Engineer

DNS

the directory turning a name into an address

What it is

DNS, the Domain Name System, is the distributed directory the internet consults before almost anything else happens. A program holding a name such as `example.com` asks it for the address to connect to, and the same system answers a handful of other questions about that name: which servers accept its mail, which text its owner has published, which servers are in charge of it.

The directory is a tree read from right to left. At the top sits the root, below it top-level domains such as `.com` or `.dk`, below those the domains people register, and below those any subdomains their owners create. No single server holds the tree. Each level hands responsibility for the next one down to named servers, and the part of the tree that one set of servers answers for is called a zone. Whoever controls a zone decides what its names resolve to.

Two kinds of server do the work. Authoritative servers hold a zone and answer for it with the final word. Recursive resolvers, run by internet providers, companies or public services, do the walking on a client's behalf: they ask a root server where `.com` lives, ask `.com` where `example.com` lives, ask that domain's servers for the record, and hand the answer back. Your operating system runs only a small stub resolver that passes the question to a recursive one and waits.

The answers are typed records, and a zone file is the classic way to write them down:

```
$ORIGIN example.com.
$TTL 3600
@     IN  SOA    ns1.example.net. hostmaster.example.com. (
                 2026101701 7200 900 1209600 300 )
@     IN  NS     ns1.example.net.
@     IN  A      192.0.2.10
```

```
@     IN  AAAA   2001:db8::10
www   IN  CNAME  example.com.
@     IN  MX     10 mail.example.net.
@     IN  TXT    "v=spf1 include:_spf.example.net -all"
```

`A` and `AAAA` give IPv4 and IPv6 addresses. `CNAME` makes one name an alias for another. `MX` names the server that receives mail, and `TXT` carries free text, which in practice means mail-authentication policies and proofs that someone controls the domain. `NS` and `SOA` describe the zone itself.

Every record carries a time to live, here 3,600 seconds, and that number is the key to how DNS behaves. A resolver that has fetched an answer may keep reusing it until its TTL runs out, without asking again. The result is a system that answers most lookups from a cache close to the client and keeps answering when an authoritative server is briefly unreachable, at the price that nobody can force the world to forget an old answer early.

Use the concept in a decision

Put “What is DNS?” to work in a real decision. We can help relate the concept to the system you are building or buying. Wi-Fi, firewalls and remote access set up so staff can work from anywhere and only the right people get in.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Managed 30 Ubuntu Linux VPS instances — 30 Linux servers managed, with recovery planned for

Managed 30 Ubuntu Linux VPS instances Get in touch →

Why it matters

Every connection to a named service starts with a lookup. When DNS is wrong, the servers can be perfectly healthy and the site, the API and the mail still look down, because nothing reaches them. When DNS is right, nobody notices it, which is why it tends to be the least documented part of a system.

DNS is also where control over a domain is exercised and proved. Mail providers read SPF, DKIM and DMARC policies from `TXT` records to decide whether a message claiming to come from your domain is genuine, so a wrong or missing record sends real mail to spam. Certificate authorities and hosted services ask for a record with a specific value as proof that you own the name. Anyone who can

edit the zone can redirect the site, intercept the mail and obtain certificates for the domain, which makes access to the DNS account as sensitive as access to the servers.

Caching shapes every change. What people call propagation is really thousands of resolvers each waiting for their own copy to expire. The way to move a site with little disruption is to lower the TTL a day or so ahead, wait for the old TTL to pass, make the change, and raise the TTL again once it has settled.

Classic DNS travels in plain text, and neither the answers nor the questions are private. DNSSEC lets a resolver check that an answer really came from the zone's owner and was not altered, and by its own specification it adds no confidentiality. Encrypted transports such as DNS over HTTPS hide the questions from the network path, while the resolver at the other end still sees every name asked for. Choosing a resolver is choosing who gets that list.

What it looks like in practice

The company's own zone shows the most common starting point and a way out of it. It held about twenty records, kept in a provider's web console, covering the web presence, a git subdomain, two personal redirects, mail routed through a hosted mailbox provider, a transactional sending domain and the authentication records for both. In a console, the current state is whatever the last person to click left behind. The zone is now data in a repository: live records, application records, transport-policy records and a separate list of records due for removal. One play reconciles that declaration against the provider's API. Two more entry points sit behind explicit opt-in tags, an audit that reports the difference without changing anything and an export that writes the zone in the standard zone-file format, so that something other than the repository can read it. The audit is the part used most, because knowing the difference is more often what is wanted than closing it.

The same provider has a deliberately narrow job. The site's credits page explains that Cloudflare answers DNS for the site and nothing else, with no proxy in front, and it prints the server's address outright, since DNS already hands that address to anyone who asks.

Reading DNS also belongs to the routine description of the estate. Among seven read-only reporting roles that render the live host into Markdown is a provider inventory that reads DNS alongside two cloud APIs and a dedicated-server API,

under the rule that no report prints a number its run did not measure.

Where it goes wrong

DNS failures are often delayed, cached or split, so the same name works for one person and fails for another. These are the ones that recur.

A change refuses to land. Someone edits a record with a TTL of a day and expects the new address to take effect at once, but resolvers that cached the old answer keep serving it until their copy expires. A quieter version catches new names: if a name was looked up before its record existed, the negative answer is cached too, for as long as the zone's `SOA` allows (RFC 2308). Lower TTLs before planned changes, and create records before anything asks for them.

An alias sits where it cannot. RFC 1034 says that a name holding a `CNAME` may hold no other data. The bare domain, `example.com` itself, always holds `SOA` and `NS` records, so a standards-following zone cannot make the apex an alias, and a `CNAME` on a subdomain that also needs `MX` or `TXT` records breaks the same rule. Providers offer flattened or `ALIAS` records as their own workaround; each behaves a little differently, so read what yours actually serves.

Mail authentication tips over. Every `include` in an SPF record causes DNS lookups, and RFC 7208 caps the total at ten. Adding one more newsletter or ticketing service can push the record past the limit, and receivers then treat the policy as a permanent error, so the mail it was meant to authenticate is judged without it. Count the lookups whenever a sender is added.

A record outlives its target. A subdomain points by `CNAME` at a cloud storage bucket or a hosted app that has since been deleted. If the provider lets anyone claim that resource name, a stranger can claim it and serve their own content under your subdomain, with your domain's reputation attached. Remove the record in the same change that removes what it points to.

The zone lives only in a console. There is no history, no review and no diff, so an accidental deletion of the `MX` record is discovered when mail stops arriving, and the previous value has to be reconstructed from memory. Exporting the zone regularly, or better, declaring it as code, gives every change a record and a way back.

Further reading

- RFC 1035: Domain Names, Implementation and Specification — record formats, the zone-file syntax and the wire protocol.
- RFC 1034: Domain Names, Concepts and Facilities — the tree, zones, delegation and how resolvers walk them.
- RFC 9499: DNS Terminology — careful definitions of the words this page uses.
- RFC 2308: Negative Caching of DNS Queries — how long a resolver may remember that a name does not exist.
- RFC 4033: DNS Security Introduction and Requirements — what DNSSEC protects, and what it leaves alone.
- RFC 8484: DNS Queries over HTTPS — carrying lookups inside ordinary encrypted web traffic.

Glossary

Choose for your actual constraints

Bring the deployment path, the operational risks and the people on call. We can scope a change with checks, a recovery route and clear ownership.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is DNS?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/dns/