Engineer Platform

Portfolio

Engineering services, with examples of delivered work.

Engineer ApS was registered in 2025. Most of the work shown here was delivered before that by our founder inside client teams; some has been delivered by the company itself, and some since in employment. The same hands and the same standards — we say which is which rather than blurring it.

Find experience worth applying

Use this record to find experience that is relevant to your project.

Explore our services Get in touch →

Read the decisions behind the record

The entries describe delivered work across services. Read a review to see the situation and decisions behind it.

Cases Get in touch →

Browse by category

SOFTWARE ENGINEERING

Backend Engineering · Frontend Engineering · Full-Stack Development · APIs & Integration · Web Development · Testing & QA · Performance Tuning · Python · Internationalization

DATA & DATABASES

Databases · PostgreSQL · SQL · Data Engineering · Data Pipelines (ETL/ELT) · Data Analytics · Data Governance · GIS / Geospatial

CLOUD, INFRASTRUCTURE & DEVOPS

Cloud · Containers (Docker/Kubernetes) · DevOps · Automation & CI/CD · Infrastructure

Linux & Servers · System Administration · Networking & VPN · Monitoring & Observability · Reliability & Backups · Security · Platform Architecture · Migrations & Modernization

CONSULTING & LEADERSHIP

Technical Leadership · Team Leadership · Agile & Scrum · Project Management · Product & Requirements · Stakeholder & Reporting · Documentation · Solution Architecture · Brand & Marketing

Everything below **was delivered** — not proposed, not planned.

Each entry names what the work changed and links the services it demonstrates.

What we offer What it costs →

{ Cloud, Infrastructure & DevOps }

Cloud Infrastructure & Migration

Where the software runs

Servers sized for what you actually use, moved into the cloud or back out of it, with the monthly bill worked out first.

Provisioned a second server from code

Automated the provisioning of a second server on a second cloud provider, creating the firewall before the machine so it is born behind one, with both providers' firewalls written directly against their REST APIs to avoid a third-party collection.

Ran containers on Podman and Quadlet

Deployed a container plane on Podman and Quadlet under systemd rather than Docker, because Docker's default bridge-port publishing can bypass UFW rules — and gave deploys an unprivileged user with one fixed command instead of root.

Public services on a small server

Hosted the company's public services on a single-core 512 MB host — a git forge, a web server for seven domains, Tor, two alternative-protocol servers and intrusion banning — treating 464 MB of usable memory as the design constraint; backup automation is a separate implementation.

Infrastructure defined in a repository

Built the company's own infrastructure as 19 Ansible playbooks and 34 roles across 12,065 lines of YAML, converging a live host to a declared state with every play idempotent.

Built the entire core infrastructure from scratch

As one of the first hires, designed and built the entire core infrastructure and supporting processes from scratch for a green-energy SaaS startup, laying the foundation for rapid growth.

Owned end-to-end platform deployments to Azure

Owned end-to-end deployments of the platform to Azure, managing releases across development, staging and production environments.

Provisioned Azure infrastructure as code

Provisioned Azure infrastructure as code with Bicep — Container Apps, PostgreSQL Flexible Server, Front Door/WAF and networking — across the development, staging and production environments.

Infrastructure simplified beyond AWS

Reworked an international client's infrastructure by removing unused services, adjusting capacity and moving workloads away from AWS. The founder reports a 10x reduction in cloud costs with no loss of productivity.

Ran 20 Docker apps on Google Cloud and AWS

Deployed and maintained 20 Docker containerized applications, troubleshooting with Podman and Kubernetes, and managing R-based apps on Google Cloud and AWS.

{ Cloud, Infrastructure & DevOps }

Infrastructure as Code

Servers you can rebuild from versioned configuration

The whole setup written down as files, so it can be recreated exactly and never lives only in one person's head.

Published a readable Tor onion mirror

Published the site as a Tor onion mirror on a readable address beginning engineer, with the key generated off the host so it never reached this repository or a laptop, and left it the one front door of four that is never counted.

Rehearsed the forge CI hook, found 2 defects

Rehearsed the forge-side CI hook and found two defects unreachable by reading the file: a fallback that put an unresolvable argument on the hook's input, and git's own environment variable following the gate into the checkout and turning 19 tests red.

Split every operational secret per host

Split the four per-host operational secrets after establishing that two hosts sharing one dead-man's switch alert less than two switches, not more, and that a shared backup passphrase can expose both hosts' backups, even in separate repositories, if someone obtains it and can read both repositories' encrypted data and key files.

Your code on your own server

Deployed the company's own git forge on Soft Serve, private by default with no web panel and its SSH port bound to loopback behind a jump host, and made the landing page in front of it a build artefact of the main site rather than a hand-kept copy.

Cut systemd sandbox exposure on every unit

Cut systemd sandbox exposure across every unit this platform installs — a dead-man's-switch service from 9.6 UNSAFE to 1.5, an internet-facing git forge from 8.3 EXPOSED to 1.5 — and added a converge-time parser check after finding a misspelled directive silently ignored in three unit templates.

Reconciled a 20-record DNS zone as code

Reconciled a 20-record DNS zone declaratively against the Cloudflare API with separate audit and BIND-export entry points, and turned the CDN proxy back off on privacy grounds after building it.

Added a one-second preflight for secrets

Added a preflight play that runs the same code as the converge against operator-local secrets in about a second, after a half-applied production run died on its ninth task with the swap settings already written to the live host.

Made Ansible check mode tell the truth

Made check mode tell the truth across the whole platform after finding six probes deciding on a value the host never gave, because Ansible's command module reports success under --check while skipping the command entirely.

Built dead-man's-switch host monitoring

Built dead-man's-switch monitoring that pings only while memory and disk are healthy, so a degraded host raises an alert by going silent — and caught six variable names saying "free" where the check correctly measured "available", an order of magnitude apart on a 464 MB box.

Backup automation with an explicit restore method

Implemented restic backup automation with retention, integrity checks and a monthly restore-test workflow; documented storage activation and live-restore verification as separate operational steps.

Hardened SSH to 24 validated directives

Hardened SSH to 24 asserted directives with three-stage validation — the candidate file, the assembled config, then the daemon's own read-back — after the read-back caught the running server silently overriding two of the twenty-four.

Implemented backups and disaster recovery as code

Configured database backup retention as infrastructure-as-code, then audited the recovery position and documented the restore procedure — naming the remaining gaps rather than leaving them to be found during an incident.

Wrote 1,000 RBAC rules to secure Linux servers

Enhanced data security by implementing 1,000 RBAC rules for developers, application instances, PostgreSQL, MS SQL, and other Linux servers, preventing unauthorized access; documented with Ansible automation.

This service is part of work listed elsewhere on this page:

Provisioned a second server from code · Public services on a small server · Infrastructure defined in a repository · Provisioned Azure infrastructure as code

{ Cloud, Infrastructure & DevOps }

DevOps & CI/CD Automation

Shipping changes without drama

Tested, released and reversible, so changes reach customers on a Tuesday afternoon rather than late on a Friday night.

Held 4 repositories to one history standard

Held four repositories to one history standard — conventional, emoji-free, no attribution trailers, enforced by a commit-message hook — alongside 48 instruction documents that govern how the work is done.

Know the licence of everything you ship

Audited 644 Rust crates for licence compatibility on every build, and proved the check fires by rewriting one crate's licence and by moving the pinned core revision without regenerating.

Reached the unused half of the core

Reached the half of the messaging core the application had never used — backup transfer, disappearing messages, message editing and resending, verified invitations, proxies and encryption policy — driving every test against the real library with no mocks.

Took the test suite to 8.9 seconds

Took the test suite from six tests over a sixty-second limit to 135 passing in 8.9 seconds by profiling the main thread and removing the two calls it sat inside for 3,989 samples out of 4,017.

Wrote a parser verifying every FFI call

Wrote a parser that reads the real 7,308-line C header and verifies every call site, every enum constant and that every pointer-owning class is final, after a hand-written placeholder header let calls to three removed functions compile, link and crash.

Found the hooks and the gate disagreeing

Found the commit hooks and the quality gate running different checks while a document promised they were the same, by comparing the two lists in a test — the five that only ever ran by hand were the ones reading the CV prose.

Wrote tests for the checkers themselves

Wrote tests for the checkers themselves after establishing that a checker fed only clean input will one day report clean because it read nothing — planting a misspelling to confirm the spell-check finds it, and taking an id range from the database rather than from a number in the test.

Enabled every linter rule as an error

Selected every rule the Python linter has as an error, working through 1,815 findings to reach zero, with each of the few exemptions carrying a written reason and two of them backed by a checker instead of a comment.

Held the generator to 981 tests and 92% coverage

Held the generator to 981 test cases at a 92% branch-coverage floor with warnings treated as failures, and asserted idempotence by running the whole build pipeline twice from an empty file and requiring the second pass to change nothing.

Linted the 10,242-line quality gate itself

Brought 10,242 lines of quality-gate JavaScript under a formatter and a linter after establishing it was the largest body of code in the repository and the only one nothing read, fixing 13 findings and suppressing none.

Fixed a sitemap with one shared timestamp

Fixed a sitemap where 172 of 176 URLs shared one modification timestamp, by taking the date from git history after establishing that the export rewrites every file on every run.

Mirrored the site to Gemini and Gopher

Built Gemini and Gopher outputs from the same website source; the achievement records 777 documents in each format at that stage, with zero HTML bytes changed by adding the formats.

A safety net that no longer costs half an hour

Cut the site's browser-driven quality gate from 1,636 seconds to 615 by scheduling its checks longest-first through a worker pool bounded to four lanes, after measuring that alphabetical order cost 320 seconds against 224.

Built a 22-linter commit gate

Built a commit gate of 22 one-line linters plus five that earn a paragraph, with no warning tier and no inline suppressions permitted, covering HTML, CSS, JavaScript, Python, YAML, Markdown, shell, links, spelling, secrets and typography.

Ran a continuous security scanning programme

Established a continuous security programme — code scanning, DAST, dependency and vulnerability checks, SBOM generation, secret scanning and SHA-pinned actions — alongside 21 written security audits.

1,022 migrations behind a schema drift gate

Kept the schema honest across 1,022 migrations with a CI gate that builds the database both ways — a fresh install, and an install plus every migration — and fails when the two disagree.

Built first-party error monitoring and tracing

Built first-party error monitoring and OpenTelemetry tracing rather than buying them — payload sanitising, spike and regression detection, symbolication and a synthetic heartbeat — behind 11 operator views.

Built a 268-check guard engine for the repo

Built the repository's guard engine — 268 registered commit checks, 277 lint rules and 15 custom ESLint rules — plus 146 tests of the guards themselves, so the build holds the standard, not review.

Built a test suite across Go, SQL and browser

Built a layered automated test suite — 981 Go tests, 543 frontend and browser specs, 494 SQL behavioural tests — with mutation testing, property-based tests and an accessibility gate.

Automated team collaboration

Automated team collaboration, password management, task and time management, and built a semi-automatic project-showcase system, raising team productivity.

Set a zero-warnings bar across six languages

Set a zero-warnings quality bar across six languages — Go, TypeScript, SQL, Python, Shell and Markdown — enforced by pre-commit hooks.

Authored 578 go-task automation targets

Authored 578 go-task automation targets spanning native, Docker and HTTPS dev modes, linting, testing, database and deployment.

Built GitHub Actions CI/CD pipelines

Built GitHub Actions CI/CD pipelines with a distroless production frontend image and multi-environment promotion.

Generated the API contract from the database out

Generated the API contract outward from the database — OpenAPI, a 44,076-line typed TypeScript client, 61 mock handlers and the limits the UI enforces — with a guard at every hop that fails on drift.

Automated data processing in Python and PL/pgSQL

Automated data processing tasks using Shell scripting, PL/pgSQL, Python, and Transact-SQL, increasing productivity and efficiency.

Saved 4,000 hours with GitHub and GitLab CI/CD

Streamlined data analysis and software development processes, saving 4,000 hours by introducing GitHub, GitLab, Bash, and Python CI/CD practices.

Streamlined CI/CD processes, saving 4,000 hours

Streamlined CI/CD processes, saving 4,000 hours by introducing automation in software development pipelines.

Automated SSL/TLS certs for 100 Docker apps

Automated SSL/TLS certificate creation for 100 Docker applications, ensuring secure connections across Ubuntu Linux hosts.

Overhauled internal processes, saving 8,000 hours

Overhauled internal processes, saving 8,000 hours by improving software architecture, systems, and scheduling efficiency.

Cut operational risk with Grafana and Prometheus

Mitigated operational risks by implementing a monitoring dashboard using Grafana and Prometheus, improving system reliability.

Automated 100 critical data backups using Barman

Automated 100 critical data backups using Barman, Google Cloud, Bash, and Python, ensuring data integrity across databases.

Automated delivery of 20 GIS data pipelines

Automated delivery of 20 GIS data pipelines and app data ETL processes, streamlining infrastructure automation and reporting.

Automated GIS SaaS deployment with GitHub Actions

Automated GIS SaaS application deployment, data processing, and reporting system using GitHub Actions CI/CD, Python, Bash, and SQL.

This service is part of work listed elsewhere on this page:

Rehearsed the forge CI hook, found 2 defects · Ran containers on Podman and Quadlet · Your code on your own server · Added a one-second preflight for secrets · Made Ansible check mode tell the truth · Infrastructure defined in a repository · Built the entire core infrastructure from scratch · Owned end-to-end platform deployments to Azure · Infrastructure simplified beyond AWS · Ran 20 Docker apps on Google Cloud and AWS

{ Cloud, Infrastructure & DevOps }

Containerization & Orchestration

Software that runs the same everywhere

Packaged to behave identically on a laptop, on a test server and in production, and to scale up when a busy week arrives.

This service is part of work listed elsewhere on this page:

Ran containers on Podman and Quadlet · Cut systemd sandbox exposure on every unit · Built GitHub Actions CI/CD pipelines · Automated SSL/TLS certs for 100 Docker apps · Ran 20 Docker apps on Google Cloud and AWS

{ Cloud, Infrastructure & DevOps }

Site Reliability & Monitoring

Knowing before your customers do

Monitoring and alerts that speak up while a fault is still small, and the figures that show how often it stays that way.

Made six collectors state what they skipped

Made six reporting collectors state the part of their input they never read — files that stayed shut, probes not installed, journal lines unrecognised — and fitted thresholds to 116 measured samples, after establishing that a blind collection and a healthy host render the same page.

Measured 13 addresses with no tracking

Measured thirteen addresses from the web server's own access log alone — no cookie, no tracker and no third-party service — masking the client network as each line is written, and folding the result into a permanent archive of 21 daily counts and 20 monthly facets.

A bounded event pipeline for a macOS app

Bounded event buffers and logging after a recorded macOS incident with 41 MB per second growth and 111 GB of compressed pages on a 36 GB machine; a recorded full test run reduced core log lines from 610,996 to 1,411.

Built 7 read-only host reporting roles

Built seven read-only reporting roles that render a live host to Markdown — facts, access, git, metrics, traffic, security and provider inventory — under a rule that no number is printed the run did not measure.

Proved the intrusion ban works every run

Proved the intrusion-banning path end to end on every hardening run by banning a reserved test address, reading the resulting kernel rule and unbanning in a guaranteed cleanup block, so a jail that stops working fails a run instead of reporting healthy.

Three locks on the door, none of them locked

Found and closed three SSH brute-force protections that had never worked: a ban jail watching port 22 while the daemon listened on 1986, a rate limit shadowed by a broader rule above it, and a ban action whose binary never resolved, so no ban had ever applied.

Built fail-closed rate limiting and anti-abuse

Built fail-closed abuse controls — 22 Redis-backed rate limiters, Cloudflare Turnstile, request idempotency and an origin lock — so the platform sheds bots and floods instead of trusting its callers.

Moved slow work onto a River job queue

Moved slow work off the request path onto a River job queue — 15 worker modules, 8 scheduled tasks and 20 pg_cron jobs — so a request returns while the work behind it carries on.

Built infrastructure that still runs years later

Planned and implemented new infrastructure functionality for internal and external systems, building solutions durable enough to still run years later with minimal change.

Kept IPTV streaming signals running non-stop

Ensured uninterrupted delivery of IPTV streaming signals between suppliers and clients, monitoring and maintaining the streaming network and IP telephony around the clock.

Ran 24/7 support for ~1,000 IPTV/OTT servers

Provided round-the-clock 24/7 infrastructure support for an IPTV/OTT streaming platform, administering ~1,000 servers plus client-owned systems for global customers in China, the US and Germany.

Built email with failover and delivery logging

Built email as a platform capability — three providers with failover, delivery webhooks, send and delivery logging, templating and campaigns — behind a startup check that will not boot without one.

A mapping product, carried end to end

Architected, developed and supported infrastructure, data processing and a map application over 2 years, covering both software delivery and operational maintenance.

Administered 40 websites on Apache and Nginx

Administered 40 websites on Ubuntu Linux hosting servers with Apache and Nginx, ensuring high availability and performance.

Ran 10 PostgreSQL and MS SQL servers on Ubuntu

Designed, deployed, and maintained 10 PostgreSQL and MS SQL servers on Ubuntu Linux VPS, ensuring optimal server performance and reliability.

Built the company's first observability dashboard

Developed and launched the company's first observability dashboard, providing real-time system performance insights and data visualization on the large office TV.

This service is part of work listed elsewhere on this page:

Split every operational secret per host · Made Ansible check mode tell the truth · Built dead-man's-switch host monitoring · Backup automation with an explicit restore method · Built first-party error monitoring and tracing · Owned end-to-end platform deployments to Azure · Cut operational risk with Grafana and Prometheus

{ Cloud, Infrastructure & DevOps }

Backup & Disaster Recovery

Getting everything back after a bad day

Backups that are tested rather than assumed, and a written plan for the morning a server does not come back on.

Managed 30 Ubuntu Linux VPS instances

Managed 30 Ubuntu Linux VPS instances, implementing disaster recovery strategies and ensuring optimal network configurations.

This service is part of work listed elsewhere on this page:

Split every operational secret per host · Backup automation with an explicit restore method · Implemented backups and disaster recovery as code · Automated 100 critical data backups using Barman

{ Cloud, Infrastructure & DevOps }

System Administration

Keeping the lights on

Day-to-day care of servers, mail, accounts and updates, so the things everybody relies on stay boring.

Verified firewall rules by position

Verified firewall rules by position rather than presence, reading the numbered rule list and the live packet-filter chain, because a rule that exists is not a rule any packet reaches.

Integrated a company-wide password manager

Integrated a company-wide password-management system, strengthening security and streamlining access control.

Administered 100 Bare Bone servers

Administered 100 Bare Bone servers, physical networks, and IP telephony systems, ensuring robust infrastructure for company growth.

Configured and deployed 1,000 Wi-Fi routers

Configured and deployed 1,000 Wi-Fi routers, improving network accessibility and performance for clients.

Administered network infrastructure

Administered network infrastructure for over 1,000 servers, ensuring optimal system deployment, security, and troubleshooting.

Managed 8 WireGuard and IPsec VPN connections

Managed and troubleshot 8 WireGuard VPN and IPsec VPN connections, ensuring secure communication across Google Cloud and Linux systems.

Led access management across M365 and SSO

Prevented security breaches by leading access management initiatives, utilizing M365, 1Password, Red Hat SSO, and OKTA SSO.

This service is part of work listed elsewhere on this page:

Published a readable Tor onion mirror · Made six collectors state what they skipped · Rehearsed the forge CI hook, found 2 defects · Your code on your own server · Built 7 read-only host reporting roles · Cut systemd sandbox exposure on every unit · Reconciled a 20-record DNS zone as code · Built dead-man's-switch host monitoring · Backup automation with an explicit restore method · Proved the intrusion ban works every run · Hardened SSH to 24 validated directives · Public services on a small server · Infrastructure defined in a repository · Built the entire core infrastructure from scratch · Built infrastructure that still runs years later · Kept IPTV streaming signals running non-stop · Ran 24/7 support for ~1,000 IPTV/OTT servers · Administered 40 websites on Apache and Nginx · Managed 30 Ubuntu Linux VPS instances · Wrote 1,000 RBAC rules to secure Linux servers · Ran 10 PostgreSQL and MS SQL servers on Ubuntu

{ Cloud, Infrastructure & DevOps }

Networking & VPN Setup

Connecting your offices and your people

Wi-Fi, firewalls and remote access set up so staff can work from anywhere and only the right people get in.

This service is part of work listed elsewhere on this page:

Provisioned a second server from code · Reconciled a 20-record DNS zone as code · Verified firewall rules by position · Three locks on the door, none of them locked · Kept IPTV streaming signals running non-stop · Provisioned Azure infrastructure as code ·

Administered 100 Bare Bone servers · Configured and deployed 1,000 Wi-Fi routers · Administered network infrastructure · Managed 8 WireGuard and IPsec VPN connections · Managed 30 Ubuntu Linux VPS instances

{ Cloud, Infrastructure & DevOps }

Security & Access Management

Deciding who can reach what

Accounts, passwords and permissions arranged so people hold exactly the access their job needs, and leavers lose it that day.

Wrote a scope rule after a credential leak

Wrote a scope rule into the repository after a restructure carried another company's inventory, firewall allowances and prose into it — and kept the quarantined residue under the secret scanner rather than excluding it.

Built company ownership claims and adjudication

Built company ownership claims end to end — a user claims a company, an administrator adjudicates, and an approval rewrites the authorization graph that decides who is allowed to edit what.

Built Stripe and in-app purchase entitlements

Built the payments and entitlements layer — Stripe alongside Apple and Google in-app purchase — gating the directory, search and export through an 11-table access model checked on the server.

Hardened the application with nonce-based CSP

Hardened the application with nonce-based CSP, HSTS, SameSite cookies, least-privilege database roles and server-side entitlement re-checks.

Built organization context switching for users

Designed an organization context-switching system with client localStorage and server-side cookie mirroring, letting users act as managed organizations while enforcing least-privilege authorization.

Improved team communication and collaboration

Improved team communication and collaboration by implementing Slack, Mattermost, 1Password, and Jira, saving 8,000 hours of labor.

This service is part of work listed elsewhere on this page:

Published a readable Tor onion mirror · Measured 13 addresses with no tracking · Know the licence of everything you ship · Reached the unused half of the core · Split every operational secret per host · Provisioned a second server from code · Ran containers on Podman and Quadlet · Cut systemd sandbox exposure on every unit · Reconciled a 20-record DNS zone as code · Added a one-second preflight for secrets ·

Verified firewall rules by position · Proved the intrusion ban works every run · Hardened SSH to 24 validated directives · Three locks on the door, none of them locked · Ran a continuous security scanning programme · Built fail-closed rate limiting and anti-abuse · Integrated a company-wide password manager · Automated team collaboration · Provisioned Azure infrastructure as code · Built email with failover and delivery logging · Automated SSL/TLS certs for 100 Docker apps · Administered network infrastructure · Managed 8 WireGuard and IPsec VPN connections · Led access management across M365 and SSO · Wrote 1,000 RBAC rules to secure Linux servers

{ Cloud, Infrastructure & DevOps }

AI Agent Security

Making sure your AI only touches what it should

Every agent's access mapped and held to what it needs, every action logged where the agent cannot alter it, and every safeguard tested rather than trusted.

This service is part of work listed elsewhere on this page:

Split every operational secret per host · Wrote a scope rule after a credential leak · Added a one-second preflight for secrets · Proved the intrusion ban works every run · Three locks on the door, none of them locked · Ran a continuous security scanning programme · Built fail-closed rate limiting and anti-abuse · Led access management across M365 and SSO · Wrote 1,000 RBAC rules to secure Linux servers

Compare your starting point

Choose a comparable piece of work and identify what differs in your own situation.

See what it costs Get in touch →

Discuss a relevant approach

Ask us to turn the relevant experience into an approach you can evaluate.

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://platform.engineer.company/portfolio/