Abschlussbericht: Labor Linux Systemadministration

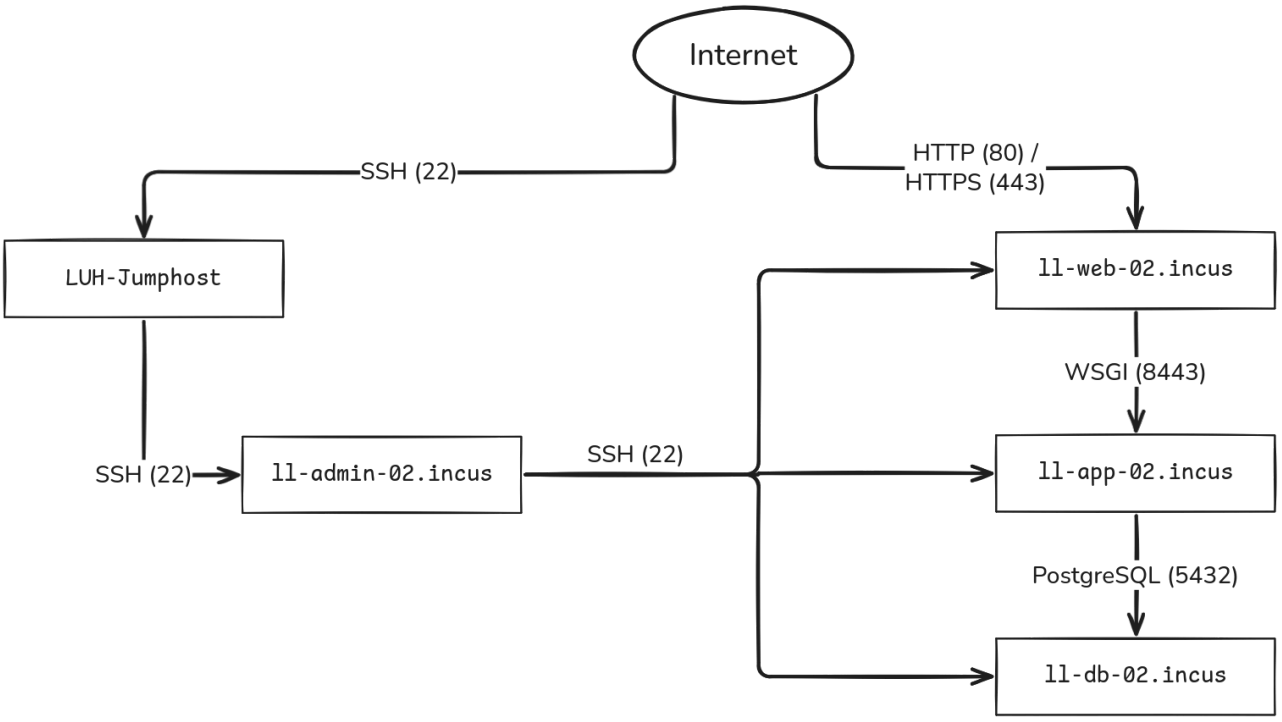
Dozent: Dr. Hans Georg Krojanski

Paul Bakenhus
Matrikelnummer: 10054696

Systemübersicht und Architektur

Dieser Bericht dokumentiert die Einrichtung und Konfiguration einer serviceorientierten Architektur für die Anwendung GNU Health auf Basis von vier dezidierten virtuellen Debian-12-Maschinen.

Systemarchitektur Diagramm



Server-Rollen und Spezifikationen

Die Infrastruktur ist auf vier Server aufgeteilt, die jeweils klar definierte Aufgaben erfüllen. Grundsätzlich laufen auf allen Servern die Dienste sshd zur Administration und nftables zur Netzwerkabsicherung.

ll-admin-02.incus (Adminserver)

Der zentrale Verwaltungs- und Sicherheitsserver. Er dient als einziger Zugangspunkt zum internen Netzwerk.

- **Offene Ports:** SSH (22) erreichbar vom LUH-Jumphost.
- **Installierte Pakete:**
    - apt: barman, cron, postgresql-client (Für die Verwaltung und Automatisierung der Datenbank-Backups).
    - pip: proteus (Zum Ausführen von Konnektivitätstests gegen den Anwendungsserver).
- **Weitere Rollen:** Beherbergt die private Certificate Authority (CA) zur Ausstellung aller TLS-Zertifikate.

ll-web-02.incus (Webserver)

Der einzige aus dem Internet erreichbare Server, der als Reverse Proxy agiert.

- **Offene Ports:** HTTP (80) mit permanenter Umleitung auf HTTPS; HTTPS (443) aus dem Internet erreichbar; SSH (22) nur vom Adminserver erreichbar.
- **Installierte Pakete:**
    - apt: nginx (Als Reverse Proxy und für die TLS-Terminierung).
    - apt: ssl-cert (Nur für die initiale Test-Einrichtung genutzt).

ll-app-02.incus (Anwendungsserver)

Der Server, auf dem die eigentliche GNU-Health-Anwendung in einer isolierten Umgebung läuft.

- **Offene Ports:** uWSGI (8443) erreichbar nur vom Webserver; SSH (22) nur vom Adminserver erreichbar.
- **Installierte Pakete:**
    - apt: python3-pip, python3-venv, libssl-dev, libpq-dev (Systemabhängigkeiten für die Python-Umgebung).
    - apt: restic (Client für das dateibasierte Backup).
    - pip: gnuhealth-all-modules, uwsgi (Die Anwendung selbst und ihr WSGI-Server, installiert im venv).

ll-db-02.incus (Datenbankserver)

Der dezidierte Server für die Persistenz der Anwendungsdaten.

- **Offene Ports:** PostgreSQL (5432) erreichbar nur vom Anwendungsserver; SSH (22) nur vom Adminserver erreichbar.
- **Installierte Pakete:**
    - apt: postgresql (Der Datenbankserver).
    - apt: barman-cli (Stellt barman-wal-archive)

Übergreifende Sicherheitsmaßnahmen

- **Firewall (nftables)**
- **SSH:** Die Authentifizierung erfolgt ausschließlich über Public-Key-Verfahren; Passwort-Logins sind systemweit deaktiviert. *(Anhang I)*
- **Private Certificate Authority**
- **SSL/TLS:**
    - **Client → Webserver:** HTTPS
    - **Webserver → Applikationsserver:** uWSGI über TLS
    - **Applikationsserver → Datenbankserver:** PostgreSQL-Protokoll über TLS

Vertiefung der einzelnen Server

ll-admin-02

Der Adminserver ist zuständig für Zugriff, Verwaltung, Testen und Wiederherstellung der drei anderen Server. Im Folgenden wird auf die einzelnen Aufgaben genauer eingegangen, auch um im Falle einer Neuinstallation die Schritte dieser nachverfolgbar zu machen.

1. **SSH**
   Sämtlicher Zugriff auf die drei Server muss über den Adminserver erfolgen. Als einziger ist er auf Port 22 vom LUH-Jumphost erreichbar, um die Angriffsfläche so klein wie möglich zu halten.
2. **CA**
   Um sichere Verbindungen zu gewährleisten, wurde auf dem Adminserver eine private Certificate Authority eingerichtet *(Anhang II)*. Es wurde sich dabei an der *Ubuntu Server* Dokumentation orientiert. Der private Schlüssel wurde mit Zugriffsrechten 400 bestmöglichst geschützt. Mit openssl wurden auf allen Servern CSRs generiert und auf dem Adminserver signiert. Auf jedem der drei Server liegt ein ssl Ordner im $HOME Verzeichnis, welche die notwendigen Dateien enthalten. Die signierten Zertifikate wurden in /etc/ssl/certs kopiert, die Schlüssel in /etc/ssl/private.
3. **PostgreSQL Backup mit barman**
   Um eine Point-in-Time-Recovery der Datenbank zu ermöglichen, sichert Barman kontinuierlich die Write-Ahead-Logs und erstellt zusätzlich per Cronjob regelmäßige physische Basis-Backups auf dem Adminserver. Notwendig dafür ist die Einrichtung eines SSH Schlüsselpaares vom Adminserver zum Datenbankserver (Nutzer barman) und vice versa (Nutzer postgres). Die Konfiguration auf Seite des Adminservers beläuft sich auf eine Konfigurationsdatei in /etc/barman.d/health.conf *(Anhang III)*. Genaueres zur Konfiguration ist im Abschnitt für ll-db-02 enthalten.
4. **GNU Health Dateibackup mit restic**
   Während barman das Backup für die Datenbank verwaltet, werden die Anwendungsdateien für GNU Health mit restic gesichert. Dazu musste ein Nutzer restic-backup auf dem Adminserver erstellt werden, auf welchen sich der gnuhealth Nutzer vom Applikationsserver mit einem Schlüsselpaar einloggen kann. Auf dem Adminserver wurde ein Ordner restic-repo im $HOME Verzeichnis des restic-backup Nutzers erstellt, in welches mit einem cronjob regelmäßige Backups vom Applikationsserver übertragen werden. Die Einrichtung des Repositorys wird im Abschnitt für ll-app-02 erklärt.
5. **Tests**
   Um kontinuierlich die Funktionalität des GNU Health Endpunktes zu prüfen, befindet sich ein Skript connect.py im gnuhealth Ordner des $HOME Verzeichnisses. Zum Testen musste zuerst das virtual environment gestartet und danach das Skript ausgeführt werden.

ll-web-02

Der Webserver ist als einziger der vier Server aus dem Internet erreichbar *(Anhang V)*. Seine Aufgabe ist das Entgegennehmen eingehender Verbindungen und Weiterleitung an den GNU Health Endpunkt.

1. **Reverse Proxy und SSL**
   Die auf dem Server laufende Instanz von nginx wurde als Reverse Proxy konfiguriert um den Applikationsserver bestmöglich nach außen hin zu schützen. Er leitet sämtliche Zugriffe von Port 80 auf den Port 443 um, damit unverschlüsselte Zugriffe nicht möglich sind. *(Anhang VI)*

ll-db-02

Auf dem Datenbankserver läuft die GNU Health zugrundeliegende PostgreSQL 15 Datenbank health.

1. **Einrichtung für GNU Health**
   Vor Installation der GNU Health Software wurde eine Datenbank health und die Rolle gnuhealth als deren Besitzer angelegt.
2. **SSL**
   Damit der Datenverkehr stets verschlüsselt abläuft, muss die Konfiguration im Anhang übernommen werden. Diese nutzt das vom Adminserver signierte Zertifikat um sicher mit dem Applikationsserver zu kommunizieren *(Anhang VII)*.
3. **Backup mit barman**
   Für eine automatische Sicherung der Datenbank wird barman verwendet. Folgende Schritte wurden durchgeführt:
   - 3.1. **PostgreSQL-Nutzer:** Es wurden zwei dezidierte Rollen erstellt:
     barman mit SUPERUSER-Rechten für die Steuerung der Backups und streaming_barman mit REPLICATION-Rechten für das WAL-Streaming.
   - 3.2. **Zugriffsregeln (pg_hba.conf):** Der Zugriff für diese beiden Nutzer wurde explizit auf verschlüsselte Verbindungen vom Adminserver beschränkt. *(Anhang VIII)*
   - 3.3. **Backup-Konfiguration (postgresql.conf):** Die WAL-Archivierung wurde durch wal_level = replica, archive_mode = on und den archive_command aktiviert, der per SSH auf barman-wal-archive auf dem Adminserver verweist. *(Anhang VII)*
   - 3.4. **Replikationsslot:** Mittels psql wurde ein physischer Replikationsslot namens barman erstellt, um sicherzustellen, dass der DB-Server die WAL-Dateien nicht löscht, bevor der Barman-Server sie empfangen hat.
   - 3.5. **Passwortdatei:** Im $HOME Verzeichnis des barman Nutzers auf dem Adminserver wurde eine .pgpass Datei zur automatischen Anmeldung erstellt. *(Anhang IV)*
   - 3.6. **Konfigurationsatei:** Die Konfigurationsdatei für den Server health wurde im entsprechenden Verzeichnis erstellt *(Anhang III)*.
   - 3.7. **Einrichtung:** Folgende Schritte wurden als barman Nutzer auf dem Adminserver ausgeführt:
     - 3.7.1. barman list-server gab erfolgreich “health - Barman - PostgreSQL Server health” zurückgeben.
     - 3.7.2. barman check health um die Verbindung mit Datenbankserver zu prüfen. *Falls PostgreSQL nicht OK zeigte, nftables Konfiguration und pg_hba.conf Einträge geprüft. Weiter waren /var/log/postgresql/postgresql-15-main.log auf dem Datenbankserver und /var/log/barman/barman.log auf dem Adminserver oft aufschlussreich.*

3.7.3. barman replication-status health um WAL-Streaming zu prüfen
3.7.4. barman backup health startet ein Backup.
Bei der Einrichtung zeigte sich, dass eine korrekte Firewall-Regel und ein archive_command entscheidend für die Funktionalität sind, da Fehler hier zu blockierenden Backup-Prozessen führen können.
3.7.5. barman list-backup health zeigt erfolgreiches Backup.

ll-app-02

Der Anwendungsserver ist verantwortlich für die Ausführung der GNU Health Anwendung in einem isolierten virtual environment.

1. **GNU Health**
   a. **Vorbereitung**
   Es wurde ein Systembenutzer gnuhealth angelegt mit dem $HOME Verzeichnis in /opt/gnuhealth. Dort wurde eine virtuelle Python-Umgebung erstellt, in welcher sämtliche Dateien für GNU Health liegen. Über pip wurden im venv die Pakete gnuhealth-all-modules mit Version 4.4.1 und uwsgi installiert, wodurch Konflikte mit systemweiten Paketen vermieden wurden.
   b. **Konfigurationsdateien**
   Es wurden Konfigurationsvorlagen gestellt, welche entsprechend den Anforderungen angepasst wurden *(Anhang IX, Anhang X)*.
   c. **Initialisierung der Datenbank**
   Mit trytond-admin wurde die Datenbank initialisiert und es wurde ein Adminpasswort festgelegt.
   d. **GNU Health als Service**
   Es wurde eine systemd-Unit-Datei (gnuhealth.service) unter /etc/systemd/system erstellt. Diese Datei definiert, wie der Prozess als Systemdienst gestartet und verwaltet wird. Durch systemctl enable --now gnuhealth.service wurde der Dienst aktiviert und gestartet. Dies gewährleistet den automatischen Start der Anwendung beim Hochfahren des Servers sowie einen automatischen Neustart im Fehlerfall.
2. **Dateibackup mit restic**
   Während die GNU Health zugrundeliegende Datenbank von barman gesichert ist, liegen im /opt/gnuhealth/var/lib wichtige Dateien, welche mit restic gesichert wurden.
   a. **Vorbereitung**
   Auf dem Anwendungsserver wurde restic installiert und ein Schlüsselpaar mit dem restic-backup Nutzer auf dem Adminserver ausgetauscht. Genaueres dazu im
   b. **Repository**
   Es wurde ein Repository auf dem Adminserver im $HOME Verzeichnis des restic-backup Nutzers erstellt und in /opt/gnuhealth/ eine .restic-pass Datei angelegt, welche das bei der Initialisierung des Respositorys festgelegte Passwort enthält. Diese ermöglicht eine automatische Authentifizierung.
   c. **Automatisierung**
   Um eine ständige Sicherung zu gewährleisten wurde mit cron ein regelmäßiges Backup eingerichten *(Anhang XI)*.
   d. **Test**
   Um die Funktionalität zu prüfen wurden Dateien im /opt/gnuhealth/var/lib Verzeichnis angelegt, ein Backup manuell angestoßen und die Dateien gelöscht. Es wurden erfolgreich die Dateien wiederhergestellt.

Neustart von Diensten

Für einen gesamten Neustart der GNU Health Anwendung oder einzelner Services sollte folgende Reihenfolge eingehalten werden:

1. **PostgreSQL Datenbank**
   debian@ll-db-02: sudo systemctl restart postgresql.service
2. **GNU Health**
   debian@ll-app-02: sudo systemctl restart gnuhealth.service
3. **Nginx**
   debian@ll-web-02: sudo systemctl restart nginx.service

Disaster recovery

PostgreSQL

Die folgenden Schritte sind durchzuführen, um eine Datenbankwiederherstellung durchzuführen:

1. **Dienste anhalten**
   debian@ll-app-02: sudo systemctl stop gnuhealth.service
   debian@ll-db-02: sudo systemctl stop postgresql.service
2. **Gegebenenfalls alte Daten verschieben:**
   debian@ll-db-02: sudo mv mv /var/lib/postgresql/15/main /var/lib/postgresql/15/main_defekt
3. **Backup wiederherstellen:**
   debian@ll-admin-02: sudo -u barman barman restore health latest /var/lib/postgresql/15/main
4. **Dienste starten:**
   debian@ll-db-02: sudo systemctl start postgresql.service
   debian@ll-app-02: sudo systemctl start gnuhealth.service

GNU Health

Die folgenden Schritte sind durchzuführen, um eine Dateiwiederherstellung durchzuführen:

1. **Dienste anhalten**
   debian@ll-app-02: sudo systemctl stop gnuhealth.service
2. **Gegebenenfalls alte Daten verschieben und Verzeichnis vorbereiten:**
   debian@ll-app-02: sudo mv /opt/gnuhealth/var/lib /opt/gnuhealth/var/lib_defekt
   debian@ll-app-02: sudo mkdir /opt/gnuhealth/var/lib
   debian@ll-app-02: sudo chown gnuhealth:gnuhealth /opt/gnuhealth/var/lib
   debian@ll-app-02: sudo -u gnuhealth restic -r sftp:restic-backup@ll-admin-02:/home/restic-backup/restic-repo --password-file /opt/gnuhealth/.restic-pass restore latest --target /
3. **Dienste starten**
   debian@ll-app-02: sudo systemctl start gnuhealth.service

Anhang

I) `/etc/ssh/sshd_config` auf allen Servern:

```
PubkeyAuthentication yes
PasswordAuthentication no
```

II) `/etc/ssl/openssl.conf` auf **`ll-admin-02`**:

```
0.organizationName_default      = Leibniz Universitaet Hannover
organizationalUnitName_default  = LinuxLab
commonName_default              = ll-admin-02.incus
emailAddress                    = paul.bakenhus@stud.uni-hannover.de

[ req_attributes ]
subjectAltName= @alt_names
```

III) `/etc/barman.conf` auf **`ll-admin-02`**:

```
[health]
description =  "Barman - PostgreSQL Server health"

conninfo = host=ll-db-02.incus user=barman dbname=health
streaming_conninfo = host=ll-db-02.incus user=streaming_barman

backup_method = postgres

streaming_archiver = on
archiver = on
slot_name = barman

minimum_redundancy = 3
last_backup_maximum_age = 1 MONTHS
retention_policy = RECOVERY WINDOW OF 30 DAYS
```

IV) `/var/lib/barman/.pgpass` auf **`ll-admin-02`**:

```
ll-db-02.incus:5432:*:barman:barman
ll-db-02.incus:5432:*:streaming_barman:streaming_barman
```

V) `/etc/nftables.conf` auf **`ll-web-02`**:

```
define ll-admin-02 = 10.100.30.21
        chain input {
                type filter hook input priority 0;
                policy drop;
                ct state established, related accept
                iif "lo" accept
                tcp dport 22 ip saddr $ll-admin-02 accept
                tcp dport 80 accept
                tcp dport 443 accept
        }
```

VI) `/etc/nginx/sites-available/gnuhealth` auf **`ll-web-02`**:

```
server {
    listen 443 ssl default_server;
    server_name ll-web-02.incus;

    include snippets/ssl.conf;
    include snippets/linuxlab.conf;
    location / {
        proxy_pass https://10.100.30.23:8443;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

VII) `/etc/postgresql/15/main/postgresql.conf` auf **`ll-db-02`**:

```
ssl = on
ssl_ca_file = '/etc/ssl/certs/cacert.pem'
ssl_cert_file = '/etc/ssl/certs/ll-db-02.incus.crt'
ssl_key_file = '/etc/ssl/private/postgresql.key'

wal_level = replica
archive_mode = on
archive_command = 'barman-wal-archive ll-admin-02.incus health %p'
restore_command = 'barman-wal-restore ll-admin-02.incus health %f %p'
```

VIII) `/etc/postgresql/15/main/pg_hba.conf` auf **`ll-db-02`**:

```
# Zugriffsrechte für Backup von Adminserver
hostssl replication streaming_barman    10.100.30.21/32     scram-sha-256
hostssl all         barman              10.100.30.21/32     scram-sha-256
# Zugriffsrechte für Datenbank von Applikationsserver
hostssl health      gnuhealth           10.100.30.23/32     scram-sha-256
```

IX) `/opt/gnuhealth/etc/trytond.conf` auf **`ll-app-02`**:

```
[database]
uri =
postgresql://gnuhealth:health@ll-db-02.incus:5432/health?sslmode=verify-full&sslrootc
ert=/etc/ssl/certs/cacert.pem
path = /opt/gnuhealth/var/lib

[web]
listen = [::]:8000

[ssl]
privatekey=/etc/ssl/private/gnuhealth.key
certificate=/etc/ssl/certs/gnuhealth.crt
```

X) `/opt/gnuhealth/etc/uwsgi_trytond.ini` auf **`ll-app-02`**:

```
[uwsgi]
# http = 0.0.0.0:8000
#wsgi supports HTTPS via the following option:
https = 0.0.0.0:8443,/etc/ssl/certs/gnuhealth.crt,/etc/ssl/private/gnuhealth.key,HIGH
wsgi-file = /opt/gnuhealth/venv/bin/trytond
env = TRYTOND_CONFIG=/opt/gnuhealth/etc/trytond.conf
env = TRYTOND_LOGGING_CONFIG=/opt/gnuhealth/etc/gnuhealth_log.conf
chdir = /opt/gnuhealth/venv
module = trytond.application:app
callable = app
stats = 127.0.0.1:9191
venv = /opt/gnuhealth/venv
logto = /opt/gnuhealth/var/log/uwsgi.log
```

XI) `/var/spool/crontab/gnuhealth` auf **`ll-app-02`**:

```
*/15 * * * * restic -r
sftp:restic-backup@ll-admin-02.incus:/home/restic-backup/restic-repo --password-file
./.restic-pass backup /opt/gnuhealth/var/lib

0 1 * * * restic -r
sftp:restic-backup@ll-admin-02.incus:/home/restic-backup/restic-repo --password-file
./.restic-pass forget --prune --keep-hourly 24 --keep-daily 7 --keep-weekly 3
--keep-monthly 3 --keep-yearly 2
```