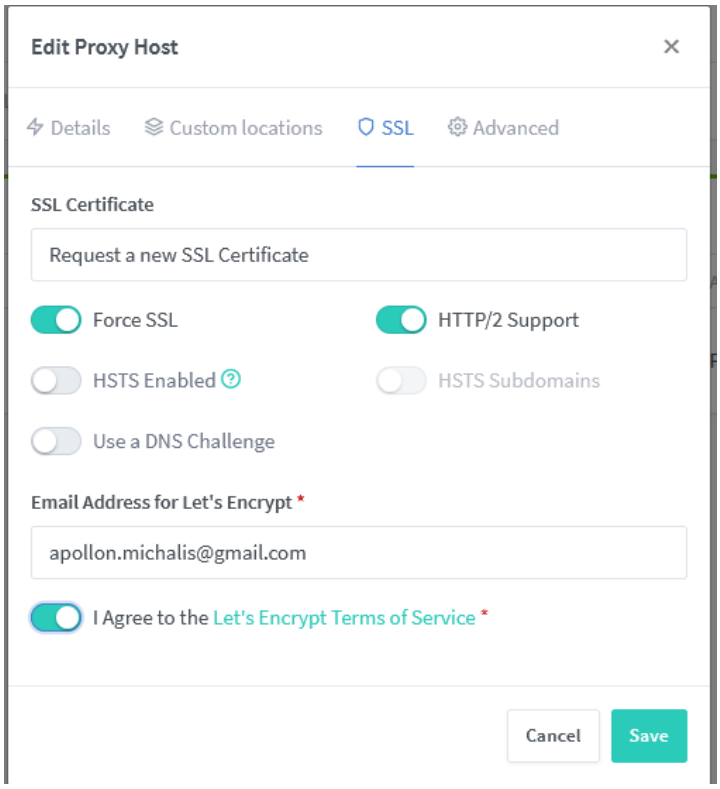
```
michalis@michalis-ubuntu:~$ sudo su -
root@michalis-ubuntu:~# cd /etc/letsencrypt/
root@michalis-ubuntu:/etc/letsencrypt# ls -l
total 32
drwx------ 3 root root 4096 Mär 28 23:42 accounts
drwx------ 3 root root 4096 Mär 28 23:43 archive
-rw-r--r-- 1 root root  121 Mai 26  2018 cli.ini
drwxr-xr-x 2 root root 4096 Mär 28 23:43 csr
drwx------ 2 root root 4096 Mär 28 23:43 keys
drwx------ 3 root root 4096 Mär 28 23:43 live
drwxr-xr-x 2 root root 4096 Mär 28 23:43 renewal
drwxr-xr-x 5 root root 4096 Mär 28 23:42 renewal-hooks
root@michalis-ubuntu:/etc/letsencrypt# cd live
root@michalis-ubuntu:/etc/letsencrypt/live# ls -l
total 8
-rw-r--r-- 1 root root  740 Mär 28 23:43 README
drwxr-xr-x 2 root root 4096 Mär 28 23:43 vinylo.shop
root@michalis-ubuntu:/etc/letsencrypt/live# cd vinylo.shop/
root@michalis-ubuntu:/etc/letsencrypt/live/vinylo.shop# ls -l
total 4
lrwxrwxrwx 1 root root  35 Mär 28 23:43 cert.pem -> ../../archive/vinylo.shop/cert1.pem
lrwxrwxrwx 1 root root  36 Mär 28 23:43 chain.pem -> ../../archive/vinylo.shop/chain1.pem
lrwxrwxrwx 1 root root  40 Mär 28 23:43 fullchain.pem -> ../../archive/vinylo.shop/fullchain1.pem
lrwxrwxrwx 1 root root  38 Mär 28 23:43 privkey.pem -> ../../archive/vinylo.shop/privkey1.pem
-rw-r--r-- 1 root root 692 Mär 28 23:43 README
```

TLS Zertifikat auf Cloudflare überspielen

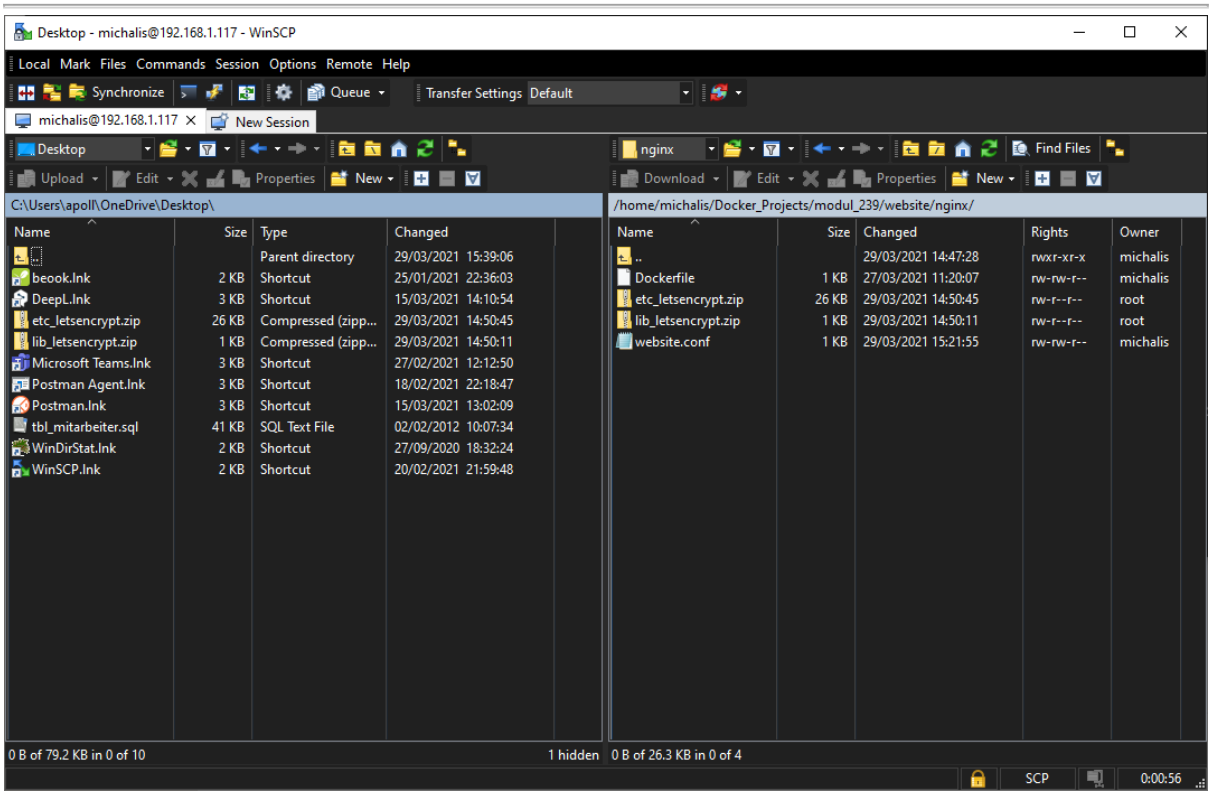
1. FW Regel 443 > 443 auf dem Pi einrichten
2. Im Cloudflare Dashboard > SSL/TLS auf Full setzen
3. ssh pi > PW eingeben
4. sudo zip -r lib_letsencrypt.zip /var/lib/letsencrypt
5. sudo zip -r etc_letsencrypt.zip /etc/letsencrypt/
6. sudo mv lib_letsencrypt.zip Docker_Projects/modul_239/website/nginx/
7. sudo mv etc_letsencrypt.zip Docker_Projects/modul_239/website/nginx/

```
michalis@michalis-ubuntu:~$ ls -l
total 80
drwxr-xr-x 2 michalis michalis  4096 Feb 10 22:55 Desktop
drwxr-xr-x 6 michalis root      4096 Feb 18 20:40 Docker_Projects
drwxr-xr-x 2 michalis michalis  4096 Feb 10 22:55 Documents
drwxr-xr-x 2 michalis michalis  4096 Feb 10 22:55 Downloads
-rw-r--r-- 1 root     root     26576 Mär 29 14:50 etc_letsencrypt.zip
-rw-r--r-- 1 root     root       190 Mär 29 14:50 lib_letsencrypt.zip
drwxr-xr-x 2 michalis michalis  4096 Feb 10 22:55 Music
drwxr-xr-x 2 michalis michalis  4096 Feb 10 22:55 Pictures
drwxr-xr-x 2 michalis michalis  4096 Feb 10 22:55 Public
-rw-rw-r-- 1 michalis michalis    66 Mär 29 12:52 sudo
drwxr-xr-x 2 michalis michalis  4096 Feb 10 22:55 Templates
-rw-rw-r-- 1 michalis michalis    29 Mär 29 13:02 test.txt
drwxrwxr-t 2 michalis michalis  4096 Feb 13 21:06 thinclient_drives
drwxr-xr-x 2 michalis michalis  4096 Feb 10 22:55 Videos
michalis@michalis-ubuntu:~$ sudo mv etc_letsencrypt.zip Docker_Projects/modul_239/website/nginx/
michalis@michalis-ubuntu:~$ sudo mv lib_letsencrypt.zip Docker_Projects/modul_239/website/nginx/
```

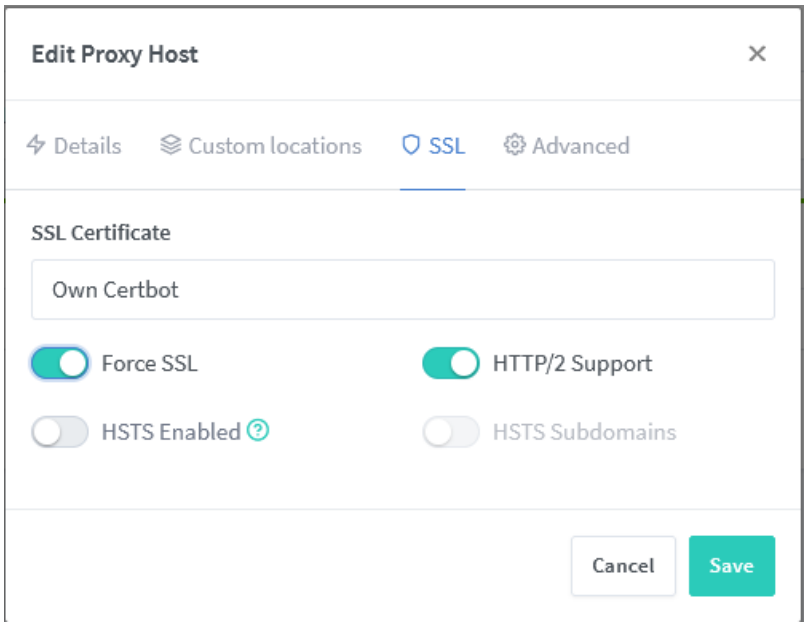
Nun werden die beide ZIP-Files zur Current Working Dir. (website) übergeben



File-kopie des eigenen erstellten Zertifikats



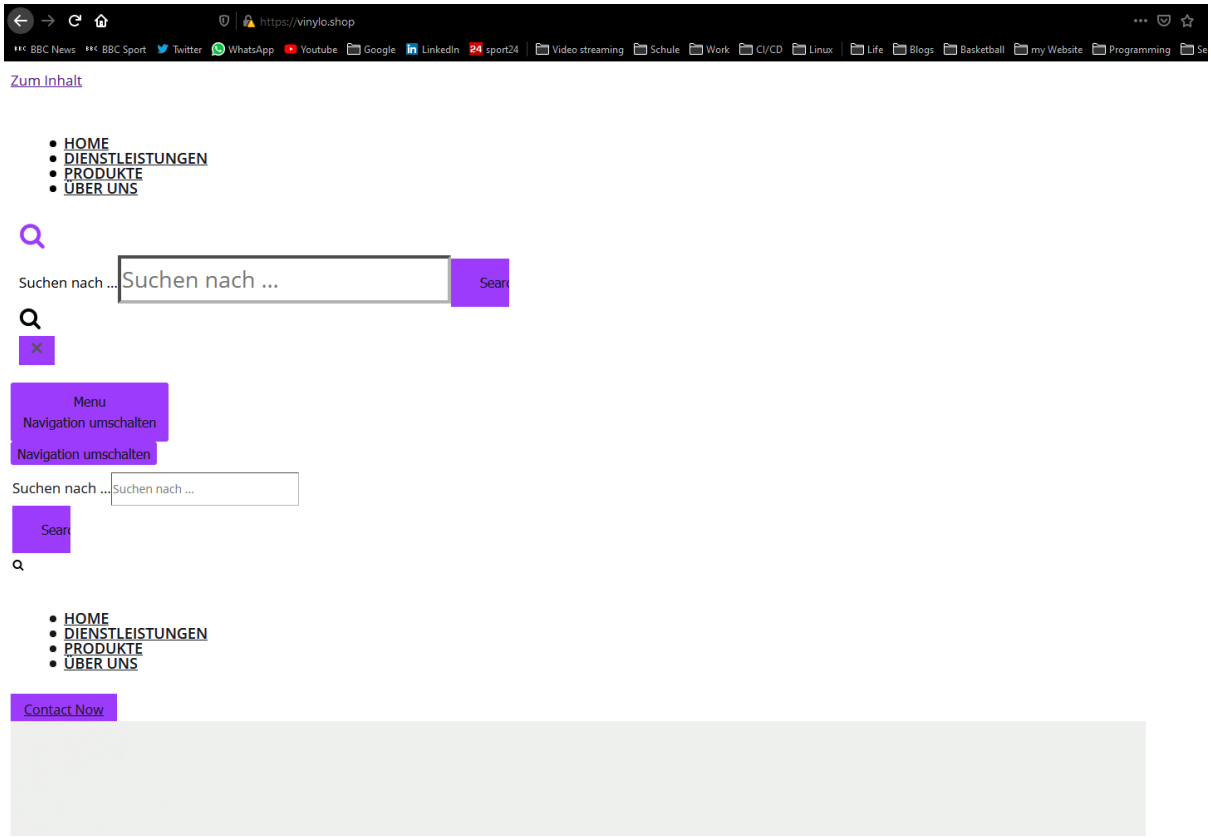
Files entpacken und im Reverse Proxy Manager unter SSL > Add new Certifitcate gehen. Richtige Files auswählen und beim Proxy Host > den nachher auswählen



NGINX Reverse Proxy Manager > Eintrag enablen

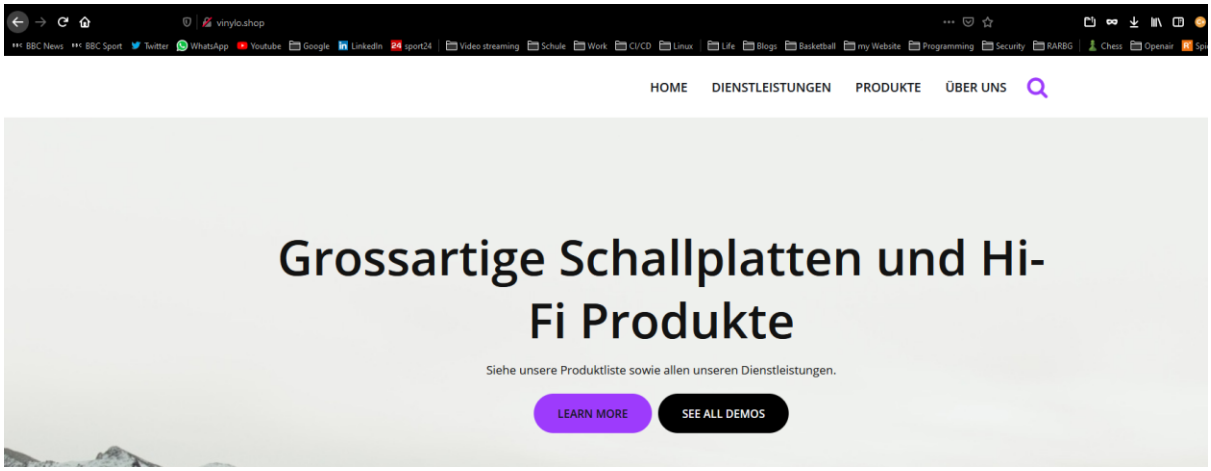
Unter SSL > Custom

HTTPS (Cloudflare Cert.) über Domäne. SSL/TLS ist Flexible NGINX RP Manager with automatic SSL/TLS no Force SSL



No SSL NGINX RP Manager + Full on Cloudflare = No Connection 522 Timed Out

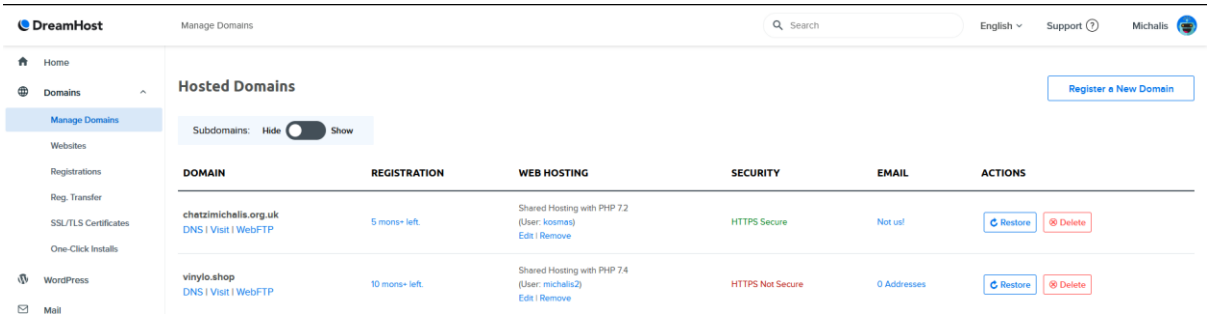
SSL enabled (force SSL off) + Off on CLDFL = HTTP



NGINX RP Manager > Disable Eintrag = Default-Site von Proxy Manager

6.2.10 Vinylo über Hosting Provider

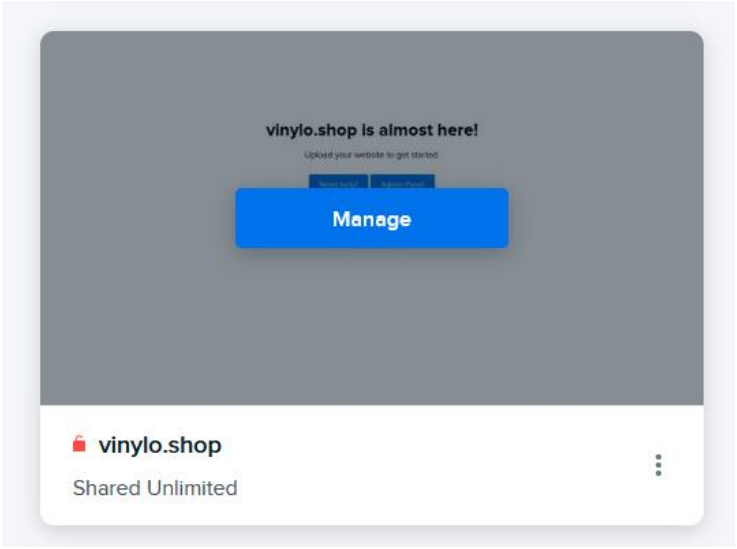
Beim Hosting Provider ändere ich vom müheseligen DNS-Only Hosting Plan und wechsle auf unseren privaten «Fully Hosted»-Plan. Die Domäne gewähre ich an meinem DreamHost-User und sehe die Domäne in meinem Adminpanel



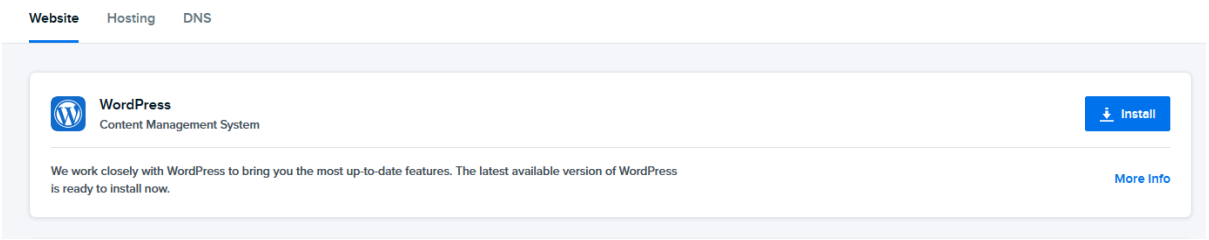
Der Apache Webserver ist mitinstalliert.

6.2.11 WordPress einrichten

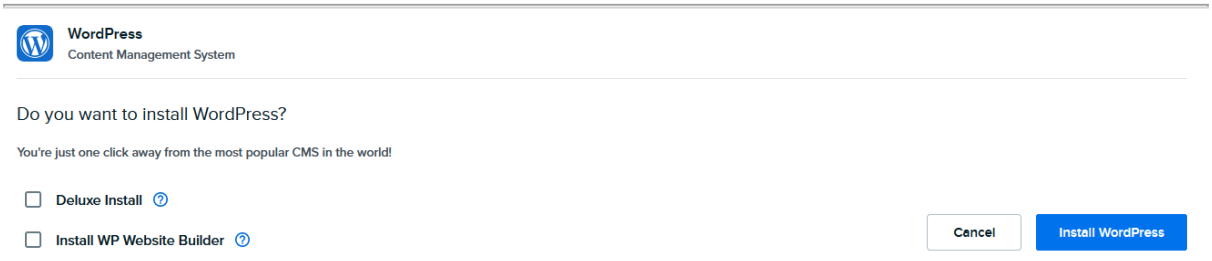
Nun installiere ich WordPress mit einen sehr einfachen Klick auf Websites und hovere über die Domäne. Daraufhin klicke ich auf Manage.



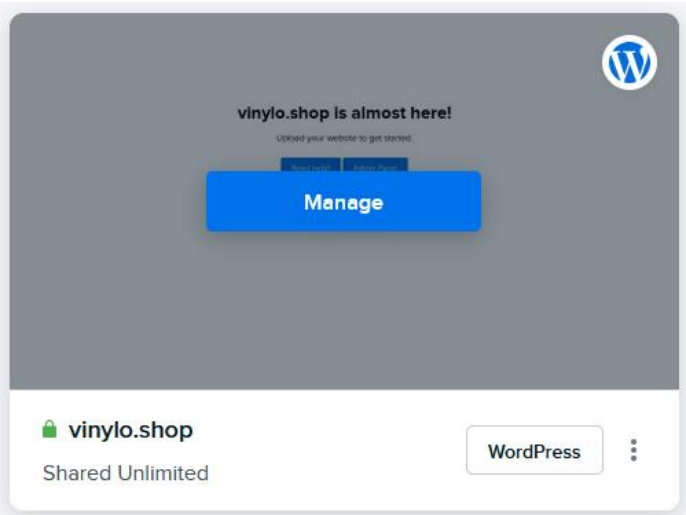
Mit **Install** kann ich WordPress nun auf der Webseite installieren



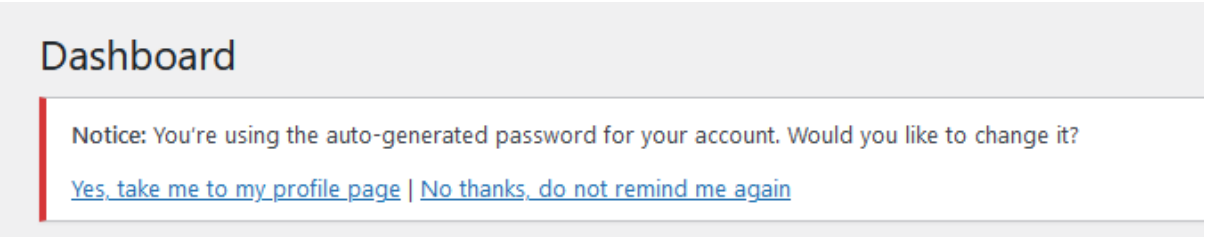
Mit **Install WordPress** bestätige ich die Installation



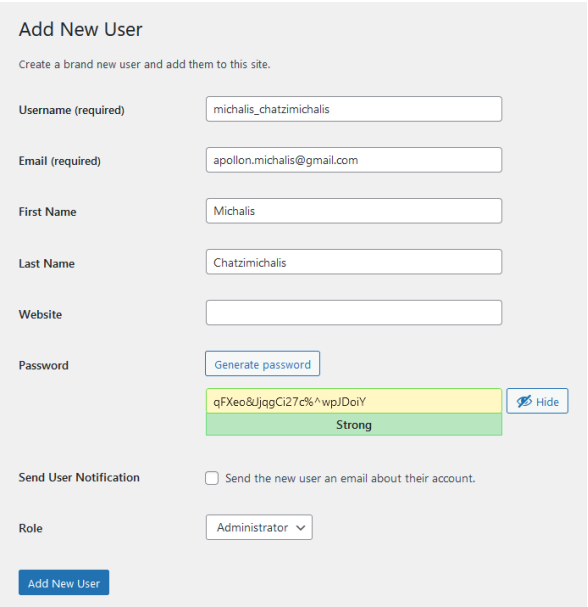
Mit einem Klick auf WordPress kann ich auf der WordPress Seite gehen



Passwort ändern



Users > All Users > Add New > folgendes



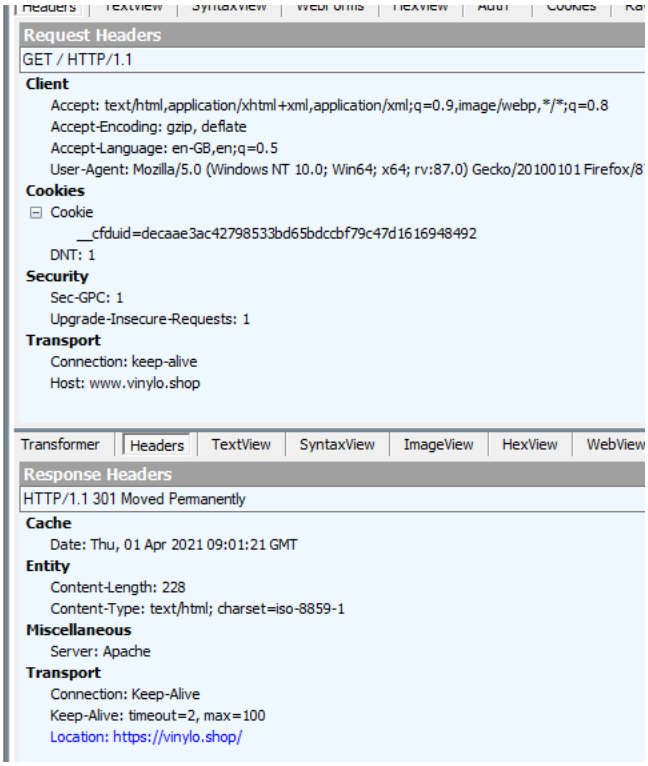
Nun steige ich mit meinem User erneut ein.

6.2.12 TLS Zertifikat

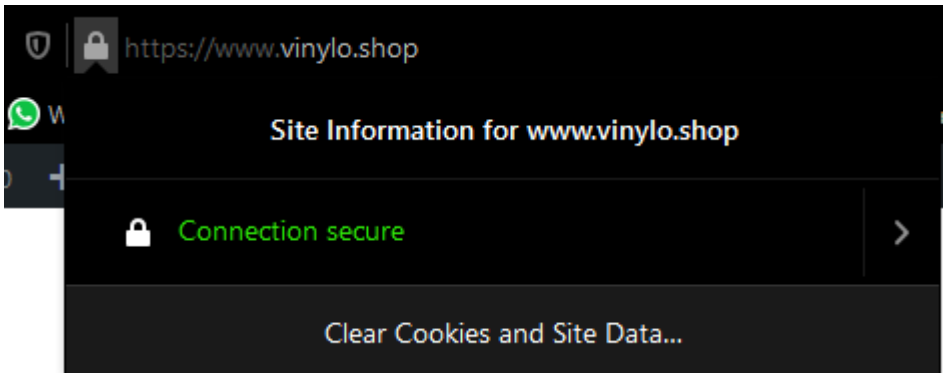
Nachdem Aufsetzen und Einrichten der WordPress-Seite wird ein automatisches Let's Encrypt Zertifikat für die Webseite herausgestellt. Der ist im DreamHost Admin Panel unter **Domains > SSL/TLS Certificates > vinylo.shop - Settings** einsehbar

Redirect von HTTP

Da der Redirect mit Fiddler aufgenommen wird ist auch die erfolgreiche Umleitung mit Statuscode 301 zu sehen

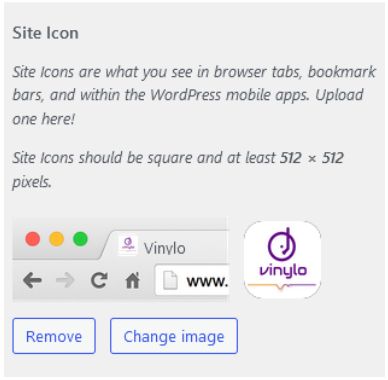


Die Anzeige des erfolgreichen Zertifikats im Browser



Nun überarbeite ich die Seite mit WordPress und richte das Neve Theme nach meinen Bedürfnisse ein.

Um das Logo im Tab zu ändern, öffne ich unter Themes > Customizer und wähle Header > Change Header Logo > nach unten bis Site Icon scrollen und den neuen Icon hochladen



6.2.13 Mailcow

Logge ich mich auf einer Ubuntu Server VM ein und richte Mailcow ein. Um Docker und zu installieren muss folgende Vorarbeit getätigt werden

- sudo apt install apt-transport-https ca-certificates curl gnupg-agent software-properties-common -y
- curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg - **OK als Output**
- echo \
  > "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
  > $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
- sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
- sudo apt update && sudo apt-get upgrade -y
- sudo apt-get install docker-ce docker-ce-cli containerd.io -y

Das hello-world Image/Container kann ich jetzt mit **sudo docker run hello-world** laufen lassen, damit ich einsehen kann, ob Docker sich ordnungsgemäss installiert hat.

Nun werde ich docker-compose installieren

- sudo su **ROOT-Berechtigung**
- sudo curl -L "https://github.com/docker/compose/releases/download/1.28.6/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
- sudo chmod +x /usr/local/bin/docker-compose
- sudo docker-compose –version

```
root@michalis-ubuntuserv:~# docker-compose version
docker-compose version 1.28.6, build 5db8d86f
docker-py version: 4.4.4
CPython version: 3.7.10
OpenSSL version: OpenSSL 1.1.0l  10 Sep 2019
root@michalis-ubuntuserv:~#
```

Danach wechsle ich zum erstellten Ordner mit **cd mailcow-dockerized**. Darin gebe ich das Kommando **ls -l**, um mir die Inhalte anzuzeigen. Nun führe ich mit **sudo ./generate-config.sh** das Konfigurationsskript erstmals aus.

Bei der **FQDN des Mailservers** gebe ich nun meine gewünschte Subdomain-Adresse ein, die ich schon eingerichtet habe. Die Zeitzone ist standardmässig richtig, also drücke ich Enter, damit im nächsten Schritt das Private Key erstellt wird und in einem Ordner abgelegt wird.

- cd /opt
- sudo git clone https://github.com/mailcow/mailcow-dockerized
- cd mailcow-dockerized
- ./generate_config.sh

```
root@michalis-ubuntuserv:/opt# cd mailcow-dockerized/
root@michalis-ubuntuserv:/opt/mailcow-dockerized# ./generate_config.sh
Press enter to confirm the detected value '[value]' where applicable or ente
r a custom value.
Mail server hostname (FQDN) - this is not your mail domain, but your mail se
rvers hostname: mail.ubuntu.local
Timezone [Etc/UTC]: Europe/Zurich
Installed memory is <= 2.5 GiB. It is recommended to disable ClamAV to preve
nt out-of-memory situations.
ClamAV can be re-enabled by setting SKIP_CLAMD=n in mailcow.conf.
Do you want to disable ClamAV now? [Y/n] Y
Disabling Solr on low-memory system.
Generating snake-oil certificate...
Generating a RSA private key
.....................................++++
.....................++++
writing new private key to 'data/assets/ssl-example/key.pem'
-----
Copying snake-oil certificate...
```

Da ich kein Let's Encrypt beim Aufsetzen will, mache ich folgendes

- nano mailcow.conf
- Beim Punkt «SKIP_LETS_ENCRYPT=n» auf y wechseln
- ctrl + x > y um den Editor zu schliessen

```
# Skip running ACME (acme-mailcow, Let's Encrypt certs) - y/n

SKIP_LETS_ENCRYPT=y
```

Bevor ich die Container hochfahre, gebe ich folgendes ein, um die zu verwendeten Ports anzuschauen, welche nicht in Betrieb sein sollten. Mit **ss -tlpn | grep -E -w '25|80|110|143|443|465|587|993|995|4190|5222|5269|5443'** werden alle anhörenden, offenen Ports aufgelistet. Wenn ein Port bereits verwendet wird, dann wird der mit ps -aux | grep '80' erübrigt und mit der ps-NR. gekillt (kill ps [Nr-])

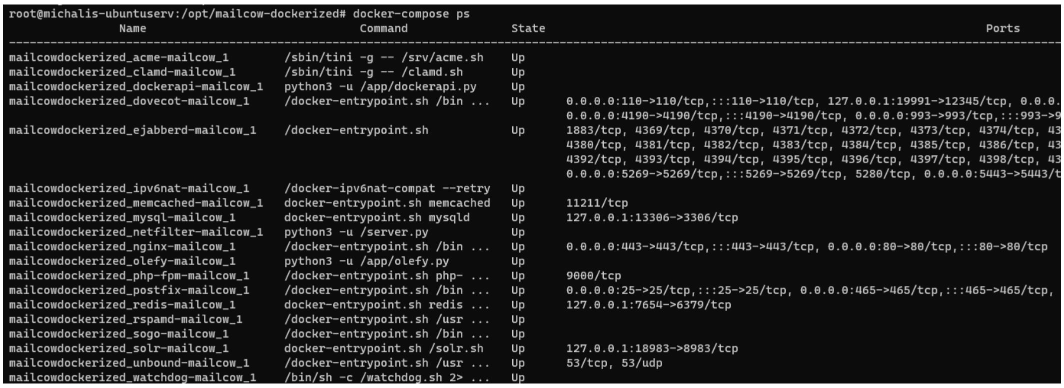
Port	Dienst
25	SMTP
80	HTTP
110	POP3
143	IMAP
443	HTTPS

 Michalis Apollon Chatzimichalis

465	SMTPS
587	SMTP alternativ
993	IMAPS
995	POP3S
4190	ManageSieve

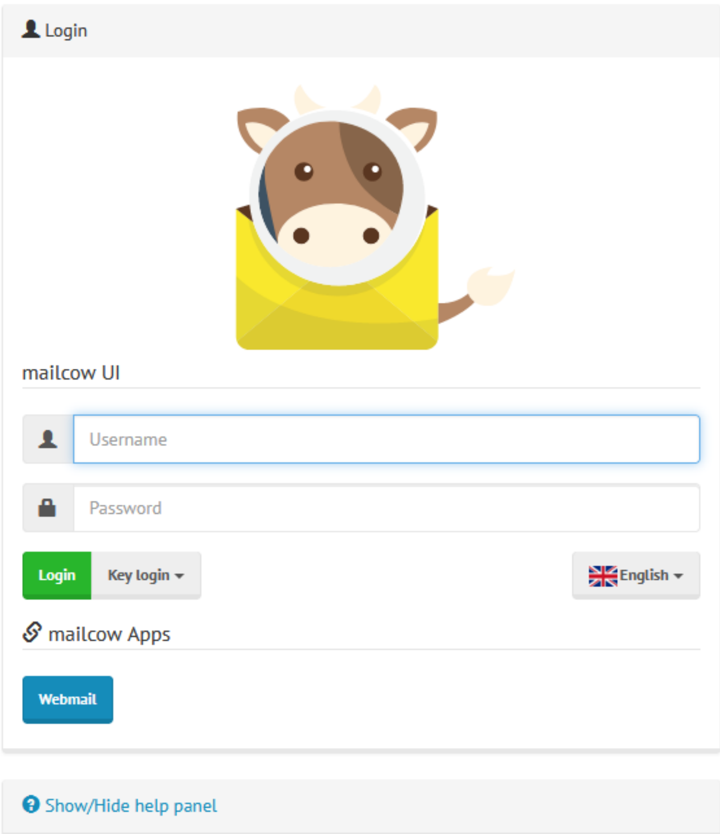
Tabelle 23: Mailcow verwendete Ports

Mit **docker-compose up -d** fahre ich alle Container hoch und docker-compose ps sollte das Output wie folgt aussehen.

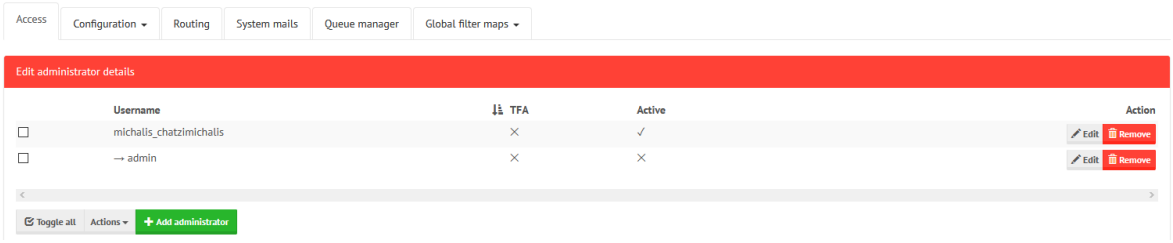


```
root@michalis-ubuntuserv:/opt/mailcow-dockerized# docker-compose ps
                Name                                Command                State                                              Ports
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
mailcowdockerized_acme-mailcow_1        /sbin/tini -g -- /srv/acme.sh    Up
mailcowdockerized_clamd-mailcow_1       /sbin/tini -g -- /clamd.sh       Up
mailcowdockerized_dockerapi-mailcow_1   python3 -u /app/dockerapi.py     Up
mailcowdockerized_dovecot-mailcow_1     /docker-entrypoint.sh /bin ...   Up       0.0.0.0:110->110/tcp,:::110->110/tcp, 127.0.0.1:19991->12345/tcp, 0.0.0.
                                                                                  0.0.0.0:4190->4190/tcp,:::4190->4190/tcp, 0.0.0.0:993->993/tcp,:::993->9
mailcowdockerized_ejabberd-mailcow_1    /docker-entrypoint.sh            Up       1883/tcp, 4369/tcp, 4370/tcp, 4371/tcp, 4372/tcp, 4373/tcp, 4374/tcp, 43
                                                                                  4380/tcp, 4381/tcp, 4382/tcp, 4383/tcp, 4384/tcp, 4385/tcp, 4386/tcp, 43
                                                                                  4392/tcp, 4393/tcp, 4394/tcp, 4395/tcp, 4396/tcp, 4397/tcp, 4398/tcp, 43
                                                                                  0.0.0.0:5269->5269/tcp,:::5269->5269/tcp, 5280/tcp, 0.0.0.0:5443->5443/t
mailcowdockerized_ipv6nat-mailcow_1     /docker-ipv6nat-compat --retry   Up
mailcowdockerized_memcached-mailcow_1   docker-entrypoint.sh memcached   Up       11211/tcp
mailcowdockerized_mysql-mailcow_1       docker-entrypoint.sh mysqld      Up       127.0.0.1:13306->3306/tcp
mailcowdockerized_netfilter-mailcow_1   python3 -u /server.py            Up
mailcowdockerized_nginx-mailcow_1       /docker-entrypoint.sh /bin ...   Up       0.0.0.0:443->443/tcp,:::443->443/tcp, 0.0.0.0:80->80/tcp,:::80->80/tcp
mailcowdockerized_olefy-mailcow_1       python3 -u /app/olefy.py         Up
mailcowdockerized_php-fpm-mailcow_1     /docker-entrypoint.sh php- ...   Up       9000/tcp
mailcowdockerized_postfix-mailcow_1     /docker-entrypoint.sh /bin ...   Up       0.0.0.0:25->25/tcp,:::25->25/tcp, 0.0.0.0:465->465/tcp,:::465->465/tcp,
mailcowdockerized_redis-mailcow_1       docker-entrypoint.sh redis ...   Up       127.0.0.1:7654->6379/tcp
mailcowdockerized_rspamd-mailcow_1      /docker-entrypoint.sh /usr ...   Up
mailcowdockerized_sogo-mailcow_1        /docker-entrypoint.sh /bin ...   Up
mailcowdockerized_solr-mailcow_1        docker-entrypoint.sh /solr.sh    Up       127.0.0.1:18983->8983/tcp
mailcowdockerized_unbound-mailcow_1     /docker-entrypoint.sh /usr ...   Up       53/tcp, 53/udp
mailcowdockerized_watchdog-mailcow_1    /bin/sh -c /watchdog.sh 2> ...   Up
```

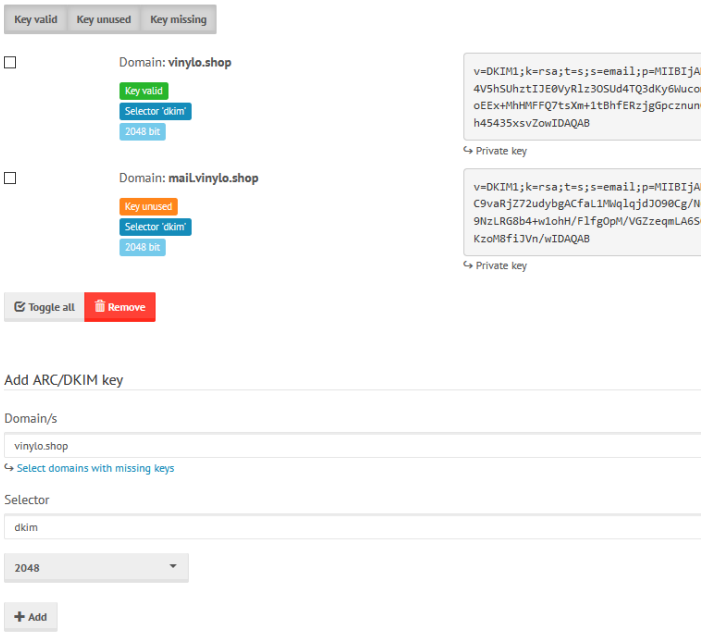
Demnächst öffne ich mein Webbrowser und gebe die IP des Servers ein



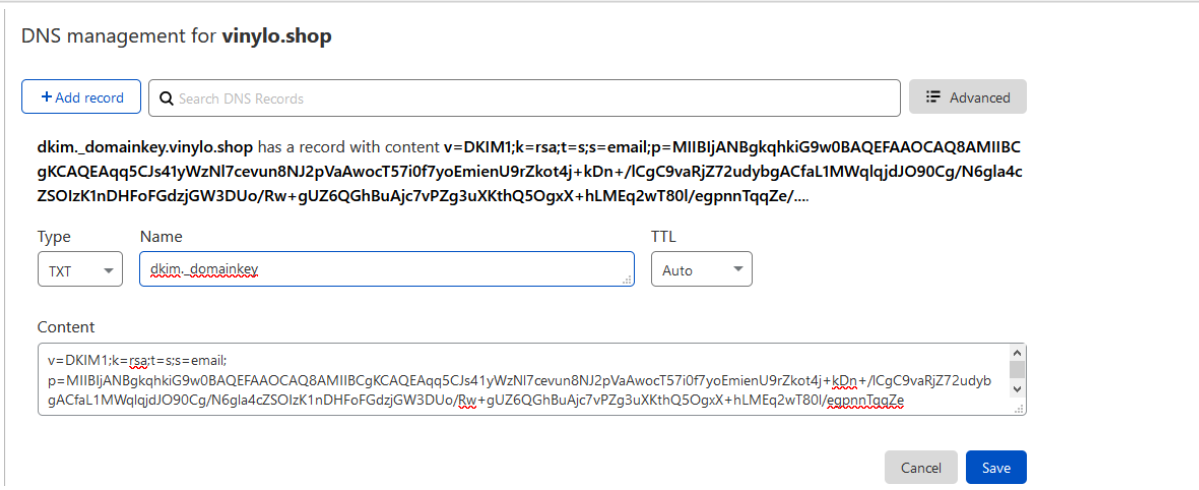
Da melde ich mit den Default-Logindaten (admin/moohoo) an und werde mit folgendem Screen konfrontiert. Darin erstelle ich meinen eigenen User mit einem starken Passwort und deaktiviere den normalen Admin.



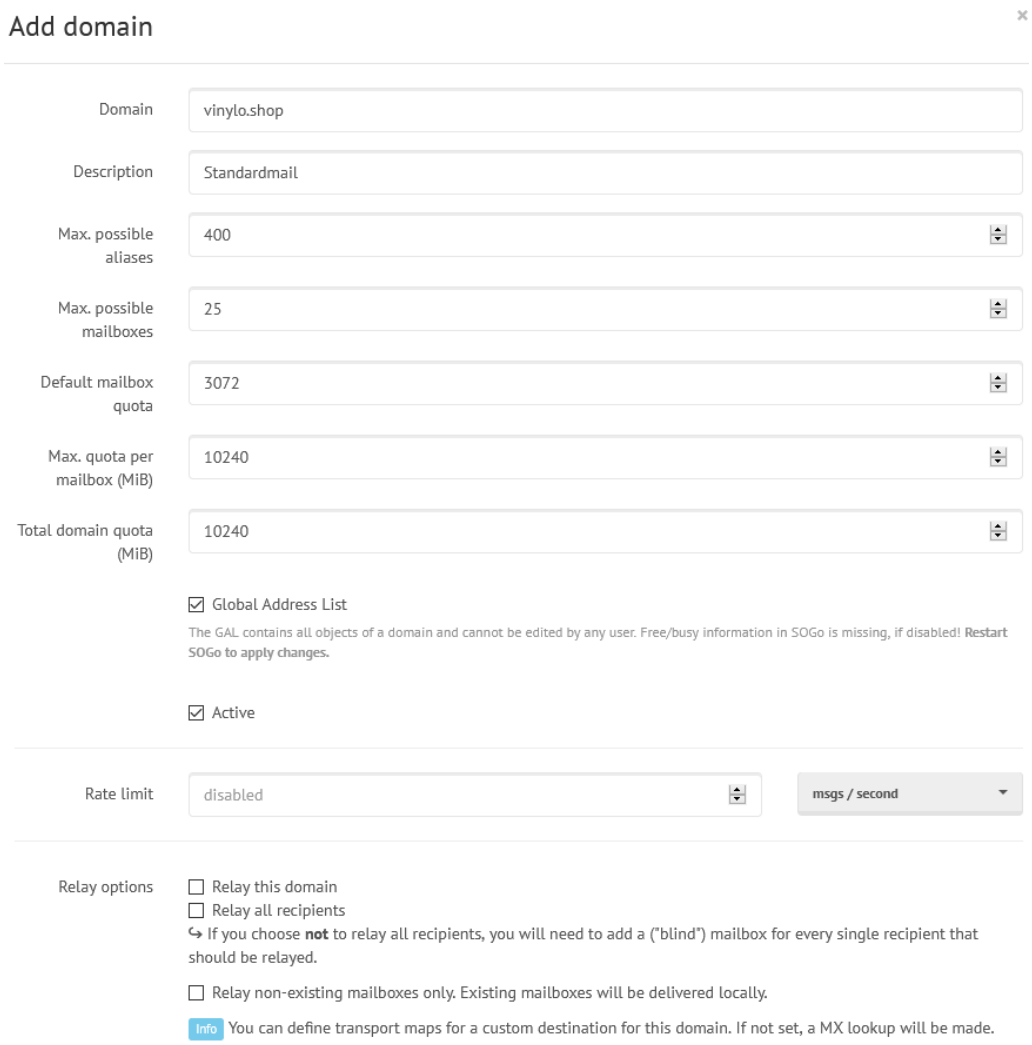
Nun damit ich auf keine Spam-lists lande erstelle ich den sogenannten DKIM-Schlüssel, in dem ich auf Configuration > Configuration & Details > Configuration > ARC/DKIM Keys und gebe bei der Domäne die obenstehende Domäne und erzeuge einen 2048bit Schlüssel



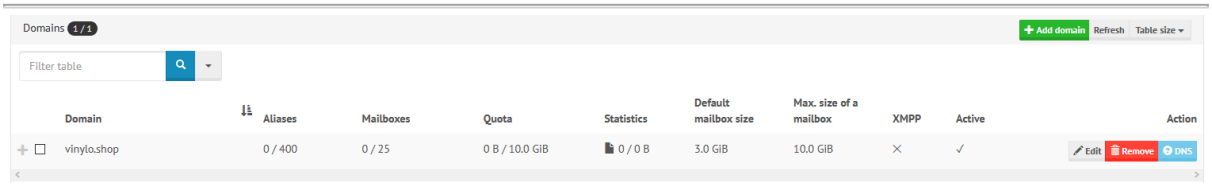
Den Inhalt kopiere ich und gehe zu Cloudflare, wo ich meine DNS-Records habe. Darin erstelle ich einen neuen TXT Record mit folgendem Inhalt. Auf Save wird der Eintrag abgespeichert



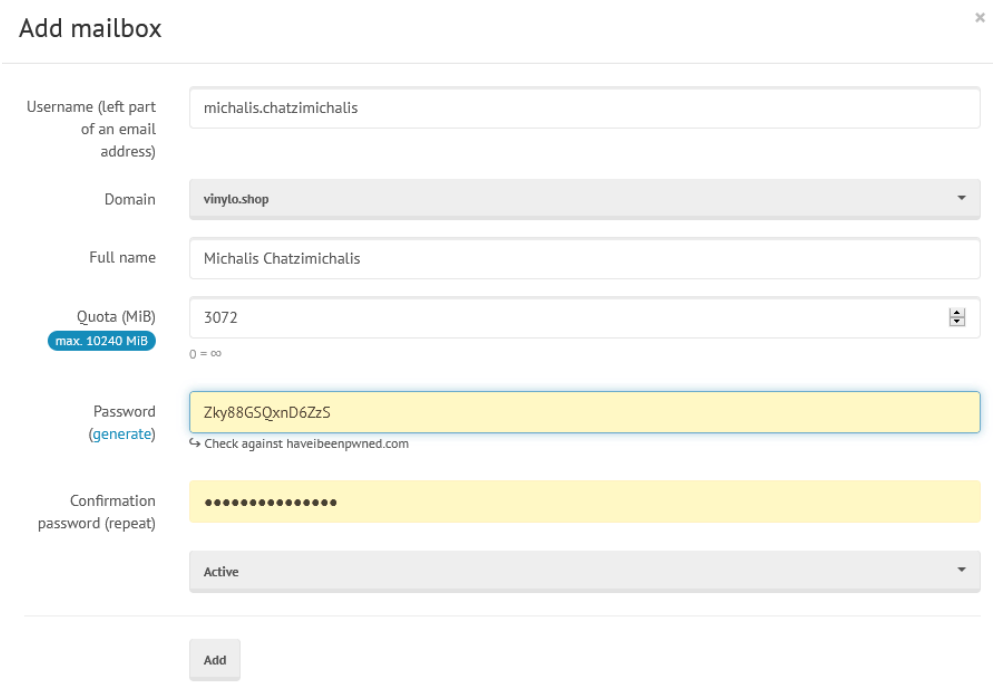
Über **Configuration > Mail-Setup** kann ich jetzt mit Add Domain meine Domäne einrichten. Mit **Add Domain and restart SoGo** starte ich die nötige Dienste neu.



Die neu-hinzugefügte Domäne

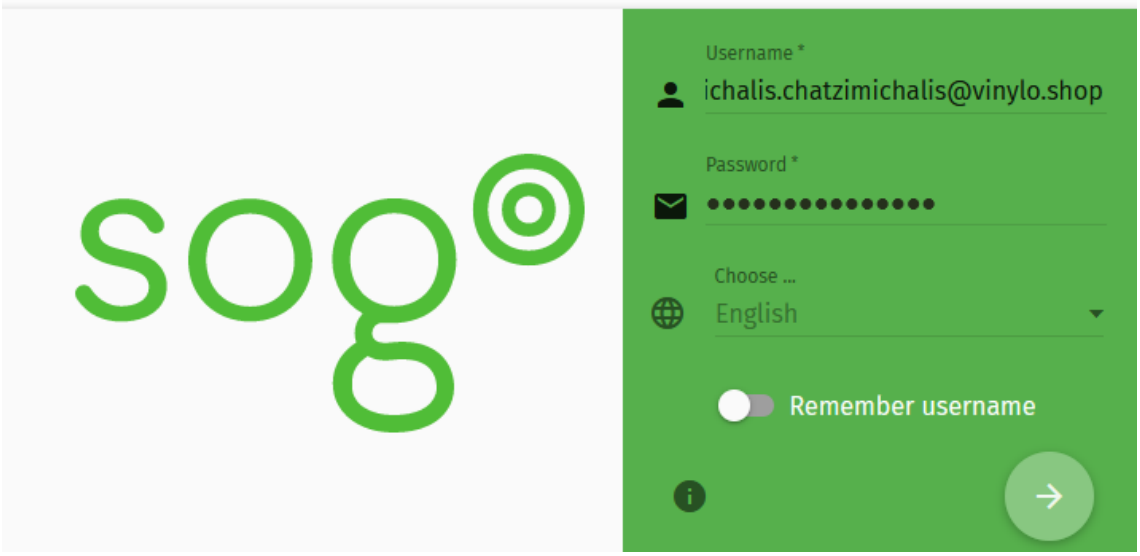


Mit Mailbox > Add Mailbox kann ich nun einen Mailbox generieren

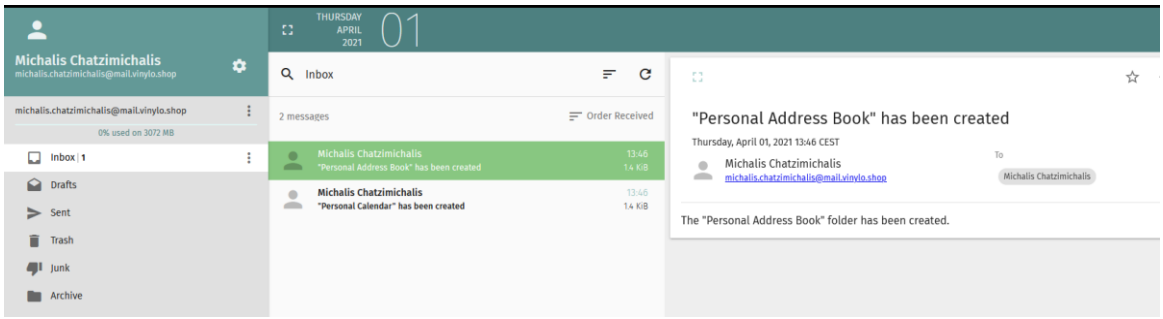


Webmail

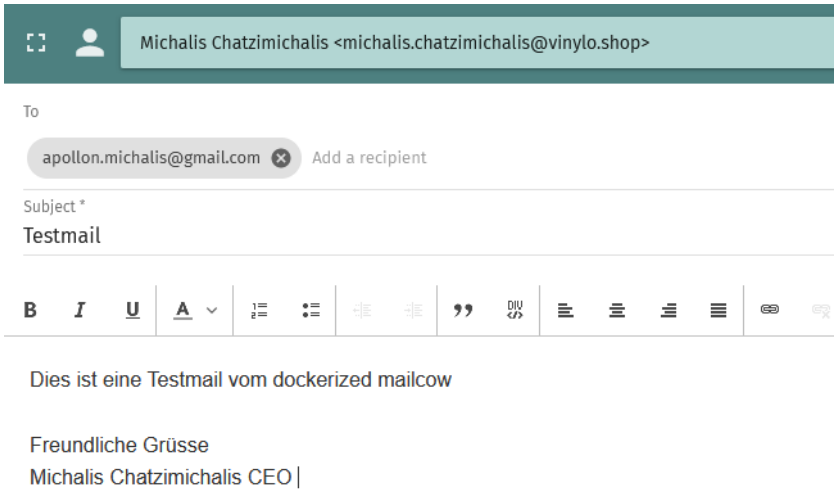
Auf Apps > Webmail wird ein neues Fenster geöffnet, in welcher ich den vordefinierter User anmelden kann.



Das Webmail-GUI sieht wie folgt aus

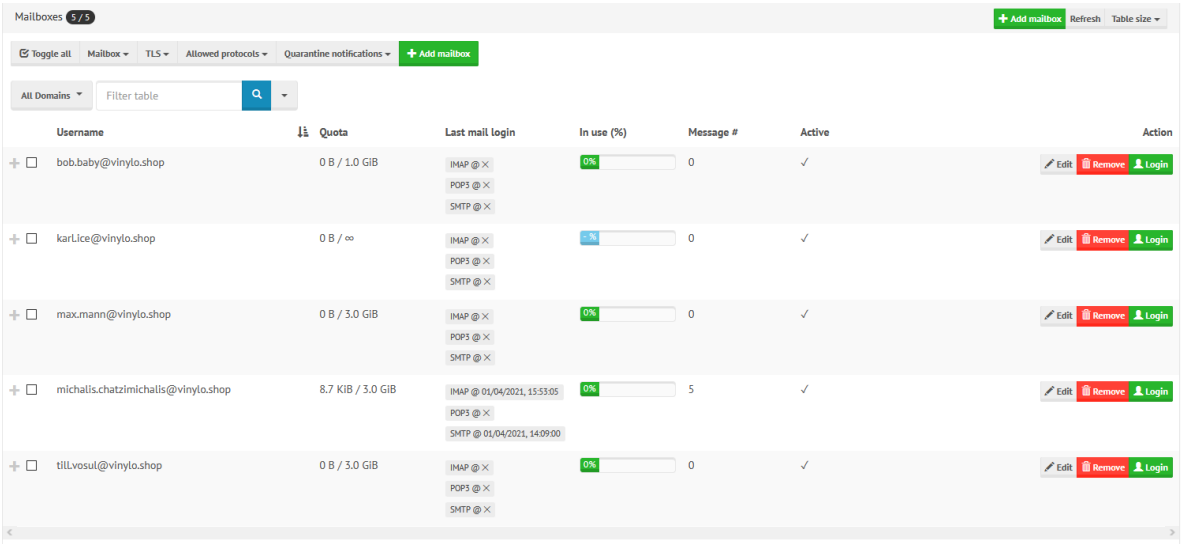


Auf dem Bleistift kann ich nun eine Mail schicken



Ich habe das Mail **nie bekommen**, jedoch auch keine Fehlermeldung im Inbox meines SoGo's Users bekommen.

Nun erstelle ich für allen Usern eine Mailbox sowie einen dementsprechenden

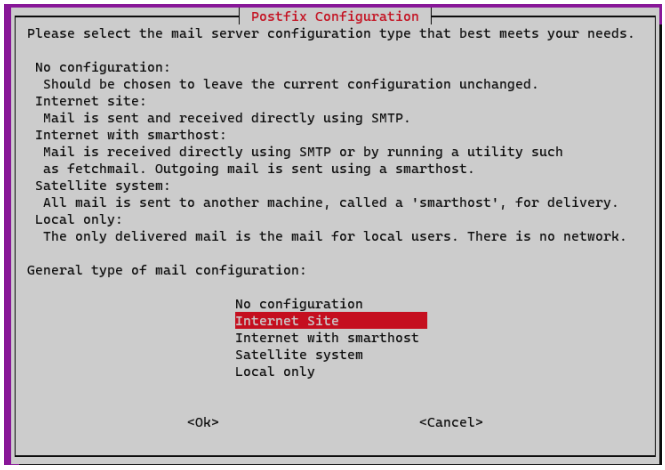


6.2.14 Postfix

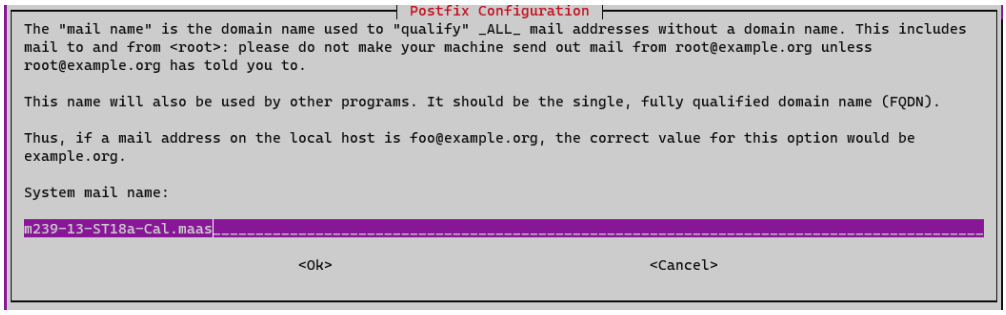
Da ich auf Port 80 nichts ändern will, werde ich die Postfix Aufgabe erledigen.

Um auf die TBZ Cloud zu kommen, aktiviere ich das WireGuard-Tunnel, welches wir im Unterricht eingerichtet haben und steige nun via ssh auf meine Server-Umgebung ein. Dafür öffne ich CMD und gebe ssh ubuntu@10.43.1.13 PW: ubuntu

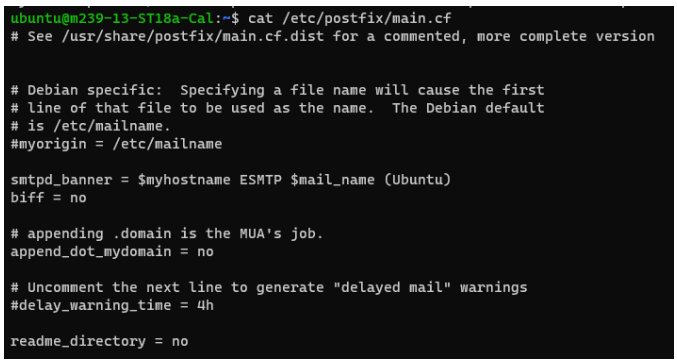
Mit einem **sudo apt-get update && sudo apt-get upgrade -y** aktualisiere ich das Betriebssystem. Demnach installieren ich Postfix mit **sudo apt install postfix**. Im Konfigurationswizard wähle ich Internet-Site



Unter System-mail name lasse ich den Standardwert und gehe weiter



Als Nächstes überprüfe ich, ob die Postfix-Konfigurationsdatei erfolgreich vorhanden ist



Zusätzlich überprüfe ich, ob das SASL (Simple Authentication and Security Layer) Verzeichnis vorhanden ist

```
ubuntu@m239-13-ST18a-Cal:/etc/postfix$ ls -l
total 128
-rw-r--r-- 1 root root    60 Mar 27 11:33 dynamicmaps.cf
drwxr-xr-x 2 root root  4096 Jul 10  2020 dynamicmaps.cf.d
-rw-r--r-- 1 root root  1493 Mar 27 11:33 main.cf
-rw-r--r-- 1 root root 27125 Mar 27 11:33 main.cf.proto
-rw-r--r-- 1 root root  4480 Jul 10  2020 makedefs.out
-rw-r--r-- 1 root root  6143 Mar 27 11:33 master.cf
-rw-r--r-- 1 root root  6143 Mar 27 11:33 master.cf.proto
-rwxr-xr-x 1 root root 29522 Jul 10  2020 post-install
-rw-r--r-- 1 root root 10170 Jul 10  2020 postfix-files
drwxr-xr-x 2 root root  4096 Jul 10  2020 postfix-files.d
-rwxr-xr-x 1 root root 10123 Jul 10  2020 postfix-script
drwxr-xr-x 2 root root  4096 Jul 10  2020 sasl
ubuntu@m239-13-ST18a-Cal:/etc/postfix$ ls sasl/
ubuntu@m239-13-ST18a-Cal:/etc/postfix$
```

Zunächst erstelle ich eine Password-Datei, wo meine Authentifizierungsdaten für mein Google-Konto und Gmail-Client sich befinden

```
ubuntu@m239-13-ST18a-Cal:/etc/postfix/sasl$ sudo nano sasl_passwd
ubuntu@m239-13-ST18a-Cal:/etc/postfix/sasl$ ls -l
total 4
-rw-r--r-- 1 root root 5 Mar 27 11:38 sasl_passwd
```

Befülle Datei

```
ubuntu@m239-13-ST18a-Cal:/etc/postfix/sasl$ sudo cat sasl_passwd
[smtp.gmail.com]:587 apollon.michalis@gmail.com:izihedmbtijpkxiz
```

Mit sudo postmap /etc/postfix/sasl/sasl_passwd erstelle ich eine dazugehörige Datenbank für diesen Eintrag

```
ubuntu@m239-13-ST18a-Cal:/etc/postfix/sasl$ ls -l
total 12
-rw-r--r-- 1 root root    65 Mar 27 11:39 sasl_passwd
-rw-r--r-- 1 root root 12288 Mar 27 11:40 sasl_passwd.db
ubuntu@m239-13-ST18a-Cal:/etc/postfix/sasl$ cat sasl_passwd.db
◆◆□◆◆□□◆O◆Y5◆6□□□ɜh^□□□apollon.michalis@gmail.com:izihedmbtijpkxiz□[smtp.gmail.com]:587
```

Mit sudo chmod 600 * sind alle Dateien in diesem Ordner nur für den Root-User zugänglich

```
ubuntu@m239-13-ST18a-Cal:/etc/postfix/sasl$ sudo chmod 600 *
ubuntu@m239-13-ST18a-Cal:/etc/postfix/sasl$ ls -l
total 12
-rw------- 1 root root    65 Mar 27 11:39 sasl_passwd
-rw------- 1 root root 12288 Mar 27 11:40 sasl_passwd.db
```

Mit sudo nano /etc/postfix/main.cf und Ctrl + W suche ich nach relayhost und ergänze die Zeile mit [smtp.gmail.com]:587

```
relayhost = [smtp.gmail.com]:587
```

Beim Punkt SASL Authentication soll folgendes eingefügt werden

```
# Enable SASL authentication
smtp_sasl_auth_enable = yes
smtp_sasl_security_options = noanonymous
smtp_sasl_password_maps = hash:/etc/postfix/sasl/sasl_passwd
smtp_tls_security_level = encrypt
```

smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt

Die Datei sieht am Schluss so aus und kann abgespeichert werden

```
relayhost = [smtp.gmail.com]:587
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
inet_protocols = all

# Enable SASL authentication
smtp_sasl_auth_enable = yes
smtp_sasl_security_options = noanonymous
smtp_sasl_password_maps = hash:/etc/postfix/sasl/sasl_passwd
smtp_tls_security_level = encrypt
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt
```

Mit ls /etc/ssl/certs/ca-certificates.crt überprüfe ich, ob das Zertifikat vorhanden ist. Wenn dies der Fall ist, hat man die SSL Zertifizierung erfolgreich eingerichtet

```
ubuntu@m239-13-ST18a-Cal:/etc/postfix/sasl$ ls /etc/ssl/certs/ca-certificates.crt
/etc/ssl/certs/ca-certificates.crt
```

Mit sudo systemctl restart postfix startet man den Systemdienst neu. Zugleich wird das Configfile noch eingelesen

```
ubuntu@m239-13-ST18a-Cal:/etc/postfix/sasl$ sudo systemctl restart postfix
ubuntu@m239-13-ST18a-Cal:/etc/postfix/sasl$ sudo systemctl status postfix
● postfix.service - Postfix Mail Transport Agent
   Loaded: loaded (/lib/systemd/system/postfix.service; enabled; vendor preset: enabled)
   Active: active (exited) since Sat 2021-03-27 11:54:56 UTC; 1s ago
  Process: 14273 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
 Main PID: 14273 (code=exited, status=0/SUCCESS)

Mar 27 11:54:56 m239-13-ST18a-Cal systemd[1]: Starting Postfix Mail Transport Agent...
Mar 27 11:54:56 m239-13-ST18a-Cal systemd[1]: Started Postfix Mail Transport Agent.
```

Um jetzt einen Mail zu senden, gebe ich nun sendmail [e-mail-Adresse] ein. Danach werde ich aufgefordert ein Betreff einzugeben, gefolgt von der Inhalt des Mailsndma

Testmail an meine zweite E-Mail-Adresse

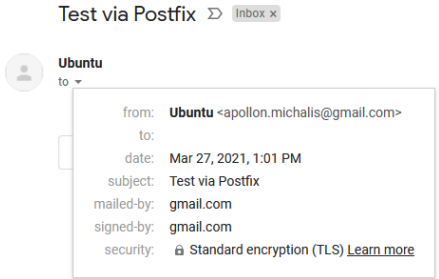
sendmail apolllon.michalis@gmail.com

Subject: Test via Postfix

Mit Enter und Ctrl + d schliesst man das Fenster und schickt das Mail

```
ubuntu@m239-13-ST18a-Cal:/etc/postfix/sasl$ sendmail apolllon.michalis@gmail.com
Subject: Test via Postfix
```

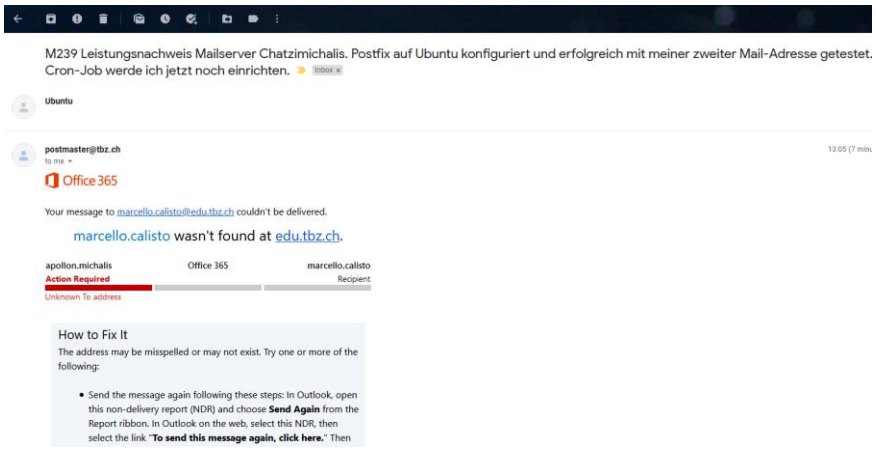
Die Bestätigung



Nun werde ich an Hr. Calisto die Bestätigung schicken, dass es bei mir funktioniert hat.

```
ubuntu@m239-13-ST18a-Cal:/etc/postfix/sasl$ sendmail marcello.calisto@edu.tbz.ch
Subject: M239 Leistungsnachweis Mailserver Chatzimichalis. Postfix auf Ubuntu konfiguriert und erfolgreich mit meiner zw
eiter Mail-Adresse getestet. Cron-Job werde ich jetzt noch einrichten.
```

Auf dieser Mail habe ich folgender Fehlermeldung gekriegt



Nun ändere ich die Mail-Adresse auf marcello.calisto@tbz.ch

```
ubuntu@m239-13-ST18a-Cal:/etc/postfix/sasl$ sendmail marcello.calisto@tbz.ch
Subject: M239 Leistungsnachweis Mailserver Chatzimichalis. Postfix auf Ubuntu konfiguriert und erfolgreich mit meiner zw
eiter Mail-Adresse getestet. Cron-Job werde ich jetzt noch einrichten.
```

Sodass Systemprozesse Mails an eine definierte Adresse schicken können (als Alarmierung) muss ich **crontab -e** eingeben, um die Auswahl der Editor zu bekommen. Da wähle ich nano, also gebe ich 1 ein

```
ubuntu@m239-13-ST18a-Cal:/etc/postfix/sasl$ crontab -e
no crontab for ubuntu - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/nano        <---- easiest
  2. /usr/bin/vim.basic
  3. /usr/bin/vim.tiny
  4. /bin/ed

Choose 1-4 [1]: 1
```

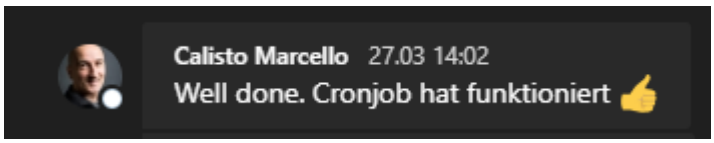
Zuerst werde ich ein Testmail an meiner zweite Mail-Adresse

```
# m h  dom mon dow   command
MAILTO="apolllon.michalis@gmail.com"
2 * * * * echo "Regelmässige Benachrichtung M.Chatzimichalis"
```

Für Hr. Calisto sieht der Input wie folgt aus

```
MAILTO="marcello.calisto@tbz.ch"
2 * * * * echo "Regelmässige Benachrichtung M.Chatzimichalis"
```

Der Cron-Job war erfolgreich



6.2.15 FTP
Installation und Benutzersetup

Zusätzlich stelle ich einen FTP-Server mit vsftpd auf. Als Erstes installiere ich den Dienst mit **sudo apt install vsftpd**.

Nun kopiere ich das Konfigurationsfile, sodass ich eine nigelnagelneue Datei habe. **sudo cp /etc/vsftpd.conf /etc/vsftpd.conf.bak**

```
michalis@michalis-ubuntu:~$ sudo cp /etc/vsftpd.conf /etc/vsftpd.conf.bak
```

Da ich für die 4/5 Mitarbeiter je User erstelle ich eine Gruppe. Mit **sudo adduser bbaby** und setzte zugleich ein sicheres Passwort.

```
michalis@michalis-ubuntu:~$ sudo adduser bbaby
Adding user `bbaby' ...
Adding new group `bbaby' (1001) ...
Adding new user `bbaby' (1001) with group `bbaby' ...
Creating home directory `/home/bbaby' ...
Copying files from `/etc/skel' ...
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: password updated successfully
Changing the user information for bbaby
Enter the new value, or press ENTER for the default
        Full Name []: Bob Baby
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
```

Um die Sicherheit zu gewährleisten, erstelle ich im Home-Ordner des neu erstellten Users einen Ordner namens ftp.

sudo mkdir /home/bbaby/ftp. Mit **sudo chown nobody:nogroup /home/bbaby/ftp**.

Mit **sudo ls -la /home/bbaby/ftp** kann ich die korrekte Berechtigungsvergabe überprüfen.

```
michalis@michalis-ubuntu:~$ sudo ls -la /home/bbaby/ftp
total 8
drwxr-xr-x 2 nobody nogroup 4096 Mär 29 12:51 .
drwxr-xr-x 3 bbaby  bbaby   4096 Mär 29 12:51 ..
```

Nun erstelle ich den Ordner für das Hochladen von Dateien und gebe die entsprechende Berechtigungen.

sudo mkdir /home/bbaby/ftp/dateien – Ordner erstellen

sudo chown bbaby:bbaby /home/bbaby/ftp/dateien – Berechtigungen erteilen

Beispielsdatei **echo "Dies ist ein vsftpd Testfile" >> sudo tee /home/bbaby/ftp/dateien/test.txt** erstellen

```
michalis@michalis-ubuntu:~$ echo "Dies ist ein vsftpd Testfile" | sudo tee /home/bbaby/ftp/dateien/test.txt
Dies ist ein vsftpd Testfile
michalis@michalis-ubuntu:~$ sudo ls -la /home/bbaby/ftp/dateien
total 12
drwxr-xr-x 2 bbaby  bbaby   4096 Mär 29 12:53 .
drwxr-xr-x 3 nobody nogroup 4096 Mär 29 12:51 ..
-rw-r--r-- 1 root   root      29 Mär 29 12:53 test.txt
```

FTP-Zugang konfigurieren

Die Konfigurationsdatei muss wie folgt überarbeitet werden. **sudo nano /etc/vsftpd.conf** eingeben.

write_enable=YES – unkommentieren

chroot_local_user=YES – unkommentieren

user_sub_token=$USER

local_root=/home/$USER/ftp

```
#
# Uncomment this to indicate that vsftpd use a utf8 filesystem.
#utf8_filesystem=YES

#User subtoken
user_sub_token=$USER
local_root=/home/$USER/ftp
```

pasv_min_port=40000

pasv_max_port=50000

```
#Port eingrenzen
pasv_min_port=40000
pasv_max_port=50000
```

userlist_enable=YES

userlist_file=/etc/vsftpd.userlist

userlist_deny=NO

```
#Userlist einlesen
userlist_enable=YES
userlist_file=/etc/vsftpd.userlist
userlist_deny=NO
```

File abspeichern und Editor schliessen

Nun füge ich den vorhin erstellter User zu der Liste der User mit **echo "bbaby" | sudo tee -a /etc/vsftpd.userlist.** Mit **cat /etc/vsftpd.userlist** kann ich die Anpassung überprüfen.

```
michalis@michalis-ubuntu:~$ echo "bbaby" | sudo tee -a /etc/vsftpd.userlist
bbaby
michalis@michalis-ubuntu:~$ cat /etc/vsftpd.userlist
bbaby
```

Mit **sudo systemctl restart vsftpd** starte ich den FTP-Dienst neu.

```
michalis@michalis-ubuntu:~$ sudo systemctl restart vsftpd
michalis@michalis-ubuntu:~$ sudo systemctl status vsftpd
● vsftpd.service - vsftpd FTP server
     Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2021-03-29 12:57:47 CEST; 4s ago
```

FTP Zugang testen

Um den FTP Zugang zuerst mit einem anonymen-User zu testen, gebe ich die Public IP ein mit dem **ftp -p 178.192.230.31** Befehl im CMD ein.

Testfälle

Mit Anonym;

```
michalis@michalis-ubuntu:~$ ftp -p 192.168.1.117
Connected to 192.168.1.117.
220 (vsFTPd 3.0.3)
Name (192.168.1.117:michalis): anonymous
530 Permission denied.
Login failed.
ftp>
```

Mit sudo_user;

```
michalis@michalis-ubuntu:~$ ftp -p 192.168.1.117
Connected to 192.168.1.117.
220 (vsFTPd 3.0.3)
Name (192.168.1.117:michalis): sudo_user
530 Permission denied.
Login failed.
ftp> bye
221 Goodbye.
```

Jetzt melde ich mit dem User **bbaby** und sein **PW** ein.

```
michalis@michalis-ubuntu:~$ ftp -p 192.168.1.117
Connected to 192.168.1.117.
220 (vsFTPd 3.0.3)
Name (192.168.1.117:michalis): bbaby
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Darin wechsle ich zum Ordner «Dateien» und Frage mit dem Get-Befehl nach der Testdatei, welche ich vorher erstellt habe.

```
ftp> cd dateien
250 Directory successfully changed.
ftp> pwd
257 "/dateien" is the current directory
ftp> get test.txt
local: test.txt remote: test.txt
227 Entering Passive Mode (192,168,1,117,167,52).
150 Opening BINARY mode data connection for test.txt (29 bytes).
226 Transfer complete.
29 bytes received in 0.00 secs (52.3481 kB/s)
```

Mit **put [Datei] [Zieldatei]** kann ich dieselbe Datei mit einem anderen Namen hochladen. Mit dem teste ich zugleich die Schreiberechte.

```
ftp> put test.txt upload.txt
local: test.txt remote: upload.txt
227 Entering Passive Mode (192,168,1,117,188,129).
150 Ok to send data.
226 Transfer complete.
29 bytes sent in 0.00 secs (151.4455 kB/s)
```

Mit **bye** schliesse ich die Session.

TLS aktivieren

Da FTP sehr unsicher ist, werde ich den Verkehr via TLS/SSL verschlüsseln. Das Zertifikat werde ich mit OpenSSL erstellen. Der Befehl lautet wie folgt; **sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/vsftpd.pem -out /etc/ssl/private/vsftpd.pem**

Folgende Infos muss ich jetzt nachtragen;

Country Code: **CH**

State: **Zurich**

Locality Name: **Zurich**

Organisation: **Vinylo**

Organisation Unit:

Common Name: **192.168.1.117**

Email: []

```
Country Name (2 letter code) [AU]:CH
State or Province Name (full name) [Some-State]:Zurich
Locality Name (eg, city) []:Zurich
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Vinylo
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:192.168.1.117
Email Address []:
```

Nachdem dieser Vorgang erfolgreich durchgegangen ist, öffne ich die Konfigurationsdatei mit **sudo nano /etc/vsftpd.conf** erneut

Folgende Änderung sind zu machen;

 # rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem - Kommentieren

 # rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key – Kommentieren

rsa_cert_file=/etc/ssl/private/vsftpd.pem – hinzufügen

rsa_private_key_file=/etc/ssl/private/vsftpd.pem – hinzufügen

ssl_enable=YES – HTTPS Datenverkehr zwingen

allow_anon_ssl=NO – Anonyme Anmeldungen unterbinden

force_local_data_ssl=YES

force_local_logins_ssl=YES

ssl_tlsv1=YES – TLS einschalten

ssl_sslv2=NO – SSL deaktivieren

ssl_sslv3=NO – SSL deaktiveren

require_ssl_reuse=NO

ssl_ciphers=HIGH – Cipher-Suites mit >128bit-Länge

Die Datei soll dann wie folgt aussehen

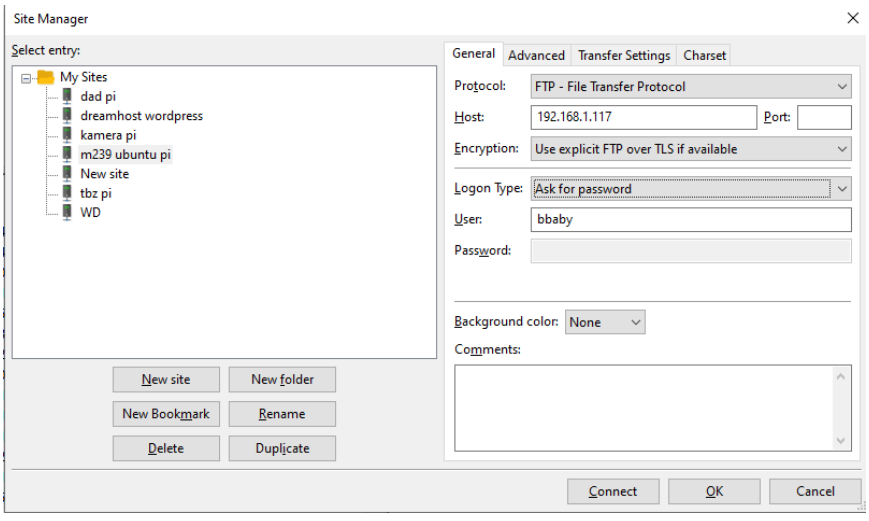
```
# This option specifies the location of the RSA certificate to use for SSL
# encrypted connections.
rsa_cert_file=/etc/ssl/private/vsftpd.pem
rsa_private_key_file=/etc/ssl/private/vsftpd.pem
ssl_enable=YES

#Disallow anon
allow_anon_ssl=NO
force_local_data_ssl=YES
force_local_logins_ssl=YES

#TLS einschalten
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO

#Cipher hinzufügen
require_ssl_reuse=NO
ssl_ciphers=HIGH
```

Mit **sudo systemctl restart vsftpd** starte ich den FTP-Dienst neu. Nun kann ich testen, ob die Shellverbindung korrekterweise unterbinden wird. Dazu gebe ich **ftp -p 178.192.230.31** ein.

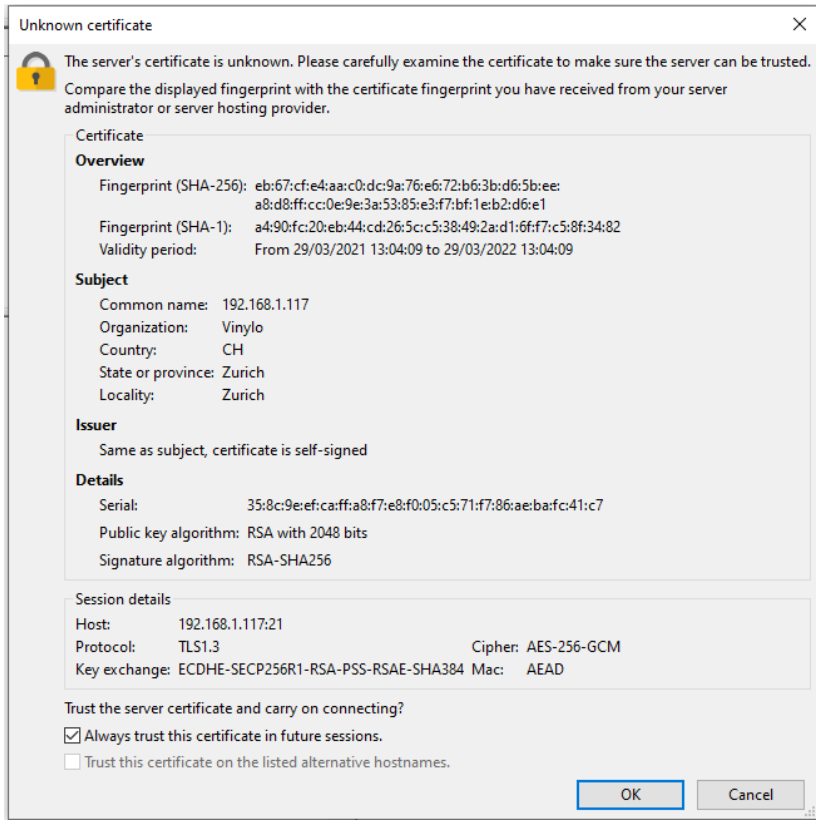
```
michalis@michalis-ubuntu:~$ ftp -p 192.168.1.117
Connected to 192.168.1.117.
220 (vsFTPd 3.0.3)
Name (192.168.1.117:michalis): bbaby
530 Non-anonymous sessions must use encryption.
Login failed.
421 Service not available, remote server has closed connection
ftp>
```

FileZilla TLS testen

Jetzt öffne ich FileZilla auf mein lokalen PC und füge eine neue Site hinzu.



Nachdem ich auf Connect klicke, werde ich aufgefordert das PW für den User einzugeben. Nach erfolgreicher Eingabe werde ich mit dem vorhin erstellten Zertifikat konfrontiert, welches ich akzeptieren muss.



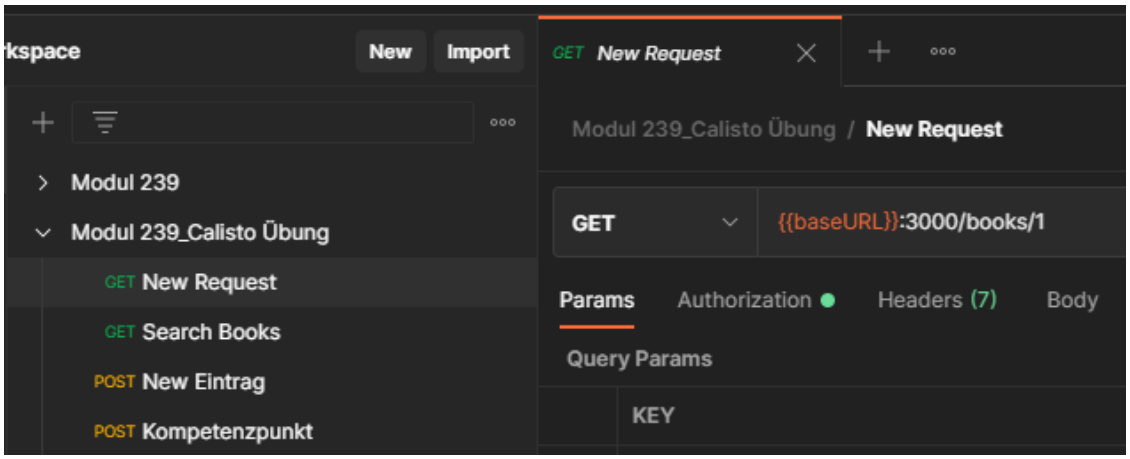
Erfolgreiche Transfer der test.txt

6.2.16 Postman

Postman ist eine simple Applikation, um mit verschiedenen APIs zu kommunizieren und eigene HTTP-Requests zu erstellen und auch zu verwalten.

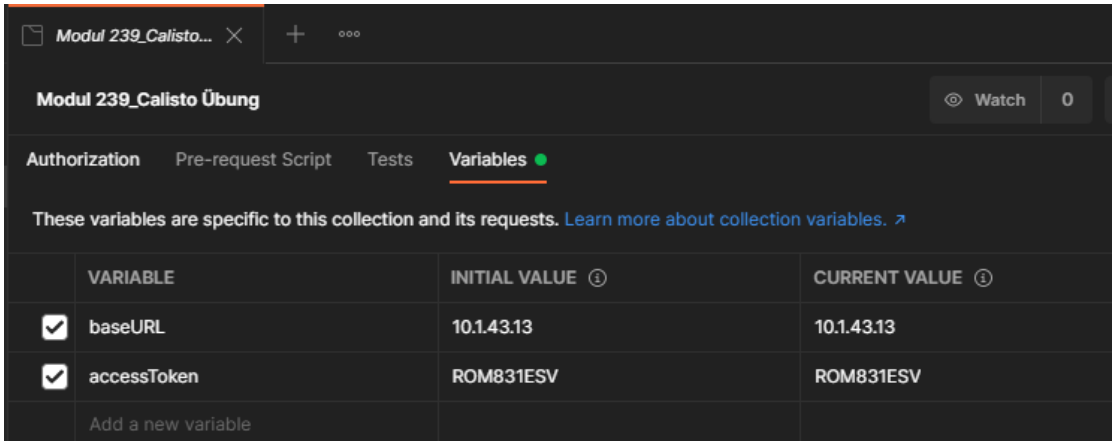
Als Erstes habe ich auf die Postman-Webseite einen Account erstellt, um meine Konfigurationen abzuspeichern. Nachdem Erstellen meines Accounts wurde ich zu meinem Workspace geführt. Gleichzeitig werde ich gefordert, den Desktop-Programm herunterzuladen, mit den Versprechen es hebe einige Limitierungen, welche Webbrowser bewusst (für XSS-Prävention) eingerichtet haben, eines davon sei das Cross-Origin-Policy oder auch Same-Origin-Policy. Das Desktop-Programm lade ich herunter und führe den Installationswizard sauber durch.

Collection erstellen



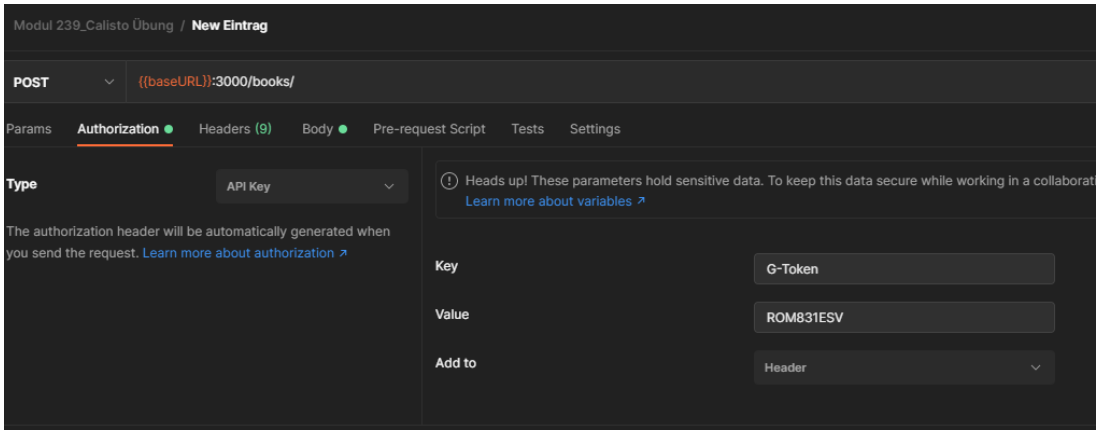
Scoped Variabeln erstellen

Auf Collection > Menüpunkte > Edit > Variabels > Neue Variable als Key-Value Pair erstellen



API token

In den nötigen Requests wird unter Authorization das definierte accessToken angegeben



6.3 Troubleshooting (Docker)

Ich wurde mit Docker schnell überfordert und hatte mit mehreren Container Problem

6.3.1 Installation WordPress, NGINX (Fehlerhaft DigitalOcean)

Der Docker Container mit allen zur Verfügung gestellten Services sieht wie folgt aus.

Der nginx Container setze ich wie folgt auf. Gebraucht wird folgendes

- Dockerfile
- Docker-Compose.yml mit Wordpress
- Ordnerstruktur (Volume Freigabe)

Als Erstes wechsle ich zur Ordnerstruktur, die ich im Kap. [Link] verwiesen habe. Da drin erstelle ich nun den Ordner wordpress und wechsle in dem. Zusätzlich erstelle ich für die Konfigurationsdateien für NGINX noch einen Ordner. Darin werde ich eine Konfigurationsdatei für die Zieldomäne einrichten

1. sudo mkdir wordpress
2. cd wordpress
3. sudo mkdir nginx-conf
4. cd nginx-conf
5. nano vinylo.conf

Das Konfigurationsfile wird mit dem folgenden Inhalt befüllt. Unterhalb der Abbildung wird das Syntax erläutert.

```
server {
    listen 80;
    listen [::]:80;

    server_name vinylo.shop www.vinylo.shop;

    index index.php index.html index.htm;

    root /var/www/html;

    location ~ /.well-known/acme-challenge {
            allow all;
            root /var/www/html;
    }

    location / {
            try_files $uri $uri/ /index.php$is_args$args;
    }

    location ~ \.php$ {
            try_files $uri =404;
            fastcgi_split_path_info ^(.+\.php)(/.+)$;
            fastcgi_pass wordpress:9000;
            fastcgi_index index.php;
            include fastcgi_params;
            fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
            fastcgi_param PATH_INFO $fastcgi_path_info;
    }

    location ~ /\.ht {
            deny all;
    }

    location = /favicon.ico {
            log_not_found off; access_log off;
    }
    location = /robots.txt {
            log_not_found off; access_log off; allow all;
    }
    location ~* \.(css|gif|ico|jpeg|jpg|js|png)$ {
            expires max;
            log_not_found off;
    }
}
```

Abbildung 14: NGINX vinylo.conf File

Die ersten 4 Befehle heissen **Directives** und die sind Grundkonfigurationen für NGINX.

- **Listen** definiert der zuhörende Port
- **server_name** beinhaltet der Domain-Name
- **index** legt die Indizierung der Dateien
- **root** spezifiert den Pfad für alle Inhalte (insbesondere der vom Wordpress Dockerfile)

Die nächsten paar **Blöcke** sind mit Location bezeichnet und deuten auf folgendes hin

- **Location ~ /.well-known/acme-challenge** verarbeitet Anfragen an das Verzeichnis .well-known, wo Certbot eine temporäre Datei ablegt, um zu überprüfen, ob der DNS für unsere Domain zu unserem Server auflöst.
- **Location / verwende** ich eine try_files-Direktive, um nach Dateien zu suchen, die mit einzelnen URI-Anfragen übereinstimmen. Anstatt jedoch standardmäßig einen 404 Not Found-Status

zurückzugeben, übergeben wir die Kontrolle an die index.php-Datei von WordPress mit den Anforderungsargumenten

- **Location ~ \.php$** Dieser Speicherortblock wird die PHP-Verarbeitung übernehmen und diese Anfragen an unseren WordPress-Container weiterleiten. Da unser WordPress-Docker-Image auf dem php:fpm-Image basieren wird, werden wir auch Konfigurationsoptionen, die spezifisch für das FastCGI-Protokoll sind, in diesen Block aufnehmen**.**
- **location ~ /\.ht** behandelt .htaccess-Dateien, da Nginx sie nicht ausliefert. Die Direktive deny_all stellt sicher, dass .htaccess-Dateien niemals an Benutzer ausgeliefert werden.
- **location = /favicon.ico, location = /robots.txt**: Diese Blöcke stellen sicher, dass Anfragen an /favicon.ico und /robots.txt nicht protokolliert werden.
- **location ~* \.(css|gif|ico|jpeg|jpg|js|png)$**: Dieser Block schaltet die Protokollierung für statische Asset-Anfragen aus und stellt sicher, dass diese Assets in hohem Maße cachefähig sind, da sie in der Regel teuer zu bedienen sind

Wieso ich das Location FastCGI Implementation lese, hier. Mehr zu den Location Server Blocks da Link.

Die Datei wird abgespeichert und vom Editor herausgegangen.

Globale Variablen definieren

Ihre Datenbank- und WordPress-Anwendungscontainer benötigen zur Laufzeit Zugriff auf bestimmte Umgebungsvariablen, damit Ihre Anwendungsdaten bestehen bleiben und für Ihre Anwendung zugänglich sind. Diese Variablen enthalten sowohl sensible als auch nicht-sensible Informationen: sensible Werte für Ihr MySQL-Root-Passwort und den Benutzer und das Passwort der Anwendungsdatenbank sowie nicht-sensible Informationen für den Namen und den Host Ihrer Anwendungsdatenbank.

Anstatt alle diese Werte in unserer Docker Compose-Datei zu setzen - der Hauptdatei, die Informationen darüber enthält, wie unsere Container ausgeführt werden - können wir die sensiblen Werte in einer .env-Datei setzen und ihre Verbreitung einschränken. Dadurch wird verhindert, dass diese Werte in unsere Projekt-Repositories kopiert und öffentlich zugänglich gemacht werden.

Dafür gehen wir im Working Directory im Ordner **wordpress** und erstellen die besagte env Datei mit **nano .env**. Darin werde ich das Root-Password fürs mySQL sowie der Benutzernamen und sein Passwort definieren.

- root - $LsgbGsYNwEX
- user – michalis
- password – Admin.1234-L

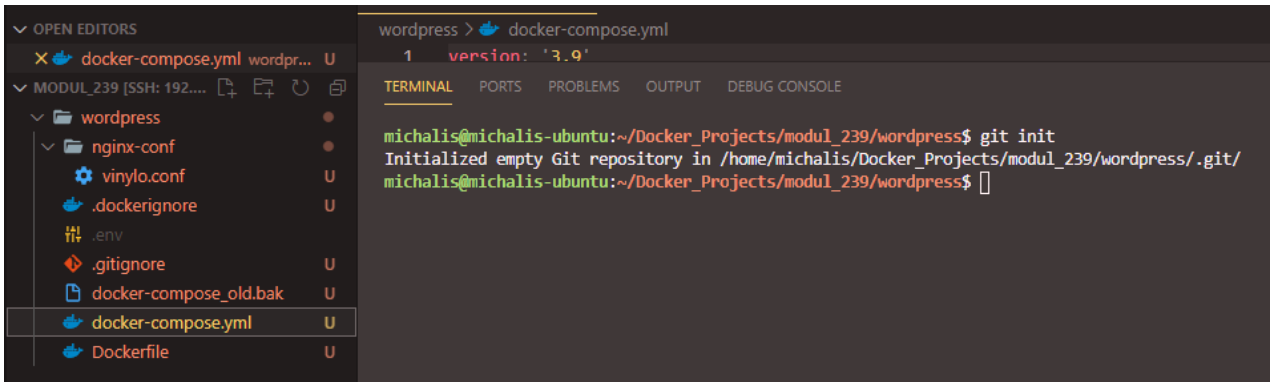
Da die .env-Datei sensible Informationen enthält, sollten Sie sicherstellen, dass sie in den Dateien .gitignore und .dockerignore Ihres Projekts enthalten ist, die Git und

Docker mitteilen, welche Dateien nicht in Ihre Git-Repositorys bzw. Docker-Images kopiert werden sollen.

Git kombinieren

Um die Ordnerdateien und den Code zu publizieren, werde ich einen Git-Repo starten. Diese iniitiere ich mit **git init**, wenn ich mich im Hauptordner wordpress befinde.

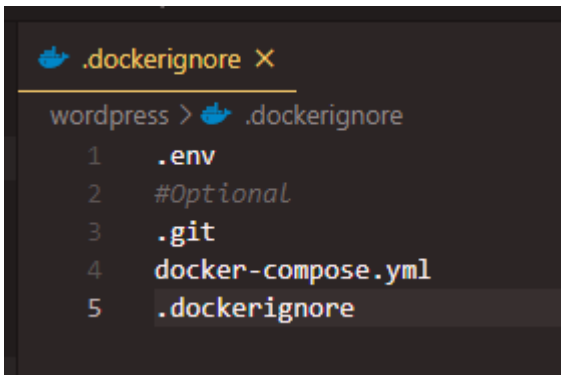
Nun sieht es im VS Code wie folgt aus



De folgenden Status-Codes sind wichtig

- U - Un

Docker ebenfalls



Um diese initialisierte Git-Repo erstmals auf GitHub zu veröffentlichen, müsse ich nun eine Git Aktion auslösen, also eine einfache Push-Aktion. Danach werde ich aufgefordert mich mit meinem GitHub-Konto zu authentifizieren, was ich schnell erledige.

6.3.2 Container Hinzufügen mit Docker-Compose

Die Docker-Compose Datei wird Allererstens mit dem Datenbank-Service mariaDB befüllt. MySQL hat wie gesagt kein arm64-Image auf Docker. Zusätzlich wird nun die natürliche Versionszahl genommen und nicht die Version :latest, da das mit Inkompatibilitäten mit den Abhängigkeiten zusammenhängt. Mehr findet man unter der [Best Practices bei Docker-Compose Dateien]

```yaml
version: '3.9'

services:
 db:
    image: mysql:8.0
    container_name: mysql_db
    restart: unless-stopped
    env_file: .env
    environment:
      - MYSQL_DATABASE=wordpress
    volumes:
      - dbdata:/var/lib/mysql
    command: '--default-authentication-plugin=mysql_native_password'
    networks:
      - app-network
```

- Version
- Services: db
- image
- container_name
- restart
- env_file ist für den vorhin definierten Variabeln
- environment
- volumes
- command – Mehr dazu hier
- networks

Zusätzlich definiere ich das Wordpress Service wie folgt

```yaml
 wordpress:
    depends_on:
      - db
    image: wordpress:5.1.1-fpm-alpine
    container_name: wordpress
    restart: unless-stopped
    env_file: .env
    environment:
      - WORDPRESS_DB_HOST=db:3306
      - WORDPRESS_DB_USER=$MYSQL_USER
      - WORDPRESS_DB_PASSWORD=$MYSQL_PASSWORD
      - WORDPRESS_DB_NAME=wordpress
    volumes:
      - wordpress:/var/www/html
    networks:
      - app-network
```

- depends_on
- environment
- volumes

Der Webserver wird als Nächstes definiert

```yaml
webserver:
  depends_on:
    - wordpress
  image: nginx:1.15.12-alpine
  container_name: webserver
  restart: unless-stopped
  ports:
    - "80:80"
  volumes:
    - wordpress:/var/www/html
    - ./nginx-conf:/etc/nginx/conf.d
    - certbot-etc:/etc/letsencrypt
  networks:
    - app-network
```

- depends_on
- image
- ports – ein Array
- volumes
- network wird auch im gemeinsamen Netzwerk gefügt

Als letzte Service ist das automatisierte SSL/TLS-Zertifizierungsstelle für NGINX, das sogenannte Certbot.

```yaml
certbot:
  depends_on:
    - webserver
  image: certbot/certbot
  container_name: certbot
  volumes:
    - certbot-etc:/etc/letsencrypt
    - wordpress:/var/www/html
  command: certonly --webroot --webroot-path=/var/www/html --
email apollon.michalis@gmail.com --agree-tos --no-eff-email --staging -
d vinylo.shop -d www.vinylo.shop
```

- depends_on
- image
- volumens
- command werden die Paramater eingegeben . Volles Kommando hier zu finden

- `--webroot`: This tells Certbot to use the webroot plugin to place files in the webroot folder for authentication. This plugin depends on the HTTP-01 validation method, which uses an HTTP request to prove that Certbot can access resources from a server that responds to a given domain name.
- `--webroot-path`: This specifies the path of the webroot directory.
- `--email`: Your preferred email for registration and recovery.
- `--agree-tos`: This specifies that you agree to ACME's Subscriber Agreement.
- `--no-eff-email`: This tells Certbot that you do not wish to share your email with the Electronic Frontier Foundation (EFF). Feel free to omit this if you would prefer.
- `--staging`: This tells Certbot that you would like to use Let's Encrypt's staging environment to obtain test certificates. Using this option allows you to test your configuration options and avoid possible domain request limits. For more information about these limits, please see Let's Encrypt's rate limits documentation.
- `-d`: This allows you to specify domain names you would like to apply to your request. In this case, we've included `example.com` and `www.example.com`. Be sure to replace these with your own domain.

Nachfolgend sind die Abhängigkeiten von den definierten Services aufgelistet, also die Volumes und das Netzwerk.

```
volumes:
    certbot-etc:
    wordpress:
    dbdata:

networks:
    app-network:
        driver: bridge
```

Unser Top-Level-Volumes-Schlüssel definiert die Volumes certbot-etc, wordpress und dbdata. Wenn Docker Volumes erstellt, wird der Inhalt des Volumes in einem Verzeichnis auf dem Host-Dateisystem, /var/lib/docker/volumes/, gespeichert, das von Docker verwaltet wird. Der Inhalt jedes Volumes wird dann von diesem Verzeichnis in jeden Container eingehängt, der das Volume verwendet. Auf diese Weise ist es möglich, Code und Daten zwischen Containern zu teilen.

Das benutzerdefinierte Bridge-Netzwerk app-network ermöglicht die Kommunikation zwischen unseren Containern, da sie sich auf demselben Docker-Daemon-Host befinden. Dies rationalisiert den Datenverkehr und die Kommunikation innerhalb der Anwendung, da alle Ports zwischen den Containern auf demselben Bridge-Netzwerk geöffnet werden, ohne dass irgendwelche Ports nach außen hin offengelegt werden. So können unsere DB-, Wordpress- und Webserver-Container miteinander kommunizieren, und wir müssen nur Port 80 für den Front-End-Zugriff auf die Anwendung freigeben.

Das endgültige docker-compose.yml File sieht wie folgt aus. Die aktualierten Versionen der jeweiligen Services sollten vorhin auf Docker Hub nachgeforscht werden und allenfalls auf den neusten Wert im YAML-File überschrieben werden.

Quelle 15: Docker Hub

```yaml
version: '3'

services:
  mariadb:
    image: mariadb:10.5
    container_name: mariadb
    restart: unless-stopped
    env_file: .env
    environment:
      - MYSQL_DATABASE=wordpress
    volumes:
      - dbdata:/var/lib/mysql
    command: '--default-authentication-plugin=mysql_native_password'
    networks:
      - app-network

  wordpress:
    depends_on:
      - mariadb
    image: wordpress:5.6.2-fpm-alpine
    container_name: wordpress
    restart: unless-stopped
    env_file: .env
    environment:
      - WORDPRESS_DB_HOST=db:3306
      - WORDPRESS_DB_USER=$MYSQL_USER
      - WORDPRESS_DB_PASSWORD=$MYSQL_PASSWORD
      - WORDPRESS_DB_NAME=wordpress
    volumes:
      - wordpress:/var/www/html
    networks:
      - app-network

  webserver:
    depends_on:
      - wordpress
    image: nginx:1.19.7-alpine
    container_name: webserver
    restart: unless-stopped
    ports:
      - "80:80"
    volumes:
```

```
      - wordpress:/var/www/html
      - ./nginx-conf:/etc/nginx/conf.d
      - certbot-etc:/etc/letsencrypt
    networks:
      - app-network

  certbot:
    depends_on:
      - webserver
    image: certbot/certbot:arm32v6-latest
    container_name: certbot
    volumes:
      - certbot-etc:/etc/letsencrypt
      - wordpress:/var/www/html
    command: certonly --webroot --webroot-path=/var/www/html --
email apollon.michalis@gmail.com --agree-tos --no-eff-email --staging -
d vinylo.shop -d www.vinylo.shop

volumes:
  certbot-etc:
  wordpress:
  dbdata:

networks:
  app-network:
    driver: bridge
```

Abbildung 15: Ganzes Docker-Compose File

Insgesamt hat dieser Docker-Compose 66 Zeilen Code und ist zugleich auf meinem GitHub-Konto zugänglich.

Mit dem Befehl docker-compose up -d werden allen Containers erstellt und in der jeweiligen, definierten Reihenfolge, im Hintergrund gestartet. Also zuerst die MySQL-Datenbank, danach Wordpress, NGINX und zuletzt Certbot. Dies braucht ungefähr 10 Minuten.

```
Status: Downloaded newer image for certbot/certbot:latest
Creating mariadb_db ... done
Creating wordpress  ... done
Creating webserver  ... done
Creating certbot    ... done
Attaching to mariadb_db, wordpress, webserver, certbot
```

Mit dem Befehl docker-compose images sehe ich jetzt alle meine erstellten Container mit dem Container-Namen, von welcher Repo das Service aufgebaut wurde, welcher Tag gegeben wurde sowie die einzigartig mit SHA-256 verschlüsselte ID und zuletzt die Grösse des jeweiligen Containers

```
michalis@michalis-ubuntu:~/Docker_Projects/modul_239/wordpress$ docker-compose images
Container     Repository          Tag             Image Id      Size
------------------------------------------------------------------------
certbot       certbot/certbot     latest          67cfe9e9c63f  95.54 MB
mariadb_db    mariadb             10.5            939d05495a90  386.7 MB
webserver     nginx               1.19.7-alpine   5a30d6b798fd  21.21 MB
wordpress     wordpress           5.6.2-fpm-alpine e0925ca46b42  229.9 MB
```

Um alle laufenden Prozesse anzuschauen, gebe ich **docker-compose ps** ein und bekomme folgendes Ergebnis

```
michalis@michalis-ubuntu:~/Docker_Projects/modul_239/wordpress$ docker-compose ps
  Name                 Command               State          Ports
-------------------------------------------------------------------------
certbot     certbot certonly --webroot ...   Exit 1
mariadb     docker-entrypoint.sh --def ...   Up       3306/tcp
webserver   /docker-entrypoint.sh ngin ...   Up       0.0.0.0:80->80/tcp
wordpress   docker-entrypoint.sh php-fpm     Up       9000/tcp
```

Da ich eine Namensänderung zum Container mariadb_db > mariadb angelegt habe, werde ich erneut das **docker-compose up -d** Befehl ausführen und sehe, dass ich mit dem Befehl **docker-compose up -d --remove-orphans** jegliche veraltete Container löschen kann. Mit den Zusatz der Option --no-deps am docker-compose up -d Befehl teilt man Compose mit, dass es den Start des Webserver-Dienstes überspringen kann, da dieser bereits läuft

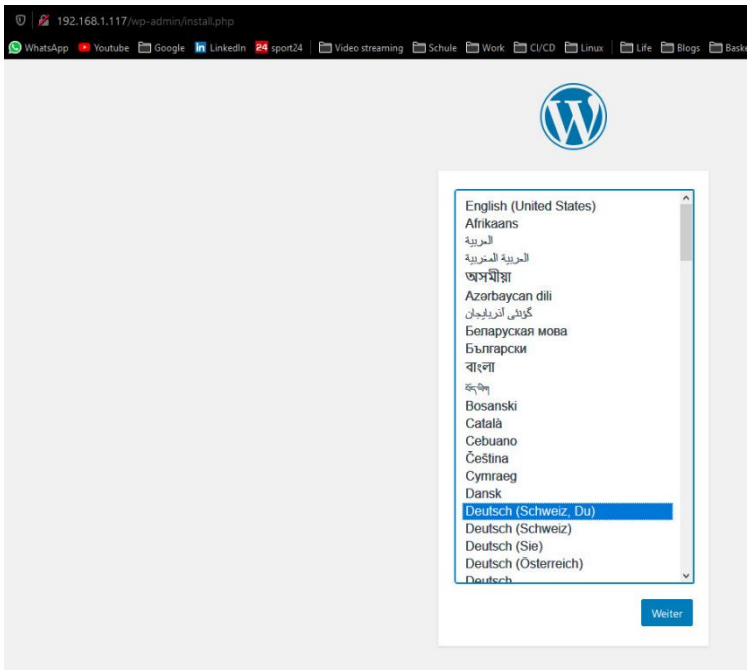
docker-compose up -d --no-deps certbot

```
michalis@michalis-ubuntu:~/Docker_Projects/modul_239/wordpress$ docker-compose ps
  Name                 Command               State          Ports
-------------------------------------------------------------------------
certbot     certbot certonly --webroot ...   Up       443/tcp, 80/tcp
mariadb     docker-entrypoint.sh --def ...   Up       3306/tcp
webserver   /docker-entrypoint.sh ngin ...   Up       0.0.0.0:80->80/tcp
wordpress   docker-entrypoint.sh php-fpm     Up       9000/tcp
```

Wenn ich mir die Logs anschaue, bekomme ich folgende Fehlermeldung bei dem HTTP (ACME) Challenge seitens certbot und Let's Encrypt

```
certbot | Performing the following challenges:
certbot | http-01 challenge for vinylo.shop
certbot | http-01 challenge for www.vinylo.shop
certbot | Using the webroot path /var/www/html for all unmatched domains.
certbot | Waiting for verification...
certbot | Challenge failed for domain www.vinylo.shop
certbot | Challenge failed for domain vinylo.shop
certbot | http-01 challenge for www.vinylo.shop
certbot | http-01 challenge for vinylo.shop
certbot | Cleaning up challenges
certbot | Some challenges have failed.
certbot | IMPORTANT NOTES:
certbot |  - The following errors were reported by the server:
certbot |
certbot |    Domain: vinylo.shop
certbot |    Type:   connection
certbot |    Detail: Fetching
certbot |    http://vinylo.shop/.well-known/acme-challenge/Apg9POjcqfmFAx6b70hiBoc0YNceTVQKcF34dVCLLGA:
certbot |    Timeout during connect (likely firewall problem)
certbot |
certbot |    To fix these errors, please make sure that your domain name was
certbot |    entered correctly and the DNS A/AAAA record(s) for that domain
certbot |    contain(s) the right IP address. Additionally, please check that
certbot |    your computer has a publicly routable IP address and that no
certbot |    firewalls are preventing the server from communicating with the
certbot |    client. If you're using the webroot plugin, you should also verify
certbot |    that you are serving files from the webroot path you provided.
certbot |  - The following errors were reported by the server:
certbot |
certbot |    Domain: www.vinylo.shop
certbot |    Type:   dns
certbot |    Detail: DNS problem: NXDOMAIN looking up A for www.vinylo.shop -
certbot |    check that a DNS record exists for this domain
certbot | Saving debug log to /var/log/letsencrypt/letsencrypt.log
certbot | Plugins selected: Authenticator webroot, Installer None
certbot | Requesting a certificate for vinylo.shop and www.vinylo.shop
```

Beim Zugriff auf der NGINX Seite und zusätzlich das der WordPress werde ich mit folgendes konfrontiert



Gerade nach der erfolgreicher Verbindung von Wordpress habe ich in einer der Konfigurationsdateien einige Zeilen angepasst und kam nach etwa 1-2h Aufwand nie ans WordPress Login. Deswegen habe ich

6.3.3 WordPress im Portainer einrichten (Fehlerhaft)

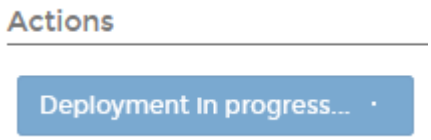
Als neues Stack wird jetzt WordPress angelegt. Unter Stacks > Add New Stack das folgende .yml File einfügen

```
version: '2'
```

```
services:
  db:
    image: mariadb
    volumes:
      - /wordpress/db:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: $MSQL_RPW
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: $MSQL_PW

  wordpress:
    image: wordpress:latest
    ports:
      - 8977:80
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: $Wordpress_PW
    volumes:
      - /wordpress:/var/www/html
```

Folgende Environment Variables definiere ich, sodass sie nicht im Klartext im Compose-File stehen

Mit **Deploy the Stack** erstelle ich die nötigen Containers



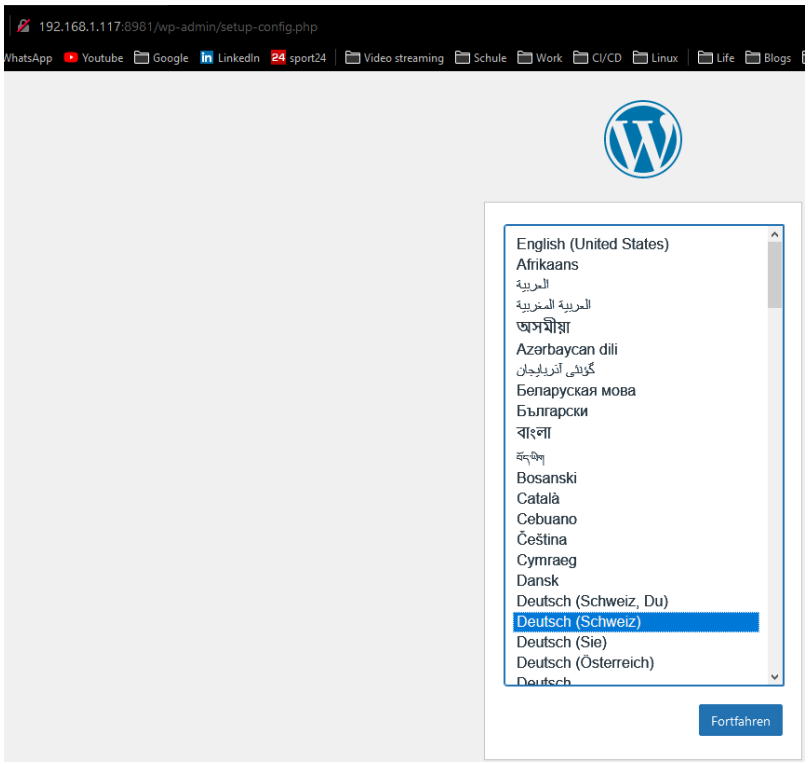
Nun ist mein WordPress-Container auf [IP-Adresse]:8977 erreichbar. Leider ist diesmal auch der NGINX WordPress Container nicht erreichbar. Obwohl das offizielle yaml-File von WordPress – Docker Hub genommen wurde, scheint es, dass das Zusammenspiel zwischen NGINX/PHP Image von WordPress und mit der MariaDB Datenbank ein Problem zu haben.

6.3.4 WordPress über CLI (Fehlerhaft, keine DB-Anbindung als eigenständiges Container)

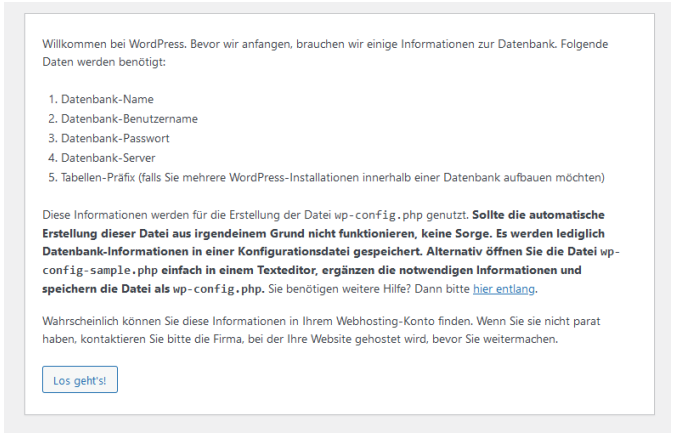
Nun habe ich über das CLI vom Pi folgendes Kommando ausgeführt, um eine herkömmliche Apache-WordPress Container aufzubringen

docker run --name wordpressCLI -p 8981:80 -d wordpress

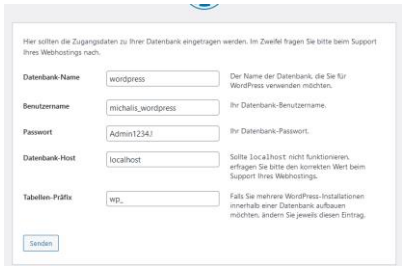
Der Zugriff auf Port 8981 ist nun möglich



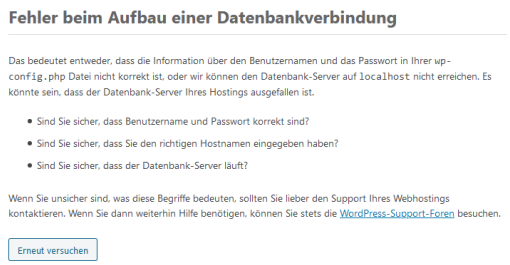
Auf Fortfahren > Los geht's



Die Felder werden wie folgt befüllt



Das Scheitern, meiner Meinung nach ist es wegen der Nicht-Anbindung einer DB und das alleinstehende Ausführen des WordPress-Containers

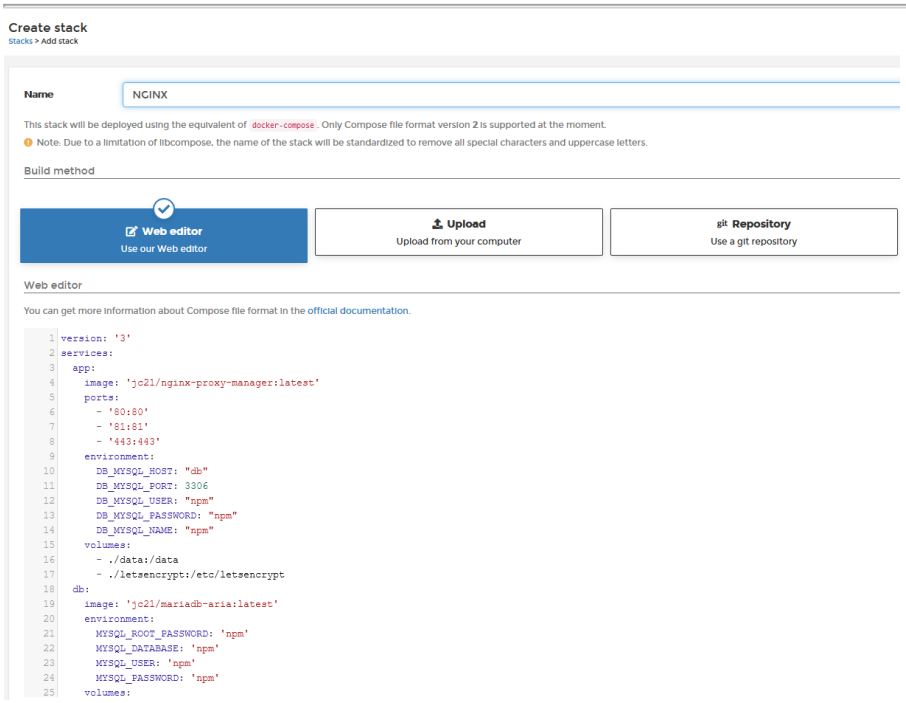


6.3.5 Obsolet NGINX RP Manager via Portainer

Auf **Stack > Add New Stack** muss ich folgendes eingeben.

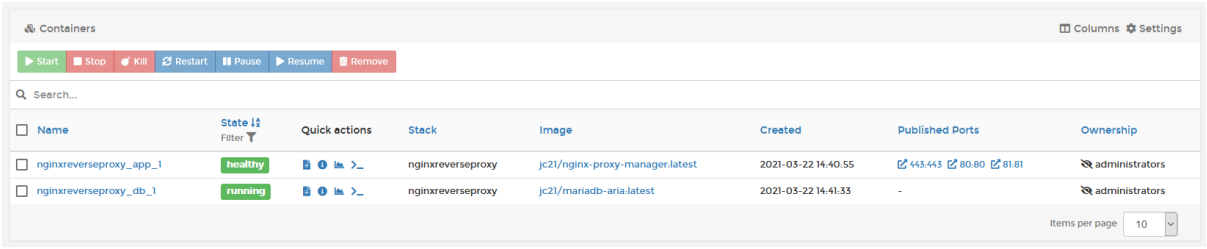
```
version: '2'
services:
  app:
    image: 'jc21/nginx-proxy-manager:latest'
    ports:
      - '80:80'
      - '81:81'
      - '443:443'
    environment:
      DB_MYSQL_HOST: "db"
      DB_MYSQL_PORT: 3306
      DB_MYSQL_USER: "npm"
      DB_MYSQL_PASSWORD: "npm"
      DB_MYSQL_NAME: "npm"
    volumes:
      - ./data:/data
      - ./letsencrypt:/etc/letsencrypt
  db:
    image: 'jc21/mariadb-aria:latest'
    environment:
      MYSQL_ROOT_PASSWORD: 'npm'
      MYSQL_DATABASE: 'npm'
      MYSQL_USER: 'npm'
      MYSQL_PASSWORD: 'npm'
    volumes:
      - ./data/mysql:/var/lib/mysql
```

Die Muss wie folgt aussehen

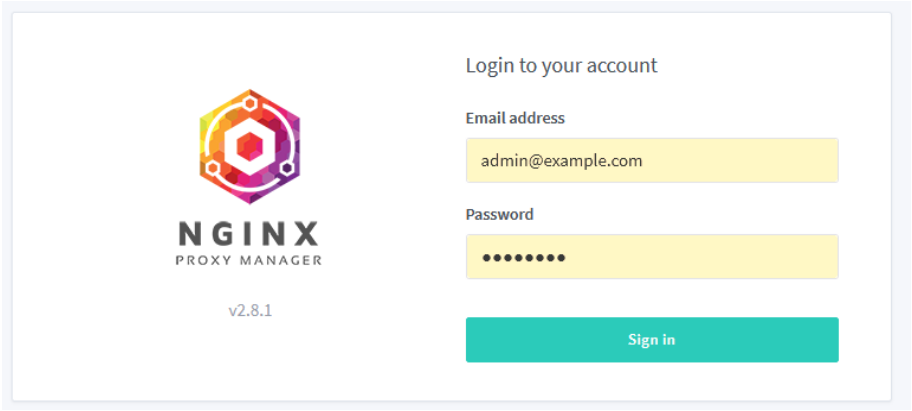


Zuletzt auf **Deploy this Stack**

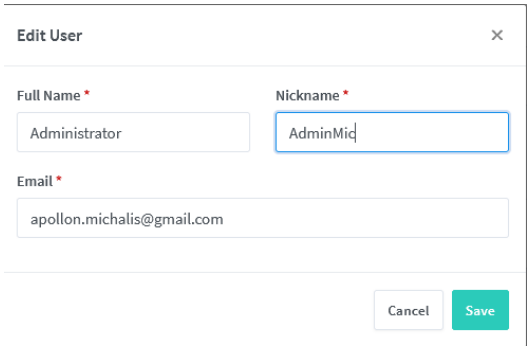
Nach dem erfolgreichen Starten des Stack, klicke ich auf dem Stack und sehe zwei erstellte Containers (Appl. und DB)



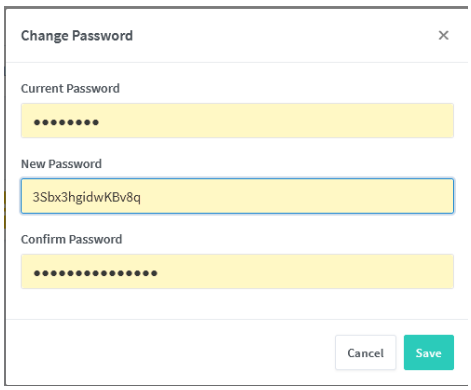
Auf den Reverse Proxy Manager kann ich jetzt via der 3 konfigurierten Ports zugreifen. Dafür gebe ich [IP-Adresse des Pis]:81 im Browser ein und lande auf folgende Loginseite. Die Logindaten sind **admin@example.com/changeme**



Daraufhin werde ich angefordert meine E-Mail zu ändern

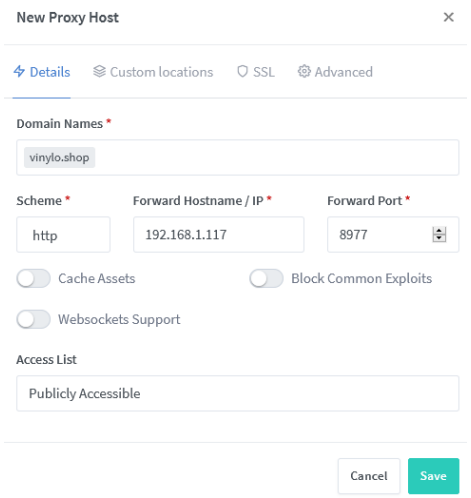


Das Passwort ändere ich ebenfalls

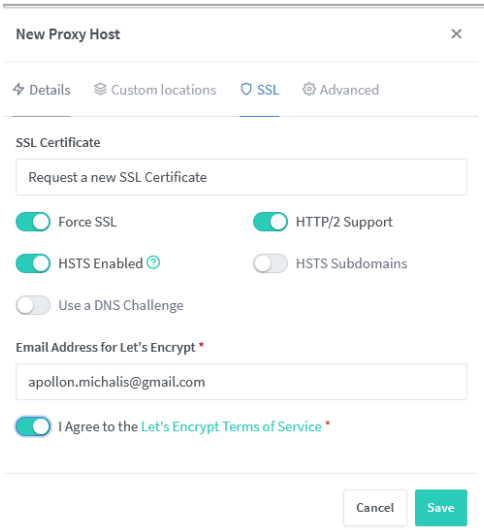


New NGINX Proxy Host

Um die Domäne über der IP-Adresse

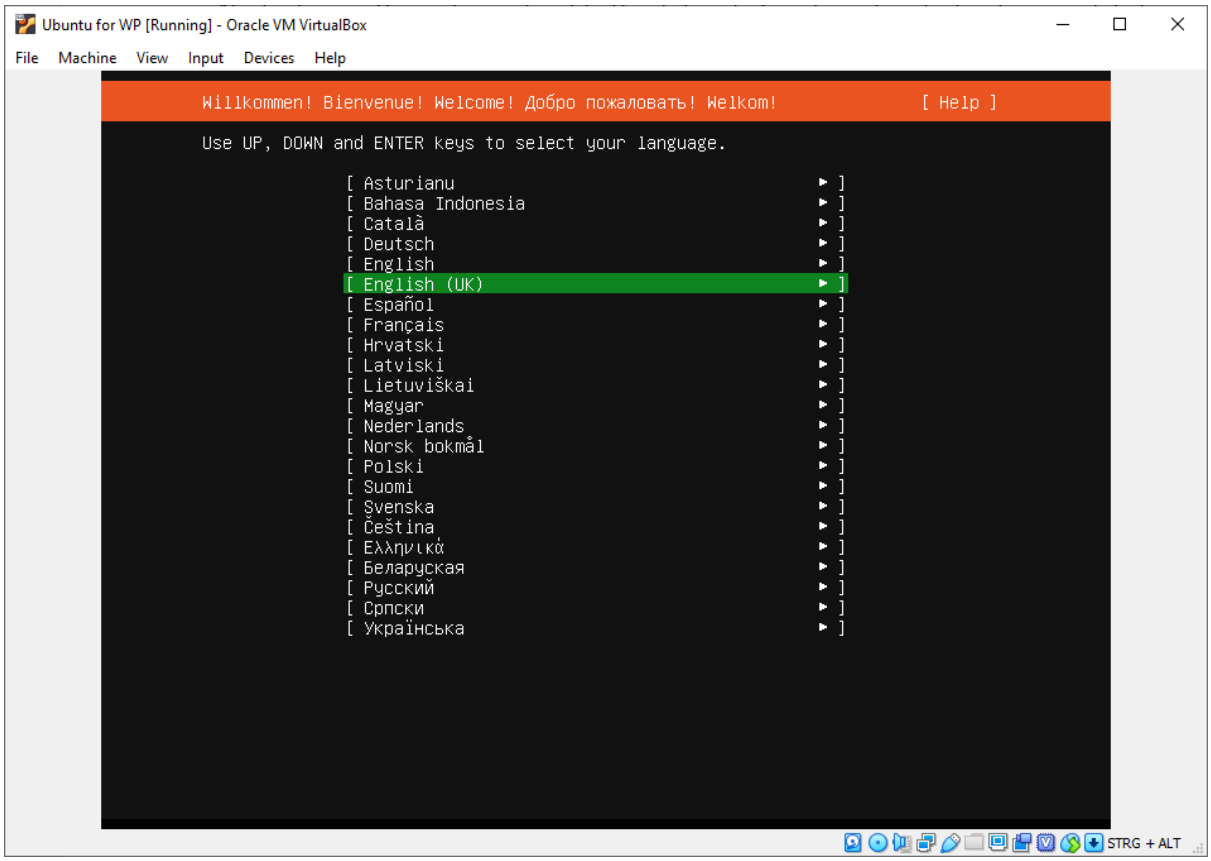


Unter SSL > folgendes ankreuzen/ausfüllen

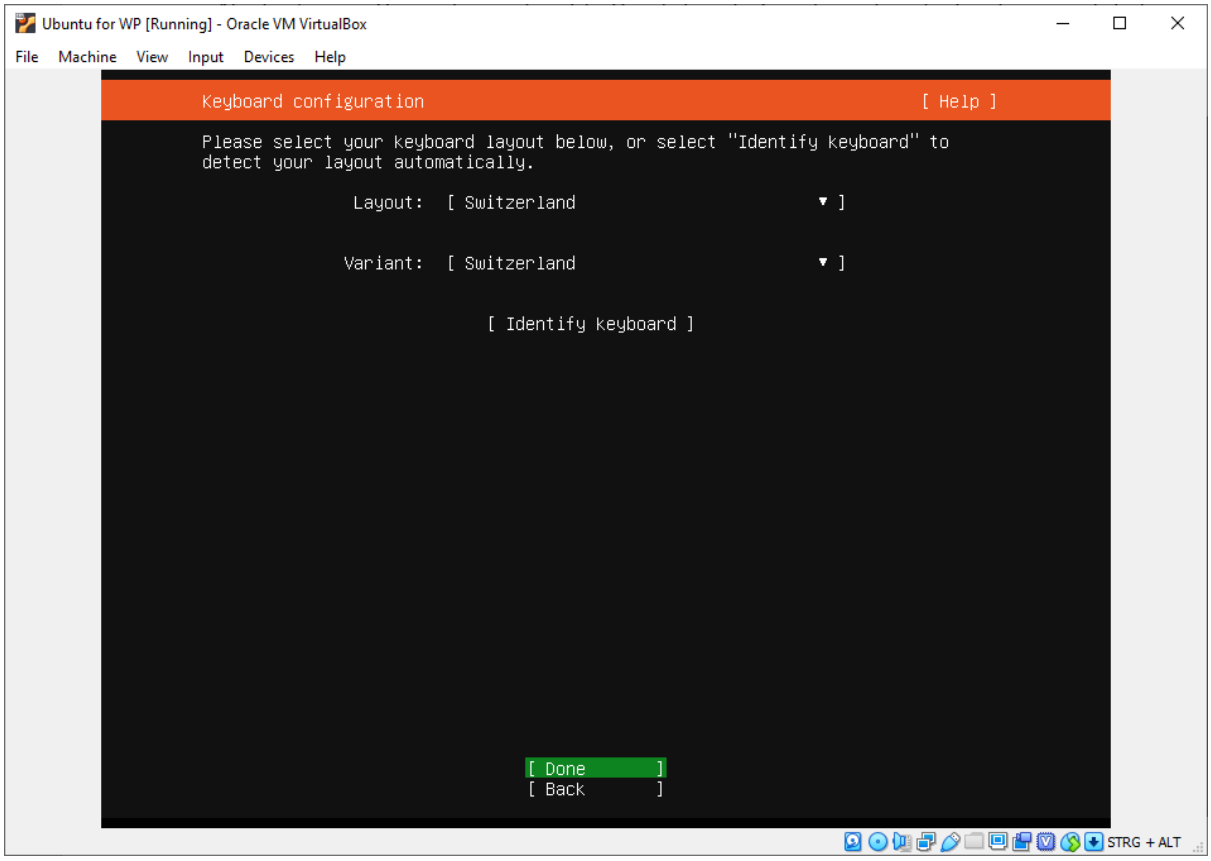


6.3.6 Lokale Installation WordPress, Apache

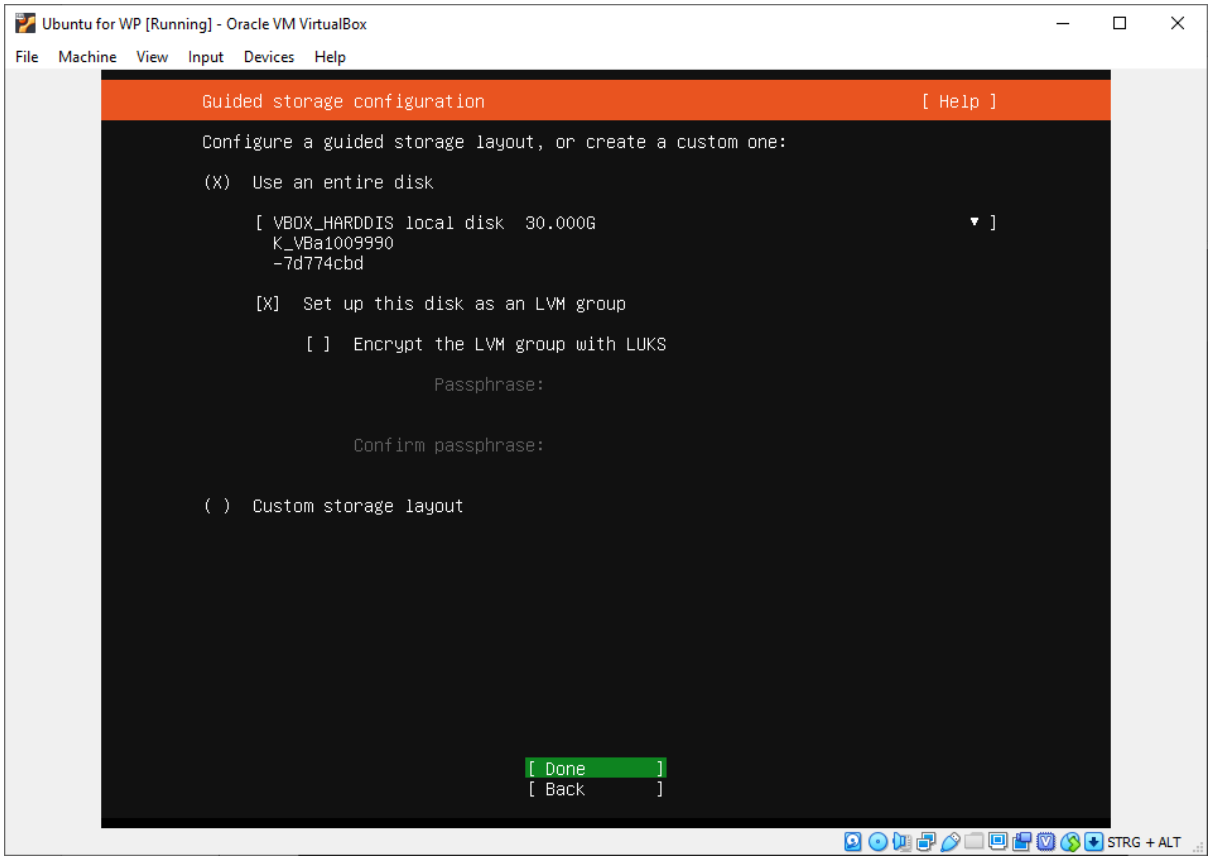
Nach etwaige Versuche und nichts-Verstehen habe ich mich entschieden eine lokale Instanz von WP/Apache und SSL auszuführen und alle meine Docker Container herunterzufahren. Server



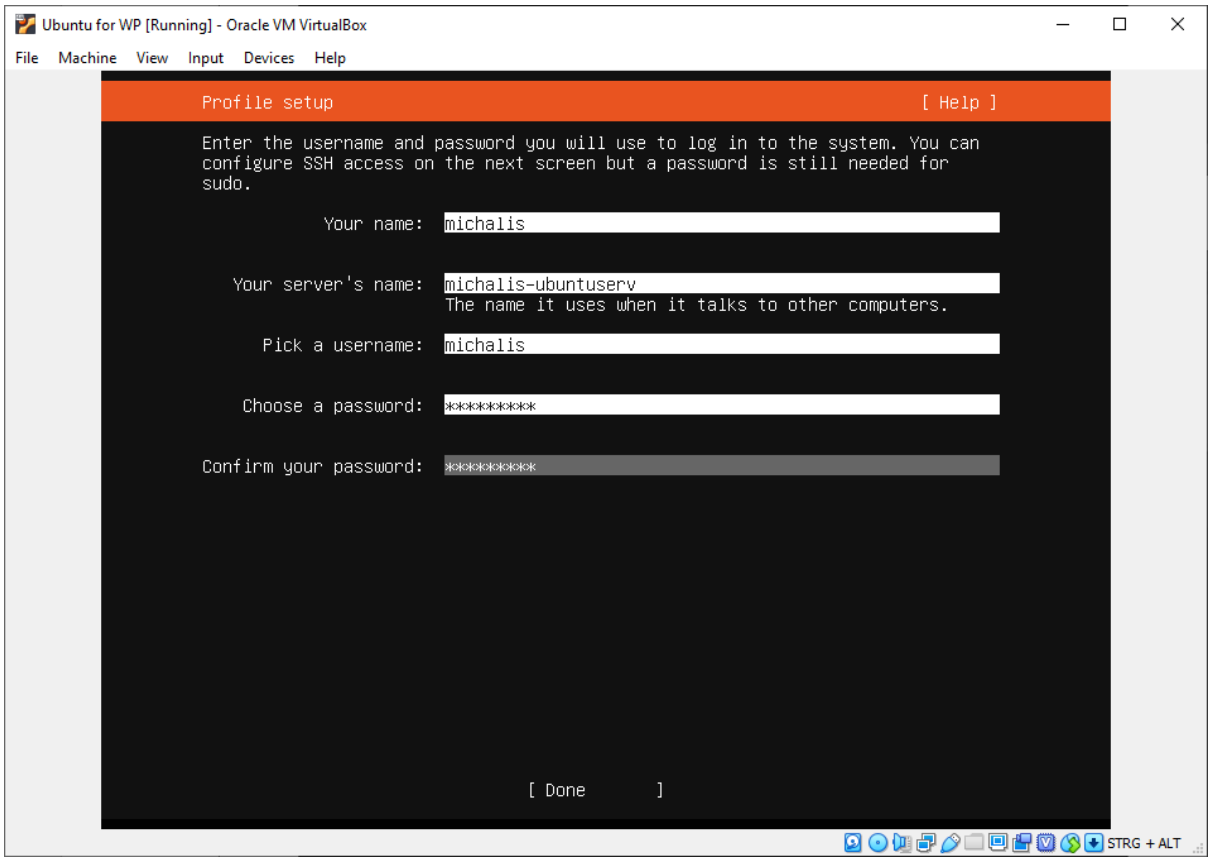
Schweiz auswählen


Ubuntu for WP [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Keyboard configuration
[Help]
Please select your keyboard layout below, or select "Identify keyboard" to detect your layout automatically.
Layout: [Switzerland ▾]
Variant: [Switzerland ▾]
[Identify keyboard]
[Done]
[Back]
STRG + ALT

Disk


Ubuntu for WP [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Guided storage configuration
[Help]
Configure a guided storage layout, or create a custom one:
(X) Use an entire disk
[VBOX_HARDDIS local disk 30.000G ▾]
K_VBa1009990
-7d774cbd
[X] Set up this disk as an LVM group
[ ] Encrypt the LVM group with LUKS
Passphrase:
Confirm passphrase:
( ) Custom storage layout
[Done]
[Back]
STRG + ALT

Name;



Netzwerkkonfigs und Installieren. Anschliessend neu starten

Anmelden und mit System aktualiseren

Statische IP-Setzung

```
  GNU nano 4.8                  /etc/netplan/00-installer-config.yaml
# This is the network config written by 'subiquity'
network:
  ethernets:
    enp0s3:
      addresses: [192.168.1.140/24]
      gateway: 192.168.1.1
      nameservers:
        addresses: [4.2.2.2, 8.8.8.8]
  version: 2
```

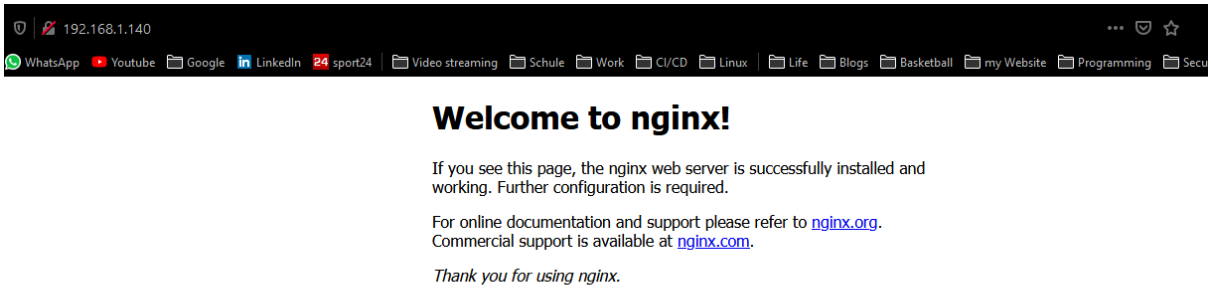
sudo netplan apply

Installation LEMP

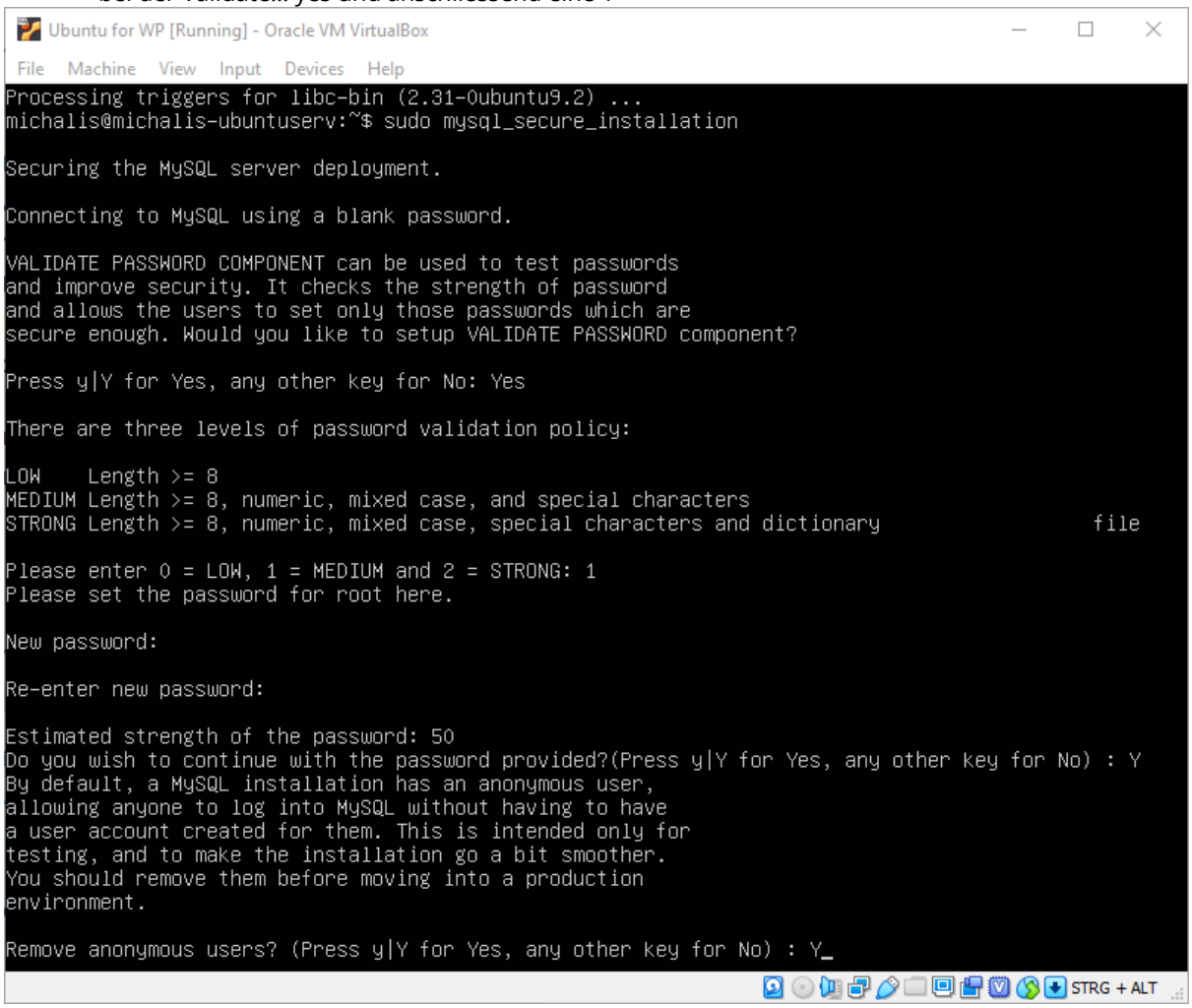
sudo apt install nginx -y

LAMP sudo apt install apache2 -y

Erfolgreiche Installation



- sudo apt install mysql-server -y
- sudo mysql_secure_installation
- bei der Validate... yes und anschliessend eine 1



- bei allen nächsten Prompts Y eingeben

Um php zu installieren sudo apt install php-fpm php-mysql.

sudo apt install php libapache2-mod-php php-mysql

Ordner für HTML-Dokumenten der Webseite erstellen

- sudo mkdir /var/www/vinylo
- sudo chown -R $USER:$USER /var/www/vinylo
- sudo nano /etc/nginx/sites-available/vinylo

Folgendes Text befüllen

```
  GNU nano 4.8                                    /etc/nginx/sites
server {
    listen 80;
    server_name vinylo.shop www.vinylo.shop;
    root /var/www/vinylo;

    index index.html index.htm index.php;

    location / {
        try_files $uri $uri/ =404;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
     }

    location ~ /\.ht {
        deny all;
    }

}
```

Aktiviere

- sudo ln -s /etc/nginx/sites-available/vinylo /etc/nginx/sites-enabled/
- sudo unlink /etc/nginx/sites-enabled/default
- sudo nginx -t

Output;

```
michalis@michalis-ubuntuserv:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Mit sudo systemctl reload nginx neu starten

- nano /var/www/vinylo/index.html

```
  GNU nano 4.8                                    /var/www/vinylo/inde
<html>
  <head>
    <title>Vinylo Shop</title>
  </head>
  <body>
    <h1>Hello World!</h1>

    <p>This is the landing page of <strong>Vinylo</strong>.</p>
  </body>
</html>
```

Erfolgreich

Hello World!

This is the landing page of **Vinylo**.

MySQL Einrichtung

- sudo mysql
- CREATE DATABASE wordpress DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;
- CREATE USER 'wordpressusr'@'localhost' IDENTIFIED BY 'koL83.!j';
- GRANT ALL ON wordpress.* TO 'wordpressusr'@'localhost';
- EXIT;

PHP Zusätze

- sudo apt install php-curl php-gd php-intl php-mbstring php-soap php-xml php-xmlrpc php-zip -y
- sudo systemctl restart php7.4-fpm

WordPress installieren

- sudo wget https://wordpress.org/latest.tar.gz
- sudo tar
- sudo chown -R www-data:www-data /var/www/html/wordpress

- sudo cp
- sudo nano wp-config.php

Jetzt werden Secret Keys gebraucht. Um neue zu holen curl -s https://api.wordpress.org/secret-key/1.1/salt/. Werte sollen jetzt mit der rechten Maustaste kopiert werden

```
michalis@michalis-ubuntuserv:/var/www/html/wordpress$ curl -s https://api.wordpress.org/secret-key/1.1/salt/
define('AUTH_KEY',         ' UW*tkZ[7TF<Z;U+rjAQZ/9QfvF@Q-QHy`F@a63+jZ2|P6?7$4P3P@4q#kM2o45z');
define('SECURE_AUTH_KEY',  'rzm|I><+$Rxt~Y-A%3iS^duae4H{<gl&@-{V4E ;9*c0?ASeD3>:hnyZFDJTo+kO');
define('LOGGED_IN_KEY',    'U%Ch;d $.3i n>Dn%Rw?awdLsN|Uk:?g*c;m`IGQ|W[=0{&.W[:!;(TvIW6S6Oi6');
define('NONCE_KEY',        '->9W_<qsVi[2v}uQ( /lS?`g-r.+cHmL-$IZSm74c5B>u6MG/8HFz&g:OvOkq/7L');
define('AUTH_SALT',        '^NrrU1D+<V++|/5~mb38i;u&dXDCqHC>?CFz[H6A8z(|X|D!?h/,+|jkT$M~Wsr$');
define('SECURE_AUTH_SALT', 'c9^JyV|[2{l;c{#|l?CZcLTAb|0|G}]Csoi )p7PS?GKbcUFxF-8xQGxB>wQK&BF');
define('LOGGED_IN_SALT',   'A.K0^Z|?~B-|dWQ;2!|/lBqM_8`_r8ErLa(,]j#{` T+h,yx?wIUm([KPsY;rd;9');
define('NONCE_SALT',       'mfgZ<in+O78JhQp<tIp|Hz$cE46RJZ^|2Hi[q|;MfO5=&V.RPW}RZ9m@K&/R&9ls');
```

Bestehende Werte löschen und die neuen Werte einfügen

```
 *
 * @since 2.6.0
 */
define('AUTH_KEY',         ' UW*tkZ[7TF<Z;U+rjAQZ/9QfvF@Q-QHy`F@a63+jZ2|P6?7$4P3P@4q#kM2o45z');
define('SECURE_AUTH_KEY',  'rzm|I><+$Rxt~Y-A%3iS^duae4H{<gl&@-{V4E ;9*c0?ASeD3>:hnyZFDJTo+kO');
define('LOGGED_IN_KEY',    'U%Ch;d $.3i n>Dn%Rw?awdLsN|Uk:?g*c;m`IGQ|W[=0{&.W[:!;(TvIW6S6Oi6');
define('NONCE_KEY',        '->9W_<qsVi[2v}uQ( /lS?`g-r.+cHmL-$IZSm74c5B>u6MG/8HFz&g:OvOkq/7L');
define('AUTH_SALT',        '^NrrU1D+<V++|/5~mb38i;u&dXDCqHC>?CFz[H6A8z(|X|D!?h/,+|jkT$M~Wsr$');
define('SECURE_AUTH_SALT', 'c9^JyV|[2{l;c{#|l?CZcLTAb|0|G}]Csoi )p7PS?GKbcUFxF-8xQGxB>wQK&BF');
define('LOGGED_IN_SALT',   'A.K0^Z|?~B-|dWQ;2!|/lBqM_8`_r8ErLa(,]j#{` T+h,yx?wIUm([KPsY;rd;9');
define('NONCE_SALT',       'mfgZ<in+O78JhQp<tIp|Hz$cE46RJZ^|2Hi[q|;MfO5=&V.RPW}RZ9m@K&/R&9ls');

/**#@-*/
```

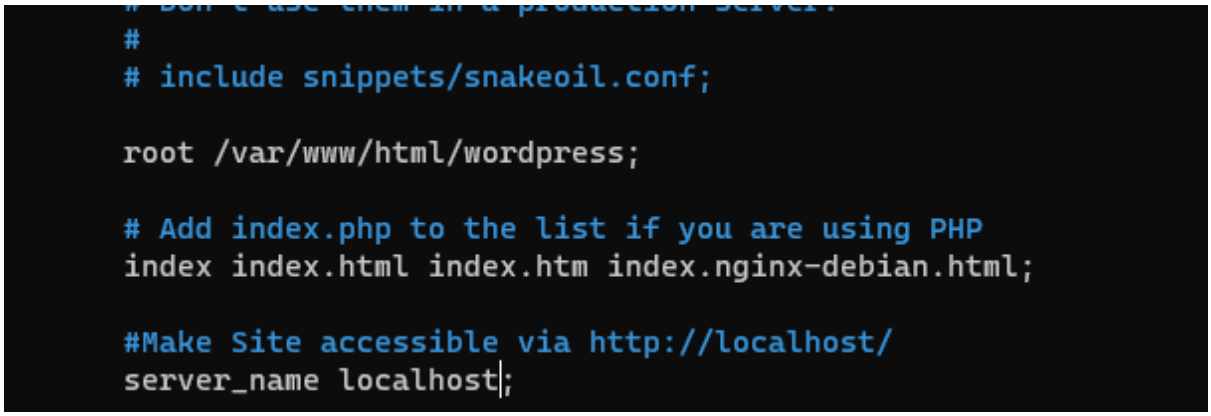
Anpassen

```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );

/** MySQL database username */
define( 'DB_USER', 'wordpressusr' );

/** MySQL database password */
define( 'DB_PASSWORD', 'koL83.!j' );

/** MySQL hostname */
define( 'DB_HOST', 'localhost' );

/** Database Charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The Database Collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );
```

Konfig file abspeichern

NGINX konfig.

In cd /etc/nginx/sites-available/ gehen und sudo cp default wordpress eingeben, um das Default-Konfigurationsfile zu duplizieren. Danach mit sudo nano wordpress in dem gehen.



Eingeben sudo ln -s /etc/nginx/sites-available/your_domain /etc/nginx/sites-enabled/

6.3.7 Mailcow über Pi (Troubleshooting)

Beim Mailserver wird eine Git-Repo auf dem lokalen Pi geklont und danach aufgesetzt. Bei der Installation wird Mailcow sich mit allen Docker Konfigurationsdateien kümmern, sodass die Installation nicht allzu komplex wird.

Als Erstes gehe ich zu der Orderstruktur /opt und gebe folgender Befehl ein sudo git clone https://github.com/mailcow/mailcow-dockerized

```
michalis@michalis-ubuntu:/opt$ umask
0002
michalis@michalis-ubuntu:/opt$ sudo su
[sudo] password for michalis:
root@michalis-ubuntu:/opt# git clone https://github.com/mailcow/mailcow-dockerized
Cloning into 'mailcow-dockerized'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 33828 (delta 1), reused 1 (delta 0), pack-reused 33823
Receiving objects: 100% (33828/33828), 22.51 MiB | 11.08 MiB/s, done.
Resolving deltas: 100% (21695/21695), done.
root@michalis-ubuntu:/opt# ls
containerd  mailcow-dockerized
root@michalis-ubuntu:/opt# ls mailcow-dockerized/
CODE_OF_CONDUCT.md  CONTRIBUTING.md  data  docker-compose.yml  generate_config.sh  helper-scripts  LICENSE  README.md  update.sh
root@michalis-ubuntu:/opt#
```

Super-Tutorial

Danach wechsle ich zum erstellten Ordner mit **cd mailcow-dockerized**. Darin gebe ich das Kommando **ls -l**, um mir die Inhalte anzuzeigen. Nun führe ich mit **sudo ./generate-config.sh** das Konfigurationsskript erstmals aus.

Bei der **FQDN des Mailservers** gebe ich nun meine gewünschte Subdomain-Adresse ein, die ich schon eingerichtet habe. Die Zeitzone ist standardmässig richtig, also drücke ich Enter, damit im nächsten Schritt das Private Key erstellt wird und in einem Ordner abgelegt wird.

```
root@michalis-ubuntu:/opt/mailcow-dockerized# ./generate_config.sh
Press enter to confirm the detected value '[value]' where applicable or enter a custom value.
Mail server hostname (FQDN) - this is not your mail domain, but your mail servers hostname: mail.ubuntu.local
Timezone [Europe/Zurich]:
Generating snake-oil certificate...
Generating a RSA private key
.....
.....++++
writing new private key to 'data/assets/ssl-example/key.pem'
-----
Copying snake-oil certificate...
```

Die erstellte Konfigurationsdatei kann ich jetzt mit sudo nano mailcow.conf anschauen, um alle Einstellungen betr. Ports/Hostname und alle andere jegliche Einstellungen eine Übersicht zu bekommen.

```
# Comma separated list without spaces! Example: ADDITIONAL_SERVER_NA

ADDITIONAL_SERVER_NAMES=

# Skip running ACME (acme-mailcow, Let's Encrypt certs) - y/n

SKIP_LETS_ENCRYPT=y

# Create seperate certificates for all domains - y/n
# this will allow adding more than 100 domains, but some email clien
# see https://wiki.dovecot.org/SSL/SNIClientSupport
ENABLE_SSL_SNI=n

# Skip IPv4 check in ACME container - y/n
```

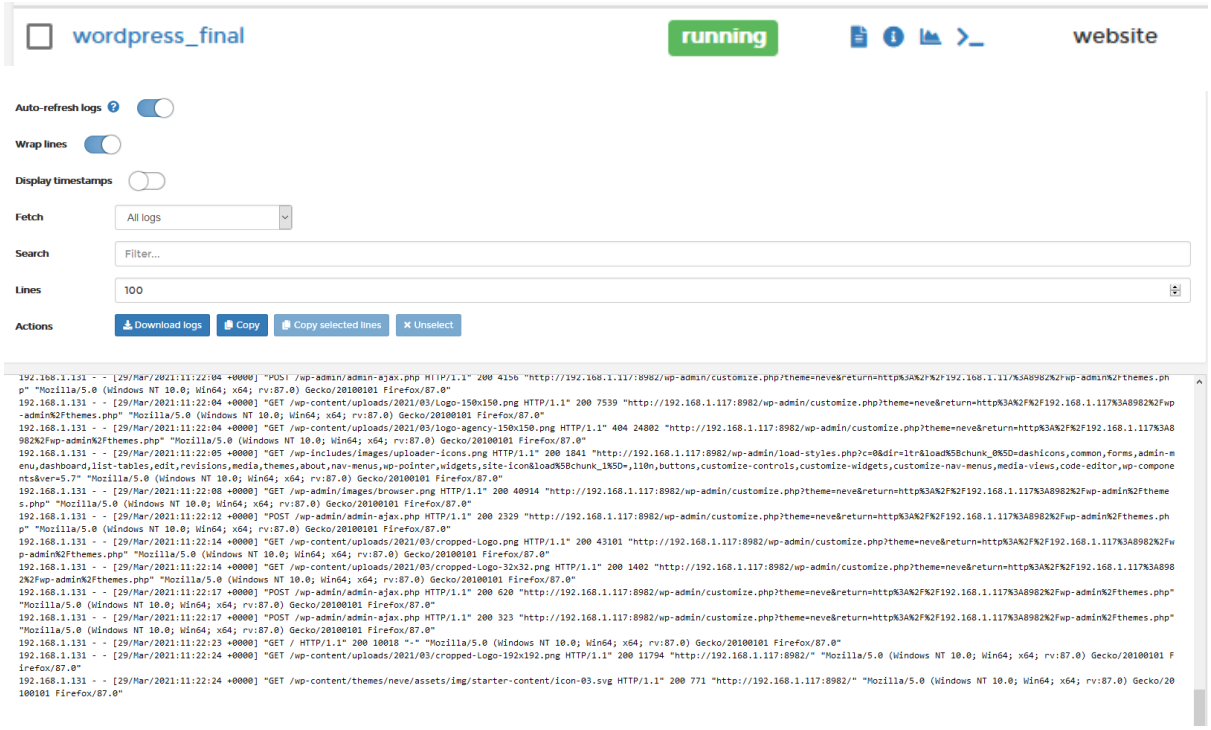
Mit **docker-compose up -d** kann ich jetzt für diese Ansammlung von Diensten ein Image aufgrund des bestehenden docker-compose.yml-File kreieren. Wenn alle Dienste erfolgreich zusammenkompiliert sind, sieht der Output wie folgt aus

```
Creating mailcowdockerized_watchdog-mailcow_1  ... done
Creating mailcowdockerized_unbound-mailcow_1   ... done
Creating mailcowdockerized_clamd-mailcow_1     ... done
Creating mailcowdockerized_memcached-mailcow_1 ... done
Creating mailcowdockerized_solr-mailcow_1      ... done
Creating mailcowdockerized_sogo-mailcow_1      ... done
Creating mailcowdockerized_olefy-mailcow_1     ... done
Creating mailcowdockerized_dockerapi-mailcow_1 ... done
Creating mailcowdockerized_redis-mailcow_1     ... done
Creating mailcowdockerized_ejabberd-mailcow_1  ... done
Creating mailcowdockerized_php-fpm-mailcow_1   ... done
Creating mailcowdockerized_mysql-mailcow_1     ... done
Creating mailcowdockerized_nginx-mailcow_1     ... done
Creating mailcowdockerized_dovecot-mailcow_1   ... done
Creating mailcowdockerized_postfix-mailcow_1   ... done
Creating mailcowdockerized_acme-mailcow_1      ... done
Creating mailcowdockerized_netfilter-mailcow_1 ... done
Creating mailcowdockerized_rspamd-mailcow_1    ... done
Creating mailcowdockerized_ipv6nat-mailcow_1   ... done
```

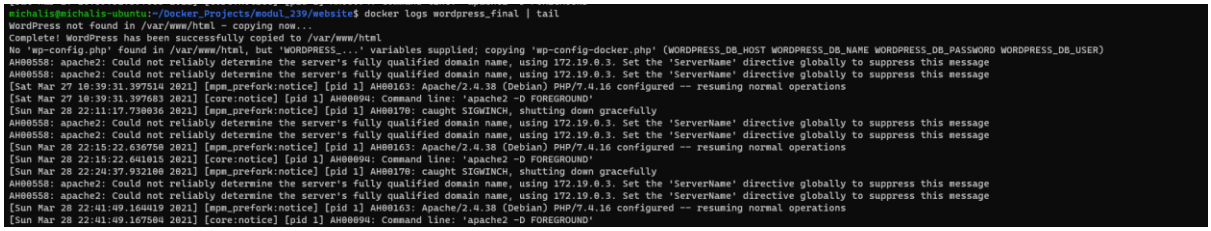
6.4 Protokollierung (SW3)

6.4.1 Ablageort Logfiles

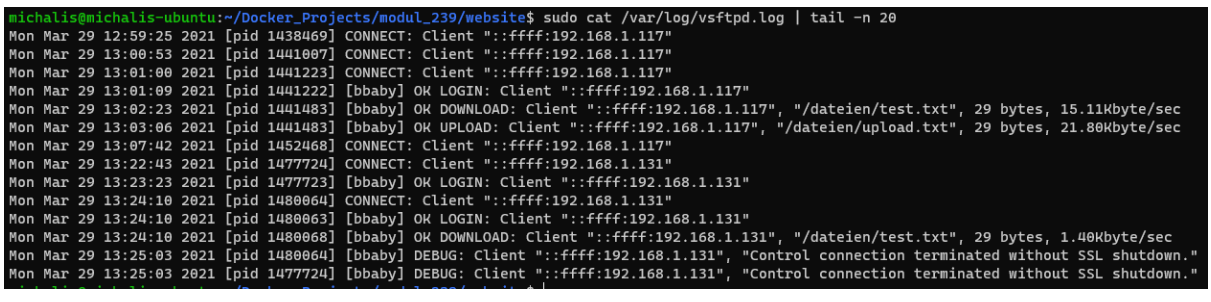
Im Portainer können Logs mit Click auf das Berichtsymbol eingesehen werden



Im CLI können die Logs wie folgt ausgelesen werden **docker logs [container-namen]** und mit den gepipeten Befehl **| tail -n 20** werden die letzten 20 Zeilen angezeigt



Die Logdateien des installiertes vsftpd-Dienst befinden sich im /var/log/vsftd.log Ordner und werden wie folgt ausgelesen (**sudo cat /var/log/vsftpd.log | tail -n 20**).



7 Kontrollieren

7.1 Funktions- und Lasttests

7.1.1 Webserver

Testfall	Erwartetes Resultat	Effektives Resultat
Ist die Webseite erreichbar	Über vinylo.shop und www.vinylo.shop erreichbar	Ja, die Webseite ist unter beiden Domänen erreichbar
Mit 5 aktive Sessions ist sie erreichbar?	Die Website sollte nicht abstürzen	Die Webseite lauft ordnungsgemäss weiter
Funktioniert die verschlüsselte Kommunikation	Richtigen TLS cert eingebunden	Ja, der HTTPS-Zugriff funktioniert
Funktioniert die Weiterleitung von HTTP auf HTTPS	301 Meldung, eine erfolgreiche Weiterleitung	Ja, der HTTP-Zugriff wird erfolgreich weitergeleitet

Tabelle 24: Funktionstest Webserver

7.1.2 WordPress (CMS)

Testfall	Erwartetes Resultat	Effektives Resultat
Ist der WordPress Panel erreichbar	Ja der WP Panel ist erreichbar	Ja, der Webpanel ist über vinylo.shop/wp-admin erreichbar
Funktioniert die verschlüsselte Kommunikation	Soll mit HTTPS verschlüsselt sein	Ja, der HTTPS-Zugriff funktioniert
Haben die Berater Schreibezugriff?	Ja sie haben. Admineinstellungen können Sie nicht vornehmen	Ja, Sie können nur Posts verfassen

Tabelle 25: Funktionstest WordPress

7.1.3 Mailcow

Testfall	Erwartetes Resultat	Effektives Resultat
Ist das Admin GUI erreichbar	Der Admin kann sich anmelden	Das Admin GUI ist über 192.168.1.140 erreichbar
Können die unterschiedliche Mitarbeiter sich im Webclient anmelden	Der Login sollte funktionieren	Ja der Login funktioniert für jeden Mitarbeiter
Funktioniert die verschlüsselte Kommunikation	Mit einem TLS Zertifikat verschlüsselte Verkehr	Nein, leider nicht
Können E-Mails versendet und empfangen werden	Ja E-Mails können sowohl versendet als empfangen werden	E-Mails können leider nicht versendet oder empfangen werden, jedoch erscheint keine Fehlermeldung beim Verfasser / Empfänger

Tabelle 26: Funktionstest Mailcow

7.1.4 FTP

Testfall	Erwartetes Resultat	Effektives Resultat
Ist der Server erreichbar	Der FTP-Server sollte über die IP-Adresse des Pis erreichbar sein	Ja, der Server ist erreichbar.
Können zusätzliche Mitarbeiter auf dem FTP-Server kommen	Mit Ihrem erstellten Login und PW sollten die MA den Zugriff auf den FTP-Server haben.	Ja, sie können im FileZilla Ihr Login und PW eingeben und sehen ihren Home-Ordner für das Hoch- und Herunterladen von Files
Können Dateien hoch- und heruntergeladen werden	Das Hoch- und Herunterladen sollte für den Usern funktionieren	Ja das Hoch- und Herunterladen funktioniert

Tabelle 27: Funktionstest FTP

8 Auswerten

8.1 Reflexion

Das Modul hat mir gefallen, da ich selbständig eine kreative Idee für einen Webauftritt einer fiktiven Firma herausdenken konnte und die Freiheit hatte, mich mit jeglichen, modernen Programmen auseinandersetzen und für das Beste entschieden. Dadurch habe ich eine Menge an hilfreiche Tools für geschäftliche sowie private Umfeld kennengelernt. Das Wissen, welches ich durch dieses Modul erworben habe, werde ich definitiv weiterverbreiten, sodass ich mich mit möglichst viele Prinzipien und Tools der IT, in allen Bereichen, auseinandersetze.

Die Auseinandersetzung mit dem Self-Hosting und eigene DNS-Records / Portforwarding war für mich lehrreich, jedoch würde es nicht gern wieder machen. Die Aufsetzung mit dem Hosting-Provider war am Anfang die einfachere Lösung, jedoch wollte ich mich herausfordern und schauen, ob bei mir das Self-Hosting funktionieren würde. Das Kompetenzraster meiner Meinung nach war zu wenig mit dem praktischen Teil belastet und mehr mit theoretischen Kompetenzen. Für zukünftige Klasen würde ich das Kompetenzraster umstrukturieren und evtl. Schüler auffordern schon bevor das M300, mit Docker/Git zu arbeiten.

8.2 Zielerreichnung

Die Ziele, welcher ich am Anfang des Moduls gesetzt habe, waren folgende

Ziel	Erledigt/Nicht erledigt	Grund falls nicht erledigt
Realisitisch, umsetzbare Idee für eine fiktive Firma herausdenken	✔	-
Mit modernen Technologien auseinandersetzen, im theoretischen sowie praktischen	😕	Docker wurde eingerichtet, jedoch Splunk/Grafana/Prometheus/Postman (richtig) konnte ich nie richtig verwirklichen
Saubere Aufsetzung mittels oben genannter Tools, erledigt	✕	Docker mit Self-Hosting war zu kompliziert und brachte viele Stolpersteinen
Saubere Lerndokumentation schreiben	✔	

Tabelle 28: Zielerreichung

8.3 Ausblick

Zukünftig werde ich im privaten Umfeld vermehrt Projekte mit Docker umsetzen, da den Betrieb und Unterhalt der Ressourcen mit dem Container perfekt gehandelt wird. Die Eigenständigkeit des Containers ermöglichen auch einen späteren, vereinfachten Prozess des Containers zu einem anderen System zu migrieren. Die Kompatibilität des Containers mit verschiedenen Linux-Distributionen sowie Betriebssysteme allgemein ist ein weiterer Vorteil und spricht für das Migrieren auch dafür.

9 Glossar

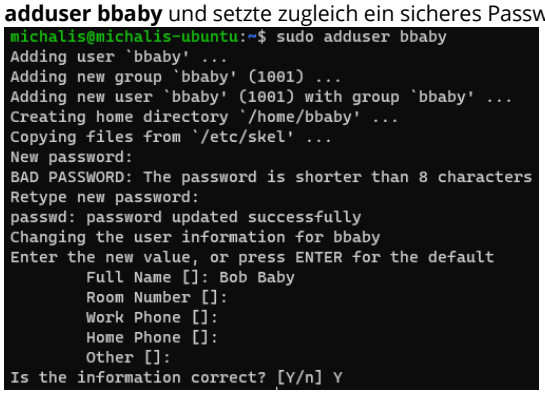
In den folgenden Tabellen findet Ihr Bezeichnungen, Abkürzungen und noch nicht weltweit bekannte Wörter, welche ich in dieser Lerndokumentation genannt habe

Alphabet	Wort	Bedeutung	Kapitel
A	ACME	Automatic Certificate Management Environment	Kap. 3.4 Webserver
	Array	Ein Array ist eine Sammlung von Daten (Ziffern, Buchstaben, Wörtern) welche mit {} [] «» angegeben werden kann	Kap . Docker Compose
B	Backend	Das Backend ist die Bezeichnung aller Geräten, die den Webservern und Applikationen mit Daten füttern. Dazu gehören Datenbanken und das Code, welche die Applikation notwendig ist	Kap. 3.2.9
D	DB	Datenbank	Kap. Prometheus
E	Endpoint	Prometheus HTTP Endpoint; Eine Quelle von Metriken, die ausgewertet werden können und normalerweise einem einzelnen	

		Prozess entsprechen.	
F	Fork	Ein Programm-Fork ist eine Abspaltung ein Programmes in zwei	Kap. 6.2.11
H	HERMES	Vorgehensmethodik für Projekte und Programme HERMES 5 ist ein eCH Standard	Kap,
I	Idempotent	Eine HTTP-Methode ist idempotent, wenn eine identische Anfrage einmal oder mehrmals hintereinander mit der gleichen Wirkung durchgeführt werden kann und dabei den Server im gleichen Zustand belässt.	Kap. 0 **LB1 HTTP Antworten (P)**
J	JSON	JavaScript Object Notation ist eine Methode, welcher APIs mit den Backend-Diensten (DBs usw.) kommunizieren können. Es stored und exchanged Data	
M	Metadata	Zusätzlich, mitgelieferte Daten einer Datei	**9.1.1 Kap. 6.2.15 FTP** **Installation und Benutzersetup** Zusätzlich stelle ich einen FTP-Server mit vsftpd auf. Als Erstes installiere ich den Dienst mit **sudo apt install vsftpd**.

Nun kopiere ich das Konfigurationsfile, sodass ich eine nigelnagelneue Datei habe. **sudo cp /etc/vsftpd.conf /etc/vsftpd.conf.bak**

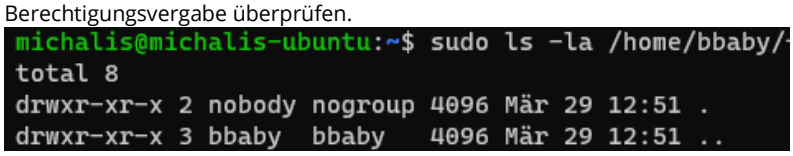
Da ich für die 4/5 Mitarbeiter je User erstelle ich eine Gruppe. Mit **sudo adduser bbaby** und setzte zugleich ein sicheres Passwort.



Um die Sicherheit zu gewährleisten, erstelle ich im Home-Ordner des neu erstellten Users einen Ordner namens ftp.

sudo mkdir /home/bbaby/ftp. Mit **sudo chown nobody:nogroup /home/bbaby/ftp**.

Mit **sudo ls -la /home/bbaby/ftp** kann ich die korrekte Berechtigungsvergabe überprüfen.

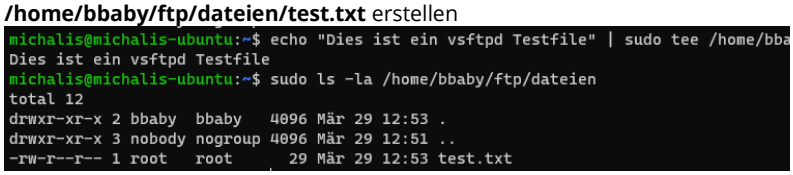


Nun erstelle ich den Ordner für das Hochladen von Dateien und gebe die entsprechende Berechtigungen.

sudo mkdir /home/bbaby/ftp/dateien – Ordner erstellen

sudo chown bbaby:bbaby /home/bbaby/ftp/dateien – Berechtigungen erteilen

Beispielsdatei **echo "Dies ist ein vsftpd Testfile" >> sudo tee /home/bbaby/ftp/dateien/test.txt** erstellen



FTP-Zugang konfigurieren

Die Konfigurationsdatei muss wie folgt überarbeitet werden. **sudo nano /etc/vsftpd.conf** eingeben.

write_enable=YES – unkommentieren

chroot_local_user=YES – unkommentieren

user_sub_token=$USER

local_root=/home/$USER/ftp

```
#
# Uncomment this to indicate that vsftpd use a utf8 filesystem.
#utf8_filesystem=YES

#User subtoken
user_sub_token=$USER
local_root=/home/$USER/ftp
```

pasv_min_port=40000

pasv_max_port=50000

```
#Port eingrenzen
pasv_min_port=40000
pasv_max_port=50000
```

userlist_enable=YES

userlist_file=/etc/vsftpd.userlist

userlist_deny=NO

```
#Userlist einlesen
userlist_enable=YES
userlist_file=/etc/vsftpd.userlist
userlist_deny=NO
```

File abspeichern und Editor schliessen

Nun füge ich den vorhin erstellter User zu der Liste der User mit **echo "bbaby" | sudo tee -a /etc/vsftpd.userlist.** Mit **cat /etc/vsftpd.userlist** kann ich die Anpassung überprüfen.

```
michalis@michalis-ubuntu:~$ echo "bbaby" | sudo tee -a /etc/vsftpd.userlist
bbaby
michalis@michalis-ubuntu:~$ cat /etc/vsftpd.userlist
bbaby
```

Mit **sudo systemctl restart vsftpd** starte ich den FTP-Dienst neu.

```
michalis@michalis-ubuntu:~$ sudo systemctl restart vsftpd
michalis@michalis-ubuntu:~$ sudo systemctl status vsftpd
● vsftpd.service - vsftpd FTP server
     Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset: enabl
     Active: active (running) since Mon 2021-03-29 12:57:47 CEST; 4s ago
```

FTP Zugang testen

Um den FTP Zugang zuerst mit einem anonymen-User zu testen, gebe ich die Public IP ein mit dem **ftp -p 178.192.230.31** Befehl im CMD ein.

Testfälle

Mit Anonym;

```
michalis@michalis-ubuntu:~$ ftp -p 192.168.1.117
Connected to 192.168.1.117.
220 (vsFTPd 3.0.3)
Name (192.168.1.117:michalis): anonymous
530 Permission denied.
Login failed.
ftp>
```

Mit sudo_user;

```
michalis@michalis-ubuntu:~$ ftp -p 192.168.1.117
Connected to 192.168.1.117.
220 (vsFTPd 3.0.3)
Name (192.168.1.117:michalis): sudo_user
530 Permission denied.
Login failed.
ftp> bye
221 Goodbye.
```

Jetzt melde ich mit dem User **bbaby** und sein **PW** ein.

```
michalis@michalis-ubuntu:~$ ftp -p 192.168.1.117
Connected to 192.168.1.117.
220 (vsFTPd 3.0.3)
Name (192.168.1.117:michalis): bbaby
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Darin wechsle ich zum Ordner «Dateien» und Frage mit dem Get-Befehl nach der Testdatei, welche ich vorher erstellt habe.

```
ftp> cd dateien
250 Directory successfully changed.
ftp> pwd
257 "/dateien" is the current directory
ftp> get test.txt
local: test.txt remote: test.txt
227 Entering Passive Mode (192,168,1,117,167,52).
150 Opening BINARY mode data connection for test.txt (29 bytes).
226 Transfer complete.
29 bytes received in 0.00 secs (52.3481 kB/s)
```

Mit **put [Datei] [Zieldatei]** kann ich dieselbe Datei mit einem anderen Namen hochladen. Mit dem teste ich zugleich die Schreiberechte.

```
ftp> put test.txt upload.txt
local: test.txt remote: upload.txt
227 Entering Passive Mode (192,168,1,117,188,129).
150 Ok to send data.
226 Transfer complete.
29 bytes sent in 0.00 secs (151.4455 kB/s)
```

Mit **bye** schliesse ich die Session.

TLS aktivieren

Da FTP sehr unsicher ist, werde ich den Verkehr via TLS/SSL verschlüsseln.

Das Zertifikat werde ich mit OpenSSL erstellen. Der Befehl lautet wie folgt;

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/vsftpd.pem -out /etc/ssl/private/vsftpd.pem

Folgende Infos muss ich jetzt nachtragen;

Country Code: **CH**

State: **Zurich**

Locality Name: **Zurich**

Organisation: **Vinylo**

Organisation Unit:

Common Name: **192.168.1.117**

Email: []

```
Country Name (2 letter code) [AU]:CH
State or Province Name (full name) [Some-State]:Zurich
Locality Name (eg, city) []:Zurich
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Vinylo
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:192.168.1.117
Email Address []:
```

Nachdem dieser Vorgang erfolgreich durchgegangen ist, öffne ich die Konfigurationsdatei mit **sudo nano /etc/vsftpd.conf** erneut

Folgende Änderung sind zu machen;

 # rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem - Kommentieren

 # rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key – Kommentieren

rsa_cert_file=/etc/ssl/private/vsftpd.pem – hinzufügen

rsa_private_key_file=/etc/ssl/private/vsftpd.pem – hinzufügen

ssl_enable=YES – HTTPS Datenverkehr zwingen

allow_anon_ssl=NO – Anonyme Anmeldungen unterbinden

force_local_data_ssl=YES

force_local_logins_ssl=YES

ssl_tlsv1=YES – TLS einschalten

ssl_sslv2=NO – SSL deaktivieren

ssl_sslv3=NO – SSL deaktiveren

require_ssl_reuse=NO

ssl_ciphers=HIGH – Cipher-Suites mit >128bit-Länge

Die Datei soll dann wie folgt aussehen

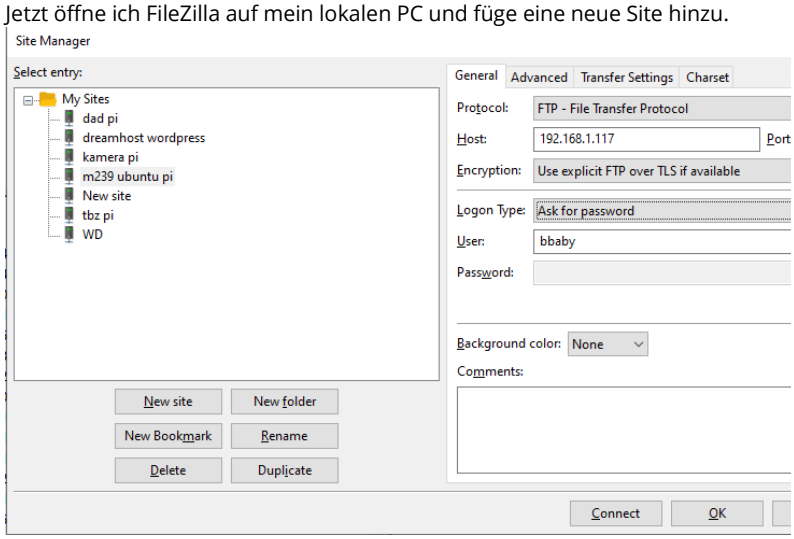
```
# This option specifies the location of the RSA certificate to use for SSL
# encrypted connections.
rsa_cert_file=/etc/ssl/private/vsftpd.pem
rsa_private_key_file=/etc/ssl/private/vsftpd.pem
ssl_enable=YES

#Disallow anon
allow_anon_ssl=NO
force_local_data_ssl=YES
force_local_logins_ssl=YES

#TLS einschalten
ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO

#Cipher hinzufügen
require_ssl_reuse=NO
ssl_ciphers=HIGH
```

Mit **sudo systemctl restart vsftpd** starte ich den FTP-Dienst neu. Nun kann ich testen, ob die Shellverbindung korrekterweise unterbinden wird. Dazu gebe ich **ftp -p 178.192.230.31** ein.

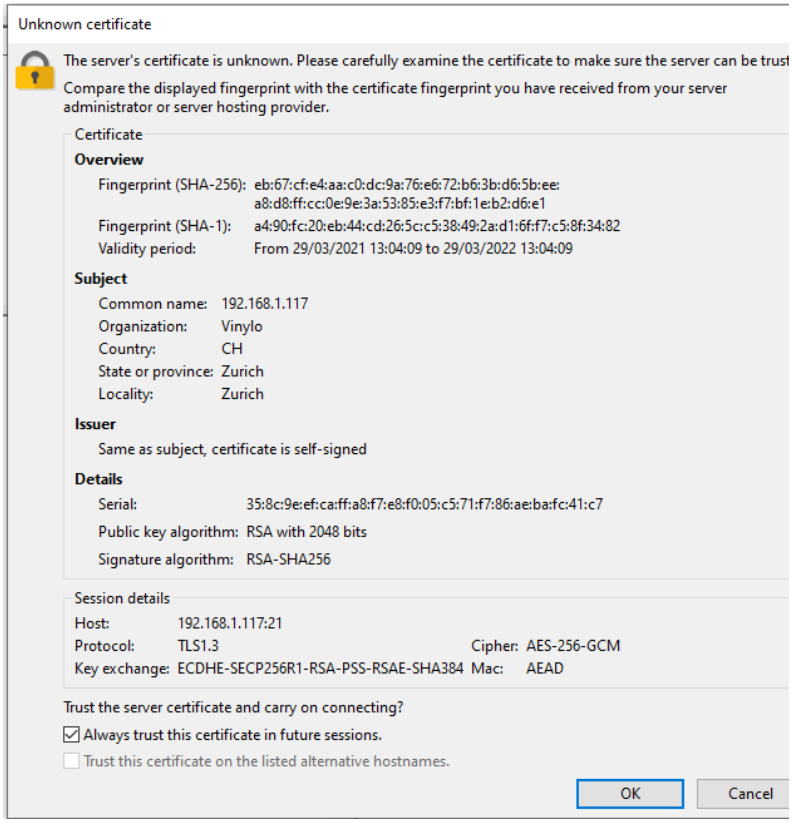
```
michalis@michalis-ubuntu:~$ ftp -p 192.168.1.117
Connected to 192.168.1.117.
220 (vsFTPd 3.0.3)
Name (192.168.1.117:michalis): bbaby
530 Non-anonymous sessions must use encryption.
Login failed.
421 Service not available, remote server has closed connection
ftp>
```

FileZilla TLS testen

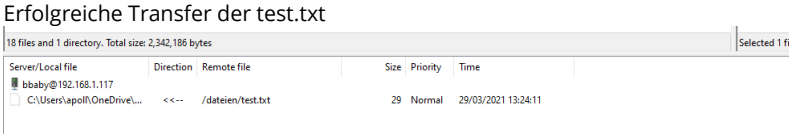
Jetzt öffne ich FileZilla auf mein lokalen PC und füge eine neue Site hinzu.



Nachdem ich auf Connect klicke, werde ich aufgefordert das PW für den User einzugeben. Nach erfolgreicher Eingabe werde ich mit dem vorhin erstellten Zertifikat konfrontiert, welches ich akzeptieren muss.



Erfolgreiche Transfer der test.txt



Postman

	monolithisch	Bezug zur Applikation, eine nicht-updateenthaltetene Applikation	
R	Registrar	Eine Domain-Registrar ist eine Hosting-Company, welche von der ICANN propagandiert wird	
	Repository/ies	Eine Repository ist eine Ansammlung von Dateiein, welches ein Programm für	Kap.

		die funktionelle Handhabung notwendig hat und für die Öffentlichkeit zum Herunterladen steht	
S	SNI	Stand	
W	WebDAV	WebDAV wird für das Herunterladen und Hochladen von Dateien im Web	
Z	Zeitreihe (time-series)	Eine Reihe von Datenpunkten, die in zeitlicher Reihenfolge indiziert/aufgelistet/grafisch dargestellt werden. Meistens eine Sequenz, die zu aufeinanderfolgenden, gleichmässig beanstandeten Zeitpunkten aufgenommen wurde. Sie ist also eine Folge von zeitdiskreten Daten.	Kap. funk. Prome

Tabelle 29: Glossar

10 Verzeichnisse

10.1 Quellenverzeichnis (L)

Quelle 1: Raspberry Pi auf Digitec 10
Quelle 2: Wikipedia URL-Encoding 15
Quelle 3: Mozilla Liste aller HTTP-Methoden 15
Quelle 4: Mozilla Liste aller HTTP Status-Codes 17
Quelle 5: Unterschied wget und curl 18
Quelle 6: POP3 im Detail 21
Quelle 7: IMAP im Detail 21
Quelle 8: SMTP im Detail 21
Quelle 9: Wie ist die NGINX Konfiguration aufgebaut 26
Quelle 10: Postman Webseite 29
Quelle 11: Prometheus Überblick 31
Quelle 12: HERMES Bundesrat 38
Quelle 13: Raspberry Pi Imager Tool 44
Quelle 14: DreamHost Domain-Status 45
Quelle 15: Docker Hub 93

10.2 Tabellenverzeichnis

Tabelle 1: Änderungsverzeichnis 2
Tabelle 2: Beispiel einer Tabelle 7
Tabelle 3: Modulidentifikation 8
Tabelle 4: Kontaktdaten der Firma 9
Tabelle 5: Dimensionierung 11
Tabelle 6: DNS-Ressource-Records 14
Tabelle 7: HTTP Methoden mit der Version 16
Tabelle 8: HTTP Versionen 16
Tabelle 9: HTTP Codes 17
Tabelle 10: Webserver Vergleich 25
Tabelle 11: Mailserver Vergleich 27
Tabelle 12: CMS Vergleich 27
Tabelle 13: Dateitransfer Vergleich 27
Tabelle 14: Prometheus im Vergleich zu anderen time-series DB Tools 32
Tabelle 15: Namenskonzept 36
Tabelle 16: Benutzerrechte 37
Tabelle 17: Gruppen 37
Tabelle 18: Berechtigungsmatrix 37
Tabelle 19: Stellen und Funktionen 39
Tabelle 20: Auflistung der Anforderungen mit Zuordnungen und Abdeckung 41
Tabelle 21: Risikoanalysetabelle 42

Tabelle 22: Risikomatrix 42
Tabelle 23: Mailcow verwendete Ports 70
Tabelle 24: Funktionstest Webserver 113
Tabelle 25: Funktionstest WordPress 113
Tabelle 26: Funktionstest Mailcow 113
Tabelle 27: Funktionstest FTP 114
Tabelle 28: Zielerreichung 115
Tabelle 29: Glossar 123

10.3 Abbildungsverzeichnis

Abbildung 1: Beispiel einer Abbildung 7
Abbildung 2: vinylo Logo 9
Abbildung 3: Team von vinylo 9
Abbildung 4: SOLL-Zustand der Dienste 10
Abbildung 5: Lastprofil der Webseite 11
Abbildung 6: Frage 11 des Tests 20
Abbildung 7: E-Mail-Verkehr 21
Abbildung 8: SSH Verbindungsaufbau 22
Abbildung 9: Asymmetrischer Schlüsselvorgang 23
Abbildung 10: Darstellung der Prometheus Komponente 31
Abbildung 11: Organigramm 38
Abbildung 12: VS Code Remote Dev Add-On 43
Abbildung 13: VS Code eingebautes Terminal 43
Abbildung 14: NGINX vinylo.conf File 87
Abbildung 15: Ganzes Docker-Compose File 94