Engineer Platform

{ 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.

Still working out what you need?

Tell us what is going wrong, or what you are trying to build. We will tell you what it would take, roughly what it would cost, and whether we are the right people for it.

From 1,400 DKK an hour, excluding VAT — and less per hour on a longer booking.

Get in touch See what it costs

Selected work demonstrating this service.

Turn the service into a result

Make “Infrastructure as Code” serve a business outcome. We can connect the technical work to the users, operations and decisions it needs to support. The whole setup written down as files, so it can be recreated exactly and never lives only in one person's head.

Get in touch

Examine the relevant work

Use the related work on this page to examine our approach. Ask about the tradeoffs and the conditions behind each result. 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 →

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

Looking for Infrastructure as Code?

Use the linked work to judge relevant experience. Tell us what you need to build; we can discuss the scope, delivery approach and handover.

Get in touch Full portfolio

Scope the work before commissioning it

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 →

Talk through your requirements

Tell us what you need from “Infrastructure as Code”, including the deadline and the team that will maintain it. We can discuss a practical way forward.

Get in touch

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/services/infrastructure-as-code/