Engineer

ACME

the protocol that issues a certificate without a human

What it is

ACME, the Automatic Certificate Management Environment, is the protocol a program uses to ask a certificate authority for a certificate, prove it controls the names it asked for, and come back for a fresh one before the old one runs out. Nobody fills in a form or answers a confirmation email along the way. RFC 8555 standardised it in March 2019, and Let's Encrypt is the authority most people meet it through.

The client starts with an account key of its own, kept apart from the key in any certificate, and signs every request with it. Getting a certificate then runs through five resources on the server:

```
newAccount   register the account key once
newOrder     list the names the certificate should cover
authz        one authorization per name, each offering challenges
finalize     send a certificate signing request once all are valid
certificate  download the issued chain, then install it
```

A challenge is how the client shows it controls a name. The server hands over a token; the client joins it to a fingerprint of the account key, which ties the proof to that one account, and publishes the result where only someone in control of the name could put it. Three challenge types matter:

- **http-01** serves the value as a file at `/.well-known/acme-challenge/<token>` on the name itself. The authority's first request for it always goes over plain HTTP to port 80.
- **dns-01** puts a hash of the value in a TXT record at `_acme-challenge` under the name. It is the only one of the three that can prove control for a wildcard such as `*.example.com` .

- **tls-alpn-01**, added by RFC 8737 in February 2020, answers on port 443 during a TLS handshake that uses a dedicated ALPN protocol. It suits a reverse proxy that owns port 443 and cannot use port 80.

RFC 9773, published in June 2025, added ACME Renewal Information, or ARI. The authority publishes a suggested window for renewing each certificate it issued, and the client asks for that window instead of guessing. That lets the authority move renewals earlier, ahead of a mass revocation for instance, without anyone editing a configuration file.

Use the concept in a decision

Put “What is ACME?” to work in a real decision. We can help relate the concept to the system you are building or buying. Accounts, passwords and permissions arranged so people hold exactly the access their job needs, and leavers lose it that day.

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. Wrote 1,000 RBAC rules to secure Linux servers — 1,000 access rules written to close a security gap

Wrote 1,000 RBAC rules to secure Linux servers Get in touch →

Why it matters

Whether a person can still renew certificates by hand has stopped being a matter of taste. It now has dates attached.

The CA/Browser Forum sets the rules every publicly trusted authority follows. Its ballot SC-081v3, adopted in April 2025, caps a certificate issued since 15 March 2026 at 200 days. The cap falls to 100 days from 15 March 2027 and to 47 days from 15 March 2029, and the time a proof of domain control may be reused falls in step, from 200 days now to 100 and then to just 10. Under the final rules, a name renewed at two thirds of its lifetime needs a new certificate roughly every month, and nearly every one of those renewals needs a fresh challenge.

Let’s Encrypt is ahead of that schedule. Its default certificates last 90 days today. From 10 February 2027 they last 64 days, with authorizations reusable for 10 days, and from 16 February 2028 they last 45 days, with reuse cut to 7 hours. Its own announcement says manual renewal is not recommended. It also stopped sending expiry reminder emails on 4 June 2025, so even the safety net that caught a forgotten renewal is gone.

Hand renewal is off the table, so ACME is the default. What is left to decide is where its moving parts live:

- **The challenge.** http-01 is the easy start for a single server that answers on port 80. dns-01 earns its extra set-up for wildcards, for several servers behind one name, and for hosts the internet cannot reach. tls-alpn-01 belongs to proxies that hold port 443 and lack port 80.
- **The client.** It can live inside the web server, run beside it with a hook that installs each new certificate, or run on a separate machine that validates centrally and copies certificates out.
- **The watcher.** Something outside the client has to notice when a renewal has not happened, because the client is the part that has just failed.

What it looks like in practice

As DevOps Engineer, the company took certificate work out of human hands for a hundred Docker applications on Ubuntu Linux hosts, where Apache and Nginx both served traffic. ACME ran every step: issuing each certificate, renewing it before expiry, deploying it. No step in the cycle waited on someone remembering a date. All hundred applications stayed on valid certificates, and the hand work went away.

Deployment is the step that is easy to underrate. Apache and Nginx read a certificate when they start or reload and keep serving that copy, so a client that only writes a new file has renewed nothing a visitor can see. Where Apache and Nginx share a fleet, each needs its own reload once a new certificate lands.

The company's own site shows the other arrangement, where the client is the web server. There, Caddy renews the site's certificate from Let's Encrypt itself. Caddy speaks ACME natively and needs no separate tooling, so there is no deploy step to forget.

Where it goes wrong

The challenge times out or returns 404, though the site loads fine in a browser. The http-01 request arrives on port 80 and something in its path is wrong: a firewall or cloud security group that opens only 443, a redirect that leads to a port other than 80 or 443, or a load balancer that sends the request to a server that never received the token. Let's Encrypt follows up to ten redirects,

and only to those two ports. Keep port 80 open, serve the challenge path from every backend or from shared storage, and switch to dns-01 where port 80 cannot be had.

The DNS API key on the web server can rewrite the whole zone. A wildcard needs dns-01, so the client needs a way to create TXT records, and the easiest credential to get is often one that can change every record the account owns. If the web server is broken into, the attacker can redirect mail and every other name as well. Use a credential limited to the `_acme-challenge` records where the provider offers one, or point `_acme-challenge` by CNAME at a zone set aside for challenges, or validate from a separate machine and copy the certificates over. DNS changes take time to spread, so the client may have to wait as long as an hour before telling the authority the challenge is ready.

After a redeploy, new containers come up without certificates. Each container asks for its own certificate when it starts and keeps it only in its own filesystem, so every restart asks again. Redeploy often enough, or start many at once, and the authority's rate limits refuse the extra orders; RFC 8555 gives the error a name of its own, `rateLimited`. Let's Encrypt warns that in the worst case none of the instances gets a certificate. Keep certificates and keys in durable storage, and have a new instance use a valid certificate from there before it orders one.

A fixed renewal timer lets certificates expire once lifetimes shrink. The job renews every 60 days, say, a pace tuned to 90-day certificates. A 64-day certificate leaves that timer four days of margin, and a 45-day one expires about two weeks before the timer fires. RFC 9773 names fixed intervals as one of the habits that keep authorities from changing lifetimes. Turn on ARI where the client supports it. Let's Encrypt's advice is to check it at least twice a day and, as a backstop, to renew when a third of the lifetime is left.

The client reports success and visitors still see the old certificate. The new file is on disk, and nothing told the web server to reload, or the hook reloaded one server out of several. The old certificate stays in service until it expires. Reload the server from the client's deploy hook, and watch the certificate that is actually served, checked from outside, rather than the file:

```
openssl s_client -connect example.com:443 -servername example.com \
    </dev/null 2>/dev/null | openssl x509 -noout -enddate
```

Further reading

- RFC 8555: Automatic Certificate Management Environment (ACME) — accounts, orders, authorizations and the http-01 and dns-01 challenges.
- RFC 9773: ACME Renewal Information (ARI) Extension — the renewal window an authority suggests for each certificate.
- RFC 8737: ACME TLS Application-Layer Protocol Negotiation (ALPN) Challenge Extension — the tls-alpn-01 challenge, answered entirely on port 443.
- Challenge Types — Let's Encrypt's own comparison of the three challenges, with the limits of each.
- Decreasing Certificate Lifetimes to 45 Days — Let's Encrypt's dates for shorter certificates, and its renewal advice.
- Ballot SC-081v3 — the CA/Browser Forum vote that sets the 200, 100 and 47-day limits.

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 ACME?" 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/acme/