Engineer Plattform

Portfölj

Ingenjörstjänster med exempel på utfört arbete.

Engineer ApS registrerades 2025. Merparten av arbetet som visas här levererades dessförinnan av vår grundare inne i kundernas egna team; en del har levererats av företaget självt, och en del sedan dess i anställning. Samma händer och samma krav — vi säger vad som är vad i stället för att sudda ut det.

Hitta erfarenhet värd att använda

Använd översikten för att hitta erfarenhet som är relevant för ert projekt.

Se våra tjänster Hör av er →

Läs besluten bakom arbetet

Exemplen beskriver levererat arbete över flera tjänster. Läs en genomgång för att se situationen och besluten.

Fallstudier Hör av er →

Bläddra efter kategori

MJUKVARUUTVECKLING

Backendutveckling · Frontendutveckling · Fullstackutveckling · API:er och integration · Webbutveckling · Test och kvalitetssäkring · Prestandaoptimering · Python · Internationalisering

DATA & DATABASER

Databaser · PostgreSQL · SQL · Data engineering · Datapipelines (ETL/ELT) · Dataanalys · Datastyrning · GIS / geodata

MOLN, INFRASTRUKTUR & DEVOPS

Molntjänster · Containrar (Docker/Kubernetes) · DevOps · Automatisering och CI/CD

Infrastruktur · Linux och servrar · Systemadministration · Nätverk och VPN · Övervakning och observerbarhet · Tillförlitlighet och säkerhetskopiering · Säkerhet · Plattformsarkitektur · Migrering och modernisering

RÅDGIVNING & LEDARSKAP

Tekniskt ledarskap · Teamledning · Agilt arbete och Scrum · Projektledning · Produkt och krav · Intressenter och rapportering · Dokumentation · Lösningsarkitektur · Varumärke och marknadsföring

Allt nedan **är levererat** — inte föreslaget, inte planerat.

Varje post berättar vad arbetet förändrade och länkar till tjänsterna den visar.

Det vi erbjuder Vad det kostar →

{ Moln, infrastruktur & DevOps }

Molninfrastruktur och migrering

Där mjukvaran körs

Servrar dimensionerade efter vad du faktiskt använder, flyttade till molnet eller tillbaka ut ur det, med månadskostnaden uträknad först.

Provisionerade en andra server från kod

Automatiserade provisioneringen av en andra server hos en andra molnleverantör och skapade brandväggen före maskinen så att den föds bakom en, med båda leverantörernas brandväggar skrivna direkt mot deras REST-API:er för att undvika en collection från tredje part.

Körde containrar på Podman och Quadlet

Driftsatte en containermiljö på Podman och Quadlet under systemd i stället för Docker, eftersom Dockers standardpublicering av portar i bridge-nätverk kan kringgå UFW-regler — och gav driftsättningarna en oprivilegierad användare med ett enda fast kommando i stället för root.

Offentliga tjänster på en liten server

Drev företagets offentliga tjänster på en värd med en kärna och 512 MB — en git-forge, en webbserver för sju domäner, Tor, två servrar för alternativa protokoll och intrångsblockering — med 464 MB användbart minne som designbegränsning; backupautomatik är en separat implementation.

Infrastruktur beskriven i ett kodarkiv

Byggde företagets egen infrastruktur som 19 Ansible-playbooks och 34 roller på 12 065 rader YAML, som konvergerar en värd i drift mot ett deklarerat tillstånd, där varje play är idempotent.

Byggde hela kärninfrastrukturen från noll

Utformade och byggde, som en av de första anställda, hela kärninfrastrukturen och de stödjande processerna från noll för en SaaS-startup inom grön energi och lade grunden för snabb tillväxt.

Ansvarade för plattformens Azure-driftsättningar

Ansvarade för plattformens driftsättningar till Azure från början till slut och hanterade releaser i utvecklings-, staging- och produktionsmiljöerna.

Provisionerade Azure-infrastruktur som kod

Provisionerade Azure-infrastruktur som kod med Bicep — Container Apps, PostgreSQL Flexible Server, Front Door/WAF och nätverk — i utvecklings-, staging- och produktionsmiljöerna.

Förenklad infrastruktur utanför AWS

Omarbetade en internationell kunds infrastruktur genom att ta bort oanvända tjänster, anpassa kapaciteten och flytta arbetslaster från AWS. Grundaren beskriver en minskning av molnkostnaderna med en faktor 10 utan förlust av produktivitet.

Drev 20 Docker-appar på Google Cloud och AWS

Driftsatte och underhöll 20 containeriserade Docker-applikationer, felsökte med Podman och Kubernetes och förvaltade R-baserade appar på Google Cloud och AWS.

{ Moln, infrastruktur & DevOps }

Infrastruktur som kod

Servrar du kan bygga upp igen från versionshanterad konfiguration

Hela uppsättningen nedskriven som filer, så att den kan återskapas exakt och aldrig bara finns i en enda persons huvud.

Publicerade en läsbar onion-spegel i Tor

Publicerade webbplatsen som en onion-spegel i Tor på en läsbar adress som börjar med engineer, med nyckeln genererad utanför värden så att den aldrig nådde detta repo eller en bärbar dator, och lät den förbli den enda av fyra ytterdörrar som aldrig räknas.

Repeterade forgens CI-hook och hittade 2 fel

Repeterade CI-hooken på forge-sidan och hittade två fel som inte gick att se genom att läsa filen: en reservväg som lade ett argument som inte gick att lösa upp i hookens indata, och en miljövariabel från git själv som följde med grinden in i utcheckningen och gjorde 19 tester röda.

Delade upp varje driftshemlighet per värd

Delade upp de fyra driftshemligheterna per värd efter att ha konstaterat att två värdar som delar ett dödmansgrepp larmar mindre än två separata grepp, inte mer, och att en delad lösenfras för säkerhetskopior kan ge tillgång till båda värdarnas kopior, även i separata repon, om någon får tag på den och kan läsa båda reponas krypterade data och nyckelfiler.

Er kod på er egen server

Driftsatte företagets egen git-forge på Soft Serve, privat som standard, utan webbpanel och med SSH-porten bunden till loopback bakom en hoppserver, och gjorde landningssidan framför den till en byggartefakt från huvudwebbplatsen i stället för en manuellt underhållen kopia.

Minskade systemd-exponeringen för varje enhet

Minskade exponeringen i systemd-sandlådan för varje enhet som plattformen installerar — en tjänst för dödmansgrepp från 9,6 UNSAFE till 1,5, en internetexponerad git-forge från 8,3 EXPOSED till 1,5 — och lade till en parserkontroll vid konvergering efter att ha hittat ett felstavat direktiv som tyst ignorerades i tre enhetsmallar.

Stämde av en DNS-zon med 20 poster som kod

Stämde av en DNS-zon med 20 poster deklarativt mot Cloudflare-API:t med separata ingångar för granskning och BIND-export, och stängde av CDN-proxyn igen av integritetsskäl efter att ha byggt den.

Kontrollerade hemligheter i förväg på en sekund

Lade till en play för förhandskontroll som kör samma kod som konvergeringen mot operatörens lokala hemligheter på ungefär en sekund, efter att en halvt tillämpad produktionskörning kraschade vid sin nionde uppgift med swapinställningarna redan skrivna till värden i drift.

Fick Ansibles check mode att tala sanning

Fick check mode att tala sanning i hela plattformen efter att ha hittat sex sonder som fattade beslut utifrån ett värde som värddatorn aldrig gav, eftersom command-modulen i Ansible rapporterar framgång under --check medan den hoppar över kommandot helt.

Byggde värdövervakning med dödmansgrepp

Byggde övervakning med dödmansgrepp som pingar bara så länge minne och disk är i gott skick, så att en försämrad värd larmar genom att tystna — och upptäckte sex variabelnamn som sa ”free” där kontrollen korrekt mätte ”available”, en storleksordning isär på en maskin med 464 MB.

Backupautomatik med en tydlig återläsningsmetod

Implementerade restic-backupautomatik med lagringspolicy, integritetskontroll och ett flöde för månatliga återläsningstest; dokumenterade aktivering av lagring och kontroll av en verklig återläsning som separata driftsteg.

Härdade SSH till 24 validerade direktiv

Härdade SSH till 24 kontrollerade direktiv med validering i tre steg — kandidatfilen, den sammansatta konfigurationen och sedan daemonens egen återläsning — efter att återläsningen hade avslöjat att den körande servern tyst åsidosatte två av de tjugofyra.

Byggde säkerhetskopior och återställning som kod

Konfigurerade lagringstiden för databasens säkerhetskopior som infrastruktur som kod, granskade sedan återställningsförmågan och dokumenterade rutinen för återläsning — och pekade ut de återstående luckorna i stället för att låta dem upptäckas under en incident.

Skrev 1 000 RBAC-regler som säkrar Linux-servrar

Stärkte datasäkerheten genom att införa 1 000 RBAC-regler för utvecklare, applikationsinstanser, PostgreSQL, MS SQL och andra Linux-servrar, vilket förhindrade obehörig åtkomst; dokumenterade reglerna som automatisering i Ansible.

Tjänsten ingår i arbete som står annanstans på sidan:

Provisionerade en andra server från kod · Offentliga tjänster på en liten server · Infrastruktur beskriven i ett kodarkiv · Provisionerade Azure-infrastruktur som kod

{ Moln, infrastruktur & DevOps }

DevOps och CI/CD-automatisering

Leverera ändringar utan dramatik

Testade, släppta och möjliga att backa, så att ändringar når kunderna en tisdagseftermiddag i stället för sent en fredagskväll.

Lät 4 repon följa en och samma historikstandard

Lät fyra repon följa en och samma historikstandard — konventionella commits, fria från emojier, utan attributionsrader, upprätthållen av en hook för commit-meddelanden — vid sidan av 48 instruktionsdokument som styr hur arbetet utförs.

Känn till licensen för allt ni levererar

Granskade licenskompatibiliteten hos 644 Rust-crates vid varje bygge och bevisade att kontrollen slår till genom att skriva om licensen för en crate och genom att flytta den låsta kärnrevisionen utan att generera om.

Tog i bruk den oanvända halvan av kärnan

Tog i bruk den hälft av meddelandekärnan som applikationen aldrig hade använt — överföring av säkerhetskopior, försvinnande meddelanden, redigering och omsändning av meddelanden, verifierade inbjudningar, proxyer och krypteringspolicy — och körde varje test mot det riktiga biblioteket utan mockar.

Tog testsviten ned till 8,9 sekunder

Tog testsviten från sex tester över en gräns på sextio sekunder till 135 godkända på 8,9 sekunder genom att profilera huvudtråden och ta bort de två anrop som den befann sig i vid 3 989 av 4 017 stickprov.

Skrev en parser som verifierar varje FFI-anrop

Skrev en parser som läser den verkliga C-headern på 7 308 rader och verifierar varje anropsställe, varje enum-konstant och att varje klass som äger en pekare är final, efter att en handskriven platshållarheader hade låtit anrop till tre borttagna funktioner kompilera, länka och krascha.

Upptäckte att hookarna och grinden var oense

Upptäckte att commit-hookarna och kvalitetsgrinden körde olika kontroller medan ett dokument lovade att de var desamma, genom att jämföra de två listorna i ett test — de fem som bara någonsin kördes för hand var just de som läste CV-texten.

Skrev tester för själva kontrollerna

Skrev tester för själva kontrollerna efter att ha konstaterat att en kontroll som bara matas med felfri indata en dag rapporterar felfritt för att den inte läste något — lade in en avsiktlig felstavning för att bekräfta att stavningskontrollen hittar den och hämtade ett id-intervall från databasen i stället för från ett tal i testet.

Aktiverade varje lintregel på felnivå

Aktiverade varje regel i Python-lintern på felnivå och arbetade igenom 1 815 fynd ned till noll, där vart och ett av de få undantagen har ett skriftligt skäl och två av dem backas upp av en kontroll i stället för en kommentar.

Gav generatorn 981 tester och 92 % täckning

Krävde 981 testfall och minst 92 % grentäckning av generatorn, med varningar behandlade som fel, och säkerställde idempotens genom att köra hela byggpipelinen två gånger från en tom fil och kräva att den andra körningen inte ändrade något.

Lintade själva kvalitetsgrinden på 10 242 rader

Förde in 10 242 rader JavaScript för kvalitetsgrinden under en formaterare och en linter efter att ha konstaterat att det var den största kodmassan i repot och den enda som inget läste, och åtgärdade 13 fynd utan att undertrycka något.

Rättade en sitemap med en enda delad tidsstämpel

Rättade en sitemap där 172 av 176 URL:er delade samma ändringstidsstämpel genom att hämta datumet ur git-historiken, efter att ha konstaterat att exporten skriver om varje fil vid varje körning.

Speglade webbplatsen till Gemini och Gopher

Byggde Gemini- och Gopher-utdata från samma webbkälla; resultatbeskrivningen anger 777 dokument i varje format vid det skedet och noll ändrade HTML-byte genom tillägget av formaten.

Ett skyddsnät som inte längre kostar en halvtimme

Kortade webbplatsens webbläsarstyrda kvalitetsgrind från 1 636 sekunder till 615 genom att schemalägga kontrollerna med den längsta först i en workerpool begränsad till fyra spår, efter att ha mätt att alfabetisk ordning kostade 320 sekunder mot 224.

Byggde en commit-grind med 22 linters

Byggde en commit-grind med 22 linters på en rad var plus fem som förtjänar ett eget stycke, utan varningsnivå och utan tillåtna inline-undantag, som täcker HTML, CSS, JavaScript, Python, YAML, Markdown, shell, länkar, stavning, hemligheter och typografi.

Drev ett löpande program för säkerhetsskanning

Etablerade ett kontinuerligt säkerhetsprogram — kodskanning, DAST, kontroller av beroenden och sårbarheter, generering av SBOM, skanning efter hemligheter och SHA-låsta actions — vid sidan av 21 skriftliga säkerhetsgranskningar.

1 022 migreringar bakom en grind mot schemadrift

Höll schemat korrekt genom 1 022 migreringar med en CI-grind som bygger databasen på båda sätten — en ny installation, och en installation plus varje migrering — och fäller när de två skiljer sig åt.

Byggde egen felövervakning och spårning

Byggde egen felövervakning och OpenTelemetry-spårning i stället för att köpa dem — sanering av nyttolaster, upptäckt av toppar och regressioner, symbolisering och ett syntetiskt hjärtslag — bakom 11 operatörsvyer.

Byggde repots kontrollmotor med 268 kontroller

Byggde repots kontrollmotor — 268 registrerade commit-kontroller, 277 lintregler och 15 egna ESLint-regler — plus 146 tester av själva kontrollerna, så att det är bygget som håller standarden, inte granskningen.

Byggde en testsvit för Go, SQL och webbläsare

Byggde en skiktad automatiserad testsvit — 981 Go-tester, 543 specifikationer för frontend och webbläsare, 494 beteendetester i SQL — med mutationstestning, egenskapsbaserade tester och en tillgänglighetsgrind.

Automatiserade teamets samarbete

Automatiserade teamets samarbete, lösenordshantering samt uppgifts- och tidshantering och byggde ett halvautomatiskt system för att visa upp projekt, vilket höjde teamets produktivitet.

Satte ribban vid noll varningar i sex språk

Satte kvalitetsribban vid noll varningar i sex språk — Go, TypeScript, SQL, Python, Shell och Markdown — upprätthållen av pre-commit-hookar.

Skrev 578 automationsmål i go-task

Skrev 578 automationsmål i go-task som täcker utvecklingslägena native, Docker och HTTPS samt lintning, testning, databas och driftsättning.

Byggde CI/CD-pipelines i GitHub Actions

Byggde CI/CD-pipelines i GitHub Actions med en distroless-image för frontenden i produktion och uppflyttning mellan flera miljöer.

Genererade API-kontraktet utåt från databasen

Genererade API-kontraktet utåt från databasen — OpenAPI, en typad TypeScript-klient på 44 076 rader, 61 mock-hanterare och de gränser som gränssnittet upprätthåller — med en spärr i varje led som fäller vid avvikelse.

Automatiserade databearbetning i Python, PL/pgSQL

Automatiserade uppgifter inom databearbetning med Shell-skript, PL/pgSQL, Python och Transact-SQL och ökade produktiviteten och effektiviteten.

Sparade 4 000 timmar med GitHub och GitLab CI/CD

Effektiviserade processerna för dataanalys och mjukvaruutveckling och sparade 4 000 timmar genom att införa CI/CD-arbetssätt med GitHub, GitLab, Bash och Python.

Effektiviserade CI/CD och sparade 4 000 timmar

Effektiviserade CI/CD-processerna och sparade 4 000 timmar genom att införa automatisering i pipelines för mjukvaruutveckling.

Automatiserade SSL/TLS-cert för 100 Docker-appar

Automatiserade skapandet av SSL/TLS-certifikat för 100 Docker-applikationer och säkerställde säkra anslutningar på alla värdar med Ubuntu Linux.

Förnyade interna processer, sparade 8 000 timmar

Gjorde en genomgripande översyn av de interna processerna och sparade 8 000 timmar genom att förbättra mjukvaruarkitekturen, systemen och effektiviteten i schemaläggningen.

Minskade driftrisken med Grafana och Prometheus

Minskade driftriskerna genom att införa en dashboard för övervakning med Grafana och Prometheus, vilket förbättrade systemets tillförlitlighet.

Automatiserade 100 kritiska säkerhetskopior

Automatiserade 100 kritiska säkerhetskopior av data med Barman, Google Cloud, Bash och Python och säkerställde dataintegriteten i samtliga databaser.

Automatiserade leveransen av 20 GIS-datapipelines

Automatiserade leveransen av 20 GIS-datapipelines och ETL-processer för applikationsdata, vilket effektiviserade infrastrukturautomatiseringen och rapporteringen.

Automatiserade GIS-driftsättning i GitHub Actions

Automatiserade driftsättningen av en GIS-applikation som SaaS, databearbetningen och rapporteringssystemet med GitHub Actions CI/CD, Python, Bash och SQL.

Tjänsten ingår i arbete som står annanstans på sidan:

Repeterade forgens CI-hook och hittade 2 fel · Körde containrar på Podman och Quadlet · Er kod på er egen server · Kontrollerade hemligheter i förväg på en sekund · Fick Ansibles check mode att tala sanning · Infrastruktur beskriven i ett kodarkiv · Byggde hela kärninfrastrukturen från noll · Ansvarade för plattformens Azure-driftsättningar · Förenklad infrastruktur utanför AWS · Drev 20 Docker-appar på Google Cloud och AWS

{ Moln, infrastruktur & DevOps }

Containerisering och orkestrering

Mjukvara som fungerar likadant överallt

Paketerad för att bete sig identiskt på en laptop, på en testserver och i produktion, och för att skala upp när en hektisk vecka kommer.

Tjänsten ingår i arbete som står annanstans på sidan:

Körde containrar på Podman och Quadlet · Minskade systemd-exponeringen för varje enhet · Byggde CI/CD-pipelines i GitHub Actions · Automatiserade SSL/TLS-cert för 100 Docker-appar · Drev 20 Docker-appar på Google Cloud och AWS

{ Moln, infrastruktur & DevOps }

Tillförlitlig drift och övervakning

Vet före dina kunder

Övervakning och larm som säger till medan ett fel fortfarande är litet, och siffrorna som visar hur ofta det förblir så.

Fick sex insamlare att ange vad de hoppade över

Fick sex rapporterande insamlare att ange den del av sin indata som de aldrig läste — filer som förblev stängda, sonder som inte installerats, journalrader som inte kändes igen — och anpassade tröskelvärden till 116 mätningar, efter att ha konstaterat att en blind insamling och en värd i gott skick ger samma sida.

Mätte 13 adresser helt utan spårning

Mätte tretton adresser enbart utifrån webbserverns egen åtkomstlogg — ingen cookie, ingen spårare och ingen tjänst från tredje part — maskerade klientens nätverk när varje rad skrivs och förde in resultatet i ett permanent arkiv med 21 dagliga räkningar och 20 månatliga fasetter.

Ett begränsat händelseflöde i en macOS-app

Begränsade händelsebuffertar och loggning efter en dokumenterad macOS-incident med en tillväxt på 41 MB i sekunden och 111 GB komprimerade sidor på en maskin med 36 GB; en dokumenterad fullständig testkörning minskade kärnans loggrader från 610 996 till 1 411.

Byggde 7 skrivskyddade rapportroller för värdar

Byggde sju skrivskyddade rapportroller som återger en värd i drift som Markdown — fakta, åtkomst, git, mätvärden, trafik, säkerhet och leverantörsinventering — under regeln att inget tal skrivs ut som körningen inte har mätt.

Bevisade att intrångsspärren fungerar varje gång

Bevisade spärrkedjan mot intrång från början till slut vid varje härdningskörning genom att spärra en reserverad testadress, läsa den kärnregel som uppstod och häva spärren i ett städblock som alltid körs, så att en jail som slutar fungera fäller körningen i stället för att rapportera att allt är i ordning.

Tre lås på dörren, inget av dem låst

Hittade och åtgärdade tre skydd mot brute force-attacker över SSH som aldrig hade fungerat: en jail som bevakade port 22 medan daemonen lyssnade på 1986, en hastighetsbegränsning som skuggades av en bredare regel ovanför den och en spärråtgärd vars binärfil aldrig gick att hitta, så att ingen spärr någonsin hade tillämpats.

Byggde hastighetsbegränsning som stänger vid fel

Byggde skydd mot missbruk som stänger vid fel — 22 hastighetsbegränsare i Redis, Cloudflare Turnstile, idempotenta anrop och ett ursprungslås — så att plattformen avvisar botar och anropsfloder i stället för att lita på sina anropare.

Flyttade långsamt arbete till en jobbkö i River

Flyttade långsamt arbete från anropsvägen till en jobbkö i River — 15 workermoduler, 8 schemalagda uppgifter och 20 pg_cron-jobb — så att ett anrop returnerar medan arbetet bakom det fortsätter.

Byggde infrastruktur som ännu körs år senare

Planerade och implementerade ny infrastrukturfunktionalitet för interna och externa system och byggde lösningar som höll tillräckligt väl för att fortfarande vara i drift flera år senare med minimala ändringar.

Höll strömmade IPTV-signaler igång oavbrutet

Säkerställde oavbruten leverans av strömmade IPTV-signaler mellan leverantörer och kunder och övervakade och underhöll strömningsnätet och IP-telefonin dygnet runt.

Gav support 24/7 för ca 1 000 IPTV/OTT-servrar

Gav infrastruktursupport dygnet runt, 24/7, för en IPTV/OTT-strömningsplattform och administrerade ca 1 000 servrar samt kundägda system för globala kunder i Kina, USA och Tyskland.

Byggde e-post med failover och leveransloggning

Byggde e-post som en förmåga i plattformen — tre leverantörer med failover, webhooks för leveranser, loggning av utskick och leveranser, mallar och kampanjer — bakom en startkontroll som vägrar starta utan en leverantör.

En kartprodukt, buren hela vägen

Utformade, utvecklade och stödde infrastruktur, databearbetning och en kartapplikation under 2 år, med både programvaruleveranser och driftunderhåll.

Administrerade 40 webbplatser på Apache och Nginx

Administrerade 40 webbplatser på värdservrar med Ubuntu Linux, Apache och Nginx och säkerställde hög tillgänglighet och prestanda.

Drev 10 PostgreSQL- och MS SQL-servrar på Ubuntu

Utformade, driftsatte och underhöll 10 PostgreSQL- och MS SQL-servrar på VPS:er med Ubuntu Linux och säkerställde optimal serverprestanda och tillförlitlighet.

Byggde företagets första observerbarhetsdashboard

Utvecklade och lanserade företagets första dashboard för observerbarhet, med realtidsinsikt i systemens prestanda och datavisualisering på den stora tv:n på kontoret.

Tjänsten ingår i arbete som står annanstans på sidan:

Delade upp varje driftshemlighet per värd · Fick Ansibles check mode att tala sanning · Byggde värdövervakning med dödmansgrepp · Backupautomatik med en tydlig återläsningsmetod · Byggde egen felövervakning och spårning · Ansvarade för plattformens Azure-driftsättningar · Minskade driftrisken med Grafana och Prometheus

{ Moln, infrastruktur & DevOps }

Säkerhetskopiering och återställning

Få tillbaka allt efter en dålig dag

Säkerhetskopior som testas i stället för att tas för givna, och en skriftlig plan för den morgon då en server inte startar igen.

Förvaltade 30 VPS-instanser med Ubuntu Linux

Förvaltade 30 VPS-instanser med Ubuntu Linux, införde strategier för katastrofåterställning och säkerställde optimala nätverkskonfigurationer.

Tjänsten ingår i arbete som står annanstans på sidan:

Delade upp varje driftshemlighet per värd · Backupautomatik med en tydlig återläsningsmetod · Byggde säkerhetskopior och återställning som kod · Automatiserade 100 kritiska säkerhetskopior

{ Moln, infrastruktur & DevOps }

Systemadministration

Håller allt igång

Daglig skötsel av servrar, e-post, konton och uppdateringar, så att det som alla förlitar sig på förblir odramatiskt.

Verifierade brandväggsregler efter position

Verifierade brandväggsregler efter position snarare än förekomst och läste den numrerade regellistan och den aktiva paketfilterkedjan, eftersom en regel som finns inte är detsamma som en regel som något paket når.

Införde en företagsgemensam lösenordshanterare

Integrerade ett företagsövergripande system för lösenordshantering, vilket stärkte säkerheten och effektiviserade åtkomstkontrollen.

Administrerade 100 Bare Bone-servrar

Administrerade 100 Bare Bone-servrar, fysiska nätverk och system för IP-telefoni och säkerställde en robust infrastruktur för företagets tillväxt.

Konfigurerade och driftsatte 1 000 Wi-Fi-routrar

Konfigurerade och driftsatte 1 000 Wi-Fi-routrar och förbättrade nätverkets tillgänglighet och prestanda för kunderna.

Administrerade nätverksinfrastrukturen

Administrerade nätverksinfrastrukturen för över 1 000 servrar och säkerställde optimal driftsättning, säkerhet och felsökning av systemen.

Förvaltade och felsökte 8 VPN-anslutningar

Förvaltade och felsökte 8 VPN-anslutningar över WireGuard och IPsec och säkerställde säker kommunikation i Google Cloud och på Linux-system.

Ledde behörighetshanteringen i M365 och SSO

Förhindrade säkerhetsintrång genom att leda initiativ för behörighetshantering med M365, 1Password, Red Hat SSO och OKTA SSO.

Tjänsten ingår i arbete som står annanstans på sidan:

Publicerade en läsbar onion-spegel i Tor · Fick sex insamlare att ange vad de hoppade över · Repeterade forgens CI-hook och hittade 2 fel · Er kod på er egen server · Byggde 7 skrivskyddade rapportroller för värdar · Minskade systemd-exponeringen för varje enhet · Stämde av en DNS-zon med 20 poster som kod · Byggde värdövervakning med dödmansgrepp · Backupautomatik med en tydlig återläsningsmetod · Bevisade att intrångsspärren fungerar varje gång · Härdade SSH till 24 validerade direktiv · Offentliga tjänster på en liten server · Infrastruktur beskriven i ett kodarkiv · Byggde hela kärninfrastrukturen från noll · Byggde infrastruktur som ännu körs år senare · Höll strömmade IPTV-signaler igång oavbrutet · Gav support 24/7 för ca 1 000 IPTV/OTT-servrar · Administrerade 40 webbplatser på Apache och Nginx · Förvaltade 30 VPS-instanser med Ubuntu Linux · Skrev 1 000 RBAC-regler som säkrar Linux-servrar · Drev 10 PostgreSQL- och MS SQL-servrar på Ubuntu

{ Moln, infrastruktur & DevOps }

Nätverk och VPN-uppsättning

Kopplar ihop dina kontor och dina medarbetare

Wi-Fi, brandväggar och fjärråtkomst uppsatta så att personalen kan arbeta varifrån som helst och bara rätt personer kommer in.

Tjänsten ingår i arbete som står annanstans på sidan:

Provisionerade en andra server från kod · Stämde av en DNS-zon med 20 poster som kod · Verifierade brandväggsregler efter position · Tre lås på dörren, inget av dem låst · Höll strömmade IPTV-signaler igång oavbrutet · Provisionerade Azure-infrastruktur som kod · Administrerade 100 Bare Bone-servrar · Konfigurerade och driftsatte 1 000 Wi-Fi-routrar · Administrerade nätverksinfrastrukturen · Förvaltade och felsökte 8 VPN-anslutningar · Förvaltade 30 VPS-instanser med Ubuntu Linux

{ Moln, infrastruktur & DevOps }

Säkerhet och behörighetshantering

Bestämmer vem som når vad

Konton, lösenord och behörigheter ordnade så att alla har exakt den åtkomst deras arbete kräver, och de som slutar förlorar den samma dag.

Skrev en avgränsningsregel efter läckta uppgifter

Skrev in en avgränsningsregel i repot efter att en omstrukturering hade fört in ett annat företags värdförteckning, brandväggsundantag och text i det — och höll de karantänsatta resterna kvar under skanningen efter hemligheter i stället för att undanta dem.

Byggde anspråk på företagsägarskap och prövning

Byggde anspråk på företagsägarskap från början till slut — en användare gör anspråk på ett företag, en administratör avgör, och ett godkännande skriver om auktoriseringsgrafen som bestämmer vem som får redigera vad.

Byggde åtkomsträttigheter via Stripe och appköp

Byggde skiktet för betalningar och åtkomsträttigheter — Stripe vid sidan av köp i appen via Apple och Google — som spärrar katalogen, sökningen och exporten bakom en åtkomstmodell i 11 tabeller som kontrolleras på servern.

Härdade applikationen med nonce-baserad CSP

Härdade applikationen med nonce-baserad CSP, HSTS, SameSite-cookies, databasroller med minsta behörighet och förnyade kontroller av åtkomsträttigheter på serversidan.

Byggde växling av organisationskontext

Utformade ett system för att växla organisationskontext med localStorage hos klienten och en speglad cookie på serversidan, så att användare kan agera som de organisationer de förvaltar medan auktorisering enligt minsta behörighet upprätthålls.

Förbättrade teamets kommunikation och samarbete

Förbättrade teamets kommunikation och samarbete genom att införa Slack, Mattermost, 1Password och Jira, vilket sparade 8 000 arbetstimmar.

Tjänsten ingår i arbete som står annanstans på sidan:

Publicerade en läsbar onion-spegel i Tor · Mätte 13 adresser helt utan spårning · Känn till licensen för allt ni levererar · Tog i bruk den oanvända halvan av kärnan · Delade upp varje driftshemlighet per värd · Provisionerade en andra server från kod · Körde containrar på Podman och Quadlet · Minskade systemd-exponeringen för varje enhet · Stämde av en DNS-zon med 20 poster som kod · Kontrollerade hemligheter i förväg på en sekund · Verifierade brandväggsregler efter position · Bevisade att intrångsspärren fungerar varje gång · Härdade SSH till 24 validerade direktiv · Tre lås på dörren, inget av dem låst · Drev ett löpande program för säkerhetsskanning · Byggde hastighetsbegränsning som stänger vid fel · Införde en företagsgemensam lösenordshanterare · Automatiserade teamets samarbete · Provisionerade Azure-infrastruktur som kod · Byggde e-post med failover och leveransloggning · Automatiserade SSL/TLS-cert för 100 Docker-appar · Administrerade nätverksinfrastrukturen · Förvaltade och felsökte 8 VPN-anslutningar · Ledde behörighetshanteringen i M365 och SSO · Skrev 1 000 RBAC-regler som säkrar Linux-servrar

{ Moln, infrastruktur & DevOps }

Säkerhet för AI-agenter

Hålla AI:n borta från det den inte ska röra

Varje agents behörigheter kartlagda och begränsade till det nödvändiga, varje handling loggad där agenten inte kan ändra loggen, och varje skydd prövat i stället för taget för givet.

Tjänsten ingår i arbete som står annanstans på sidan:

Delade upp varje driftshemlighet per värd · Skrev en avgränsningsregel efter läckta uppgifter · Kontrollerade hemligheter i förväg på en sekund · Bevisade att intrångsspärren fungerar varje gång · Tre lås på dörren, inget av dem låst · Drev ett löpande program för säkerhetsskanning · Byggde hastighetsbegränsning som stänger vid fel · Ledde behörighetshanteringen i M365 och SSO · Skrev 1 000 RBAC-regler som säkrar Linux-servrar

Jämför ert utgångsläge

Välj ett jämförbart arbete och identifiera skillnaderna i er egen situation.

Se vad det kostar Hör av er →

Diskutera ett relevant angreppssätt

Be oss göra relevant erfarenhet till ett angreppssätt ni kan bedöma.

Se våra tjänster

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://platform.engineer.company/sv/portfolio/