

[zurück](#)

# Proxy-Server - Grundlagen (Forward Proxy & Reverse Proxy)

Ein Proxy-Server ist ein Vermittler zwischen Client und Server.

Er verändert, filtert oder kontrolliert den Datenverkehr und bietet Sicherheit, Anonymisierung oder Lastverteilung.

Es gibt zwei Hauptarten:

- **Forward Proxy**
- **Reverse Proxy**

Diese werden oft verwechselt – diese Seite erklärt die Unterschiede klar.

---

## 1. Forward Proxy

Der **Forward Proxy** steht vor den Clients und vermittelt deren Anfragen nach außen.

Clients → Proxy → Internet

ASCII:

```
Client → Forward Proxy → Internet
```

### Einsatzbereiche

- Unternehmensnetzwerke
- Jugendschutz / Filter
- Logging & Monitoring
- Caching (Webseiten lokal zwischenspeichern)
- Zugriffskontrolle / Whitelisting
- Anonymisierung (Tor, VPN)

### Beispiele

- Squid Proxy
- Microsoft Proxy / TMG (veraltet)

- Webfilter (Sophos, FortiGate, Palo Alto)
- Content-Filter-Proxy

## Funktionen

- URL-Filterung
- Benutzer- und Gruppenfilter (LDAP/AD)
- Caching
- Malware-Prüfung
- Zugriffskontrolle (welcher User darf wohin?)
- Traffic-Optimierung

## Vorteile

- zentral kontrollierter Internetzugang
- Bandbreitenersparnis (durch Caching)
- Sicherheit (Filter, Blocklisten)

## Nachteile

- zusätzliche Latenz
- Privacy-Themen
- komplexe Regelwerke

---

## 2. Reverse Proxy

Der **Reverse Proxy** schützt und verwaltet **Server**. Er steht vor Webanwendungen, APIs oder Diensten.

Internet → Reverse Proxy → interne Server

ASCII:

```
Internet → Reverse Proxy → Webserver / API / Container
```

## Einsatzbereiche

- Webhosting

- Microservices
- Cloud
- Docker-Stacks
- SSL/TLS-Management
- Load Balancing
- Sicherheit (WAF, Rate Limits)

## Beispiele

- Nginx
- Traefik
- HAProxy
- Apache mod\_proxy
- Envoy
- Cloudflare (Reverse Proxy CDN)

## Funktionen

- TLS-Terminierung (Zertifikate zentral verwalten)
- Routing nach URLs & Domains
- Load Balancing (L7)
- Caching
- Web Application Firewall (WAF)
- Rate Limiting
- Authentifizierung (OIDC, JWT, Basic Auth)
- Verbindung mehrerer Backend-Systeme

## Vorteile

- Backend-Server bleiben privat
- öffentliche IP muss nur der Reverse Proxy haben
- zentrale Zertifikatsverwaltung
- erhöht Sicherheit & Performance

## Nachteile

- ein Reverse Proxy = Single Point of Failure (wenn kein HA)
- HTTPS-Offloading braucht CPU

---

# 3. Unterschiede Forward vs Reverse Proxy

| Merkmal  | Forward Proxy                 | Reverse Proxy                            |
|----------|-------------------------------|--|
| Position | vor dem Client                | vor dem Server                           |
| Zweck    | Kontrolle des Client-Traffics | Schutz & Optimierung von Backend-Servern |

| Merkmal           | Forward Proxy            | Reverse Proxy                |
|-------------------|--------------------------|------------------------------|
| Beispiele         | Squid, Webfilter         | Nginx, Traefik, HAProxy      |
| Benutzer          | Clients im LAN           | externe Benutzer (Internet)  |
| Sicherheit        | Filtert outgoing Traffic | schützt Server vor Angriffen |
| Authentifizierung | Nutzer-Login             | Webservice-Login, OIDC, JWT  |
| Caching           | Webseiten von außen      | Inhalte der eigenen Server   |

Kurz:

- Forward Proxy = „Ich gehe über den Proxy ins Internet.“
- Reverse Proxy = „Das Internet geht über den Proxy auf meine Server.“

## 4. SSL/TLS-Offloading (Reverse Proxy)

Reverse Proxys übernehmen oft die Entschlüsselung von HTTPS.

ASCII:

```
Client → HTTPS → Reverse Proxy → HTTP → Backend
```

Vorteile:

- weniger Last auf Backend-Servern
- ein zentraler Ort für Zertifikate
- bessere Sichtbarkeit für IDS/IPS
- L7 Routing möglich

## 5. Reverse Proxy in Docker-Umgebungen

Moderne Stacks nutzen Reverse Proxys intensiv.

Beispiel Aufbau:

```
Traefik →  
  /portainer  
  /vaultwarden
```

```
/nextcloud  
/matrix  
/mail
```

Funktionen:

- Weiterleitung nach Subdomain
- TLS mit ACME
- Middlewares (Redirect, Auth, IP-Whitelist)

---

## 6. Sicherheit durch Reverse Proxy

- DDoS-Reduzierung
- Rate Limits
- Web Application Firewall
- Header-Hardening (HSTS, X-Frame, CSP)
- IP-Blocklisten (z. B. CrowdSec)
- geobasiertes Blocking
- Bot Detection

---

## 7. Content Caching

Reverse Proxys können Inhalte zwischenspeichern:

- Bilder
- statische Inhalte
- API-Antworten

Dadurch wird der Backend-Server entlastet.

---

## 8. Forward Proxy vs NAT

Viele verwechseln es:

- NAT = Netzwerkadressübersetzung (Layer 3)
- Proxy = Anwenderprotokolle werden aktiv vermittelt (Layer 7 oder L4)

Proxy ist *viel höher* in der Protokollkette angesiedelt.

# Zusammenfassung

- Forward Proxy → kontrolliert ausgehenden Traffic
- Reverse Proxy → schützt und verteilt eingehenden Traffic
- Reverse Proxy ist standard in Docker & Webhosting
- Traefik / Nginx / HAProxy → mächtige Reverse Proxys
- Forward Proxys filtern, cachen und steuern Benutzerzugriffe
- Reverse Proxys übernehmen TLS, Routing, Auth & WAF

From:  
<http://wiki.nctl.de/dokuwiki/> - ☐ **Veni. Vidi. sudo rm -rf / vici.**

Permanent link:  
<http://wiki.nctl.de/dokuwiki/doku.php?id=it-themen:grundlagen:netzwerkdienste:proxy>

Last update: **04.12.2025 10:47**

