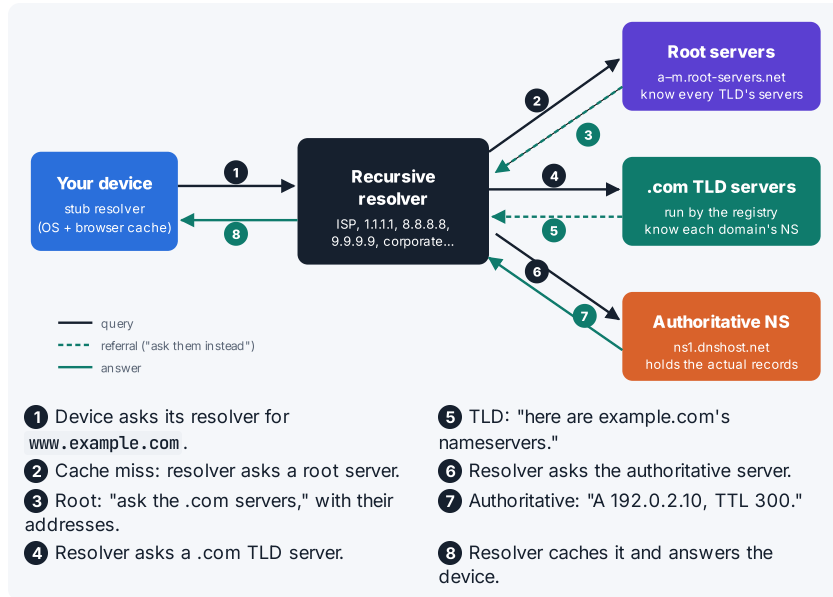


How DNS Works

\$ dig www.example.com A

How a name becomes an IP address, who answers along the way, and what can go wrong. ;; ANSWER: www.example.com. 300 IN A 192.0.2.10

1 RESOLVING A NAME, STEP BY STEP



2 WHO'S WHO

- Stub resolver**
on your device
- Asks one question and expects a final answer. Checks the browser and OS caches first.
- Recursive resolver**
does the legwork
- Chases referrals from the root down, validates DNSSEC, and caches answers for everyone who uses it. Most lookups never leave its cache.
- Root servers**
13 names, 1,900+ sites
- Serve the root zone, which lists the nameservers for every TLD. Anycast puts copies near nearly everyone.
- TLD servers**
registry-operated
- Hold the delegation (NS + DS records) for every name in the TLD. That's what your registrar edits when you change nameservers.
- Authoritative NS**
your DNS host
- Hold the zone file with your actual records. Can be your registrar, a CDN, a cloud provider, or your own servers.

Delegation & glue

Each zone delegates to the next with **NS records** held by the parent. If a domain's nameservers live inside the domain itself (`ns1.example.com`), the parent also stores **glue** records with their IPs to break the chicken-and-egg loop.

3 COMMON RECORD TYPES

Type	What it does	Example
A / AAAA	Maps a name to an IPv4 / IPv6 address	<code>www A 192.0.2.10 / AAAA 2001:db8::10</code>
CNAME	Alias to another name; can't coexist with other records, so not at the apex	<code>shop CNAME shops.myplatform.com.</code>
MX	Mail servers for the domain, with priority (lower wins)	<code>@ MX 10 mail.example.com.</code>
TXT	Free text: SPF, DKIM, DMARC, ownership verification	<code>@ TXT "v=spf1 include:_spf.host.com -all"</code>
NS / SOA	Authoritative servers / zone metadata (serial, timers, negative TTL)	<code>@ NS ns1.dnshost.net.</code>
CAA	Which certificate authorities may issue TLS certificates	<code>@ CAA 0 issue "letsencrypt.org"</code>
SRV / HTTPS	Service host and port / browser connection hints (HTTP/3, ECH)	<code>_sip._tcp SRV 10 5 5060 sip.example.com.</code>
PTR	Reverse DNS (IP to name), set by whoever controls the IP block	<code>10.2.0.192.in-addr.arpa. PTR www.example.com.</code>
DS / DNSKEY	DNSSEC: zone's public keys / key hash held at the parent	<code>example.com. DS 2371 13 2 1F98... (at .com)</code>

4 CACHING & TTL why changes aren't instant

Browser	OS	Resolver	Authoritative
seconds-minutes	per TTL	per TTL, shared	source of truth

- Every record has a **TTL** (seconds) that caches honor until it expires.
- "Propagation" is really **caches expiring**. Lower TTLs a day before a planned change.
- Nameserver changes are slower: TLD NS records often carry 1-2 day TTLs.
- NXDOMAIN ("doesn't exist") is cached too, per the SOA.

5 DNSSEC CHAIN OF TRUST

- Root zone:** signed with the root key, which resolvers trust out of the box
↓ root holds a DS record for .com
- .com zone:** signed by Verisign's keys
↓ .com holds a DS for example.com (submitted via your registrar)
- example.com:** records signed with your DNSKEY (RRSIG records)

Proves answers are authentic; does **not** encrypt them. A stale DS after switching DNS hosts breaks resolution on validating resolvers.

On the wire

- Classic DNS: **UDP/TCP port 53**, plaintext.
- **DoT** (853), **DoH** (443), and **DoQ** encrypt device-to-resolver traffic.
- **QNAME minimization** limits what resolvers reveal upstream.

What goes wrong

- **Hijacking** via stolen registrar or DNS-host logins. Use MFA and registry lock.
- **Dangling CNAMEs** to deleted cloud resources others can claim.
- **Cache poisoning** with forged answers; DNSSEC blocks it for signed zones.

Try it yourself

```
dig example.com A -basic lookup
dig +short example.com MX -just the answer
dig @1.1.1.1 example.com -ask a specific resolver
dig +trace example.com -walk root → TLD → auth
dig NS example.com @a.gtld-servers.net -see the delegation
```