Server einrichten

Vollständige Anfänger-Anleitung von A bis Z
Neutrale Version für Debian 13 Linux

Hinweis: Diese Anleitung ist eine technische Arbeits- und Testanleitung. Sie ersetzt keine Rechts-, Datenschutz- oder IT-Sicherheitsberatung. Niemals echte Passwörter, Schlüssel oder private Daten in diese PDF eintragen.

1. Was am Ende bereitsteht

Ein sicher erreichbarer Debian-Server mit Domain, HTTPS, Webserver, Website, geschützten Apps, Benutzeranmeldung, Zwei-Faktor-Schutz, Backups und Protokollen.

2. Programme ohne Werbung

Aufgabe	Programmtyp
SSH/SFTP-Verbindung	beliebiger SSH- und SFTP-Client
Passwörter	Passwortmanager mit 2FA
Einmalcodes	Authenticator-App für TOTP
Dateien bearbeiten	Texteditor mit UTF-8-Unterstützung
Tests	moderner Browser und privates Fenster

3. Platzhalter ausfüllen

DOMAIN=deine-domain.tld
SERVER_IP=203.0.113.10
SERVER_USER=adminuser
ADMIN_EMAIL=admin@deine-domain.tld
APP_NAME=meine-app
APP_PORT=3000

Die Beispiel-IP ist nur Dokumentationsmaterial. Immer die eigenen Werte verwenden.

4. Voraussetzungen

Der Server läuft bereits mit Debian 13, ist über SSH erreichbar und hat eine feste öffentliche IP. Die Domain kann DNS-Einträge verwalten. Du hast ein zweites Gerät für TOTP und ein aktuelles Backup-Konzept.

5. Erste Verbindung

```
ssh SERVER_USER@SERVER_IP
```

Beim ersten Fingerabdruck nur bestätigen, wenn der Server eindeutig deiner Verwaltung zugeordnet ist. Danach das System aktualisieren:

```
sudo apt update
sudo apt full-upgrade -y
sudo reboot
```

6. Grundschutz

```
sudo apt install -y sudo curl wget unzip git ca-certificates ufw fail2ban nginx
sudo adduser deploy
sudo usermod -aG sudo deploy
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow OpenSSH
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw enable
sudo systemctl enable --now fail2ban
```

Mit einem zweiten Terminal testen, dass der neue Benutzer sich anmelden kann, bevor der alte Zugang eingeschränkt wird. Niemals den einzigen Zugang sperren.

7. SSH sicherer machen

Optional einen SSH-Schlüssel einrichten und testen:

```
ssh-keygen -t ed25519
ssh-copy-id deploy@SERVER_IP
ssh deploy@SERVER_IP
```

Erst nach erfolgreichem Test in einer separaten Konfigurationsdatei Passwortanmeldung und direkte Root-Anmeldung deaktivieren. Danach:

```
sudo sshd -t
```

```
sudo systemctl reload ssh
```

Wenn du dich nicht mehr anmelden kannst: Konsole des Serverkontos verwenden. Nicht gleichzeitig mehrere SSH-Einstellungen ändern.

8. Domain und DNS

Einen A-Eintrag für die Domain und optional für www auf SERVER_IP setzen. Prüfen:

```
getent hosts deine-domain.tld
curl -I http://deine-domain.tld
```

DNS kann je nach Netzwerk verzögert sichtbar werden. Erst weitergehen, wenn die Domain auf den richtigen Server zeigt.

9. Webserver einrichten

```
sudo mkdir -p /var/www/DOMAIN/public
sudo chown -R deploy:deploy /var/www/DOMAIN
echo '<h1>Testseite</h1>' > /var/www/DOMAIN/public/index.html
sudo nano /etc/nginx/sites-available/DOMAIN
```

Beispiel für die Konfiguration:

```
server {
    listen 80;
    server_name DOMAIN www.DOMAIN;
    root /var/www/DOMAIN/public;
    index index.html;
    location / { try_files $uri $uri/ =404; }
}

sudo ln -s /etc/nginx/sites-available/DOMAIN /etc/nginx/sites-enabled/DOMAIN
sudo nginx -t
sudo systemctl reload nginx
```

10. HTTPS

Ein Zertifikat für die Domain mit dem Zertifikatswerkzeug des Systems einrichten. Nur die offiziellen Pakete der eigenen Debian-Version verwenden. Danach testen:

```
curl -I https://DOMAIN
sudo systemctl status nginx --no-pager
```

HTTP soll auf HTTPS weiterleiten. Zertifikate müssen automatisch erneuert werden; die Erneuerung regelmäßig testen.

11. Website hochladen

ZIP per SFTP in einen nicht öffentlichen Ordner übertragen:

```
mkdir -p /home/DEPLOY_USER/uploads /home/DEPLOY_USER/staging
unzip -q /home/DEPLOY_USER/uploads/PROJEKT.zip -d /home/DEPLOY_USER/staging/PROJEKT
find /home/DEPLOY_USER/staging/PROJEKT -maxdepth 3 -name index.html -print
```

Erst prüfen, dann in den Webroot kopieren:

```
sudo rsync -a --dry-run /home/DEPLOY_USER/staging/PROJEKT/ /var/www/DOMAIN/public/
sudo rsync -a /home/DEPLOY_USER/staging/PROJEKT/ /var/www/DOMAIN/public/
sudo nginx -t && sudo systemctl reload nginx
```

12. Node.js-Projekte

Nur nötig, wenn package.json und ein Startskript vorhanden sind:

```
node --version
npm --version
npm ci
npm run build
```

Bei einer statischen Ausgabe (dist oder build) nur diese Ausgabe veröffentlichen. Ein laufender Node-Dienst benötigt einen eigenen Systemdienst, einen lokalen Port und einen Nginx-Reverse-Proxy. Den Dienst nicht mit Root-Rechten betreiben. Abhängigkeiten nur aus vertrauenswürdigen Quellen installieren.

13. Geschützte Apps

Geschützte Inhalte getrennt vom öffentlichen Webroot speichern:

```
sudo mkdir -p /var/www/DOMAIN-private/apps/APP_NAME
sudo chown -R deploy:deploy /var/www/DOMAIN-private
sudo rsync -a /home/DEPLOY_USER/staging/APP_NAME/ /var/www/DOMAIN-private/apps/APP_NAME/
```

Nie nur den Ordner /apps/ verlinken, wenn dort keine index.html liegt. Immer den konkreten App-Pfad verwenden.

14. Anmeldung und TOTP - ausführlich

Der Schutz besteht aus zwei getrennten Aufgaben: Der Proxy schützt die Website und fragt Authentik, ob ein Benutzer freigegeben ist. Authentik führt den Anmelde-Flow aus und prüft Passwort sowie den Einmalcode der Authenticator-App.

14.1 Benutzer vorbereiten

1. In der Benutzerverwaltung für jede Person ein eigenes Konto anlegen.
2. Eine Gruppe, zum Beispiel **Tester**, erstellen.
3. Nur die gewünschten Benutzer der Gruppe hinzufügen.
4. Das Administratorkonto nicht als erstes Testkonto verwenden.

14.2 TOTP-Einrichtung anlegen

1. Einen **TOTP Authenticator Setup Stage** erstellen.
2. Sechsstellige Codes verwenden; das ist die kompatibelste Einstellung.
3. Einen **Authenticator Validation Stage** erstellen.
4. Bei Geräteklassen nur **totp** auswählen.
5. Bei „Not configured action“ **Configure** auswählen.
6. Den TOTP-Setup-Stage als Konfigurations-Stage eintragen.

14.3 Flow richtig sortieren

Im Authentication-Flow muss die Reihenfolge ungefähr so aussehen:

Reihenfolge	Stage	Aufgabe
1	Identification	Benutzername erkennen
2	Password	Passwort prüfen
3	Authenticator Validation	TOTP-Code prüfen
4	User Login	Sitzung freigeben

Die TOTP-Validation muss nach Passwort und vor User Login liegen. Eine Gruppenregel für Tester gehört direkt an das Stage Binding der TOTP-Validation, nicht unabsichtlich an den gesamten Flow.

14.4 Provider, Anwendung und Outpost

1. Einen Proxy Provider im Modus „Forward auth - einzelne Anwendung“ anlegen.
2. Als externe URL die geschützte HTTPS-Adresse eintragen.
3. Eine Anwendung erstellen und den Provider zuweisen.
4. Den verwalteten oder eigenen Outpost der Anwendung zuweisen.
5. In Nginx die frei erreichbare Weiterleitung **/outpost.goauthentik.io/** zum Outpost einrichten.
6. Danach `sudo nginx -t` und erst bei Erfolg `sudo systemctl reload nginx` ausführen.

14.5 Erster Benutzer-Test

1. Adminfenster geöffnet lassen.
2. Privates Browserfenster öffnen.

3. Geschützte URL aufrufen.
4. Mit einem Testbenutzer anmelden.
5. Beim ersten Mal QR-Code mit einer TOTP-App scannen.
6. Den sechsstelligen Code eingeben.
7. Prüfen, dass die geschützte Seite erst danach erscheint.
8. Abmelden und erneut testen.

Bildablauf: private Browserseite → Proxy → Authentik → Benutzername → Passwort → TOTP-App → Outpost → geschützte Website
Fehler vermeiden: Wird „Anfrage wurde verweigert“ angezeigt, zuerst Gruppen- und Stage-Bindings prüfen. Bei „auth request unexpected status: 404“ die Outpost-Weiterleitung prüfen. Bei einem alten „Server Apps“-Passwortdialog ist möglicherweise noch Basic Auth aktiv.

15. Backups

```
sudo mkdir -p /var/backups/mein-server
sudo tar -czf /var/backups/mein-server/web-$(date +%F).tar.gz /var/www/DOMAIN
sudo nginx -T > /var/backups/mein-server/nginx-$(date +%F).txt
```

Datenbank separat exportieren, Backups außerhalb des Webroots speichern und mindestens einmal testweise wiederherstellen. Snapshots ersetzen kein unabhängiges Datei-Backup.

16. Datenschutz und Impressum

Verantwortliche Stelle, Kontakt, Hosting, Cookies, Analyse, Protokollierung und Löschfristen ehrlich dokumentieren. Nur notwendige Cookies ohne Einwilligung einsetzen. Rechtliche Texte prüfen lassen.

17. Abschlussprüfung

1. Domain mit HTTPS öffnen.
2. Website im privaten Fenster testen.
3. App ohne Login aufrufen - sie muss gesperrt sein.
4. Mit Testkonto anmelden und TOTP prüfen.
5. Abmelden und erneut testen.
6. Nginx-Konfiguration testen.
7. Backup-Datei und freien Speicher prüfen.
8. Logs auf Fehler prüfen.

18. Fehlerbehebung

Domain nicht erreichbar

DNS, Schreibweise, HTTPS, Serverstatus, Firewall und ein anderes Netzwerk prüfen.

502 Bad Gateway

Bei Node-Diensten prüfen, ob der Dienst läuft und auf dem erwarteten lokalen Port lauscht. Nginx-Reverse-Proxy und Firewall prüfen.

500/404 bei Anmeldung

Provider, Anwendung, Outpost und /outpost.goauthentik.io/ prüfen. Danach Logs ansehen und Konfiguration testen.

Passwort oder TOTP ungültig

Benutzername exakt prüfen, Uhrzeit des Smartphones automatisch synchronisieren, Passwort neu setzen und nicht zu viele Versuche nacheinander machen.

Upload fehlt

Mit find den tatsächlichen Pfad suchen und prüfen, ob die ZIP wirklich entpackt wurde.

19. Notfall

1. Betroffenen Benutzer deaktivieren.
2. Passwort und TOTP neu setzen.
3. Aktive Sitzungen beenden.
4. Letztes Backup sichern.
5. Fehlerhafte Konfigurationsänderung zurücknehmen.
6. Nach jeder Änderung privat testen.

Nie in öffentliche Dokumente: Passwörter, TOTP-Codes, SSH-Private-Keys, Datenbankpasswörter, API-Schlüssel, interne IPs und .env-Dateien.

Diese Anleitung gilt für Debian 13 Linux. Vor Weitergabe an das eigene System und den eigenen Sicherheitsbedarf anpassen. Feedback zu Fehlern und Verbesserungsvorschlägen über das eigene Kontaktformular geben.