ADMIN:DOCS

IPv4- und IPv6-Subnetting verstehen: Präfixe und Netzgrenzen einfach erklärt

Lerne, wie du IPv4- und IPv6-Subnetting verstehst und präzise Netzgrenzen festlegst: Präfixe, Netzmasken und Adressbereiche für Linux-Infrastrukturen methodisch einteilen und am Prompt verifizieren.

Kategorie: Netzwerk **Veröffentlicht:** 17.10.2026

Sobald du ein Netzwerk aufbaust, stehst du schnell vor derselben Hürde: Arbeitsplatzrechner, Server und Gäste sollen nicht unkontrolliert durcheinanderfunken, sondern eigene Bereiche bekommen. Spätestens wenn du deinem Router oder DHCP-Server mitteilen musst, wo diese Netze anfangen und aufhören, stolperst du über Angaben wie `/24` oder `/64`.

Eine IP-Adresse ist keine reine Hausnummer. Sie erledigt immer zwei Aufgaben auf einmal: Der vordere Teil bestimmt das Netzwerk, der hintere die konkrete Schnittstelle darin. Das Präfix zieht genau diese Trennlinie. Setzt du diese Grenze falsch, überschneiden sich deine Adressbereiche oder Geräte suchen ihr Ziel auf der falschen Seite des Gateways.

Das Grundprinzip funktioniert bei `IPv4` und `IPv6` identisch, der praktische Umgang damit unterscheidet sich jedoch grundlegend: Bei `IPv4` teilst du knappe Adressblöcke passgenau auf und rechnest aus, wie viele Schnittstellen überhaupt hineinpassen. Bei `IPv6` zählst du keine Hostadressen, sondern nimmst ein zugewiesenes Präfix und zerlegst es direkt in mehrere /64-Subnetze.

Warum Netzwerke in Subnetze aufteilen?

Wenn du ein Dutzend Rechner, zwei Drucker und einen Server an denselben Switch steckst, funktioniert das meistens auf Anhieb. Vergib fortlaufende IP-Adressen, und jedes Gerät kann

Quelle: https://admindocs.de/de/netzwerk/ipv4-und-ipv6-subnetting-verstehen-praefixe-und-netzgrenzen-einfach-erklaert

jedes andere erreichen. Das fühlt sich bequem an, verwandelt das Netzwerk mit wachsender Geräteanzahl aber schnell in eine unübersichtliche Rumpelkammer.

Der wichtigste Grund für Subnetze liegt darin, wie das Betriebssystem über den Weg eines Datenpakets entscheidet. Will ein Rechner Daten verschicken, vergleicht der Kernel die Ziel-IP mit der eigenen Adresse und der konfigurierten Subnetzmaske. Wie die TCP/IP-Protokollfamilie Daten strukturiert und über Schnittstellen weiterleitet, bildet dafür das Fundament. Liegt das Ziel im selben Subnetz, ignoriert er das Standard-Gateway komplett. Er fragt per ARP direkt ins lokale Netz nach der MAC-Adresse des Ziels und schickt den Frame unmittelbar über den Switch ab. Bei `IPv6` funktioniert das vergleichbar, allerdings nutzt das Betriebssystem hier das Neighbor Discovery Protocol nach `RFC 4861` anstelle von ARP.

Das bedeutet für die Praxis:

Solange also die Arbeitsplatz-Rechner, Gäste und interne Server im selben Subnetz hängen, bekommt deine Firewall von dieser Kommunikation nicht ein einziges Bit zu sehen. Die Systeme unterhalten sich direkt auf der OSI Schicht 2. Wenn du Zugriffe regeln, bestimmte Dienste sperren oder Verbindungen protokollieren willst, musst du die Geräte in unterschiedliche Subnetze stecken. Erst wenn der Kernel erkennt, dass die Ziel-IP außerhalb des eigenen Netzes liegt, übergibt er das Paket an das Gateway – und erst dort hast du mit Firewall-Regeln überhaupt eine Handhabe.

Dazu kommt das ständige Grundrauschen.

In einem ungeteilten IPv4-Netz wird jeder ARP-Request ("Wer hat IP 192.0.2.50?"), jeder DHCP-Ruf und jede automatische Dienstsuche als Broadcast an ausnahmslos alle Ports weitergeleitet. Jede einzelne Netzwerkkarte muss den Frame entgegennehmen und der Kernel muss ihn verarbeiten, nur um festzustellen, dass die Anfrage für ein ganz anderes System gedacht war. Bei dreißig Geräten fällt das kaum ins Gewicht; bei Hunderten erzeugt dieser Verkehr spürbare Last auf den Schnittstellen.

> □□ **Ein klassischer Denkfehler:** Ein Subnetz ist eine reine logische Konvention auf Schicht 3, keine Sicherheitsmauer. Wenn du dem Gastnetz die Adressen `192.0.2.128/25` gibst und deinen internen Servern `192.0.2.0/25`, beide aber ungetrennt auf demselben

> Standard-Switch laufen, ist technisch nichts geschützt.
>
> Jeder Teilnehmer mit administrativen Rechten auf seinem Gerät kann sich manuell eine zweite IP aus dem Server-Bereich eintragen oder Datenverkehr mitschneiden. Subnetting definiert nur die logischen Grenzen fürs Routing – die echte Trennung darunter braucht eigene physische Schnittstellen oder sauber konfigurierte VLANs.

Subnetze schaffen also die logische Struktur, auf der Router und Paketfilter aufsetzen können. Sie sorgen dafür, dass Systeme nicht unkontrolliert aneinander vorbeireden, ersetzen für sich allein aber keine Netzwerk-Isolation.

IP-Adresse und Präfix: Wo endet das Netz?

Wenn du auf eine IP-Adresse schaust – egal ob `192.0.2.45` bei IPv4 oder `2001:db8::45` bei IPv6 –, wirkt das zunächst wie eine einzelne, zusammenhängende Nummer. Für dein Betriebssystem ist sie das aber nicht.

Der Kernel teilt jede Adresse gedanklich immer in zwei Hälften:

- **Der Netzanteil:** Die vorderen Bits. Sie bestimmen, zu welchem logischen Teilnetz das System gehört. Jede Schnittstelle im selben Subnetz teilt sich exakt denselben Anfang.
- **Der Hostanteil:** Die verbleibenden Bits am Ende. Sie bilden die individuelle Kennung der konkreten Schnittstelle innerhalb genau dieses Netzes.

An der reinen Adresse `192.0.2.45` kannst du allerdings nicht ablesen, wo dieser Schnitt verläuft. Sie ist ohne Zusatzinformation schlicht unvollständig. Erst das Präfix oder die Subnetzmaske teilt dem System mit, wie viele Bits von links gezählt das Netz festlegen und ab welchem Bit die einzelnen Hosts adressiert werden.

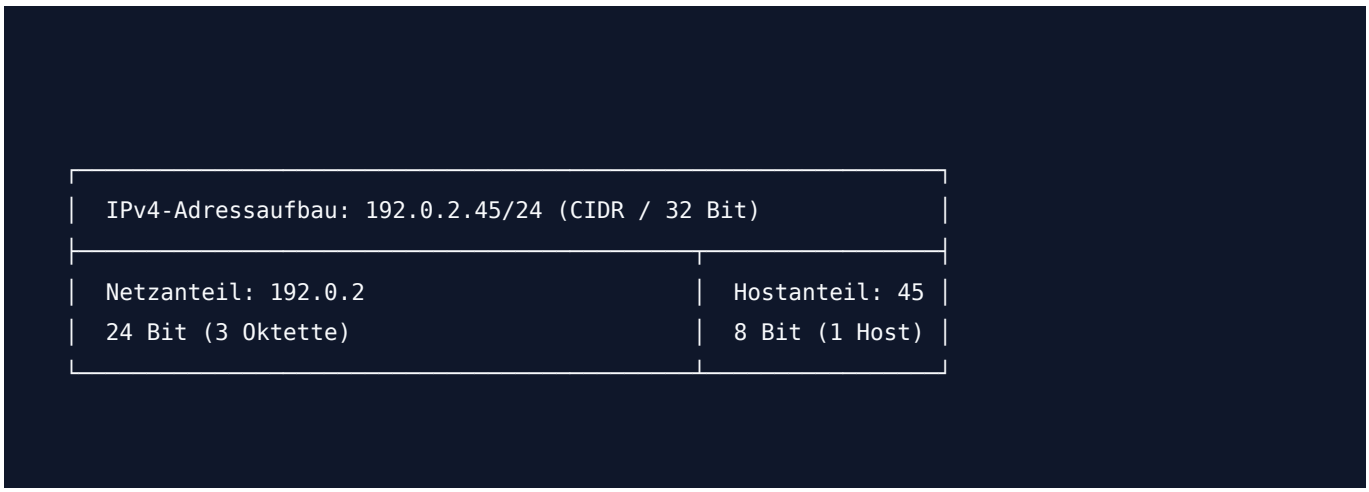
Schrägstrich statt Zahlenkolonne:

Die CIDR-Notation

Im Alltag begegnet dir heute fast überall die Schreibweise mit dem Schrägstrich, die CIDR-Notation (RFC 4632){.badge-link-text}. Sie spart Tipparbeit und nennt direkt die Anzahl der Bits,

die für das Netz reserviert sind.

Eine IPv4-Adresse ist exakt 32 Bit lang, aufgeteilt in vier Blöcke (Oktette) zu je 8 Bit:

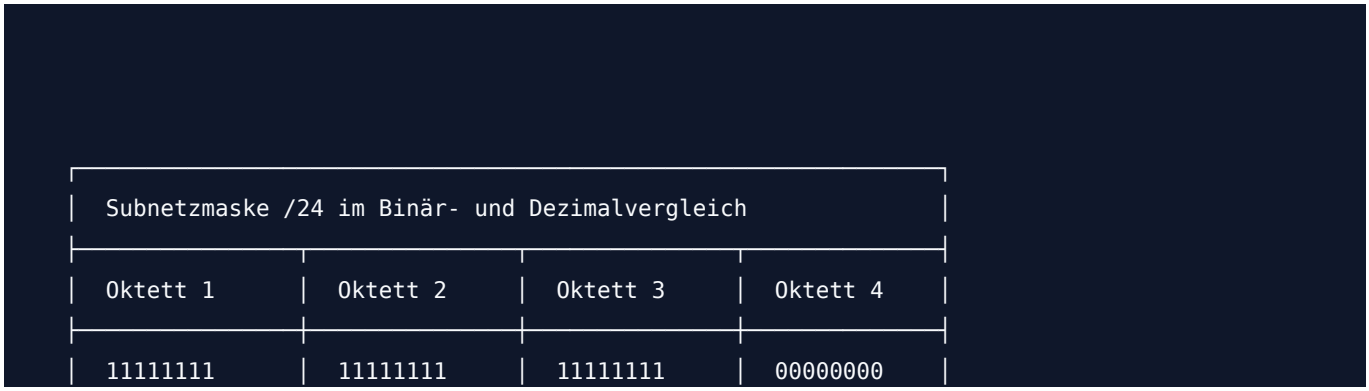


Die Angabe `/24` ist eine eindeutige Anweisung an den Kernel: Die ersten 24 Bits – also exakt die ersten drei Zahlen `192.0.2` – gehören dem Netz. Jedes System, dessen Adresse mit diesen drei Zahlen beginnt, liegt im selben Segment. Für deine Schnittstellen bleiben die letzten 8 Bits übrig, also die Werte von `0` bis `255` im vierten Block.

Warum dir trotzdem noch 255.255.255.0 begegnet

In vielen Web-Oberflächen von Routern oder älteren Konfigurationsdateien stolperst du statt `/24` immer noch über `255.255.255.0`. Das ist keine andere Logik, sondern exakt dieselbe Information in einer unhandlicheren Schreibweise.

Die Subnetzmaske ist ein 32-Bit-Filter, der von links nach rechts mit Einsen aufgefüllt wird. Wo eine `1` steht, gehört das Bit zum Netz; wo eine `0` steht, darfst du Schnittstellen adressieren:





Ein einzelnes Oktett besteht aus 8 Bits. Schaltest du alle 8 Bits ein (lauter Einsen), ergibt das nach den Regeln des Binärsystems zusammengerechnet den maximal möglichen Wert: 255 (128 + 64 + 32 + 16 + 8 + 4 + 2 + 1). Die Maske `255.255.255.0` sagt dem Betriebssystem also dreimal hintereinander: *Diese 8 Bits sind voll für das Netz belegt* – macht zusammen 24 Bits.

`/24` und `255.255.255.0` meinen haargenau dasselbe. Solange die Netzgrenze wie hier glatt an einem Punkt zwischen den Oktetten liegt, lässt sie sich mit bloßem Auge ablesen. Rechnerisch interessant wird es erst, wenn die Trennlinie mitten durch einen 8-Bit-Block schneidet.

IPv6: Gleiches Spiel, sauberere Kanten

Bei IPv6 hat man den Ballast der ausgeschriebenen Masken komplett gestrichen.

Hier gibt es ausschließlich die Präfix-Schreibweise.

Eine IPv6-Adresse ist 128 Bit lang (RFC 4291){.badge-link-text}. Steht dort ein `/64`, bedeutet das: Die vordere Hälfte (64 Bits) ist das Subnetz, die hinteren 64 Bits bilden den Interface Identifier für das Interface. Da vier Hexadezimalziffern immer einen 16-Bit-Block bilden, fällt ein `/64` exakt hinter den vierten Block. Du musst also nicht mitten in einer Hex-Ziffer rechnen, sondern hast eine saubere, lesbare Grenze.

IPv4-Subnetting Schritt für Schritt

Bisher lag die Grenze bequem hinter der dritten Zahl. Ein `/24`-Netz hat 256 Adressen – von `0` bis `255`. Wenn du dieses Netz nun aufteilst, nimmst du schlicht ein Messer und schneidest diesen Block in kleinere Stücke.

Nehmen wir als praktisches Beispiel das offizielle Dokumentationsnetz `192.0.2.0/24` (RFC 5737){.badge-link-text}. In realen Unternehmensnetzen nutzt du für interne Segmente stattdessen öffentliche vs. private IP-Adressen nach RFC 1918 (wie `10.0.0.0/8` oder `192.168.0.0/16`). Die ersten drei Blöcke (`192.0.2.`) fassen wir in unserem Beispiel gar nicht an.

Quelle: https://admindocs.de/de/netzwerk/ipv4-und-ipv6-subnetting-verstehen-praefixe-und-netzgrenzen-einfach-erklaert

Alles, was jetzt passiert, spielt sich ausschließlich in der letzten Zahl ab.

Das Netz halbieren: Aus /24 wird /25

Wenn du aus einem Netz zwei getrennte Bereiche machen willst, schiebst du die Trennlinie um genau ein Bit nach rechts: aus `/24` wird somit `/25`.

Ein Bit kann genau zwei Zustände haben: `0` oder `1`.

Damit halbiert sich der bisherige Raum:

- Bisher (`/24`): 256 Adressen in einem einzigen Topf.
- Jetzt (`/25`): Zwei getrennte Töpfe mit jeweils 128 Adressen.

Weil das erste Bit im letzten Oktett für den Wert `128` steht, teilt sich der Bereich exakt an dieser Marke: Der erste Block läuft von `0` bis `127`, der zweite von `128` bis `255`. In der alten Masken-Schreibweise wird aus `255.255.255.0` jetzt `255.255.255.128`.

> **Wichtige Grundregel:** Du kannst in einem IPv4-Subnetz niemals alle vorhandenen Adressen an Rechner vergeben. Die allererste und die allerletzte Adresse eines Blocks sind immer fest für das Protokoll reserviert.

Zwei Adressen sind immer reserviert:

- **Die Netzadresse (die erste Zahl):** Sie identifiziert das Subnetz selbst.
- **Die Broadcast-Adresse (die letzte Zahl):** Schickst du ein Paket an diese Adresse, geht es als Rundruf an ausnahmslos alle Geräte im Subnetz (RFC 919){.badge-link-text}.

Subnetz	Adressbereich	Nutzbar für Schnittstellen	Broadcast
1. Netz (`192.0.2.0/25`)	`192.0.2.0` – `192.0.2.127`	`192.0.2.1` bis `192.0.2.126` (126 IPs)	`192.0.2.127`
2. Netz (`192.0.2.128/25`)	`192.0.2.128` – `192.0.2.255`	`192.0.2.129` bis `192.0.2.254` (126 IPs)	`192.0.2.255`

Quelle: https://admindocs.de/de/netzwerk/ipv4-und-ipv6-subnetting-verstehen-praefixe-und-netzgrenzen-einfach-erklaert

Versuchst du auf einem Linux-System, einer Schnittstelle die `.0` oder `.127` als statische IP zuzuweisen, meckert der Kernel entweder direkt oder du wunderst dich später über unberechenbares Routing. In der Praxis teilst du den nutzbaren Bereich sauber auf: Gateways und Server erhalten feste IPs, während Clients dynamisch per DHCP bedient werden – wie du das planst, zeigt der Überblick zu dynamischen vs. statischen IP-Adressen. Zieh im Kopf von der Blockgröße einfach immer `2` ab, dann hast du deine nutzbaren Schnittstellen.

Weiter zerlegen: Aus /24 werden vier /26-Netze

Brauchst du mehr als zwei getrennte Bereiche – zum Beispiel für dein Büro, dein Server, dann für Gäste und Management –, halbierst du die Blöcke einfach noch einmal. Du leihst dir ein zweites Bit: Aus `/25` wird `/26`.

- Ein Bit mehr verdoppelt die Anzahl der Netze: Aus `2` Netzen werden `4`.
- Gleichzeitig halbiert sich die Größe jedes Blocks: Aus 128 Adressen werden 64 Adressen pro Netz.

Die Blockgröße ist jetzt also 64.

Damit musst du gar nichts mehr umständlich in Binärzahlen umrechnen, sondern zählst einfach im 64er-Takt hoch: `0`, `64`, `128`, `192`. Das sind deine vier Netzadressen.

Schauen wir uns das zweite Viertel (`192.0.2.64/26`) im Detail an:

- Die **Netzadresse** ist die `192.0.2.64`.
- Die **Broadcast-Adresse** ist die letzte Zahl vor dem nächsten Block, also die `192.0.2.127`.
- Dazwischen liegen deine **nutzbaren Hosts:** `192.0.2.65` bis `192.0.2.126` – exakt 62 Adressen (64 − 2).

Der Schnelltest im Kopf

Wenn du im Alltag auf einem Server die Konfiguration `192.0.2.135/26` siehst, weißt du sofort, wo er hingehört:

Die Schrittweite ist hier `64`. Die Blöcke starten bei `0`, `64`, `128` und `192`. Die Zahl `135` liegt sauber im dritten Block zwischen `128` und `191`.

Quelle: https://admindocs.de/de/netzwerk/ipv4-und-ipv6-subnetting-verstehen-praefixe-und-netzgrenzen-einfach-erklaert

Siehst du auf einem anderen Rechner die `192.0.2.70/26`, liegt er im Block davor (64 bis 127). Ohne einen Router dazwischen werden die beiden kein einziges Paket austauschen können, obwohl sie im selben Raum an denselben Dosen hängen.

IPv6-Subnetting: Präfixe statt Hostadressen zählen

Vergiss bei `IPv6` erst einmal alles, was du dir bei `IPv4` mühsam über Hostbereiche gemerkt hast. Bei IPv4 knauserst du mit jedem einzelnen Bit, ziehst die Zwei für Netz und Broadcast ab und überlegst dreimal, ob dir 62 Adressen für ein Segment reichen.

Bei IPv6 funktioniert die Welt völlig anders:

Du zählst keine einzelnen Schnittstellen mehr – du zählst ganze Netze.

Der Standard für jedes reguläre Subnetz, ob nun LAN, WLAN oder ein Server-Segment, ist ein `/64` Subnetz. Daran wird im Normalfall auch nicht gerüttelt. Ein einzelnes `/64`-Subnetz umfasst unvorstellbare 2^{64} (über 18 Trillionen) Adressen in einem einzigen Segment. Du denkst dir bestimmt jetzt, das klingt nach gigantischer Verschwendung, oder? **Das hat aber einen ganz praktischen technischen Grund:** Autokonfigurations-Mechanismen wie SLAAC (Stateless Address Autoconfiguration, RFC 4862){.badge-link-text} setzen exakt diese 64 Bit für den Hostteil (die Interface ID) voraus. Schneidest du ein Netz kleiner als `/64`, bricht diese automatische Adressvergabe für Endgeräte schlichtweg ab.

Deine Aufgabe als Administrator ist bei IPv6 also nicht, ein Subnetz passgenau auf zwanzig Rechner zuzuschneiden. Deine Aufgabe ist es, einen größeren Block, den du von deinem Provider oder deiner IT-Abteilung bekommst, in mehrere `/64`-Häppchen zu zerlegen.

Ein typisches Szenario

Aus einem /56 machst du dein Netzwerk

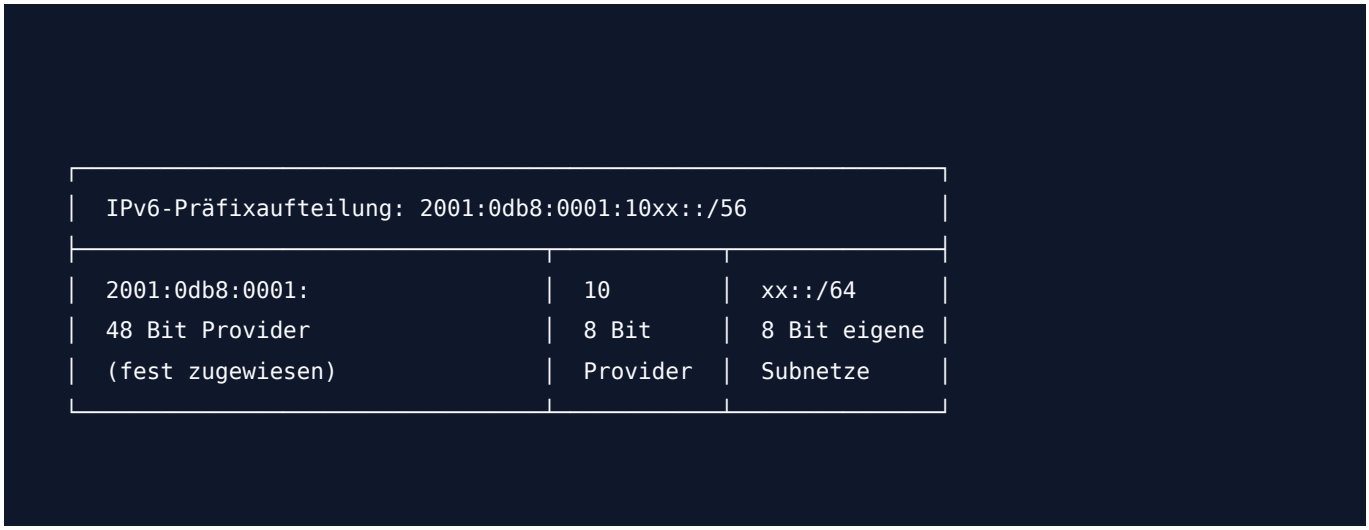
Welche Blockgröße dir zur Verfügung steht, entscheidet der Provider: Manche Anschlüsse erhalten ein großzügiges `/48`, viele Internetanschlüsse werden mit einem `/56` geschaltet, und manche Tarife knausern mit einem einzelnen `/64`.

Nehmen wir als praktisches Beispiel an, dein Router erhält ein `/56`-Präfix aus dem offiziellen Dokumentationsbereich (RFC 3849){.badge-link-text}: **`2001:db8:1:1000::/56`**

Eine IPv6-Adresse besteht aus acht Blöcken zu je 16 Bit, getrennt durch Doppelpunkte. Jedes einzelne Zeichen ist eine Hexadezimalziffer (0 bis 9 und a bis f) und entspricht exakt 4 Bit:

- Die ersten drei Blöcke (`2001:db8:1:`) sind komplett belegt: 3 × 16 = 48 Bit.
- Bis zur Grenze von `/56` fehlen noch genau 8 Bit (56 − 48 = 8). Das entspricht exakt den ersten beiden Hex-Ziffern im vierten Block.

Schauen wir uns den vierten Block ungekürzt mit seinen vier Hex-Stellen an, siehst du die Kante sofort:



Bis zum `/56` ist der Bereich fest durch den Provider vorgegeben – daran kannst du nichts ändern. Doch zwischen deiner Zuteilung (`/56`) und dem Standard-Subnetz (`/64`) liegen genau 8 freie Bits:

64 − 56 = 8 Bit

Mit diesen 8 Bits kannst du $2^8 = 256$ unterschiedliche Zustände darstellen. Du baust aus diesem einen Anschluss also ohne jede Binärrechnerei exakt 256 eigenständige `/64`-Subnetze.

Die Hex-Stellen hochzählen

Weil 8 Bit genau zwei Hex-Ziffern entsprechen, bewegst du schlicht die beiden hinteren Zeichen im vierten Block von `00` bis `ff` (also dezimal von 0 bis 255):

Quelle: https://admindocs.de/de/netzwerk/ipv4-und-ipv6-subnetting-verstehen-praefixe-und-netzgrenzen-einfach-erklaert

Subnetz	Hex-Stelle	Typischer Einsatzbereich	Zuweisung (`/64`)
Subnetz 0	`00`	Management & Infrastruktur	`2001:db8:1:1000::/64`
Subnetz 1	`01`	Büro-Clients	`2001:db8:1:1001::/64`
Subnetz 2	`02`	Server-Segment	`2001:db8:1:1002::/64`
Subnetz 3	`03`	Gäste-WLAN	`2001:db8:1:1003::/64`
Subnetz 10	`0a`	Beliebige Erweiterung	`2001:db8:1:100a::/64`
Subnetz 255	`ff`	Letzter Subnetz-Block	`2001:db8:1:10ff::/64`

Jedes dieser 256 Netze ist ein völlig autarkes Segment. Du weist der VLAN-Schnittstelle deiner Firewall für das Gastnetz beispielsweise die Adresse `2001:db8:1:1003::1/64` zu. Der Router verteilt das Präfix `2001:db8:1:1003::/64` per Router Advertisement ins Gastnetz, und die Geräte dort erzeugen sich ihre eigenen Schnittstellen-Adressen selbst.

Beim Aufschreiben von IPv6-Adressen dürfen führende Nullen innerhalb eines 16-Bit-Blocks weggelassen werden (aus `:0001:` wird `:1:`). In unserem Beispiel steht im vierten Block jedoch `1001` – die Eins am Anfang bleibt stehen, wodurch die Struktur der Subnetze optisch übersichtlich und auf einen Blick vergleichbar bleibt.

Keine Broadcast-Adresse, keine Kopfschmerzen

Ein handfester Vorteil im Alltag:

Bei IPv6 gibt es keine Broadcast-Adresse mehr. Niemand muss sich merken, welche Adresse die letzte im Bereich ist, um sie bloß nicht an einen Host zu vergeben. Rundrufe an alle gibt es nicht mehr; wenn Systeme alle Nachbarn ansprechen wollen, nutzen sie gezielte Multicast-Gruppen (RFC 4291){.badge-link-text}.

Die allererste Adresse eines Subnetzes (also lauter Nullen im Interface-Teil, wie `2001:db8:1:1001::0`) ist zwar formal als *Subnet-Router Anycast* reserviert, aber darüber musst du im Betrieb nicht lange grübeln. Du gibst deinem Gateway meist pragmatisch die `::1`, deinen statischen Servern feste Endungen wie `::10` oder `::53`, und überlässt den Rest der

Quelle: https://admindocs.de/de/netzwerk/ipv4-und-ipv6-subnetting-verstehen-praefixe-und-netzgrenzen-einfach-erklaert

Adressierung der Autokonfiguration.

Merkhilfe: Ein Bit mehr, doppelt so viele Netze

Wenn du dir beim Subnetting eine einzige Faustformel dauerhaft merken willst, dann diese: **Verschiebst du die Grenze um ein Bit nach rechts, verdoppelt sich die Anzahl deiner Netze – und jedes einzelne Netz wird genau halb so groß.**

Das ist kein Zufall, sondern die elementare Logik des Binärsystems. Ein zusätzliches Bit kennt exakt zwei Zustände (`0` oder `1`). Sobald du dir dieses Bit nimmst, teilt es jeden vorhandenen Block sauber in zwei Hälften.

Präfix-Verschiebung	Anzahl der Teilnetze	Adressen pro Subnetz
Präfix +1 Bit	Netze × 2	Adressen pro Netz ÷ 2
Präfix +2 Bit	Netze × 4	Adressen pro Netz ÷ 4
Präfix +3 Bit	Netze × 8	Adressen pro Netz ÷ 8

Die Regel an unseren Beispielen durchgespielt

Schau dir an, was wir in den vorherigen Abschnitten gerechnet haben – es ist bei beiden Protokollen dieselbe Mechanik:

- **IPv4:** Du startest mit einem `/24` (256 Adressen).
- Machst du ein `/25` daraus (+1 Bit), verdoppeln sich die Netze von 1 auf 2. Die Blockgröße halbiert sich von 256 auf 128.
- Gehst du auf ein `/26` (+2 Bit gegenüber `/24`), verdoppelt sich das noch einmal: 4 Netze mit jeweils 64 Adressen.

- **IPv6:** Dein Router bekommt ein `/56`.
- Ein Subnetz hat `/64`. Der Abstand beträgt 64 − 56 = 8 Bit.
- Verschiebst du die Grenze um 8 Bit, verdoppelst du den Raum achtmal hintereinander: 2 × 2 × 2 × 2 × 2 × 2 × 2 × 2 = 256 Subnetze.

Der häufigste Denkfehler: Große Zahl gleich großes Netz?

Vor allem Einsteiger fallen regelmäßig auf eine optische Täuschung herein:

Sie sehen ein `/28` und denken unwillkürlich, das sei „mehr“ oder „größer“ als ein `/24`.

Die Zahl hinter dem Schrägstrich zählt die Bits, die für das **Netz** gesperrt sind – nicht den Platz für deine Rechner. Eine größere Zahl bedeutet: Das Netz beansprucht mehr Bits für sich, also bleibt weniger Raum für Schnittstellen übrig. Ein `/28` ist kein großes Netz, sondern ein winziges Viertel eines `/26` mit gerade einmal 16 Adressen (14 nutzbaren Hosts).

> □□ Merk dir den Schrägstrich einfach wie eine Schranke auf einer 32- oder 128-spurigen Autobahn: Je weiter die Schranke nach rechts rückt (größere Zahl), desto schmaler wird der Streifen, der für deine Endgeräte übrig bleibt.

Selbst prüfen: Zwei typische Praxisfallen

Auf dem Papier sieht Subnetting oft logisch aus. Richtig interessant wird es aber erst, wenn du vor der Shell oder im Webinterface sitzt und eine IP eintippen musst. Hier sind zwei klassische Situationen aus dem Alltag, an denen du deinen Blick für Netzgrenzen direkt testen kannst.

Fall 1: Der neue Server im IPv4-Segment

Stell dir vor, du sollst einen neuen Backup-Server in ein bestehendes Server-VLAN hängen. In der Dokumentation deines Vorgängers steht für dieses Segment schlicht: **`Netz: 198.51.100.128/27`**

Weil du dir die Adressen gerne sauber merkst, willst du dem neuen Server die glatte IP `198.51.100.160` geben. Klingt auf den ersten Blick nach einer freien, unverfänglichen Nummer.

Bevor du aber die IP in die Netzwerkkonfiguration schreibst:

- Funktioniert das in diesem Subnetz?
- Und wo liegen die Grenzen?

Die Auflösung am Prompt:

Quelle: https://admindocs.de/de/netzwerk/ipv4-und-ipv6-subnetting-verstehen-praefixe-und-netzgrenzen-einfach-erklaert

Wenn du die `/27` siehst, machst du im Kopf denselben kurzen Dreisatz wie zuvor:

1. **Die Schrittweite ermitteln:** Bis zur vollen `/24` fehlen genau 3 Bits (27 − 24 = 3). Zählst du die Wertigkeiten im Oktett herunter (128 → 64 → 32), landest du bei einer Schrittweite von **32**. Das bedeutet: Alle Subnetze in diesem Bereich sind exakt 32 Adressen groß.
2. **Die Grenzen abstecken:** Das Netz beginnt laut Dokumentation bei `.128`. Zählst du 32 Zähler weiter, startet das nächste Teilnetz exakt bei `198.51.100.160` (128 + 32).
3. **Der nutzbare Bereich:**

- Die allererste Adresse (`.128`) ist die Netzadresse.
- Die allerletzte Zahl vor dem nächsten Block (`.159`) ist die Broadcast-Adresse.
- Deine nutzbaren Server-IPs reichen also exakt von **`198.51.100.129`** **bis** **`198.51.100.158`** – insgesamt 30 Adressen (32 − 2).

Warum die `.160` eine Bruchlandung wäre:

Die `.160` gehört überhaupt nicht mehr in dieses Subnetz. Sie ist bereits die offizielle Netzadresse des *darauffolgenden* `/27`-Blocks. Trägst du dem Server `198.51.100.160/27` ein, glaubt sein Kernel, er befände sich in einem ganz anderen Netz. Versucht er nun, das Standard-Gateway des Server-Segments (das meist auf `.129` liegt) direkt per ARP zu erreichen, weigert sich der Stack: Das Gateway liegt aus seiner Sicht im fremden Subnetz, und ohne passenden Router dazwischen geht kein einziges Paket raus.

Fall 2: Der Router-Anschluss mit /60-Präfix

Ein Kollege richtet eine Außenstelle ein. Der dortige Internetanbieter teilt dem Router kein großzügiges `/56`, sondern ein sparsameres `/60`-Präfix aus dem Dokumentationsbereich zu: **`2001:db8:beef:00a0::/60`**

Deine Aufgabe ist es also, du sollst für den Standort vier getrennte Zonen einrichten:

- Management
- Arbeitsplätze
- Gast-WLAN
- Überwachungskameras

Quelle: https://admindocs.de/de/netzwerk/ipv4-und-ipv6-subnetting-verstehen-praefixe-und-netzgrenzen-einfach-erklaert

Die Frage lautet: **Reicht das zugewiesene `/60` überhaupt aus, um vier vollwertige `/64`-Subnetze zu bauen? Und wie lauten die konkreten Präfixe für die Zonen?**

Die Auflösung ohne Rechnerei:

Bei IPv6 vergisst du sofort das Zählen von Hostadressen und schaust nur auf den Abstand zum `/64`:

1. **Die freien Bits:** Zugewiesen ist ein `/60`, deine Zielnetze brauchen für sauberes Routing und SLAAC ein `/64`. Der Unterschied beträgt exakt:

64 − 60 = 4 Bit

1. **Anzahl der möglichen Netze:** Ein Bit mehr verdoppelt den Raum. Bei 4 Bits bedeutet das: $2^4 = 16$. Du kannst aus diesem Anschluss also problemlos **16 eigenständige `/64`-Netze** herausschneiden. Vier Zonen sind damit locker abgedeckt.
2. **Die Präfixe bilden:** 4 Bits entsprechen im Hexadezimalsystem exakt einer einzigen Ziffer ($2^4 = 16$, von `0` bis `f`). Im vierten Block (`00a0`) ist also schlicht die letzte Ziffer dein eigener Zähler.

Schreiben wir die Zonen direkt sauber auf (führende Nullen im Block lassen wir wie üblich weg, aus `00a0` wird `a0`):

- **Management:** `2001:db8:beef:a0::/64` (Zähler 0)
- **Arbeitsplätze:** `2001:db8:beef:a1::/64` (Zähler 1)
- **Gäste-WLAN:** `2001:db8:beef:a2::/64` (Zähler 2)
- **Kameras:** `2001:db8:beef:a3::/64` (Zähler 3)

Bis `2001:db8:beef:af::/64` hättest du noch Luft für zwölf weitere Netze. Du siehst: Sobald du in Hex-Stellen denkst, musst du bei IPv6 weder Potenzen wälzen noch Masken umrechnen – du zählst einfach die letzte Ziffer hoch.

Befehlsreferenz (Cheatsheet)

Wenn du vor einer Linux-Kiste sitzt und wissen willst, in welchem Netz das System tatsächlich gelandet ist, helfen dir keine bunten Web-UIs. Auf der Konsole siehst du direkt, was der Kernel

konfiguriert hat und wo seine Routing-Grenzen liegen.

Die alten `net-tools` (`ifconfig`, `route`) gehören seit Jahren aufs Altenteil. Unter modernem Linux regelt das ausschließlich `iproute2`. Hier sind die wenigen Befehle, die du für die Diagnose von Subnetzen und Schnittstellen wirklich brauchst:

Befehl	Wofür du ihn brauchst	Worauf du in der Ausgabe achten musst
`ip -4 address show`	Zeigt alle konfigurierten IPv4-Adressen an.	Hinter `inet` steht deine IP samt Präfix (z. B. `198.51.100.130/27`). Direkt daneben siehst du mit `brd` die vom Kernel automatisch berechnete Broadcast-Adresse für dieses Segment.
`ip -6 address show`	Zeigt alle IPv6-Adressen des Systems.	Hinter `inet6` siehst du deine globale Unicast-Adresse samt Präfix (meist `/64`). Achte auf `scope global` (fürs Routing im Netz) und unterscheide sie von der `fe80::`-Adresse (`scope link`), die nur lokal auf dem Kabel gilt.
`ip route show`	Zeigt die IPv4-Routing-Tabelle des Kernels.	Zeilen mit `proto kernel scope link` markieren deine lokalen Subnetze. Alles, was hier matcht, schickt der Rechner direkt per ARP ins Segment – alles andere geht an die Zeile mit `default via`.
`ip -6 route show`	Zeigt die IPv6-Routing-Tabelle.	Suche nach deinem zugewiesenen `/64`-Präfix mit `proto kernel metric 256`. Das ist deine lokale Kante. Die Standard-Route ins Internet steht hinter `default via fe80::...`.

Der schnelle Blick: Was sieht der Kernel?

□□ Schnittstellen-Details prüfen

Wenn du wissen willst, ob die Netzmaske auf dem Interface `eth0` stimmt, fragst du gezielt nur dieses Device ab:

```
ip address show dev eth0
```

Eine typische, saubere Ausgabe sieht so aus:

```
eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
qlen 1000
    inet 198.51.100.130/27 brd 198.51.100.159 scope global dynamic eth0
       valid_lft 86340sec preferred_lft 86340sec
    inet6 2001:db8:beef:a1::10/64 scope global dynamic mngtmpaddr
       valid_lft 86340sec preferred_lft 14340sec
    inet6 fe80::5054:ff:fe12:3456/64 scope link
       valid_lft forever preferred_lft forever
```

Hier siehst du schwarz auf weiß, wie die Puzzleteile ineinandergreifen:

- Bei IPv4 hat der Kernel aus der `/27` völlig eigenständig die passende Broadcast-Adresse `198.51.100.159` abgeleitet (`brd`).
- Bei IPv6 siehst du deine globale Adresse mit `/64` – ohne Broadcast-Eintrag, weil es den bei IPv6 nicht gibt.

□□ Routen verifizieren: Geht das Paket lokal oder zum Gateway?

Wenn du dir unsicher bist, welchen Weg der Kernel für eine bestimmte IP wählt, musst du nicht raten oder die Routing-Tabelle manuell zerlegen. Frag den Stack direkt:

```
ip route get 198.51.100.140
```

Liegt die Ziel-IP im selben Subnetz, antwortet Linux mit einer direkten Verbindung über das Interface:

```
198.51.100.140 dev eth0 src 198.51.100.130 uid 1000
    cache
```

Fragst du dagegen eine Adresse ab, die außerhalb deiner `/27` liegt (zum Beispiel die `198.51.100.170` aus dem nächsten Subnetz), schwenkt die Ausgabe sofort um:

```
ip route get 198.51.100.170
```

```
198.51.100.170 via 198.51.100.129 dev eth0 src 198.51.100.130 uid 1000
    cache
```

Der Kernel meldet dir unmissverständlich: *„Kenne ich nicht lokal, geht über `via 198.51.100.129` an das Gateway.“* Schneller und zuverlässiger kannst du deine Netzgrenzen in der Praxis nicht kontrollieren. Wenn Pakete trotz korrekter Maske verworfen werden oder Routen klemmen, hilft dir unser Leitfaden, wie du Netzwerkprobleme unter Linux systematisch analysieren kannst.

Weiterführende Ressourcen

Wenn du bei Netzwerkgrenzen, Adressstandards oder Protokollfeinheiten sichergehen willst, verlasse dich nicht auf Forenhalbwissen oder alte Blogbeiträge. Im Netzwerkbetrieb zählen am Ende die Standards der IETF – die Requests for Comments (RFCs).

Quelle: https://admindocs.de/de/netzwerk/ipv4-und-ipv6-subnetting-verstehen-praefixe-und-netzgrenzen-einfach-erklaert

Hier sind die zentralen Primärquellen, die genau festlegen, wie Subnetting, Dokumentationsadressen und die Präfix-Logik unter der Haube definiert sind:

RFC 4632: Classless Inter-domain Routing (CIDR) Die technische Grundlage für Subnetting ohne starre Netzklassen. Definiert die Schrägstrich-Schreibweise und wie Router Adresspräfixe aggregieren.

RFC 4291: IPv6 Addressing Architecture Beschreibt die 128-Bit-Struktur von IPv6, die Aufteilung in Präfix und Interface ID sowie Multicast- und Anycast-Adressen.

RFC 4862: IPv6 Stateless Address Autoconfiguration (SLAAC) Erklärt das Zusammenspiel von Router Advertisements und Client-Adressen – und warum /64 die feste Grenze für Endgeräte-Subnetze ist.

RFC 5737: IPv4 Address Blocks Reserved for Documentation Definiert die offiziellen IPv4-Testbereiche 192.0.2.0/24, 198.51.100.0/24 und 203.0.113.0/24, die in keiner echten Routing-Tabelle auftauchen dürfen.

RFC 3849: IPv6 Address Prefix Reserved for Documentation Reserviert das Präfix 2001:db8::/32 verbindlich für Beispiele, Konfigurationsanleitungen und Schulungsmaterial.

RFC 9637: Clarifications on IPv6 Subnet Sizing Moderne Klarstellung der IETF zur Zuweisung von IPv6-Präfixen an Endkunden und zum praktischen Umgang mit /64-Grenzen.

RFC 919: Broadcasting Internet Datagrams Der historische, aber bis heute gültige Ursprung der Broadcast-Adressierung in IPv4-Subnetzen.