Engineer

Tech stack

The technologies Engineer ApS builds with: languages, data platforms, cloud and infrastructure, and how we decide what to use.

A tool is a means. What a client buys is the judgement that picks the right one and the discipline to run it well — but you asked what we work with, so here it is, and every line of it is backed by work in the portfolio.

We write in **the languages your next hire already knows** — so the code is never ours alone.



Python Go TypeScript JavaScript PHP

Fit the tools to the work

Choose tools for the work your team needs to do and the system it will maintain.

Explore our services Get in touch →

Examine the technology decisions

Follow the work behind a technology choice and ask why it fit those particular constraints.

Cases Get in touch →

Languages and runtimes

Python for data work, automation and services. **Go** where a single static binary and low overhead matter. **TypeScript** and modern JavaScript on the web tier. **SQL** as a first-class language rather than something an ORM generates — most of the performance work we are called in for lives there. **Bash** for the glue, written to be read.

Data

PostgreSQL is the default, and we go a long way with it before reaching for anything else — partitioning, replication, extensions and the query planner. We also run **MS SQL Server**, **BigQuery**, **SQLite** and **Redis**, and the spatial stack on top: **PostGIS**, **QGIS**, **GDAL** and **ArcGIS**. Pipelines are built to be re-runnable and idempotent, because the question after an incident is always “can we replay it”.

Tools chosen **for the work** — with the trade-offs explained.

The choice depends on your existing systems, constraints and the work you need delivered.

Cloud and infrastructure

Google Cloud and **AWS**, with **Azure** where the client already lives there. **Linux** underneath everything — Debian and Ubuntu. **Docker** and **Kubernetes** for the workloads that earn them, and plain systemd units for the ones that do not. **Ansible** and **Terraform** describe the machines, so an environment is a document rather than an archaeology project. **Caddy** and **nginx** at the front.

Running it in production

Prometheus and **Grafana** for metrics, structured logging for the rest, and alerts that fire on symptoms a user would notice rather than on every twitch of a graph. CI jobs that build, test, scan and deploy without a person in the loop. Backup automation and separate restore-test workflows.

The web tier

Static-first where it fits, because a static page can be served without a request-time application layer: **Hugo** builds this site. **Django** and **PHP** where there is an application to serve. Semantic HTML, hand-written CSS, and accessibility treated as a requirement rather than an audit finding.

How we choose

Boring technology, chosen deliberately, beats novel technology chosen by default. We pick for the problem, for what your team can operate after we leave, and for how the thing fails rather than how it demonstrates. When the right answer is the stack you already have, we say so — and when it is not, we say that too, with the reasoning written down.

This site is its own worked example: the credits page names every piece of technology behind it, down to the typeface and the backup tool.

If you would rather have a supplier with opinions than one with a preference list, tell us what you are building.

We work with **the platforms** your business already runs on.


Apple


Microsoft


DigitalOcean



AWS


Google

Explore our services

On the server, we work with **open-source Linux** — no licences to pay for, and nothing hidden.


Ubuntu


Debian

How we run servers

A list of tools **is not a qualification** — knowing which one your problem wants is.

Every technology named on this page is behind work you can open and read.

See it in the work What we offer →

Start from the stack you have

Bring your existing stack, operational requirements and the skills available in your team.

See what it costs Get in touch →

Discuss a justified technical change

Discuss the smallest useful change before committing to a new platform or framework.

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/tech-stack/