POP ROCKET

Abschlussprüfung Frühjahr 2024
Fachinformatiker Systemintegration

Dokumentation zur betrieblichen Projektarbeit

Thema	Automatisierte Einrichtung eines Root Servers inklusive zweier Webapplikationen mittels Ansible
Prüfungsbewerber	Jan Ming Sulga Rapsweg 4 22549 Hamburg
Prüflingsnummer:	54185
Abgabedatum	13.12.2023
Ausbildungsbetrieb	Pop Rocket Labs GmbH Gastr. 14 22761 Hamburg
Ausbilder:	Timm Wimmers, Tel. 040 - 688 786 90

Inhaltsverzeichnis

1 Einleitung

Im Rahmen der Abschlussarbeit zum Fachinformatiker Fachrichtung Systemintegration wurde das folgende Projekt im Zeitraum vom 25.10.2023 bis zum 17.11.2023 durchgeführt und dokumentiert. Die beschriebene Lösung wurde speziell für die Systemadministration des Ausbildungsbetriebs entwickelt.

Die Firma Pop Rocket Labs GmbH ist eine Full-Service Digitalagentur, die sich auf innovative Lösungen in den Bereichen Digitalmarketing und Softwareentwicklung spezialisiert hat. Ihr Schwerpunkt liegt auf der Entwicklung von maßgeschneiderten Anwendungen, die den neuesten technologischen Trends entsprechen. Pop Rocket Labs zeichnet sich durch eine ausgeprägte Expertise in Gamification und Change Involvement aus.

2 Projektplanung

2.1 Ist-Analyse

In der Vergangenheit wurde das Hosting von Projekten auf mehreren leistungsfähigen Servern parallel laufend durchgeführt. Die Einrichtung wurde oft auf Basis vorheriger Projekte per Copy & Paste vorgenommen. Dieses Vorgehen ist fehleranfällig, wenig modular und bietet für die Auslieferung von Projekten wenig Raum für zeitgemäße Strategien.

Anpassungen fanden oft auf Zuruf und händisch direkt am System statt. Dadurch kann es ggf. zu Abweichungen bei der Dokumentation und den Repositories für die betroffenen Systeme kommen.

Beim Hosting auf unterschiedlichen Systemen (Environments) für Entwicklung, Qualitätssicherung oder Produktionssystem (`dev`, `stage`, `production`) kann es darüberhinaus zu Abweichungen kommen, die ggfs. neue und schwer zu identifizierende Fehler verursachten.

Die Handhabung von vertraulichen Information („Secrets“) wie z.B. SMTP-Zugangsdaten oder Access Tokens für API-Endpunkte, wurde zwar zentral geregelt, ließen sich bisher aber nur schwer innerhalb des Repositories verwenden ohne diese zu gefährden.

2.2 Soll-Konzept

Zukünftig sollen Server für dedizierte Projekte automatisiert, modular, konsistent und ohne direkten Eingriff eines Administrators auf dem Zielhost eingerichtet werden. Änderungen, Konfiguration und Auslieferung sollen ausschließlich auf Basis eines einzigen Repositories vorgenommen und mit Hilfe zeitgemäßer DevOps Strategien ausgeliefert werden.

Das Betriebssystem des Servers selbst, wird so minimal wie möglich gehalten und alle zusätzlichen Dienste sollen in Containern (hier Docker) gekapselt werden. Das verhindert fehlerhafte oder nicht kompatible Abhängigkeiten und vereinfacht die Wartung und Aktualisierung aller Komponenten. Die Wahl des Betriebssystem fiel aus strategischen Gründen auf Debian, dies sind z.B. die lange Verfügbarkeit von Sicherheitsupdates, die Bereitstellung eines minimalen Basissystems und die eher konservative Einführung neuer Features. Als Container Runtime

wurde Docker gewählt, weil hier betriebsintern das meiste Know How vorhanden ist und keine neuen Konzepte oder Philosophien erlernt werden müssen.

Die Definition und Verwaltung von Containern und deren Ressourcen wie Volumes (Datenpersistenz) oder Netzwerke (Datentransfer) wird mit Hilfe von Docker Compose realisiert.

Die Konfiguration der Container zur Laufzeit geschieht über separate Environment Dateien (Dotenv) und Docker Compose. Wenn Dotenv Dateien vertrauliche Informationen enthalten, müssen diese zwingend verschlüsselt im Repository abgelegt werden.

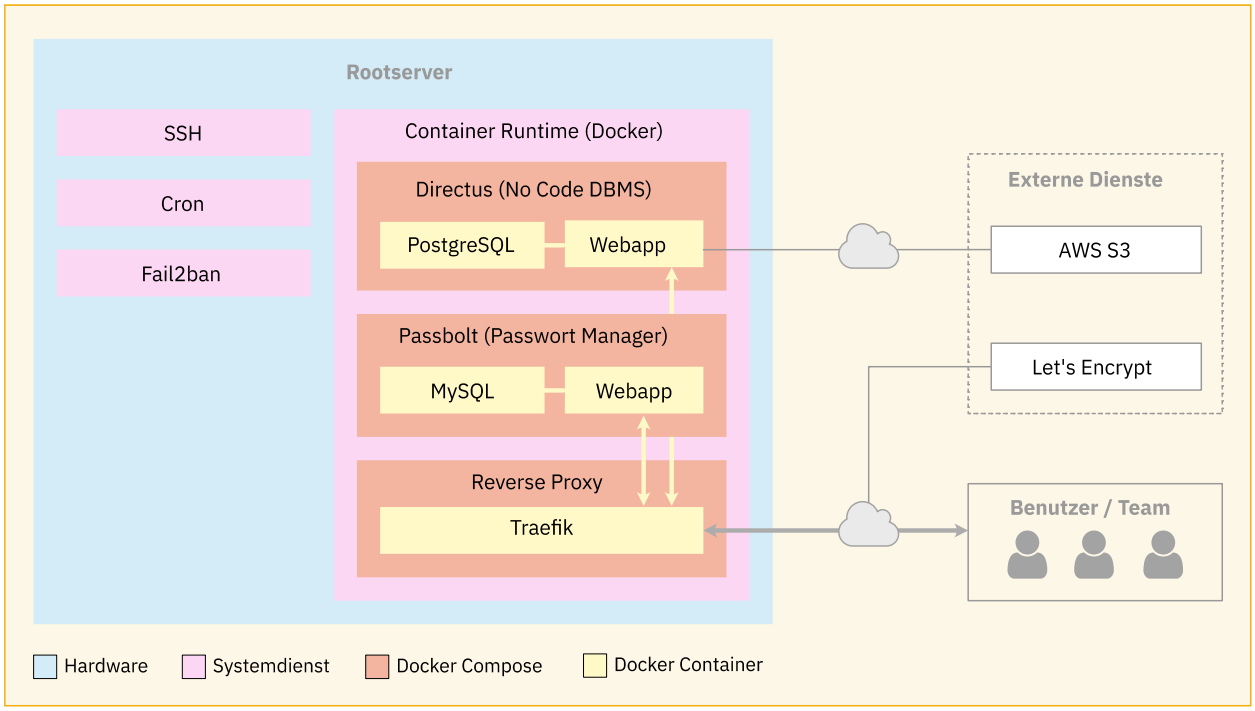

Abbildung 1: Projektüberblick

Dieses Projekt sieht zunächst vollständige und modulare Automatisierungen für die nachfolgend aufgeführten drei Bereiche (Module) vor und soll als Basis neuer Projekte deren Einrichtung vereinfachen und vor allem standardisieren:

1. **Prepare**
   Grundeineinrichtung des minimalen Betriebssystems und die weitere Konfiguration für die Verwendung von Ansible mit Hilfe eines unprivilegierten Verwaltungsbenutzers.

2. **Setup**
   Systemaktualisierung und Absicherung von SSH gegen Brut-Force-Angriffe, sowie die Installation der Docker Container Runtime Umgebung.

3. **Traefik**
   Reverse Proxy als HTTP und HTTPS Einstiegspunkt für das Routing auf Applikationsebene und TLS-Terminierung (früher SSL).

Bis hier hin kann das Projekt als saubere Basis für zukünftige Projekte, welche ein eigenes dediziertes Hosting auf einem Rootserver benötigen, betrachtet werden. Dies ist die Kernaufgabe dieser Projektarbeit und wird in der Kosten-Nutzen-Analyse (Abschnitt 4) berücksichtigt.

Darüber hinaus werden im Rahmen dieser Arbeit zwei weitere Webapplikationen installiert. Damit sollen Ablauf und Einrichtung von Multi-Container-Anwendungen, die Konfiguration von Regeln und Middleware des Revers Proxies sowie die Beschaffung valider Zertifikate umgesetzt werden:

4. **Directus**
   Installation einer „No Code“ Datenbankplattform.

5. **Passbolt**
   Webanwendung für die Verwaltung von Passwörtern im Team.

2.3 Werkzeuge und Anwendungen

2.3.1 Ansible

Ansible ist ein leistungsstarkes Open-Source-Automatisierungstool, das in der Welt der IT-Verwaltung und DevOps weit verbreitet ist. Ansible ermöglicht die Automatisierung von sich wiederholenden Aufgaben, wie z.B. Konfigurationsverwaltung, die Bereitstellung von Anwendungen auf unterschiedlichen Systemen und vieles mehr.

Mit Ansible definieren Systemadministratoren Aufgaben und Workflows (sogenannte Playbooks) mit Hilfe von YAML-Dateien (ein leicht zugänglicher und einfach strukturierter Mark-Up-Dialekt). Auf den Zielsystemen wird kein Agent benötigt. Ansible benutzt für die Ausführung der Playbooks auf den Zielsystemen vorhandene Standardwerkzeuge wie z.B. `su` oder `sudo`. Für die am häufigsten anfallenden Aufgaben bedient sich Ansible aus einer breiten Palette von integrierten und externen Community Modulen.

Darüberhinaus bringt Ansible ein Werkzeug (`ansible-vault`) mit, welches das verschlüsselte Erstellen, Betrachten und Editieren von sensiblen Daten vereinfacht und damit die Gefahr exponierter Daten im Repository reduziert.

2.3.2 Traefik

Traefik ist ein moderner Open Source Edge Router, der als Reverse Proxy und/oder als Loadbalancer eingerichtet werden kann. Als Reverse Proxy nimmt Traefik Anfragen von Clients auf Basis von Regeln entgegen, manipuliert ggf. durch eine Middleware die Requests und leitet diese an den gewünschten Service weiter. Als Loadbalancer kann Traefik den Datenverkehr auf mehrere gleiche Container verteilen. Dadurch kann die Last horizontal skalieren. Ebenfalls wird die Ausfallsicherheit erhöht, da für die anfallenden Requests mehrere Container zur Verfügung stehen. Traefik wird hier ohne Loadbalancing konfiguriert.

2.3.3 Directus

Directus ist ein „No Code“ Datenbank-Management-System (DBMS) und ermöglicht die Entwicklung mehr oder weniger komplexer Datenbanken durch den Anwender auch ohne erweiterte Programmierkenntnisse. Dadurch kann die Verwendung von verteilten und schwer zu pflegenden Tabellen im Unternehmen ersetzt werden.

2.3.4 Passbolt

Passbolt ist eine Open Source Passwortverwaltung speziell für Teams. Es ermöglicht die sichere Organisation von Passwörter im Browser. Dabei findet die Ver- und Entschlüsselung über eine Erweiterung im Browser statt. Dieses Vorgehen stellt sicher, dass auf dem Server selbst (innerhalb der Datenbank) und im Transit ausschließlich verschlüsselte Daten transportiert werden. Dadurch ist es unproblematisch, die Daten öffentlich zu hosten.

3 Realisierung

3.1 Voraussetzungen

Für dieses Projekt werden einige Ressourcen benötigt, welche seitens der hausinternen Systemadministration für diese Projekt zur Verfügung gestellt werden:

3.1.1 Rootserver

Ein Server, welcher bei einem Hoster oder Cloud-Provider auf Basis eines minimalen Linux (Debian 11) zur Verfügung gestellt wird. Diese VM ist über eine öffentliche IP-Adresse per Secure Shell (SSH) und unter Verwendung des Benutzers `root` und eines initialen Passwortes ansprechbar. Diese Art des Zugangs gilt allerdings nicht als Best Practice und eine der ersten Aufgaben wird sein, einen weniger privilegierten Verwaltungsbenutzer anzulegen und diesen dann für alle nachfolgenden Aufgaben zu verwenden.

3.1.2 Amazon AWS Simple Storage Service (S3)

Directus bietet neben der klassischen Verwaltung von Datensätzen (Tabellen) auch die Möglichkeit, größere Dokumente wie z.B. Bilddateien oder Dokumentenformate (PDF, DOCX o.ä.) zu verwalten - solche Dateien nennt man Binary Large Objects (BLOB) und werden traditionell auf einem Dateisystem gespeichert. Amazon AWS S3 ersetzt dieses Dateisystem durch ein Object Storage, ein sogenanntes Bucket, welches die Verwaltung von BLOBs durch einfache HTTP-Requests ermöglicht. Hierzu werden ein `ACCESS_KEY_ID` und ein `ACCESS_KEY_SECRET` benötigt. Directus unterstützt S3 standardmäßig.

3.1.3 Amazon AWS Simple Email Service (SES)

Directus und Passbolt versenden Transaktionsmails (Passwort Reset, Benachrichtigungen an Benutzer u.ä.), hierfür wird entweder ein lokaler Mailserver oder ein SMTP-Zugang auf einem entfernten Mailserver benötigt. AWS SES ist (vereinfacht gesagt) ein solcher entfernter E-Mailserver. Je nach Nutzung, wird hier entweder ein SMTP-Benutzer inkl. Passwort, oder ein `ACCESS_KEY_ID` und `ACCESS_KEY_SECRET` benötigt.

3.1.4 DNS-Einträge

Die hier dargestellten Services werden unter der Subdomain `mars.poprocket.com` gekapselt. Um die Arbeit mit dem System zu vereinfachen (und im späteren Verlauf auch für die Zertifikatsverwaltung), werden für die IP-Adresse des Rootserver ein A-Record, sowie für die verschiedenen Services entsprechende CNAME-Records in der DNS-Zone `poprocket.com` benötigt:

```
A-RECORD     mars.poprocket.com    ${ROOTSERVER_IP_ADDRESS}
CNAME        proxy.mars            mars.poprocket.com
CNAME        pass.mars             mars.poprocket.com
CNAME        cms.mars              mars.poprocket.com
```

3.2 Konfiguration des lokalen Repository

Zu Beginn werden drei Dotfiles angelegt, welche die Arbeit mit dem Repository vereinfachen, bzw. sicherstellen, dass bestimmte Dateien nicht in die Versionskontrolle übernommen werden:

1. `.envrc`
   Diese Datei erlaubt es der Shell, beim Wechsel in oder aus einem Verzeichnis Kommandos auszuführen. Hier wird festgestellt ob die Datei `.avp` existiert und die Umgebungsvariable `ANSIBLE_VAULT_PASSWORD_FILE` entsprechend gesetzt und im Anschluss exportiert. Dadurch wird die Verschlüsselung, das Betrachten und das Bearbeiten von Dateien mit sensiblen Daten per `ansible-vault` Kommandos vereinfacht.

   Der Inhalt der Datei besteht aus einer einzigen Zeile:

   ```
   test -f .avp && export ANSIBLE_VAULT_PASSWORD_FILE="${PWD}/.avp"
   ```

2. `.avp`
   Akronym für `ANSIBLE_VAULT_PASSWORD`. Enthält das Secret mit dem das Kommando `ansible-vault` Dateien ver- und entschlüsseln kann. Muss zwingend von der Versionskontrolle ausgeschlossen werden (und muss bei allen beteiligten Entwicklern den gleichen Inhalt haben).

3. `.gitignore`
   Teil der Versionskontrolle (Git). Konfiguriert, welche Dateien nicht nachverfolgt werden.

3.3 Anlegen der Projektverzeichnisse

Die Ansible-Playbooks werden gemäß Konvention in einem Verzeichnis `Ansible` abgelegt und konfiguriert. Reverse Proxy (Traefik), Directus und Passbold erhalten jeweils ein eigenes Projektverzeichnis außerhalb von `Ansible` um eine saubere Modularisierung zu gewährleisten:

```
$ mkdir {Ansible,traefik,directus,passbolt}
```

Verzeichnisbaum

```
.
├── Ansible
├── directus
├── passbolt
└── traefik
```

3.4 Docker Compose Projekte

Docker Compose definiert Ressourcen wie Volumes, Netzwerke und Service-Container in einer YAML-Datei (`docker-compose.yml`, Compose Datei). Diese Datei kann mit dem Kommando `docker compose up -d` innerhalb des Projektverzeichnisses (Docker Context) gestartet werden. Dabei werden die einzelnen definierten Dienste als Systemservices auf dem Zielsystem gestartet. Die Compose Datei kann durch Umgebungsvariablen konfiguriert werden, `docker` liest dazu die Datei `.env` automatisch ein und nutzt die dort definierten Key Value Paare an den vorgesehen Stellen in der Compose Datei.

In den einzelnen Projektverzeichnissen (`traefik`, `directus`, `passbolt`) liegt daher jeweils mindestens eine `docker-compose.yml` und eine `.env.compose`. Letztere wird lokal ggf. verschlüsselt und später durch Ansible während des Deployments entschlüsselt und im Projektverzeichnis auf dem Zielsystem als `.env` abgelegt. Ein entsprechender Ansible-Task für diese beiden Dateien sieht beispielsweise wie folgt aus (Auszug aus dem Playbook):

```
    - name: Deploy Docker Compose file
      copy:
        src: "../{{ PROJECT_NAME }}/docker-compose.yml"
        dest: "{{ CONTAINER_ROOT }}/{{ PROJECT_NAME }}/docker-compose.yml"
    - name: Deploy Docker Compose configuration
      copy:
        src: "../{{ PROJEKT_NAME }}/.env.compose"
        dest: "{{ CONTAINER_ROOT }}/{{ PROJEKT_NAME }}/.env"
        decrypt: true
```

Der hier gezeigte Block kann für jedes Teilprojekt verwendet werden und zeigt gut, wie die Abstrahierung mit ausgelagerten Variablen umgesetzt werden kann. Im Abschnitt 3.5 wird detailliert auf die verwendeten Ansible Playbooks eingegangen.

3.4.1 Reverse Proxy (Traefik)

Traefik als Reverse Proxy benötigt neben der Environment- und Compose Datei noch weitere Konfigurationen welche als Volumes in den Container gemountet werden. Bei der Ausführung des Playbooks muss unbedingt darauf geachtet werden, dass diese Dateien und Verzeichnisse mit den richtigen Rechten angelegt werden. Der Zugriff für unprivilegierte Benutzer muss eingeschränkt sein.

1. **Statische Konfiguration**
   Mit Hilfe der Datei `traefik.yml` (siehe Anhang) werden die Standard- und systemnahen Einstellungen konfiguriert. Insbesondere wird hier bereits standardmäßig der Einstiegspunkt `web` (normales HTTP, 80) für alle Requests an `websec` (HTTPS, 443) weitergeleitet. Dadurch muss dies nicht mehr bei den einzelnen Services berücksichtigt

werden. Die Konfiguration wird beim Starten des Containers eingelesen und kann nur geändert werden, indem Traefik bzw. der Container neu gestartet wird.

Darüberhinaus werden hier die Anbindung an den Docker Service des Hosts, die Einstiegspunkte des Reverse Proxies, der Speicherort und die Provider für die Zertifikate konfiguriert. Siehe Anhang 7.1.1: Traefik statische Konfiguration.

Die statische Konfiguration wird auf dem (Docker-) Host im Projektverzeichnis `/opt/containers/traefik/traefik.yml` abgelegt und im Dateisystem des Containers als `/traefik.yml` eingebunden (Bind-Mount).

2. **Dynamische Konfigurationen**
   Auf dem Host wird das Verzeichnis `/etc/traefik/conf.d` als dynamisches Konfigurationsverzeichnis in den Container eingebunden. Dort werden für die Absicherung des Traefik Dashboards und ggf. für andere Services die Middleware `protected-auth` und `protected-ips` durch ein Ansible Playbook angelegt. Container, die diese Middleware bei ihrer Traefik-Konfiguration einbinden, können so den Zugriff für unbekannte Benutzer und/oder unbekannte IP-Adressen verhindern.

3. **Storage für Zertifikate**
   Auf dem Host wird das Verzeichnis `/etc/traefik/acme.d` als Zertifikatsspeicher eingebunden. Im Container des Reverse Proxies steht dieses Verzeichnis unter `/acme.d` zur Verfügung. Traefik speichert zur Laufzeit in der Datei `/acme.d/crt_store.json` die ausgestellten Zertifikate ab. Daher müssen hier die Lese- und Schreibrechte durch das Playbook bei der Erstellung des Verzeichnisses auf den Benutzer `root` eingeschränkt werden. Siehe Anhang 7.5.3 (`pb3_traefik.yml`).

Auf die Funktionsweise und das Zusammenspiel von Environment- und Compose YAML-Dateien wird im nachfolgendem Abschnitt 3.4.2 detaillierter eingegangen.

3.4.2 Directus

Directus benötigt einen Speicher um Assets wie z.B. Bilder, Dokumente usw. abzulegen (siehe 3.1.2), sowie einen SMTP-Zugang für Transaktionsemails (siehe 3.1.3). Diese Zugangsdaten sind vertraulich und dürfen im Repository daher nur verschlüsselt vorliegen. Da Ansible die Ver- und Entschlüsselung transparent regelt, legen wir mit dem Kommando `ansible-vault create $REPOSITORY/directus/.env.compose` eine verschlüsselte Datei an in der die Umgebungsvariablen als Schlüsselwertpaare für Docker Compose mit Hilfe von `ansible-vault edit $REPOSITORY/directus/.env.compose` angelegt werden. Das Passwort für die Ver- und Entschlüsselung wird wie in Abschnitt 3.2 beschrieben automatisch eingelesen.

In der Datei `${REPOSITORY}/directus/docker-compose.yml` werden die passenden Schlüsselwertpaare referenziert und beim Starten der Dienste auf dem Zielhost innerhalb der Container entsprechend gesetzt.

Docker Compose legt bei Bedarf für das Projekt (auch Docker Kontext genannt) ein eigenes virtuelles Netzwerk an (hier mit Namen `stack`). Mit diesem Netzwerk erreichen sich die Container untereinander unter dem Namen des Service (innerhalb des Kontextes).

Die PostgreSQL URL `postgres://directus:P4sSw0rD@database:5432` würde beispielsweise wie erwartet innerhalb des Kontext eine Verbindung als Benutzer `directus` erfolgreich zum Container `database` aufbauen können.

In der `docker-compose.yml` werden drei Dienste unterhalb der Sektion `services` definiert: `database`, `redis` und `directus`. Für die Datenbank und Redis werden noch zwei Docker Volumes definiert um die Datenpersistenz nach einem Neustart der Container zu gewährleisten.

Der Service Directus soll nicht nur innerhalb des Containers mit den anderen Diensten kommunizieren können, sondern auch über den Reverse Proxy unter einer URL von außen ansprechbar sein. Dazu wird im Kontext ein externes Docker-Netzwerk eingebunden welches die Anbindung an den Reverse Proxy (Traefik) gewährleistet. Dieses externe Netzwerk wird im Ansible Playbook `pb3_traefik.yml` angelegt und konfiguriert (siehe Abschnitte 3.5.4).

Die Konfiguration des Containers für Traefik findet im Compose File über Docker Labels statt. Hier werden die Regeln für den (Traefik-) Entrypoint, Domainnamen (FQDN), Zertifikatsanbieter (Provider) und ggfs. Middlewares definiert oder referenziert. Traefik kümmert sich dabei vollautomatisch um die Beschaffung eines gültigen Zertifikates bei dem konfiguriertem Zertifikatsprovider. Siehe statische Konfiguration von Traefik im Anhang 7.1.1.

Alle Umgebungsvariablen werden in der `docker-compose.yml` als Umgebungsvariable nach dem Muster `${ENV_KEY}` referenziert. Beispielsweise die Labels wie folgt:

```
    labels:
      - "traefik.enable=true“
      - "traefik.http.routers.${_ID_}.entrypoints=websec"
      - "traefik.http.routers.${_ID_}.rule=Host(`${FQDN}`)"
      - "traefik.http.routers.${_ID_}.tls=true"
      - "traefik.http.routers.${_ID_}.tls.certresolver=le-http-01"
      - "traefik.http.routers.${_ID_}.service=${_ID_}"
      - "traefik.http.services.${_ID_}.loadbalancer.server.port=${DIRECTUS_PORT}"
```

Die vollständige Compose Datei befindet sich im Anhang (Abschnitt 7.2.1)

3.4.3 Passbolt

Diese Webanwendung ist ähnlich wie Directus im Abschnitt zuvor aufgebaut, mit dem Unterschied, dass es hier zwei Environment Dateien angelegt wurden. Eine dieser Dateien ist im Repository verschlüsselt und wird bei der Installation mit Ansible während des Auslieferns (Deployment) „on the fly“ entschlüsselt und auf dem Zielsystem an der richtigen Stelle abgelegt. Siehe auch das Playbook im Anhang 3.5.6.

3.5 Ansible Playbooks

3.5.1 Konfiguration und Abstraktion

- Die Datei `ansible.cfg` (Anhang 7.4.1) ist die zentrale Konfigurationsdatei für Ansible. In dieser Datei werden Einstellungen festgelegt, die Ansible bei der Ausführung von Aufgaben auf Zielsystemen steuert. Wichtige Aspekte sind Optionen wie Standardwerte für Module, die Verbindung zu den Zielsystemen und der Name des Inventories.

Durch die Option `inventory = hosts.ini` wird der Pfad zur Inventardatei festgelegt. Das Inventar dient dazu, die Zielsysteme zu definieren, auf denen Ansible Aufgaben ausführen soll. Die Wahl einer spezifischen Inventardatei ermöglicht es, gezielt auf bestimmte Systeme oder Gruppen zuzugreifen.

Die Option `remote_user = toor` legt den Standard-Benutzernamen fest, der bei der Verbindung zum Zielsystemen verwendet wird. In diesem Fall wird `toor` als Remote-Benutzer festgelegt. Dieser Benutzer wird bei SSH-Verbindungen und der Ausführung von Aufgaben auf den Zielsystemen genutzt. Dieser Benutzer kann ggf. mit sudo privilegierte Änderungen am System vornehmen.

Mit der Option `interpreter_python = /usr/bin/python3` wird der Pfad zum Python-Interpreter auf dem Zielsystem explizit angegeben, um den versehentlichen Aufruf einer veralteten Python Version 2 zu verhindern.

- Die Datei `hosts.ini` (Anhang 7.4.2) ist ein wichtiger Bestandteil eines Ansible-Projekts und bietet eine übersichtliche Möglichkeit, Zielsysteme einzeln oder in Gruppen zu organisieren. In diesem Fall ist die Datei sehr einfach aufgebaut, da nur ein Zielhost angesprochen wird.
- Die Datei `config.yml` (Anhang 7.4.3) legt die projektspezifischen Variablen fest. Dadurch bleiben die verwendeten Playbooks abstrakt und können für Folgeprojekte verwendet werden.

3.5.2 Vorbereitung des Basissystems

Mit diesem Ansible-Playbook (Anhang 7.5.1, `pb1_prepare.yml`) wird das minimale Debian System konfiguriert. Dieses Playbook führt auf dem Zielhost die Kommandos als `root` aus (`become: true`).

- **Installation von `sudo`**

  Zuerst wird das Playbook aufgerufen, um sicherzustellen, dass das System das Paket `sudo` installiert hat. Dies ermöglichte die Ausführung von Befehlen mit erhöhten Berechtigungen für Mitglieder der Gruppe `sudo`.

- **Konfiguration des Hostnamen**

  Als nächstes wird der Hostname des Systems in `mars` geändert. Dieser Schritt kann je nach den Anforderungen des Projekts über die zuvor beschrieben `config.yml` angepasst werden.

- **Einlesen von Umgebungsvariablen**

  Das Playbook liest Umgebungsvariablen aus der Datei `config.yml` ein. Dadurch werden die einzelnen Playbooks abstrakt gehalten und vollständig parametrisiert.

- **Erstellung und Konfiguration des Verwaltungsbenutzers**

  Hier wird ein neuer Benutzer mit dem Namen `toor` erstellt und der Gruppe `sudo` hinzugefügt, um ggf. privilegierte Kommandos auszuführen. Es wird eine Standard-Shell für

den Benutzer festgelegt. Das Passwort des Benutzer wird als gehashter Wert für die Datei `/etc/passwd` aus der `config.yml` gelesen.

- **Anpassung der Namensauflösung**

  Die IPv4-Adresse des Servers wird auf den korrekten Hostnamen gesetzt. Dazu wird in der Datei `/etc/hosts` auf dem Zielsystem durch Suchen und Ersetzen in der entsprechende Zeile der Standardwert des Hosting-Anbieters angepasst.

3.5.3 Aktualisierung, Fail2Ban und Container Runtime

Die Aufgabe dieses Ansible-Playbooks (Anhang 7.5.2, `pb2_setup.yml`) ist das Aktualisieren des Systems, das Einrichten von Fail2Ban und die Installation von Docker als Container Runtime. Dieses Playbook führt auf dem Zielsystem die Kommandos als `root` aus (`become: true`).

- **Systemaktualisierung**

  Das Playbook aktualisiert den Paketkatalog und ggf. werden eventuell veraltete Pakete des Betriebssystems ebenfalls aktualisiert.

- **Fail2Ban**

  Anschließend wird Fail2Ban installiert um den Rechner vor Brute-Force-Angriffen per SSH zu schützen.

- **Docker-Installation**

  Damit Docker aus den aktuelleren Paketquellen des Herstellers installiert werden kann, muss das Repository des Herstellers als Paketquelle hinzugefügt werden. Dafür werden zunächst die CA-Zertifikate aktualisiert und im Anschluss `curl` und `gnupg` entweder aktualisiert oder nachinstalliert.

  Anschließend wird der GPG-Schlüssel für die Docker-Repository-Signatur heruntergeladen und in ein passendes Format konvertiert. Darüber hinaus wird die Architektur und die Debian-Version des Systems ermittelt, um die Docker-Repository-Konfiguration zu generieren. Abschließend wird Docker inkl. Erweiterungen (z.B. Docker Compose) aus den Quellen des Herstellers (Docker) installiert.

- **Docker-Benutzergruppe**

  Dieser Ansible-Task fügt den Verwaltungsbenutzer `toor` zur Docker-Benutzergruppe hinzu, damit Docker Kommandos ohne die Notwendigkeit von `sudo` verwendet werden können.

3.5.4 Traefik

Mit diesem Playbook (Anhang 3.5.4, `pb3_traefik.yml`) wird der Reverse Proxy Traefik auf dem Zielsystem eingerichtet. Hier gibt es die Besonderheit, dass im Verzeichnis `/etc/traefik` auf dem Zielsystem die dynamische Konfiguration und der Zertifikatsspeicher abgelegt werden. Insbesondere der Zertifikatsspeicher `/etc/traefik/acme.d` muss gegenüber un-

privilegierten Benutzern eingeschränkt werden. Dieses Playbook führt auf `mars` die Kommandos als `root` aus (`become: true`).

- **Erstellung Docker Network**

  Es wird ein Docker-Network namens `proxy` erstellt. Alle Container die über den Reverse Proxy angesteuert werden sollen, müssen ein Node dieses Netzwerkes sein – dies wird in der jeweiligen `docker-compose.yml` wie folgt festgelegt:

  ```
  networks:
    proxy:
      external: true
  ```

- **Erstellung des Konfigurationsverzeichnisses**

  Mit diesem Task wird das Verzeichnis `/etc/traefik` auf dem Zielsystem vorbereitet.

- **Anlegen des Verzeichnises für die dynamische Konfiguration**

  Anlegen des Verzeichnis `/etc/traefik/conf.d` auf dem Zielhost. Wird später durch die `docker-compose.yml` in den Container als `/conf.d` eingebunden.

- **Erstellen des Verzeichnisses für den Zertfikatsspeichers**

  Der Task legt das Verzeichnis `/etc/traefik/acme.d` auf dem Zielhost an. Wird später durch `docker-compose.yml` in den Container als `/acme.d` eingebunden. Traefik legt dort in der Datei `crt_store.json` die TLS-Zertifikate der jeweiligen Services ab.

- **Traefik BasicAuth**

  In diesem Task wird die Middleware (Anhang 7.1.2) für den Reverse Proxy im Verzeichnis für die dynamische Konfiguration des Zielsystems ausgeliefert.

- **Traefik IP-Passlist**

  Dieser Task überträgt die Middleware (Anhang 7.1.3) für den Reverse Proxy in das Verzeichnis für die dynamische Konfiguration.

- **Starten des Docker Compose Stacks**

  Der Reverse Proxy wird für alle Folgedienste zwingend benötigt und wird direkt gestartet. Dem Kommando `docker compose` wird dazu der Kontext des Stacks mit dem Parameter `-f <file>` übergeben. Die Option `--force-recreate` sorgt dafür, dass ggf. angepasste Variablen des Environments neu eingelesen werden (ein ggf. bereits laufende Container wird dafür gelöscht und im Anschluss direkt neu erstellt und gestartet).

3.5.5 Directus

Dieses Playbook (Anhang 3.5.5) richtet die Services für Directus (siehe 2.3.3) ein und muss nicht mehr als Benutzer root ausgeführt werden.

- **Erstellen des Projektverzeichnisses**
- **Übertragen des Docker Compose Files**

- **Entschlüsseln und übertragen der Directus Konfiguration**
- **Starten des Docker Compose Stack**

3.5.6 Passbolt

Das Playbook (Anhang 3.5.6) richtet analog zu Directus die Dienste für Passbolt (Details 2.3.4) ein. Auch diese Tasks müssen nicht als `root` ausgeführt werden.

- **Erstellen des Projektverzeichnisses**
- **Übertragen des Docker Compose Files**
- **Übertragen des Docker Compose Environment**
- **Entschlüsseln und übertragen der Passbold Konfiguration**
- **Starten des Docker Compose Stack**

3.6 Deployment

Das Abspielen der Playbooks erfolgt hier auf dem lokalen System des Administrators. Da das Projekt in einem Git-Repository vorliegt, kann es aber auch auf einem dedizierten System für die Ausführung von Ansible Playbooks erfolgen.

Gemäß Konvention wird im Projekt eine Shell aufgerufen und in das Verzeichnis `Ansible` gewechselt. Im Anschluss werden die einzelnen Playbooks in der richtigen Reihenfolge (`pb1 – pb5`) erstmalig ausgeführt. Spätere Änderungen an einzelnen Komponenten werden dann mit dem einzelnen Abspielen des betroffenen Playbooks durchgeführt.

Das erste Playbook erfordert die Angabe des Passwortes für den Benutzer `root`, die zwei nachfolgenden Playbooks das Passwort des Verwaltungsbenutzers `toor` und alle weiteren benötigen keine privilegierte Ausführung der Ansible Tasks.

```
$ cd Ansible
$ ansible-playbook pb1_prepare.yml –ask-become-pass
$ ansible-playbook pb2_setup.yml –ask-become-pass
$ ansible-playbook pb3_traefik.yml –ask-become-pass
$ ansible-playbook pb4_directus.yml
$ ansible-playbook pb5_passbolt.yml
```

Bei fehlerfreier Ausführung ist das Zielsystem erwartungsgemäß konfiguriert und das Dashboard von *Traefik*, die Webanwendungen *Directus* und *Passbolt* stehen unter den konfigurierten URLs zur Verfügung (siehe Abschnitt 5, Funktionstest).

4 Kosten-Nutzen-Analyse

4.1 Zusammenfassung

Die automatisierte Einrichtung eines Servers mittels Ansible bietet, im Vergleich zur manuellen Konfiguration durch einen Systemadministrator, erhebliche Vorteile in Bezug auf Zeitersparnis.

Die internen Kosten eines Systemadministrators können mit 30,00 bis 50,00 Euro oder höher angesetzt werden (Junior, Senior Administrator). Bei der manuellen Einrichtung und Konzeption eines dedizierten Servers können je nach Komplexität und Umfang mehrere Stunden bis Tage anfallen.

Im Gegensatz dazu ermöglicht die Nutzung von Ansible eine schnelle und insbesondere standardisierte Bereitstellung der Server. Die einmalige Erstellung der Playbooks erfordert anfänglich zwar mehr Entwicklungszeit, jedoch wird die Entwicklung weniger projektbezogen betrachtet und führt dadurch zwangsläufig zu höherer Standardisierung bei wiederholter Bereitstellung von dedizierten Servern.

Durch die fortlaufende Automatisierung wird für das Projektmanagement darüber hinaus eine stabilere Kalkulationssicherheit sichergestellt.

Die nachfolgende Gegenüberstellung zeigt deutlich, dass die Zeit als auch die Kosten für interne Projekte auf ein Viertel, für einfache Kundenprojekte auf ein gutes Drittel und für komplexere Kundenprojekte auf ein knappes Drittel gesenkt werden können.

4.2 Gegenüberstellung

Tabelle 1: Manuelle Bereitstellungen p.A.

Projektart	Anzahl	Interner Stundensatz	Zeitaufwand	Kosten in €
Intern	3	40,00 €	8,0 Std.	960,00 €
Kunde einfach	7	40,00 €	12,0 Std.	3360,00 €
Kunde komplex	3	40,00 €	24,0 Std.	2880,00 €

Tabelle 2: Bereitstellungen via Ansible p.A.

Projektart	Anzahl	Interner Stundensatz	Zeitaufwand	Kosten in €
Intern	3	40,00 €	2,0 Std.	240,00 €
Kunde einfach	7	40,00 €	3,5 Std.	980,00 €
Kunde komplex	3	40,00 €	8,0 Std.	960,00 €

5 Funktionstest

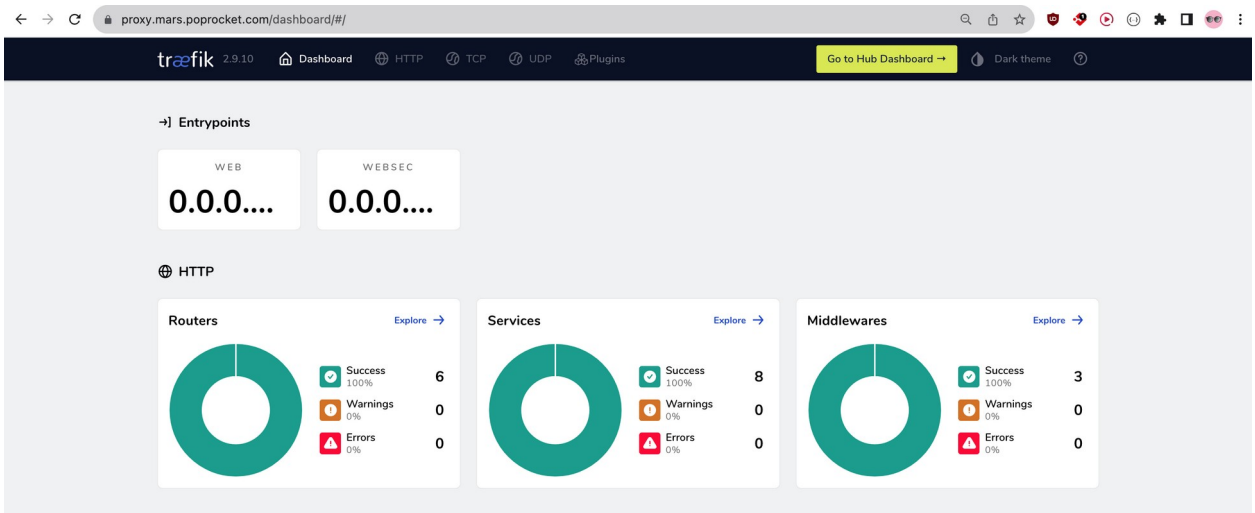

Abbildung 2: Traefik Dashboard

Die ordnungsgemäße Installation bzw. Bereitstellung der hier beschriebenen Services und Anwendungen, kann am besten überprüft werden, wenn das Dashboard des Reverse Proxies auf-

gerufen wird (https://proxy.mars.poprocket.com). Dies kann ausschließlich von den erlaubten IP-Adressen der Middleware `protected-ips` (siehe Anhang 7.1.3) aufgerufen werden. Es werden weiterhin die BasicAuth Zugangsdaten wie sie in der Middleware `protected-auth` (Anhang 7.1.2) konfiguriert wurden. Das Bildschirmfoto zeigt keine Warnungen oder Fehler. Die Anzahl der konfigurierten Router, Services und Middlewares entspricht den Erwartungen.

Im nachfolgenden Bildschirmfoto werden die Zertifikatsdetails für die URL des Proxies angezeigt. Auch hier entspricht das Ergebnis den Erwartungen, Zertifikate wurden also erfolgreich vom Zertifikatsprovider ausgestellt.

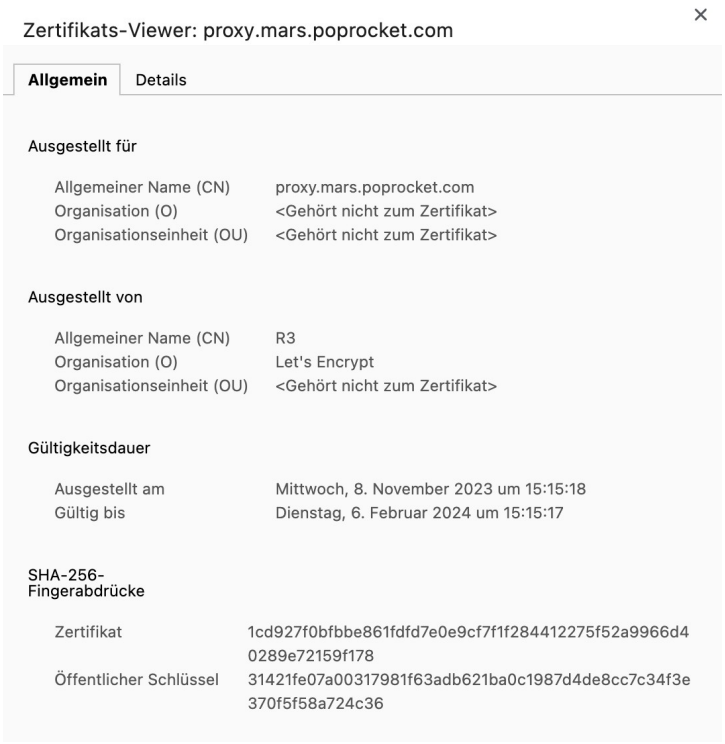

Abbildung 3: Browser Zertifikatdetails

Im Anhang 7.6 zeigen weitere Bildschirmfotos zu den Webanwendungen die Anmeldung von Directus und den Start des Onboarding-Prozesses von Passbolt.

6 Schlussbemerkung

6.1 Referenzen

Als Referenzen wurde hauptsächlich auf die Dokumentation der verwendeten Technologien und Anwendungen zurückgegriffen.

- **Ansible** https://docs.ansible.com
- **Traefik** https://doc.traefik.io/traefik
- **Docker** https://docs.docker.com/reference
- **Debian** https://www.debian.org/doc
- **Directus** https://docs.directus.io
- **Passbolt** https://help.passbolt.com/hosting/install/ce/docker.html
- **Fail2Ban** https://fail2ban.readthedocs.io/en/latest

6.2 Dokument- und Dateivorlagen

Für dieses Projekt wurde eine hausinterne Dokumentvorlage für Libre Office verwendet. Die Docker Compose YAML-Dateien basieren ebenfalls zum Teil auf hausinternen Vorlagen oder früheren Projekten, wurden im Rahmen dieser Projektarbeit jedoch umfassend standardisiert.

Die Verwendung von Ansible, das Erstellen der Ansible Playbooks und deren Abstrahierung wurde hier vollständig neu konzipiert und entwickelt. Ansible selbst fand bisher betriebsintern keine Anwendung, daher wird diese Projektarbeit im Ausbildungsbetrieb als Startpunkt für die weitere Automatisierung angesehen. Es ist jedoch auch klar, dass dafür die hier entwickelten Playbooks weiter vorangetrieben werden müssen. Zum Beispiel müssen dann auch Themen wie Monitoring, Datensicherung oder Ausfallsicherheit berücksichtigt werden.

6.3 Hinweis zu verschlüsselten Inhalten im Anhang

Wegen der vertraulichen Natur einiger Dotenv-Dateien, werden im Anhang verschlüsselte Docker Compose Environments, oder Service Konfigurationen nur konzeptionell und ohne Darstellung des Inhalts aufgeführt.

7 Anhang

7.1 Traefik

7.1.1 Statische Konfiguration

Repo: ./traefik/traefik.yml

Host: /opt/containers/traefik/traefik.yml

```
global:
  checkNewVersion: false
  sendAnonymousUsage: false
api:
  debug: false
  dashboard: true
log:
  level: INFO
entryPoints:
  web:
    address: '0.0.0.0:80'
    http:
      redirections:
        entryPoint:
          to: websec
          scheme: https
  websec:
    address: '0.0.0.0:443'
providers:
  docker:
    endpoint: "unix:///var/run/docker.sock"
    network: proxy
    exposedByDefault: false
  file:
    directory: /conf.d
    watch: true
certificatesResolvers:
  le-http-01:
    acme:
      email: dns@poprocket.com
      storage: /acme.d/crt_store.json
      # Uncomment to use Let's Encrypt's staging server, leave commented for production
      #caServer: "https://acme-staging-v02.api.letsencrypt.org/directory"
      httpChallenge:
        entryPoint: web
```

7.1.2 Dynamische Middleware BasicAuth

Repo: ./traefik/protected-auth.yml

Host: /etc/traefik/conf.d/protected-auth.yml

```
http:
  middlewares:
    protected-auth:
      basicAuth:
        realm: "Restricted Access"
        headerField: "X-WebAuth-User"
        users:
          - 'toor:--REDACTED PASSWORD HASH --'
```

7.1.3 Dynamisch Middleware IP-Passlist

```
Repo: ./traefik/protected-ips.yml
```

```
Host: /etc/traefik/conf.d/protected-ips.yml
```

```
http:
  middlewares:
    protected-ips:
      ipWhiteList:
        sourceRange
          - "10.10.10.100/29“     # Office Pop Rocket Labs (-- anonymisiert --)
          - "10.10.20.220"        # VPN-HO Pop Rocket Labs (-- anonymisiert --)
```

7.1.4 Docker Compose File

```
Repo: ./traefik/docker-compose.yml
```

```
Host: /opt/containers/traefik/docker-compose.yml
```

```
services:
  traefik:
    image: traefik:${TRAEFIK_VERSION:-2.9}
    container_name: ${_TS_}
    restart: always
    networks:
      - proxy
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - ./traefik.yml:/traefik.yaml:ro      # boot, init, static
      - /etc/traefik/conf.d:/conf.d:ro      # runtime, dynamic
      - /etc/traefik/acme.d:/acme.d:rw      # certificate storage
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.${_TS_}.entrypoints=websec"
      - "traefik.http.routers.${_TS_}.rule=Host(`${FQDN}`)"
      - "traefik.http.routers.${_TS_}.tls=true"
      - "traefik.http.routers.${_TS_}.tls.certresolver=le-http-01"
      - "traefik.http.routers.${_TS_}.service=api@internal"
      - "traefik.http.routers.${_TS_}.middlewares=protected-auth@file,protected-ips@file"
networks:
  proxy:
    external: true
```

7.1.5 Docker Compose Environment

```
Repo: ./traefik/.env.compose
```

```
Host: /opt/containers/traefik/.env
```

```
_TS_=traefik
TRAEFIK_VERSION=2.9
FQDN=proxy.mars.poprocket.com
```

7.2 Directus

7.2.1 Docker Compose File

Repo: ./directus/docker-compose.yml

Host: /opt/containers/directus/docker-compose.yml

```
name: "directus-cms"
networks:
  stack:
    internal: true
  proxy:
    external: true
volumes:
  postgresql:
  redisdumps:
services:
  database:
    container_name: ${_ID_}-database
    hostname: ${_ID_}-datebase
    image: postgis/postgis:${POSTGIS_IMAGE_VERSION}
    restart: always
    user: postgres:postgres
    volumes:
      - postgresql:/var/lib/postgresql/data
    networks:
      - stack
    environment:
      POSTGRES_DB: ${DB_DATABASE}
      POSTGRES_USER: ${DB_USER}
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    healthcheck:
      test: [ "CMD-SHELL", "pg_isready", "--quiet" ]
      start_period: 15s
      interval: 30s
      timeout: 5s
      retries: 5
  redis:
    container_name: ${_ID_}-redis
    hostname: ${_ID_}-redis
    image: redis:6
    restart: always
    volumes:
      - redisdumps:/data
    networks:
      - stack
    healthcheck:
      test: [ "CMD-SHELL", "redis-cli ping | grep PONG" ]
      start_period: 15s
      interval: 30s
      timeout: 5s
      retries: 5
  directus:
    container_name: ${_ID_}-directus
    hostname: ${_ID_}-directus
    image: directus/directus:${DIRECTUS_IMAGE_VERSION}
    restart: always
    networks:
      - stack
      - proxy
    healthcheck:
      test:  [  ↵
             "CMD-SHELL", "wget -O /dev/null --spider --tries=1 ↵
             --quiet http://localhost:${DIRECTUS_PORT}/server/health 2&>1" ↵
             ]
      start_period: 15s
      interval: 10s
```

```
      timeout: 5s
      retries: 3
    depends_on:
      redis:
        condition: service_started
      database:
        condition: service_started
    env_file:
      - .env
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.${_ID_}.entrypoints=websec"
      - "traefik.http.routers.${_ID_}.rule=Host(`${FQDN}`)"
      - "traefik.http.routers.${_ID_}.tls=true"
      - "traefik.http.routers.${_ID_}.tls.certresolver=le-http-01"
      - "traefik.http.routers.${_ID_}.service=${_ID_}"
- "traefik.http.services.${_ID_}.loadbalancer.server.port=${DIRECTUS_PORT}"
```

7.2.2 Docker Compose Environment

```
Repo: ./directus/.env.compose
```

```
Host: /opt/containers/directus.env
```

```
--- Redacted AES256 encrypted content ---
```

7.3 Passbolt

7.3.1 Docker Compose File

```
Repo: ./passbolt/docker-compose.yml
```

```
Host: /opt/containers/passbolt/docker-compose.yml
```

```
networks:
  proxy:
    external: true
  stack:
    driver: bridge
    internal: true
volumes:
  database:
  gpg_volume:
  jwt_volume:
services:
  database:
    image: mariadb:10.3
    container_name: ${_ID_}-database
    restart: always
    environment:
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    volumes:
      - database:/var/lib/mysql
    networks:
      - stack
  passbolt:
    image: passbolt/passbolt:latest-ce-non-root
    container_name: ${_ID_}
    restart: always
```

```
    tty: true
    depends_on:
      - database
    environment:
      DATASOURCES_DEFAULT_HOST: database
      DATASOURCES_DEFAULT_PORT: 3306
      DATASOURCES_DEFAULT_DATABASE: ${MYSQL_DATABASE}
      DATASOURCES_DEFAULT_USERNAME: ${MYSQL_USER}
      DATASOURCES_DEFAULT_PASSWORD: ${MYSQL_PASSWORD}
      EMAIL_TRANSPORT_DEFAULT_USERNAME: ${SES_SMTP_USER}
      EMAIL_TRANSPORT_DEFAULT_PASSWORD: ${SES_SMTP_PASS}
    env_file:
      - .env.passbolt
    volumes:
      - gpg_volume:/etc/passbolt/gpg
      - jwt_volume:/etc/passbolt/jwt
    command: ["/usr/bin/wait-for.sh", "database:3306", "--", "/docker-entrypoint.sh"]
    networks:
      - proxy
      - stack
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.${_ID_}.entrypoints=websec"
      - "traefik.http.routers.${_ID_}.rule=Host(`${FQDN}`)"
      - "traefik.http.routers.${_ID_}.tls=true"
      - "traefik.http.routers.${_ID_}.tls.certresolver=le-http-01"
      - "traefik.http.routers.${_ID_}.service=${_ID_}"
      - "traefik.http.services.${_ID_}.loadbalancer.server.port=8080"
```

7.3.2 Docker Compose Environment

```
Repo: ./passbolt/.env.compose
```

```
Host: /opt/containers/passbolt/.env
```

```
--- Redacted AES256 encrypted content ---
```

7.3.3 Passbolt Environment

```
Repo: ./passbolt/.env.passbolt
```

```
Host: /opt/containers/passbolt/.env.passbolt
```

```
APP_FULL_BASE_URL=https://pass.mars.poprocket.com
PASSBOLT_KEY_NAME=FiSi AP24 - Passbolt (Mars)
PASSBOLT_KEY_EMAIL=gpg+fisi-ap24@poprocket.com
PASSBOLT_GPG_SERVER_KEY_PUBLIC=/etc/passbolt/gpg/serverkey.asc
PASSBOLT_GPG_SERVER_KEY_PRIVATE=/etc/passbolt/gpg/serverkey_private.asc
PASSBOLT_SSL_FORCE=false
PASSBOLT_REGISTRATION_PUBLIC=false
PASSBOLT_META_TITLE=PASS
PASSBOLT_META_DESCRIPTION=mars.poprocket.com
PASSBOLT_PLUGINS_JWT_AUTHENTICATION_ENABLED=TRUE
PASSBOLT_PLUGINS_MOBILE_ENABLED=TRUE
EMAIL_DEFAULT_FROM=noreply@poprocket.com
EMAIL_TRANSPORT_DEFAULT_TLS=true
EMAIL_TRANSPORT_DEFAULT_PORT=587
EMAIL_TRANSPORT_DEFAULT_TIMEOUT=30
EMAIL_TRANSPORT_DEFAULT_HOST=email-smtp.eu-central-1.amazonaws.com
```

7.4 Ansible Metadaten

7.4.1 Konfiguration

```
Repo: ./Ansible/ansible.cfg
```

```
[defaults]
inventory = hosts.ini
remote_user = toor
interpreter_python = /usr/bin/python3
```

7.4.2 Inventory

```
Repo: ./Ansible/hosts.ini
```

```
[mars]
mars.poprocket.com
```

7.4.3 Variablen

```
Repo: ./Ansible/config.yml
```

```
# hostname for target host
HOSTNAME: mars
HOSTS_INI_TARGET: mars

# Reconfiguration patterns of '/etc/hosts'
SEARCH_LINE: '^5\.45\.103\.43\s+.*$'
REPLACE_LINE: "5.45.103.43       {{ HOSTNAME }}.poprocket.com   {{ HOSTNAME }}"

# path for Docker Compose project directories
CONTAINER_ROOT: /opt/containers

# Well known Reverse Proxy Network
PROXY_NETWORK_NAME: proxy
PROXY_NETWORK_CIDR: 172.30.0.0/16
PROXY_NETWORK_GATEWAY: 172.30.0.1

# admin user configuration
ADMIN_USER_NAME: toor
ADMIN_USER_SSH_KEY: "${HOME}/.ssh/id_rsa_poprocket_2023.pub"
ADMIN_USER_PASSWORD_HASH: "-- REDACTED --"
```

7.5 Ansible Playbooks

7.5.1 Prepare

```
Repo: ./Ansible/pb1_prepare.yml
```

```
- name: Prepare Requirements
  vars_files:
    - config.yml
  hosts: "{{ HOSTS_INI_TARGET }}"
  gather_facts: false
  remote_user: root
  become: true
  become_method: su
  tasks:
```

```
    - name: Change desired hostname
      hostname:
        name: "{{ HOSTNAME }}"
    - name: Remove host default entry from hosting provider
      lineinfile:
        path: /etc/hosts
        state: absent
        regexp: "{{ SEARCH_LINE }}"
    - name: Add correct host default entry
      lineinfile:
        path: /etc/hosts
        line: "{{ REPLACE_LINE }}"
    - name: Install sudo package
      apt:
        name: sudo
        state: present
        update_cache: yes
    - name: Create admin user "{{ ADMIN_USER_NAME }}"
      user:
        name: "{{ ADMIN_USER_NAME }}"
        groups: sudo
        append: yes
        shell: /bin/bash
        password: "{{ ADMIN_USER_PASSWORD_HASH }}"
    - name: Deploy SSH public key for "{{ ADMIN_USER_NAME }}"
      authorized_key:
        user: "{{ ADMIN_USER_NAME }}"
        key: "{{ lookup('file', \"{{ ADMIN_USER_SSH_KEY }}\") }}"
```

7.5.2 Setup

```
Repo: ./Ansible/pb2_setup.yml
```

```
- name: Install Fail2Ban and Docker
  vars_files:
    - config.yml
  hosts: "{{ HOSTS_INI_TARGET }}"
  gather_facts: false
  become: true
  tasks:
    - name: System catalog update and upgrade outdated
      register: updatesys
      apt:
        name: "*"
        state: latest
        update_cache: yes
    - name: Display the last line of the previous task to check the stats
      debug:
        msg:  "{{updatesys.stdout_lines|last}}"
    - name: Install fail2ban package
      apt:
        name: fail2ban
        state: present
    - name: Install Docker dependencies
      apt:
        name:
          - ca-certificates
          - curl
          - gnupg
        state: present
    - name: Create directory /etc/apt/keyrings
      file:
        path: /etc/apt/keyrings
        state: directory
        mode: '0755'
    - name: Check if Docker GPG key file exists
      stat:
        path: /etc/apt/keyrings/docker.gpg
```

```
      register: docker_gpg_stat
    - name: Download Docker GPG key and convert to ASCII format
      shell: >-
        curl -fsSL https://download.docker.com/linux/debian/gpg
        | gpg --dearmor -o /etc/apt/keyrings/docker.gpg
      when: docker_gpg_stat.stat.exists == false
    - name: Change permissions for Docker GPG key
      file:
        path: /etc/apt/keyrings/docker.gpg
        mode: '0644'
    - name: Determine architecture
      command: dpkg --print-architecture
      register: architecture
    - name: Determine Debian version codename
      command: /bin/bash -c '. /etc/os-release && echo "$VERSION_CODENAME"'
      register: debian_codename
    - name: Add Docker apt repository
      copy:
        content: "deb [arch={{ architecture.stdout }} ↵
                  signed-by=/etc/apt/keyrings/docker.gpg] ↵
                  https://download.docker.com/linux/debian ↵
                  {{ debian_codename.stdout }} stable"
        dest: /etc/apt/sources.list.d/docker.list
        mode: '0644'
    - name: Install Docker engine packages
      apt:
        name:
          - docker-ce
          - docker-ce-cli
          - containerd.io
          - docker-buildx-plugin
          - docker-compose-plugin
        state: present
    - name: Add user "{{ ADMIN_USER_NAME }}" to the docker group
      user:
        name: "{{ ADMIN_USER_NAME }}"
        groups: docker
        append: yes
    - name: Create "{{ CONTAINER_ROOT }}" directory
      file:
        path: "{{ CONTAINER_ROOT }}"
        state: directory
        owner: "{{ ADMIN_USER_NAME }}"
        group: "{{ ADMIN_USER_NAME }}"
```

7.5.3 Traefik

```
Repo: ./Ansible/pb3_traefik.yml
```

```
- name: Install Traefik
  vars:
    PROJEKT_NAME: traefik
  vars_files:
    - config.yml
  hosts: "{{ HOSTS_INI_TARGET }}"
  remote_user: "{{ ADMIN_USER_NAME }}"
  gather_facts: false
  become: true
  tasks:
    - name: Create Reverse Proxy Docker network
      docker_network:
        name: "{{ PROXY_NETWORK_NAME }}"
        state: present
        ipam_config:
          - subnet: "{{ PROXY_NETWORK_CIDR }}"
            gateway: "{{ PROXY_NETWORK_GATEWAY }}"
    - name: Create "/etc" entrypoint
      file:
```

```
        path: "/etc/{{ PROJEKT_NAME }}"
        state: directory
    - name: Create directory for dynamic configuration
      file:
        path: "/etc/{{ PROJEKT_NAME }}/conf.d"
        state: directory
    - name: Create directory for certificates storage
      file:
        path: "/etc/{{ PROJEKT_NAME }}/acme.d"
        state: directory
        mode: '0700'  # crts should be accessible by root only
    - name: Create project directory
      file:
        path: "{{ CONTAINER_ROOT }}/{{ PROJEKT_NAME }}"
        state: directory
        owner: "{{ ADMIN_USER_NAME }}"
        group: "{{ ADMIN_USER_NAME }}"
    - name: Deploy Docker Compose file
      copy:
        src: "../{{ PROJEKT_NAME }}/docker-compose.yml"
        dest: "{{ CONTAINER_ROOT }}/{{ PROJEKT_NAME }}/docker-compose.yml"
    - name: Deploy Docker Compose configuration
      copy:
        src: "../{{ PROJEKT_NAME }}/.env.compose"
        dest: "{{ CONTAINER_ROOT }}/{{ PROJEKT_NAME }}/.env"
        owner: "{{ ADMIN_USER_NAME }}"
        group: "{{ ADMIN_USER_NAME }}"
    - name: Deploy Traefik static configuration
      copy:
        src: "../{{ PROJEKT_NAME }}/traefik.yml"
        dest: "{{ CONTAINER_ROOT }}/{{ PROJEKT_NAME }}/traefik.yml"
        owner: "{{ ADMIN_USER_NAME }}"
        group: "{{ ADMIN_USER_NAME }}"
    - name: Deploy Traefik Whitelist middleware
      copy:
        src: "../{{ PROJEKT_NAME }}/protected-ips.yml"
        dest: "/etc/{{ PROJEKT_NAME }}/conf.d/protected-ips.yml"
        mode: '0640'  # only editable by root
    - name: Deploy Traefik BasicAuth middleware
      copy:
        src: "../{{ PROJEKT_NAME }}/protected-auth.yml"
        dest: "/etc/{{ PROJEKT_NAME }}/conf.d/protected-auth.yml"
        mode: '0640'  # only editable by root
    - name: Start stack
      command:  >-
        docker compose -f "{{ CONTAINER_ROOT }}/{{ PROJEKT_NAME }}/docker-compose.yml"
        up -d –force-recreate
```

7.5.4 Directus

```
Repo: ./Ansible/pb4_directus.yml
```

```
- name: Install Directus
  vars:
    PROJECT_NAME: directus
  vars_files:
    - config.yml
  hosts: "{{ HOSTS_INI_TARGET }}"
  remote_user: "{{ ADMIN_USER_NAME }}"
  gather_facts: false
  tasks:
    - name: Create project directory
      file:
        path: "{{ CONTAINER_ROOT }}/{{ PROJECT_NAME }}"
        state: directory
    - name: Copy Docker Compose file
      copy:
        src: "../{{ PROJECT_NAME }}/docker-compose.yml"
```

```
        dest: "{{ CONTAINER_ROOT }}/{{ PROJECT_NAME }}/docker-compose.yml"
    - name: Copy Docker Compose configuration
      copy:
        src: "../{{ PROJECT_NAME }}/.env.compose"
        dest: "{{ CONTAINER_ROOT }}/{{ PROJECT_NAME }}/.env"
        decrypt: true
    - name: Start stack
      command: >-
        docker compose -f "{{ CONTAINER_ROOT }}/{{ PROJECT_NAME }}/docker-compose.yml"
        up -d --force-recreate
```

7.5.5 Passbolt

```
Repo: ./Ansible/pb5_passbolt.yml
```

```
- name: Install Passbolt
  vars:
    PROJECT_NAME: passbolt
  vars_files:
    - config.yml
  hosts: "{{ HOSTS_INI_TARGET }}"
  remote_user: "{{ ADMIN_USER_NAME }}"
  gather_facts: false
  tasks:
    - name: Create project directory
      file:
        path: "{{ CONTAINER_ROOT }}/{{ PROJECT_NAME }}"
        state: directory
    - name: Deploy Docker Compose file
      copy:
        src: "../{{ PROJECT_NAME }}/docker-compose.yml"
        dest: "{{ CONTAINER_ROOT }}/{{ PROJECT_NAME }}/docker-compose.yml"
    - name: Deploy Passbolt configuration
      copy:
        src: "../{{ PROJECT_NAME }}/.env.passbolt"
        dest: "{{ CONTAINER_ROOT }}/{{ PROJECT_NAME }}/.env.passbolt"
    - name: Deploy Docker Compose configuration
      copy:
        src: "../{{ PROJECT_NAME }}/.env.compose"
        dest: "{{ CONTAINER_ROOT }}/{{ PROJECT_NAME }}/.env"
        decrypt: true
    - name: Start stack
      command: >-
        docker compose -f "{{ CONTAINER_ROOT }}/{{ PROJECT_NAME }}/docker-compose.yml"
        up -d –force-recreate
```

7.6 Funktionstest Webanwendungen

7.6.1 Directus

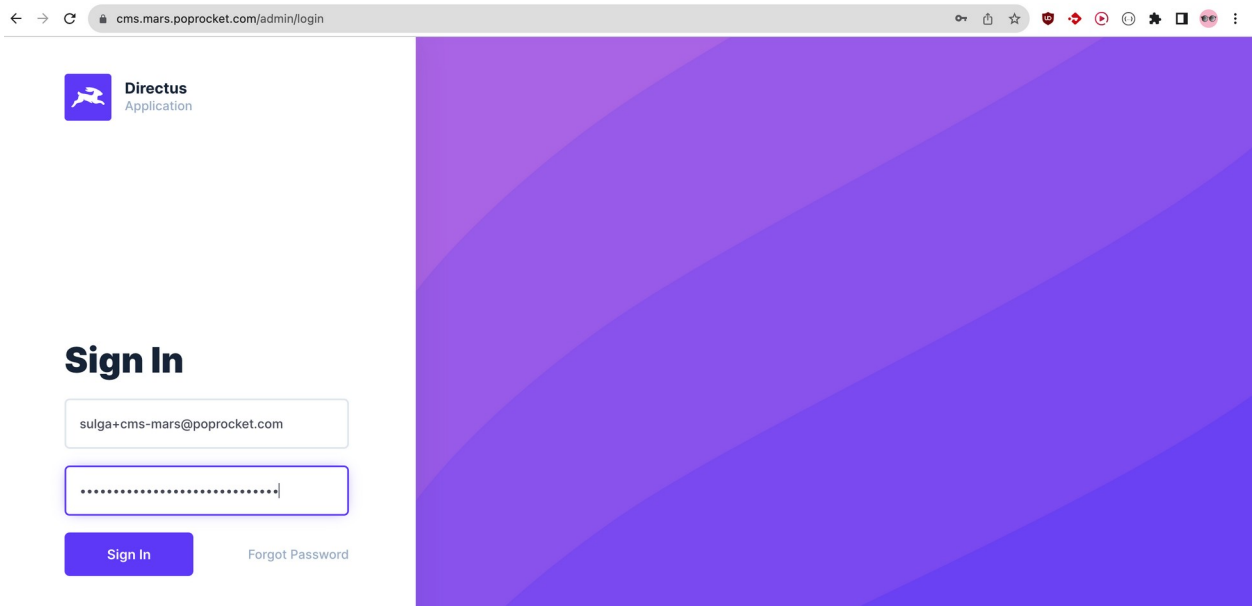

Abbildung 4: Directus Login Screen

7.6.2 Passbolt

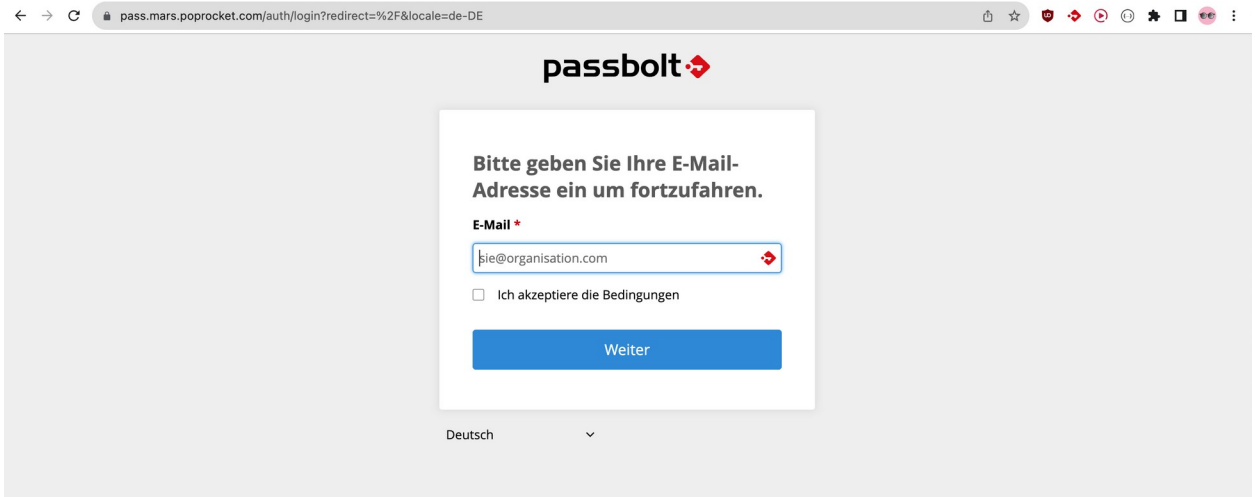

Abbildung 5: Passbolt Onboarding