Engineer Plattform

Portfolio

Ingenieurleistungen mit Beispielen umgesetzter Arbeiten.

Engineer ApS wurde 2025 eingetragen. Die meisten hier gezeigten Arbeiten hat unsere Gründerin zuvor in Kundenteams umgesetzt; einige hat das Unternehmen selbst erbracht, andere die Gründerin seitdem in Anstellungsverhältnissen. Dieselben Hände und dieselben Standards — wir benennen den jeweiligen Kontext, statt die Grenzen zu verwischen.

Erfahrung finden, die sich nutzen lässt

Nutzen Sie diese Arbeitsnachweise, um Erfahrung zu finden, die für Ihr Projekt relevant ist.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Entscheidungen hinter den Arbeiten lesen

Die Einträge beschreiben umgesetzte Arbeiten über verschiedene Leistungen hinweg. Lesen Sie eine Rückschau, um die Situation und Entscheidungen dahinter zu verstehen.

Fallstudien Kontakt aufnehmen →

Nach Kategorie stöbern

SOFTWAREENTWICKLUNG

Backend-Entwicklung · Frontend-Entwicklung · Full-Stack-Entwicklung · APIs und Integration · Webentwicklung · Tests und Qualitätssicherung · Performanceoptimierung · Python · Internationalisierung

DATEN & DATENBANKEN

Datenbanken · PostgreSQL · SQL · Data Engineering · Datenpipelines (ETL/ELT) · Datenanalyse · Data Governance · GIS / Geodaten

CLOUD, INFRASTRUKTUR & DEVOPS

Cloud · Container (Docker/Kubernetes) · DevOps

Automatisierung und CI/CD

Infrastruktur | Linux und Server | Systemadministration

Netzwerke und VPN | Monitoring und Observability

Zuverlässigkeit und Sicherungen | Sicherheit

Plattformarchitektur | Migrationen und Modernisierung

BERATUNG & FÜHRUNG

Technische Leitung | Teamführung | Agile Methoden und Scrum | Projektmanagement

Produkt und Anforderungen | Stakeholder und Berichterstattung | Dokumentation

Lösungsarchitektur | Marke und Marketing

Alles, was folgt, **wurde umgesetzt** —
nicht vorgeschlagen, nicht geplant.

Jeder Eintrag benennt, was die Arbeit verändert hat, und verlinkt die Leistungen, die sie belegt.

Was wir anbieten Was es kostet →

{ Cloud, Infrastruktur und DevOps }

Cloud-Infrastruktur und Cloud-Migration

Wo die Software läuft

Server werden nach dem tatsächlichen Bedarf dimensioniert und in die Cloud oder aus ihr heraus verlagert — die monatlichen Kosten stehen vorher fest.

Einen zweiten Server aus Code provisioniert

Die Provisionierung eines zweiten Servers bei einem zweiten Cloud-Anbieter automatisiert und die Firewall vor dem Rechner angelegt, damit er von Anfang an geschützt ist; die Firewalls beider Anbieter direkt über deren REST-APIs gesteuert, um eine Drittanbieter-Collection zu vermeiden.

Container mit Podman und Quadlet betrieben

Eine Containerebene mit Podman und Quadlet unter systemd statt Docker bereitgestellt, da Dockers standardmäßige Portfreigaben im Bridge-Netz UFW-Regeln umgehen können; für Bereitstellungen einen unprivilegierten Nutzer mit einem festen Befehl statt root eingerichtet.

Öffentliche Dienste auf einem kleinen Server

Die öffentlichen Unternehmensdienste auf einem Single-Core-Host mit 512 MB gehostet — Git-Forge, Webserver für sieben Domains, Tor, zwei Server für alternative Protokolle und Angriffssperren — mit 464 MB nutzbarem Arbeitsspeicher als Entwurfsgrenze; Sicherungsautomatisierung ist eine getrennte Implementierung.

Infrastruktur in einem Repository definiert

Die eigene Unternehmensinfrastruktur als 19 Ansible-Playbooks und 34 Rollen mit 12.065 YAML-Zeilen aufgebaut, die einen laufenden Host mit ausschließlich idempotenten Plays in den deklarierten Sollzustand überführen.

Die gesamte Kerninfrastruktur neu aufgebaut

Als eine der ersten Mitarbeiterinnen die gesamte Kerninfrastruktur und die unterstützenden Prozesse eines Green-Energy-SaaS-Start-ups von Grund auf entworfen und aufgebaut und damit die Grundlage für schnelles Wachstum gelegt.

Azure-Plattformbereitstellungen verantwortet

Die durchgängige Bereitstellung der Plattform auf Azure verantwortet und Releases über Entwicklungs-, Staging- und Produktivumgebungen hinweg gesteuert.

Azure-Infrastruktur als Code provisioniert

Azure-Infrastruktur mit Bicep als Code provisioniert — Container Apps, PostgreSQL Flexible Server, Front Door/WAF und Netzwerke — für Entwicklungs-, Staging- und Produktivumgebung.

Infrastruktur vereinfacht und AWS verlassen

Die Infrastruktur eines internationalen Kunden überarbeitet, ungenutzte Dienste entfernt, Kapazitäten angepasst und Workloads von AWS weg verlagert. Die Gründerin berichtet von einer Senkung der Cloudkosten um den Faktor 10 ohne Produktivitätsverlust.

20 Docker-Apps auf Google Cloud und AWS betrieben

20 in Docker containerisierte Anwendungen bereitgestellt und gewartet, Fehler mit Podman und Kubernetes behoben und R-basierte Anwendungen auf Google Cloud und AWS verwaltet.

{ Cloud, Infrastruktur und DevOps }

Infrastructure as Code

Server, die Sie aus versionierter Konfiguration neu aufbauen können

Die gesamte Einrichtung wird in Dateien festgehalten, damit sie sich exakt wiederherstellen lässt und nicht nur im Kopf einer einzelnen Person existiert.

Lesbaren Tor-Onion-Spiegel veröffentlicht

Die Website als Tor-Onion-Spiegel unter einer lesbaren, mit engineer beginnenden Adresse veröffentlicht, den Schlüssel außerhalb des Hosts erzeugt, sodass er weder dieses Repository noch einen Laptop erreichte, und diesen einen von vier Zugängen von der Zählung ausgenommen.

Forge-CI-Hook geprobt und 2 Fehler gefunden

Den CI-Hook auf der Forge probeweise ausgeführt und zwei beim bloßen Lesen unerreichbare Fehler gefunden: einen Fallback mit einem nicht auflösbaren Eingabeargument und eine Git-Umgebungsvariable, die der Qualitätsprüfung in den Checkout folgte und 19 Tests scheitern ließ.

Jedes Betriebsgeheimnis je Host getrennt

Die vier betrieblichen Geheimnisse je Host getrennt, nachdem sich gezeigt hatte, dass zwei Hosts mit einer gemeinsamen Totmannschaltung weniger zuverlässig alarmieren als zwei getrennte Schaltungen und eine gemeinsame Sicherungspassphrase die Sicherungen beider Hosts auch in getrennten Repositorys offenlegen kann, wenn jemand die Passphrase erlangt und die verschlüsselten Daten und Schlüsseldateien beider Repositorys lesen kann.

Ihr Code auf Ihrem eigenen Server

Die eigene Git-Forge auf Soft Serve bereitgestellt, standardmäßig privat, ohne Weboberfläche und mit an Loopback gebundenem SSH-Port hinter einem Sprunghost; die vorgeschaltete Startseite als Build-Artefakt der Hauptwebsite statt als manuell gepflegte Kopie umgesetzt.

Alle systemd-Units stärker isoliert

Die Angriffsfläche der systemd-Sandboxen für jede von der Plattform installierte Unit verringert — einen Totmannschaltungsdienst von 9,6 UNSAFE auf 1,5, eine internetseitig erreichbare Git-Forge von 8,3 EXPOSED auf 1,5 — und eine Parserprüfung beim Konvergenzlauf ergänzt, nachdem eine falsch geschriebene Direktive in drei Unit-Vorlagen stillschweigend ignoriert worden war.

DNS-Zone mit 20 Einträgen als Code abgeglichen

Eine DNS-Zone mit 20 Einträgen deklarativ über die Cloudflare-API abgeglichen, mit separaten Einstiegspunkten für Audit und BIND-Export, und den zuvor eingerichteten CDN-Proxy aus Datenschutzgründen wieder abgeschaltet.

Sekunden-Preflight für Geheimnisse ergänzt

Ein Preflight-Play ergänzt, das denselben Code wie der Konvergenzlauf mit den lokal beim Betreiber liegenden Geheimnissen in etwa einer Sekunde ausführt, nachdem ein teilweise durchgelaufener Produktivlauf bei Aufgabe neun abbrach, während die Swap-Einstellungen bereits auf dem Produktivhost geschrieben waren.

Ansibles Prüfmodus aussagekräftig gemacht

Den Prüfmodus auf der gesamten Plattform aussagekräftig gemacht, nachdem sechs Sonden anhand eines vom Host nie gelieferten Werts entschieden hatten: Ansibles command-Modul meldet unter --check Erfolg, obwohl es den Befehl vollständig überspringt.

Totmannüberwachung für Hosts entwickelt

Eine Totmannüberwachung entwickelt, die nur bei gesundem Speicher- und Plattenzustand Signale sendet, sodass ein beeinträchtigter Host durch sein Verstummen alarmiert — und sechs Variablennamen mit "free" gefunden, deren Prüfungen korrekt "available" maßen, auf einem Rechner mit 464 MB um eine Größenordnung verschieden.

Sicherungsautomatisierung mit klarem Wiederherstellungsweg

restic-Sicherungsautomatisierung mit Aufbewahrung, Integritätsprüfungen und monatlichem Wiederherstellungstest umgesetzt; Speicheraktivierung und Nachweis einer tatsächlichen Wiederherstellung als getrennte Betriebsschritte dokumentiert.

SSH mit 24 geprüften Direktiven gehärtet

SSH auf 24 geprüfte Direktiven gehärtet und dreistufig validiert — Kandidatendatei, zusammengesetzte Konfiguration und anschließend die Rückmeldung des Daemons selbst — nachdem diese gezeigt hatte, dass der laufende Server zwei der vierundzwanzig Direktiven stillschweigend überschrieb.

Sicherungen und Notfallwiederherstellung als Code

Die Aufbewahrung von Datenbanksicherungen als Infrastructure as Code konfiguriert, anschließend die Wiederherstellbarkeit geprüft und das Wiederherstellungsverfahren dokumentiert — mit ausdrücklich benannten Lücken, statt sie erst im Ernstfall entdecken zu lassen.

1.000 RBAC-Regeln für sichere Linux-Server

Die Datensicherheit durch 1.000 RBAC-Regeln für Entwickler, Anwendungsinstanzen, PostgreSQL, MS SQL und weitere Linux-Server erhöht, unberechtigte Zugriffe verhindert und die Regeln mit Ansible-Automatisierung dokumentiert.

Diese Leistung ist Teil von Arbeiten, die an anderer Stelle auf dieser Seite aufgeführt sind:

Einen zweiten Server aus Code provisioniert · Öffentliche Dienste auf einem kleinen Server · Infrastruktur in einem Repository definiert · Azure-Infrastruktur als Code provisioniert

{ Cloud, Infrastruktur und DevOps }

DevOps- und CI/CD-Automatisierung

Änderungen ohne Aufregung ausliefern

Geprüft, veröffentlicht und rückgängig zu machen: Änderungen erreichen Ihre Kunden am Dienstagnachmittag statt spät am Freitagabend.

4 Repositorys auf einen Historienstandard geeint

Vier Repositorys auf einen einheitlichen Historienstandard verpflichtet — Conventional Commits, ohne Emojis und Attribution-Trailer, durch einen Commit-Message-Hook erzwungen — ergänzt durch 48 Anweisungsdokumente zur Arbeitsweise.

Die Lizenz aller ausgelieferten Komponenten kennen

644 Rust-Crates bei jedem Build auf Lizenzverträglichkeit geprüft und die Wirksamkeit durch das Ändern einer Crate-Lizenz sowie das Verschieben der festgelegten Kernrevision ohne Neuerzeugung nachgewiesen.

Die ungenutzte Hälfte des Kerns erschlossen

Die zuvor ungenutzte Hälfte des Messaging-Kerns erschlossen — Sicherungsübertragung, verschwindende Nachrichten, Nachrichtenbearbeitung und erneutes Senden, verifizierte Einladungen, Proxys und Verschlüsselungsrichtlinien — und jeden Test ohne Mocks gegen die echte Bibliothek ausgeführt.

Testsuite auf 8,9 Sekunden beschleunigt

Die Testsuite von sechs Tests jenseits eines Limits von sechzig Sekunden auf 135 erfolgreiche Tests in 8,9 Sekunden gebracht, indem der Hauptthread profiliert und die zwei Aufrufe entfernt wurden, in denen er bei 3.989 von 4.017 Stichproben festhing.

Parser zur Prüfung aller FFI-Aufrufe geschrieben

Einen Parser geschrieben, der den echten C-Header mit 7.308 Zeilen liest und jede Aufrufstelle, jede Enum-Konstante sowie final für jede pointerbesitzende Klasse prüft, nachdem ein handgeschriebener Platzhalterheader Aufrufe dreier entfernter Funktionen kompilieren und linken ließ, die zur Laufzeit abstürzten.

Abweichende Hooks und Qualitätsprüfung gefunden

Durch den Vergleich beider Listen in einem Test festgestellt, dass Commit-Hooks und Qualitätsprüfung unterschiedliche Prüfungen ausführten, obwohl ein Dokument Gleichheit versprach — die fünf ausschließlich von Hand ausgeführten Prüfungen waren gerade diejenigen für die Lebenslauftexte.

Die Prüfwerkzeuge selbst mit Tests geprüft

Tests für die Prüfwerkzeuge selbst geschrieben, nachdem klar war, dass ein nur mit fehlerfreien Eingaben getestetes Werkzeug irgendwann auch nach dem Lesen von nichts Erfolg meldet — einen Tippfehler eingebaut, um die Rechtschreibprüfung zu prüfen, und einen ID-Bereich aus der Datenbank statt einer Testkonstante übernommen.

Jede Lint-Regel als Fehler aktiviert

Sämtliche Regeln des Python-Linters als Fehler aktiviert und 1.815 Befunde bis auf null abgearbeitet; jede der wenigen Ausnahmen schriftlich begründet und zwei davon durch eine Prüfung statt nur einen Kommentar abgesichert.

Generator mit 981 Tests und 92 % Abdeckung

Den Generator mit 981 Testfällen und mindestens 92 % Zweigabdeckung geprüft, Warnungen als Fehler behandelt und Idempotenz durch zwei vollständige Pipeline-Durchläufe aus einer leeren Datei nachgewiesen, wobei der zweite nichts ändern darf.

Qualitätsprüfung mit 10.242 Zeilen selbst geprüft

10.242 Zeilen JavaScript der Qualitätsprüfung unter Formatter und Linter gebracht, nachdem sie sich als größter und zugleich ungeprüfter Codebestand des Repositorys erwiesen hatten; 13 Befunde behoben und keinen unterdrückt.

Sitemap mit einheitlichem Zeitstempel korrigiert

Eine Sitemap korrigiert, in der 172 von 176 URLs denselben Änderungszeitpunkt trugen, indem das Datum nach der Feststellung aus der Git-Historie gelesen wurde, dass der Export jede Datei bei jedem Lauf neu schreibt.

Website nach Gemini und Gopher gespiegelt

Gemini- und Gopher-Ausgaben aus derselben Website-Quelle entwickelt; die Leistung dokumentiert für diesen Stand 777 Dokumente je Format, wobei durch das Hinzufügen der Formate null HTML-Bytes geändert wurden.

Ein Sicherheitsnetz, das keine halbe Stunde mehr kostet

Die browsergestützte Qualitätsprüfung der Website von 1.636 auf 615 Sekunden verkürzt, indem die Prüfungen in einem auf vier parallele Ausführungen begrenzten Worker-Pool nach längster Laufzeit zuerst eingeplant werden, nachdem die alphabetische Reihenfolge 320 statt 224 Sekunden benötigt hatte.

Commit-Prüfung mit 22 Lintern aufgebaut

Eine verbindliche Commit-Prüfung aus 22 einzeiligen Lintern und fünf ausführlicheren Prüfungen aufgebaut, ohne Warnungsstufe und ohne Inline-Unterdrückungen, für HTML, CSS, JavaScript, Python, YAML, Markdown, Shell, Links, Rechtschreibung, Geheimnisse und Typografie.

Kontinuierliche Sicherheitsprüfungen etabliert

Ein kontinuierliches Sicherheitsprogramm aufgebaut — Codeanalyse, DAST, Abhängigkeits- und Schwachstellenprüfungen, SBOM-Erzeugung, Geheimnissuche und per SHA festgelegte Actions — ergänzt durch 21 schriftliche Sicherheitsaudits.

1.022 Migrationen mit Schemaabweichungsprüfung

Die Konsistenz des Schemas über 1.022 Migrationen mit einer CI-Prüfung abgesichert, die die Datenbank auf zwei Wegen aufbaut — Neuinstallation sowie Installation mit sämtlichen Migrationen — und bei Unterschieden fehlschlägt.

Eigene Fehlerüberwachung und Tracing entwickelt

Eigene Fehlerüberwachung und OpenTelemetry-Tracing entwickelt statt eingekauft — Bereinigung von Nutzdaten, Erkennung von Fehlerspitzen und Regressionen, Symbolauflösung und synthetisches Lebenszeichen — mit 11 Betriebsansichten.

Repository-Prüfengine mit 268 Checks entwickelt

Die Prüfengine des Repositorys entwickelt — 268 registrierte Commit-Prüfungen, 277 Lint-Regeln und 15 eigene ESLint-Regeln — sowie 146 Tests der Prüfungen selbst, damit der Build den Standard durchsetzt und nicht erst das Review.

Testsuite für Go, SQL und Browser aufgebaut

Eine mehrschichtige automatisierte Testsuite aufgebaut — 981 Go-Tests, 543 Frontend- und Browserspezifikationen, 494 verhaltensprüfende SQL-Tests — mit Mutationstests, eigenschaftsbasierten Tests und einer verbindlichen Barrierefreiheitsprüfung.

Teamzusammenarbeit automatisiert

Teamzusammenarbeit, Passwortverwaltung sowie Aufgaben- und Zeitmanagement automatisiert und ein halbautomatisches System zur Projektpräsentation entwickelt, um die Teamproduktivität zu steigern.

Warnungsfreiheit für sechs Sprachen durchgesetzt

Für sechs Sprachen — Go, TypeScript, SQL, Python, Shell und Markdown — einen durch Pre-Commit-Hooks durchgesetzten Qualitätsstandard ohne Warnungen festgelegt.

578 go-task-Automatisierungsziele verfasst

578 go-task-Automatisierungsziele für native, Docker- und HTTPS-Entwicklungsmodi, Linting, Tests, Datenbank und Bereitstellung verfasst.

GitHub-Actions-CI/CD-Pipelines aufgebaut

GitHub-Actions-CI/CD-Pipelines mit einem Distroless-Frontend-Image für den Produktivbetrieb und der Übernahme von Releases über mehrere Umgebungen entwickelt.

Den API-Vertrag aus der Datenbank generiert

Den API-Vertrag von der Datenbank aus nach außen generiert — OpenAPI, einen typisierten TypeScript-Client mit 44.076 Zeilen, 61 Mock-Handler und die von der UI durchgesetzten Grenzen — mit einer Prüfung an jedem Übergang, die Abweichungen zurückweist.

Datenarbeit mit Python und PL/pgSQL automatisiert

Datenverarbeitungsaufgaben mit Shell-Skripten, PL/pgSQL, Python und Transact-SQL automatisiert und Produktivität sowie Effizienz gesteigert.

4.000 Stunden durch GitHub-/GitLab-CI/CD gespart

Datenanalyse und Softwareentwicklung durch CI/CD-Praktiken mit GitHub, GitLab, Bash und Python verbessert und 4.000 Stunden eingespart.

CI/CD vereinfacht und 4.000 Stunden gespart

CI/CD-Prozesse verbessert und durch die Automatisierung von Softwareentwicklungspipelines 4.000 Stunden eingespart.

SSL/TLS für 100 Docker-Apps automatisiert

Die Erstellung von SSL/TLS-Zertifikaten für 100 Docker-Anwendungen automatisiert und sichere Verbindungen auf Ubuntu-Linux-Hosts sichergestellt.

Abläufe überarbeitet und 8.000 Stunden eingespart

Interne Abläufe überarbeitet und durch bessere Softwarearchitektur, Systeme und Terminplanung 8.000 Stunden eingespart.

Betriebsrisiken mit Grafana/Prometheus gesenkt

Betriebsrisiken durch ein Monitoring-Dashboard mit Grafana und Prometheus verringert und die Systemzuverlässigkeit verbessert.

100 wichtige Sicherungen mit Barman automatisiert

100 geschäftskritische Datensicherungen mit Barman, Google Cloud, Bash und Python automatisiert und die Datenintegrität über die Datenbanken hinweg abgesichert.

20 GIS-Datenpipelines automatisiert ausgeliefert

Die Bereitstellung von 20 GIS-Datenpipelines und ETL-Prozessen für Anwendungsdaten automatisiert und dadurch Infrastrukturautomatisierung und Berichterstattung vereinfacht.

GIS-SaaS-Deployments mit GitHub automatisiert

Bereitstellung, Datenverarbeitung und Berichtswesen einer GIS-SaaS-Anwendung mit GitHub Actions CI/CD, Python, Bash und SQL automatisiert.

Diese Leistung ist Teil von Arbeiten, die an anderer Stelle auf dieser Seite aufgeführt sind:

Forge-CI-Hook geprobt und 2 Fehler gefunden · Container mit Podman und Quadlet betrieben · Ihr Code auf Ihrem eigenen Server · Sekunden-Preflight für Geheimnisse ergänzt · Ansibles Prüfmodus aussagekräftig gemacht · Infrastruktur in einem Repository definiert · Die gesamte Kerninfrastruktur neu aufgebaut · Azure-Plattformbereitstellungen verantwortet · Infrastruktur vereinfacht und AWS verlassen · 20 Docker-Apps auf Google Cloud und AWS betrieben

{ Cloud, Infrastruktur und DevOps }

Containerisierung und Orchestrierung

Software, die überall gleich läuft

So verpackt, dass sie auf dem Laptop, dem Testserver und im Produktivbetrieb gleich funktioniert und bei höherer Auslastung mitwächst.

Diese Leistung ist Teil von Arbeiten, die an anderer Stelle auf dieser Seite aufgeführt sind:

Container mit Podman und Quadlet betrieben · Alle systemd-Units stärker isoliert · GitHub-Actions-CI/CD-Pipelines aufgebaut · SSL/TLS für 100 Docker-Apps automatisiert · 20 Docker-Apps auf Google Cloud und AWS betrieben

{ Cloud, Infrastruktur und DevOps }

Site Reliability und Monitoring

Bescheid wissen, bevor Ihre Kunden es tun

Monitoring und Alarme melden sich, solange ein Fehler noch klein ist; Kennzahlen zeigen, wie oft das gelingt.

Sechs Sammler weisen übersprungene Eingaben aus

Sechs Berichtssammler um Angaben zu ungelesenen Eingaben ergänzt — ungeöffnete Dateien, nicht installierte Sonden, unbekannte Journalzeilen — und Schwellenwerte anhand von 116 gemessenen Stichproben bestimmt, nachdem eine blinde Erfassung und ein gesunder Host dieselbe Seite ergeben hatten.

13 Adressen ohne Tracking gemessen

Dreizehn Adressen ausschließlich anhand des eigenen Webserver-Zugriffsprotokolls gemessen — ohne Cookie, Tracker oder Drittanbieterdienst — das Clientnetz beim Schreiben jeder Zeile maskiert und das Ergebnis dauerhaft mit 21 täglichen Zählwerten und 20 monatlichen Facetten archiviert.

Eine begrenzte Ereignispipeline für eine macOS-App

Ereignispuffer und Protokollierung nach einem dokumentierten macOS-Vorfall mit 41 MB Wachstum pro Sekunde und 111 GB komprimierten Speicherseiten auf einem Rechner mit 36 GB begrenzt; ein dokumentierter vollständiger Testlauf reduzierte Kernprotokollzeilen von 610.996 auf 1.411.

7 lesende Host-Berichtsrollen entwickelt

Sieben schreibgeschützte Berichtsrollen entwickelt, die den Zustand eines laufenden Hosts in Markdown abbilden — Fakten, Zugriff, Git, Metriken, Verkehr, Sicherheit und Anbieterinventar — nach der Regel, dass keine Zahl erscheint, die der Lauf nicht gemessen hat.

Angriffssperren bei jedem Lauf nachgewiesen

Den Pfad zur Angriffssperre bei jedem Härtungslauf durchgängig nachgewiesen: eine reservierte Testadresse gesperrt, die entstandene Kernelregel ausgelesen und die Sperre in einem garantierten Aufräumblock aufgehoben, sodass eine ausgefallene Jail den Lauf scheitern lässt, statt einen gesunden Zustand zu melden.

Drei Türschlösser, keines davon abgeschlossen

Drei nie funktionierende SSH-Brute-Force-Schutzmechanismen gefunden und korrigiert: eine Sperr-Jail für Port 22 bei einem auf 1986 lauschenden Daemon, eine durch eine allgemeinere vorherige Regel verdeckte Ratenbegrenzung und eine Sperraktion, deren Programm nie gefunden wurde, sodass nie eine Sperre griff.

Bei Fehlern sperrenden Missbrauchsschutz gebaut

Bei Fehlern sperrende Missbrauchsschutzmechanismen aufgebaut — 22 Redis-gestützte Ratenbegrenzer, Cloudflare Turnstile, Anfrage-Idempotenz und eine Ursprungssperre — damit die Plattform Bots und Überlastungsangriffe abweist, statt ihren Aufrufern zu vertrauen.

Langsame Aufgaben in River-Jobqueue verlagert

Langsame Verarbeitung aus dem Anfragepfad in eine River-Jobqueue verlagert — 15 Worker-Module, 8 geplante Aufgaben und 20 pg_cron-Jobs — sodass Anfragen zurückkehren, während die zugehörige Arbeit weiterläuft.

Infrastruktur entwickelt, die Jahre später läuft

Neue Infrastrukturfunktionen für interne und externe Systeme geplant und umgesetzt und dabei langlebige Lösungen geschaffen, die Jahre später mit minimalen Änderungen weiterlaufen.

IPTV-Signale unterbrechungsfrei übertragen

Die unterbrechungsfreie Übertragung von IPTV-Streaming-Signalen zwischen Anbietern und Kunden sichergestellt und Streaming-Netzwerk sowie IP-Telefonie rund um die Uhr überwacht und gewartet.

24/7-Support für rund 1.000 IPTV/OTT-Server

Rund um die Uhr 24/7-Infrastruktursupport für eine IPTV/OTT-Streaming-Plattform geleistet und rund 1.000 Server sowie kundeneigene Systeme für internationale Kunden in China, den USA und Deutschland administriert.

E-Mail mit Failover und Zustellprotokoll gebaut

E-Mail als Plattformfunktion entwickelt — drei Anbieter mit Ausfallsicherung, Zustell-Webhooks, Versand- und Zustellprotokollen, Vorlagen und Kampagnen — hinter einer Startprüfung, die ohne eingerichteten Anbieter den Systemstart verhindert.

Ein Kartenprodukt durchgängig verantwortet

Infrastruktur, Datenverarbeitung und eine Kartenanwendung über 2 Jahre konzipiert, entwickelt und betreut, einschließlich Softwareauslieferung und betrieblicher Wartung.

40 Websites mit Apache und Nginx administriert

40 Websites auf Ubuntu-Linux-Hostingservern mit Apache und Nginx administriert und hohe Verfügbarkeit und Performance sichergestellt.

10 PostgreSQL-/MS-SQL-Server auf Ubuntu

10 PostgreSQL- und MS-SQL-Server auf Ubuntu-Linux-VPS entworfen, bereitgestellt und gewartet und dabei Serverperformance und Zuverlässigkeit sichergestellt.

Das erste Observability-Dashboard aufgebaut

Das erste Observability-Dashboard des Unternehmens entwickelt und eingeführt, das Systemperformance und Datenvisualisierungen in Echtzeit auf dem großen Bürofernseher anzeigt.

Diese Leistung ist Teil von Arbeiten, die an anderer Stelle auf dieser Seite aufgeführt sind:

Jedes Betriebsgeheimnis je Host getrennt · Ansibles Prüfmodus aussagekräftig gemacht · Totmannüberwachung für Hosts entwickelt · Sicherungsautomatisierung mit klarem Wiederherstellungsweg · Eigene Fehlerüberwachung und Tracing entwickelt · Azure-Plattformbereitstellungen verantwortet · Betriebsrisiken mit Grafana/Prometheus gesenkt

{ Cloud, Infrastruktur und DevOps }

Datensicherung und Notfallwiederherstellung

Nach einem schlechten Tag alles zurückbekommen

Sicherungen, deren Funktion geprüft und nicht vorausgesetzt wird, und ein schriftlicher Plan für den Morgen, an dem ein Server nicht wieder startet.

30 Ubuntu-Linux-VPS-Instanzen verwaltet

30 Ubuntu-Linux-VPS-Instanzen verwaltet, Strategien zur Notfallwiederherstellung umgesetzt und geeignete Netzwerkkonfigurationen sichergestellt.

Diese Leistung ist Teil von Arbeiten, die an anderer Stelle auf dieser Seite aufgeführt sind:

Jedes Betriebsgeheimnis je Host getrennt · Sicherungsautomatisierung mit klarem Wiederherstellungsweg · Sicherungen und Notfallwiederherstellung als Code · 100 wichtige Sicherungen mit Barman automatisiert

{ Cloud, Infrastruktur und DevOps }

Systemadministration

Den Betrieb am Laufen halten

Server, E-Mail, Konten und Updates werden im Alltag betreut, damit die Dinge, auf die alle angewiesen sind, unauffällig funktionieren.

Firewallregeln nach Position geprüft

Firewallregeln nach ihrer Position statt bloß ihrer Existenz geprüft, anhand der nummerierten Regelliste und der aktiven Paketfilterkette, denn eine vorhandene Regel wird nicht zwingend von einem Paket erreicht.

Unternehmensweiten Passwortmanager integriert

Ein unternehmensweites Passwortverwaltungssystem integriert, die Sicherheit erhöht und die Zugriffskontrolle vereinfacht.

100 Bare-Bone-Server administriert

100 Bare-Bone-Server, physische Netzwerke und IP-Telefoniesysteme administriert und eine tragfähige Infrastruktur für das Unternehmenswachstum bereitgestellt.

1.000 WLAN-Router konfiguriert und bereitgestellt

1.000 WLAN-Router konfiguriert und bereitgestellt und die Netzwerkverfügbarkeit und Performance für Kunden verbessert.

Netzwerkinfrastruktur administriert

Die Netzwerkinfrastruktur für über 1.000 Server administriert und eine geeignete Systembereitstellung, Sicherheit und Fehlerdiagnose sichergestellt.

8 WireGuard-/IPsec-VPN-Verbindungen verwaltet

8 WireGuard-VPN- und IPsec-VPN-Verbindungen verwaltet und entstört und eine sichere Kommunikation zwischen Google-Cloud- und Linux-Systemen sichergestellt.

Zugriffsverwaltung mit M365 und SSO geleitet

Durch die Leitung von Vorhaben zur Zugriffsverwaltung mit M365, 1Password, Red Hat SSO und OKTA SSO Sicherheitsverletzungen verhindert.

Diese Leistung ist Teil von Arbeiten, die an anderer Stelle auf dieser Seite aufgeführt sind:

Lesbaren Tor-Onion-Spiegel veröffentlicht · Sechs Sammler weisen übersprungene Eingaben aus · Forge-CI-Hook geprobt und 2 Fehler gefunden · Ihr Code auf Ihrem eigenen Server · 7 lesende Host-Berichtsrollen entwickelt · Alle systemd-Units stärker isoliert · DNS-Zone mit 20 Einträgen als Code abgeglichen · Totmannüberwachung für Hosts entwickelt · Sicherungsautomatisierung mit klarem Wiederherstellungsweg · Angriffssperren bei jedem Lauf nachgewiesen · SSH mit 24 geprüften Direktiven gehärtet · Öffentliche Dienste auf einem kleinen Server · Infrastruktur in einem Repository definiert ·

Die gesamte Kerninfrastruktur neu aufgebaut · Infrastruktur entwickelt, die Jahre später läuft · IPTV-Signale unterbrechungsfrei übertragen · 24/7-Support für rund 1.000 IPTV/OTT-Server · 40 Websites mit Apache und Nginx administriert · 30 Ubuntu-Linux-VPS-Instanzen verwaltet · 1.000 RBAC-Regeln für sichere Linux-Server · 10 PostgreSQL-/MS-SQL-Server auf Ubuntu

{ Cloud, Infrastruktur und DevOps }

Netzwerke und VPN-Einrichtung

Ihre Standorte und Mitarbeitenden verbinden

WLAN, Firewalls und Fernzugriff werden so eingerichtet, dass Mitarbeitende von überall arbeiten können und nur Berechtigte Zugang erhalten.

Diese Leistung ist Teil von Arbeiten, die an anderer Stelle auf dieser Seite aufgeführt sind:

Einen zweiten Server aus Code provisioniert · DNS-Zone mit 20 Einträgen als Code abgeglichen · Firewallregeln nach Position geprüft · Drei Türschlösser, keines davon abgeschlossen · IPTV-Signale unterbrechungsfrei übertragen · Azure-Infrastruktur als Code provisioniert · 100 Bare-Bone-Server administriert · 1.000 WLAN-Router konfiguriert und bereitgestellt · Netzwerkinfrastruktur administriert · 8 WireGuard-/IPsec-VPN-Verbindungen verwaltet · 30 Ubuntu-Linux-VPS-Instanzen verwaltet

{ Cloud, Infrastruktur und DevOps }

Sicherheits- und Zugriffsverwaltung

Festlegen, wer worauf zugreifen darf

Konten, Passwörter und Berechtigungen werden so geordnet, dass alle genau den beruflich erforderlichen Zugriff haben und ihn am Tag ihres Ausscheidens verlieren.

Zuständigkeit nach Geheimnisleck klar geregelt

Eine Regel zum Zuständigkeitsumfang im Repository festgeschrieben, nachdem eine Umstrukturierung Inventar, Firewallfreigaben und Texte eines anderen Unternehmens hineingetragen hatte; die isolierten Rückstände weiterhin auf Geheimnisse prüfen lassen, statt sie auszuschließen.

Unternehmensbesitzansprüche und Prüfung umgesetzt

Die Bearbeitung von Unternehmensbesitzansprüchen durchgängig umgesetzt: Ein Nutzer beansprucht ein Unternehmen, ein Administrator entscheidet und eine Genehmigung verändert den Berechtigungsgraphen, der die Bearbeitungsrechte festlegt.

Stripe- und In-App-Kaufberechtigungen entwickelt

Die Zahlungs- und Berechtigungsschicht aufgebaut — Stripe sowie In-App-Käufe von Apple und Google — und Verzeichnis, Suche und Export durch ein serverseitig geprüftes Zugriffsmodell mit 11 Tabellen abgesichert.

Anwendung mit Nonce-basierter CSP gehärtet

Die Anwendung mit Nonce-basierter CSP, HSTS, SameSite-Cookies, Datenbankrollen mit minimalen Berechtigungen und erneuten serverseitigen Zugangsprüfungen gehärtet.

Organisationswechsel für Nutzer entwickelt

Einen Wechsel des Organisationskontexts mit clientseitigem localStorage und serverseitiger Cookie-Spiegelung entworfen, sodass Nutzer im Namen verwalteter Organisationen handeln können, während das Prinzip minimaler Berechtigungen gilt.

Teamkommunikation und Zusammenarbeit verbessert

Teamkommunikation und Zusammenarbeit durch die Einführung von Slack, Mattermost, 1Password und Jira verbessert und 8.000 Arbeitsstunden eingespart.

Diese Leistung ist Teil von Arbeiten, die an anderer Stelle auf dieser Seite aufgeführt sind:

Lesbaren Tor-Onion-Spiegel veröffentlicht · 13 Adressen ohne Tracking gemessen · Die Lizenz aller ausgelieferten Komponenten kennen · Die ungenutzte Hälfte des Kerns erschlossen · Jedes Betriebsgeheimnis je Host getrennt · Einen zweiten Server aus Code provisioniert · Container mit Podman und Quadlet betrieben · Alle systemd-Units stärker isoliert · DNS-Zone mit 20 Einträgen als Code abgeglichen · Sekunden-Preflight für Geheimnisse ergänzt · Firewallregeln nach Position geprüft · Angriffssperren bei jedem Lauf nachgewiesen · SSH mit 24 geprüften Direktiven gehärtet · Drei Türschlösser, keines davon abgeschlossen ·

Kontinuierliche Sicherheitsprüfungen etabliert · Bei Fehlern sperrenden Missbrauchsschutz gebaut · Unternehmensweiten Passwortmanager integriert · Teamzusammenarbeit automatisiert · Azure-Infrastruktur als Code provisioniert · E-Mail mit Failover und Zustellprotokoll gebaut · SSL/TLS für 100 Docker-Apps automatisiert · Netzwerkinfrastruktur administriert · 8 WireGuard-/IPsec-VPN-Verbindungen verwaltet · Zugriffsverwaltung mit M365 und SSO geleitet · 1.000 RBAC-Regeln für sichere Linux-Server

{ Cloud, Infrastruktur und DevOps }

Sicherheit von KI-Agenten

Ihre KI darf nur tun, was sie soll

Der Zugriff jedes Agenten wird erfasst und auf das Nötige begrenzt, jede Handlung außerhalb seiner Änderungsbefugnis protokolliert und jede Schutzmaßnahme geprüft statt bloß vorausgesetzt.

Diese Leistung ist Teil von Arbeiten, die an anderer Stelle auf dieser Seite aufgeführt sind:

Jedes Betriebsgeheimnis je Host getrennt · Zuständigkeit nach Geheimnisleck klar geregelt · Sekunden-Preflight für Geheimnisse ergänzt · Angriffssperren bei jedem Lauf nachgewiesen · Drei Türschlösser, keines davon abgeschlossen · Kontinuierliche Sicherheitsprüfungen etabliert · Bei Fehlern sperrenden Missbrauchsschutz gebaut · Zugriffsverwaltung mit M365 und SSO geleitet · 1.000 RBAC-Regeln für sichere Linux-Server

Ihre Ausgangslage vergleichen

Wählen Sie eine vergleichbare Arbeit und bestimmen Sie, was in Ihrer eigenen Situation anders ist.

Preise ansehen Kontakt aufnehmen →

Ein passendes Vorgehen besprechen

Bitten Sie uns, aus der passenden Erfahrung ein Vorgehen abzuleiten, das Sie beurteilen können.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://platform.engineer.company/de/portfolio/