Engineer Plattform

Portefølje

Ingeniørtjenester med eksempler på utført arbeid.

Engineer ApS ble registrert i 2025. Mesteparten av arbeidet som vises her, ble levert før det av gründeren vår inne i kundenes egne team; noe er levert av selskapet selv, og noe siden i ansettelsesforhold. De samme hendene og de samme standardene — vi sier hva som er hva i stedet for å viske det ut.

Finn erfaring det er verdt å bruke

Bruk oversikten til å finne erfaring som er relevant for prosjektet deres.

Se tjenestene våre Ta kontakt →

Les beslutningene bak arbeidet

Eksemplene beskriver levert arbeid på tvers av tjenester. Les en gjennomgang for å se situasjonen og beslutningene.

Prosjekter Ta kontakt →

Bla etter kategori

PROGRAMVAREUTVIKLING

Backend-utvikling · Frontend-utvikling · Fullstack-utvikling · API-er og integrasjon · Webutvikling · Testing og kvalitetssikring · Ytelsesoptimalisering · Python · Internasjonalisering

DATA & DATABASER

Databaser · PostgreSQL · SQL · Data engineering · Datapipelines (ETL/ELT) · Dataanalyse · Datastyring · GIS / geodata

SKY, INFRASTRUKTUR & DEVOPS

Skytjenester · Containere (Docker/Kubernetes) · DevOps · Automatisering og CI/CD · Infrastruktur

Linux og servere · Systemadministrasjon · Nettverk og VPN · Overvåking og observerbarhet · Pålitelighet og sikkerhetskopier · Sikkerhet · Plattformarkitektur · Migrering og modernisering

RÅDGIVNING & LEDELSE

Teknisk ledelse · Teamledelse · Smidig og Scrum · Prosjektledelse · Produkt og krav · Interessenter og rapportering · Dokumentasjon · Løsningsarkitektur · Merkevare og markedsføring

Alt nedenfor **er levert** — ikke foreslått, ikke planlagt.

Hver oppføring forteller hva arbeidet endret, og lenker til tjenestene den viser.

Det vi tilbyr Hva det koster →

{ Sky, infrastruktur & DevOps }

Skyinfrastruktur og migrering

Der programvaren kjører

Servere dimensjonert etter det du faktisk bruker, flyttet inn i skyen eller ut av den igjen, med månedsregningen regnet ut på forhånd.

Klargjorde en server nummer to fra kode

Automatiserte klargjøringen av en server nummer to hos en annen skyleverandør, der brannmuren opprettes før maskinen slik at den blir født bak en, og begge leverandørenes brannmurer er skrevet direkte mot deres REST-API-er for å slippe en collection fra tredjepart.

Kjørte containere på Podman og Quadlet

Satte et containerlag på Podman og Quadlet under systemd i produksjon i stedet for Docker, fordi Dockers standardpublisering av porter på bridge-nettverk kan omgå UFW-regler — og ga produksjonssettinger en uprivilegert bruker med én fast kommando i stedet for root.

Offentlige tjenester på en liten server

Driftet selskapets offentlige tjenester på en vert med én kjerne og 512 MB — en git-forge, en webserver for sju domener, Tor, to servere for alternative protokoller og utestenging av inntrengere — med 464 MB brukbart minne som designbegrensning; backupautomatisering er en separat implementasjon.

Infrastruktur beskrevet i et kodelager

Bygde selskapets egen infrastruktur som 19 Ansible-playbooks og 34 roller på 12 065 linjer YAML, som konvergerer en vert i drift mot en deklarert tilstand der hver play er idempotent.

Bygde hele kjerneinfrastrukturen fra bunnen av

Designet og bygde, som en av de første ansatte, hele kjerneinfrastrukturen og de tilhørende prosessene fra bunnen av for en SaaS-oppstartsbedrift innen grønn energi, og la grunnlaget for rask vekst.

Hadde ansvar for Azure-utrulling av plattformen

Hadde ansvaret for produksjonssettingen av plattformen til Azure fra ende til annen, og styrte utgivelser gjennom utviklings-, staging- og produksjonsmiljøene.

Klargjorde Azure-infrastruktur som kode med Bicep

Klargjorde Azure-infrastruktur som kode med Bicep — Container Apps, PostgreSQL Flexible Server, Front Door/WAF og nettverk — i utviklings-, staging- og produksjonsmiljøene.

Forenklet infrastruktur utenfor AWS

La om infrastrukturen til en internasjonal kunde ved å fjerne ubrukte tjenester, tilpasse kapasiteten og flytte arbeidslaster bort fra AWS. Grunnleggeren beskriver en reduksjon i skykostnadene med en faktor på 10 uten tap av produktivitet.

Driftet 20 Docker-apper på Google Cloud og AWS

Satte i produksjon og vedlikeholdt 20 containeriserte Docker-applikasjoner, feilsøkte med Podman og Kubernetes og administrerte R-baserte applikasjoner på Google Cloud og AWS.

{ Sky, infrastruktur & DevOps }

Infrastruktur som kode

Servere du kan bygge opp igjen fra versjonsstyrt konfigurasjon

Hele oppsettet skrevet ned som filer, slik at det kan gjenskapes nøyaktig og aldri bare finnes i hodet på én person.

Publiserte et lesbart onion-speil på Tor

Publiserte nettstedet som et Tor onion-speil på en lesbar adresse som begynner med engineer, med nøkkelen generert utenfor verten slik at den aldri havnet i dette repoet eller på en bærbar PC, og lot det være den ene av fire inngangsdører som aldri telles.

Prøvekjørte CI-hooken i forgen og fant 2 feil

Prøvekjørte CI-hooken på forge-siden og fant to feil som ikke kunne oppdages ved å lese filen: en reserveløsning som la et uoppløselig argument på hookens inndata, og gits egen miljøvariabel som fulgte kvalitetsporten inn i utsjekkingen og gjorde 19 tester røde.

Delte opp hver driftshemmelighet per vert

Delte opp de fire driftshemmelighetene per vert etter å ha slått fast at to verter som deler én dødmannsknapp, varsler mindre enn to knapper, ikke mer, og at en felles passordfrase for sikkerhetskopier kan gi tilgang til begge vertenes kopier, selv i separate repoer, hvis noen får tak i den og kan lese de krypterte dataene og nøkkelfilene i begge repoene.

Koden deres på deres egen server

Satte selskapets egen git-forge på Soft Serve i produksjon, privat som standard, uten webpanel og med SSH-porten bundet til loopback bak en hoppvert, og gjorde landingssiden foran den til en byggeartefakt fra hovednettstedet i stedet for en kopi som vedlikeholdes for hånd.

Strammet inn systemd-sandkassen for hver enhet

Reduserte eksponeringen i systemd-sandkassen for hver enhet plattformen installerer — tjenesten for dødmannsknappen fra 9,6 UNSAFE til 1,5, en internetteksponert git-forge fra 8,3 EXPOSED til 1,5 — og la til en parserkontroll ved konvergering etter å ha funnet et feilstavet direktiv som ble ignorert i det stille i tre enhetsmaler.

Avstemte en DNS-sone med 20 poster som kode

Avstemte en DNS-sone med 20 poster deklarativt mot Cloudflare-API-et med egne inngangspunkter for revisjon og BIND-eksport, og slo CDN-proxyen av igjen av personvernhensyn etter å ha bygd den.

La til ett sekunds preflight for hemmeligheter

La til en preflight-play som kjører den samme koden som konvergeringen mot operatørens lokale hemmeligheter på omtrent ett sekund, etter at en halvveis gjennomført produksjonskjøring stoppet på den niende oppgaven med swap-innstillingene allerede skrevet til verten i drift.

Fikk Ansibles check-modus til å si sannheten

Fikk check-modus til å fortelle sannheten på tvers av hele plattformen etter å ha funnet seks sonder som avgjorde ut fra en verdi verten aldri hadde gitt, fordi Ansibles command-modul melder suksess under --check og samtidig hopper helt over kommandoen.

Bygde vertsovervåking med dødmannsknapp

Bygde overvåking med dødmannsknapp som bare sender ping så lenge minne og disk er i orden, slik at en svekket vert utløser et varsel ved å bli stille — og fanget opp seks variabelnavn som sa «free» der kontrollen korrekt målte «available», en størrelsesorden fra hverandre på en maskin med 464 MB.

Backupautomatisering med en tydelig gjenopprettingsmetode

Implementerte restic-backupautomatisering med lagringsregler, integritetskontroll og en arbeidsflyt for månedlig gjenopprettingstest; dokumenterte aktivering av lagring og kontroll av en virkelig gjenoppretting som separate driftsoppgaver.

Herdet SSH til 24 validerte direktiver

Herdet SSH til 24 kontrollerte direktiver med validering i tre trinn — kandidatfilen, den sammensatte konfigurasjonen og så daemonens egen tilbakelesing — etter at tilbakelesingen avslørte at den kjørende serveren i det stille overstyrte to av de tjuefire.

La sikkerhetskopi og katastrofeberedskap i kode

Konfigurerte lagringstiden for databasens sikkerhetskopier som infrastruktur som kode, gjennomgikk deretter gjenopprettingsberedskapen og dokumenterte fremgangsmåten for gjenoppretting — og navnga de gjenværende hullene i stedet for å la dem bli oppdaget under en hendelse.

Skrev 1 000 RBAC-regler for å sikre Linux-servere

Styrket datasikkerheten ved å innføre 1 000 RBAC-regler for utviklere, applikasjonsinstanser, PostgreSQL, MS SQL og andre Linux-servere, og forhindret uautorisert tilgang; dokumentert med Ansible-automatisering.

Tjenesten inngår i arbeid som står andre steder på siden:

Klargjorde en server nummer to fra kode · Offentlige tjenester på en liten server · Infrastruktur beskrevet i et kodelager · Klargjorde Azure-infrastruktur som kode med Bicep

{ Sky, infrastruktur & DevOps }

DevOps og CI/CD-automatisering

Endringer ut i produksjon uten drama

Testet, utgitt og mulig å reversere, slik at endringer når kundene en tirsdag ettermiddag i stedet for sent en fredagskveld.

Holdt 4 repoer til én historikkstandard

Holdt fire repoer til én historikkstandard — konvensjonell, uten emojier og uten attribusjonslinjer, håndhevet av en hook for commit-meldinger — sammen med 48 instruksjonsdokumenter som styrer hvordan arbeidet gjøres.

Kjenn lisensen til alt dere leverer

Reviderte 644 Rust-crates for lisenskompatibilitet ved hvert bygg, og beviste at kontrollen slår ut ved å skrive om lisensen til én crate og ved å flytte den låste kjernerevisjonen uten å regenerere.

Nådde den ubrukte halvdelen av kjernen

Tok i bruk den halvdelen av meldingskjernen applikasjonen aldri hadde brukt — overføring av sikkerhetskopier, forsvinnende meldinger, redigering og ny sending av meldinger, verifiserte invitasjoner, proxyer og krypteringspolicy — og kjørte hver test mot det ekte biblioteket uten mocks.

Fikk testsuiten ned i 8,9 sekunder

Tok testsuiten fra seks tester over en grense på seksti sekunder til 135 beståtte på 8,9 sekunder ved å profilere hovedtråden og fjerne de to kallene den satt fast i under 3 989 av 4 017 prøver.

Skrev en parser som verifiserer hvert FFI-kall

Skrev en parser som leser den faktiske C-headeren på 7 308 linjer og verifiserer hvert kallsted, hver enum-konstant og at hver klasse som eier en peker, er final, etter at en håndskrevet plassholder-header lot kall til tre fjernede funksjoner kompilere, lenke og krasje.

Fant at hookene og porten var uenige

Fant ut at commit-hookene og kvalitetsporten kjørte ulike kontroller mens et dokument lovet at de var de samme, ved å sammenligne de to listene i en test — de fem som bare noen gang ble kjørt manuelt, var de som leste CV-teksten.

Skrev tester for selve kontrollverktøyene

Skrev tester for selve kontrollene etter å ha slått fast at en kontroll som bare får ren inndata, en dag vil melde rent fordi den ikke leste noe — plantet en stavefeil for å bekrefte at stavekontrollen finner den, og hentet et id-område fra databasen i stedet for fra et tall i testen.

Aktiverte alle linterregler som feil

Slo på alle reglene i Python-linteren på feilnivå og arbeidet seg gjennom 1 815 funn ned til null, med en skriftlig begrunnelse for hvert av de få unntakene og to av dem sikret av en kontroll i stedet for en kommentar.

Holdt generatoren til 981 tester og 92 % dekning

Holdt generatoren til 981 testtilfeller med et minstekrav på 92 % grendekning og advarsler behandlet som feil, og verifiserte idempotens ved å kjøre hele byggepipelinen to ganger fra en tom fil og kreve at den andre kjøringen ikke endret noe.

Lintet selve kvalitetsporten på 10 242 linjer

Satte 10 242 linjer JavaScript i kvalitetsporten under en formaterer og en linter etter å ha slått fast at det var den største kodemengden i repoet og den eneste ingenting leste, og rettet 13 funn uten å undertrykke noen.

Rettet et sitemap med ett felles tidsstempel

Rettet et sitemap der 172 av 176 URL-er delte ett endringstidspunkt, ved å hente datoen fra git-historikken etter å ha slått fast at eksporten skriver om hver fil ved hver kjøring.

Speilet nettstedet til Gemini og Gopher

Bygde Gemini- og Gopher-utdata fra samme nettstedskilde; resultatbeskrivelsen oppgir 777 dokumenter i hvert format på det stadiet og null endrede HTML-byte ved tillegget av formatene.

Et sikkerhetsnett som ikke lenger koster en halvtime

Kuttet nettstedets nettleserdrevne kvalitetsport fra 1 636 sekunder til 615 ved å kjøre kontrollene med de lengste først gjennom en worker-pool begrenset til fire baner, etter å ha målt at alfabetisk rekkefølge kostet 320 sekunder mot 224.

Bygde en commit-port med 22 lintere

Bygde en commit-port med 22 enlinjes lintere pluss fem som fortjener et avsnitt, uten advarselsnivå og uten tillatte undertrykkinger i koden, som dekker HTML, CSS, JavaScript, Python, YAML, Markdown, shell, lenker, staving, hemmeligheter og typografi.

Drev et løpende program for sikkerhetsskanning

Etablerte et kontinuerlig sikkerhetsprogram — kodeskanning, DAST, kontroll av avhengigheter og sårbarheter, SBOM-generering, skanning etter hemmeligheter og SHA-låste actions — sammen med 21 skriftlige sikkerhetsrevisjoner.

1 022 migreringer bak en port for skjemaavvik

Holdt skjemaet ærlig gjennom 1 022 migreringer med en CI-port som bygger databasen begge veier — en ny installasjon, og en installasjon pluss alle migreringer — og feiler når de to ikke stemmer overens.

Bygde egen feilovervåking og sporing

Bygde egen feilovervåking og OpenTelemetry-sporing i stedet for å kjøpe dem — rensing av nyttelast, oppdagelse av topper og regresjoner, symbolisering og et syntetisk hjerteslag — bak 11 operatørvisninger.

Bygde repoets kontrollmotor med 268 kontroller

Bygde repoets kontrollmotor — 268 registrerte commit-kontroller, 277 lint-regler og 15 egne ESLint-regler — pluss 146 tester av selve kontrollene, slik at det er bygget, ikke gjennomgangen, som holder standarden.

Bygde en testsuite for Go, SQL og nettleser

Bygde en lagdelt automatisert testsuite — 981 Go-tester, 543 spesifikasjoner for frontend og nettleser, 494 atferdstester i SQL — med mutasjonstesting, egenskapsbaserte tester og en kvalitetsport for universell utforming.

Automatiserte samarbeid i teamet

Automatiserte samarbeid i teamet, passordhåndtering, oppgave- og tidsstyring, og bygde et halvautomatisk system for å vise frem prosjekter, noe som økte teamets produktivitet.

Satte et krav om null advarsler på seks språk

Satte en kvalitetsstandard med null advarsler for seks språk — Go, TypeScript, SQL, Python, Shell og Markdown — håndhevet av pre-commit-hooks.

Skrev 578 automatiseringsmål i go-task

Skrev 578 go-task-automatiseringsmål som dekker native-, Docker- og HTTPS-utviklingsmoduser, linting, testing, database og produksjonssetting.

Bygde CI/CD-pipelines i GitHub Actions

Bygde CI/CD-pipelines i GitHub Actions med et distroless-produksjonsimage for frontend og promotering gjennom flere miljøer.

Genererte API-kontrakten med databasen som kilde

Genererte API-kontrakten fra databasen og utover — OpenAPI, en typet TypeScript-klient på 44 076 linjer, 61 mock-handlere og grensene brukergrensesnittet håndhever — med en kontroll i hvert ledd som feiler ved avvik.

Automatiserte databehandling i Python og PL/pgSQL

Automatiserte databehandlingsoppgaver med Shell-skripting, PL/pgSQL, Python og Transact-SQL, og økte produktiviteten og effektiviteten.

Sparte 4 000 timer med GitHub og GitLab CI/CD

Effektiviserte prosessene for dataanalyse og programvareutvikling og sparte 4 000 timer ved å innføre CI/CD-praksis med GitHub, GitLab, Bash og Python.

Effektiviserte CI/CD og sparte 4 000 timer

Effektiviserte CI/CD-prosessene og sparte 4 000 timer ved å innføre automatisering i pipelines for programvareutvikling.

Automatiserte SSL/TLS for 100 Docker-apper

Automatiserte opprettelsen av SSL/TLS-sertifikater for 100 Docker-applikasjoner og sikret trygge forbindelser på tvers av Ubuntu Linux-verter.

Fornyet interne prosesser og sparte 8 000 timer

Fornyet interne prosesser og sparte 8 000 timer ved å forbedre programvarearkitekturen, systemene og effektiviteten i planleggingen.

Reduserte driftsrisiko med Grafana og Prometheus

Reduserte driftsrisikoen ved å innføre et overvåkingsdashbord med Grafana og Prometheus, og forbedret systemets pålitelighet.

Automatiserte 100 kritiske sikkerhetskopier

Automatiserte 100 kritiske sikkerhetskopier av data med Barman, Google Cloud, Bash og Python, og sikret dataintegriteten på tvers av databasene.

Automatiserte leveransen av 20 GIS-datapipelines

Automatiserte leveransen av 20 GIS-datapipelines og ETL-prosesser for applikasjonsdata, og effektiviserte infrastrukturautomatisering og rapportering.

Automatiserte GIS-SaaS-utrulling i GitHub Actions

Automatiserte produksjonssettingen, databehandlingen og rapporteringssystemet for en GIS SaaS-applikasjon med GitHub Actions CI/CD, Python, Bash og SQL.

Tjenesten inngår i arbeid som står andre steder på siden:

Prøvekjørte CI-hooken i forgen og fant 2 feil · Kjørte containere på Podman og Quadlet · Koden deres på deres egen server · La til ett sekunds preflight for hemmeligheter · Fikk Ansibles check-modus til å si sannheten · Infrastruktur beskrevet i et kodelager · Bygde hele kjerneinfrastrukturen fra bunnen av · Hadde ansvar for Azure-utrulling av plattformen · Forenklet infrastruktur utenfor AWS · Driftet 20 Docker-apper på Google Cloud og AWS

{ Sky, infrastruktur & DevOps }

Containerisering og orkestrering

Programvare som kjører likt overalt

Pakket slik at den oppfører seg helt likt på en bærbar PC, på en testserver og i produksjon, og skalerer opp når en travel uke kommer.

Tjenesten inngår i arbeid som står andre steder på siden:

Kjørte containere på Podman og Quadlet · Strammet inn systemd-sandkassen for hver enhet · Bygde CI/CD-pipelines i GitHub Actions · Automatiserte SSL/TLS for 100 Docker-apper · Driftet 20 Docker-apper på Google Cloud og AWS

{ Sky, infrastruktur & DevOps }

Pålitelighet og overvåking

Vite det før kundene dine gjør det

Overvåking og varsler som sier fra mens en feil fortsatt er liten, og tallene som viser hvor ofte den holder seg liten.

Fikk seks innsamlere til å si hva de hoppet over

Fikk seks rapporteringsinnsamlere til å oppgi den delen av inndataene de aldri leste — filer som forble lukket, sonder som ikke var installert, journallinjer som ikke ble gjenkjent — og tilpasset terskelverdier til 116 målinger, etter å ha slått fast at en blind innsamling og en vert der alt er i orden, gir den samme siden.

Målte 13 adresser uten sporing

Målte tretten adresser ut fra webserverens egen tilgangslogg alene — ingen informasjonskapsel, ikke noe sporingsverktøy og ingen tredjepartstjeneste — maskerte klientnettverket idet hver linje skrives, og samlet resultatet i et permanent arkiv med 21 daglige tellinger og 20 månedlige fasetter.

En avgrenset hendelsesflyt i en macOS-app

Avgrenset hendelsesbuffere og logging etter en registrert macOS-hendelse med vekst på 41 MB i sekundet og 111 GB komprimerte sider på en maskin med 36 GB; en registrert full testkjøring reduserte kjernens logglinjer fra 610 996 til 1 411.

Bygde 7 skrivebeskyttede rapportroller for verter

Bygde sju skrivebeskyttede rapporteringsroller som gjengir en vert i drift som Markdown — fakta, tilgang, git, metrikker, trafikk, sikkerhet og leverandørens inventar — etter en regel om at ingen tall skrives ut som kjøringen ikke har målt.

Beviste at utestengingen virker ved hver kjøring

Beviste at utestengingen virker fra ende til annen ved hver herdingskjøring ved å utestenge en reservert testadresse, lese den resulterende kjerneregelen og oppheve utestengingen i en garantert oppryddingsblokk, slik at en jail som slutter å virke, får kjøringen til å feile i stedet for å melde at alt er i orden.

Tre låser på døren, ingen av dem låst

Fant og rettet tre SSH-vern mot brute force som aldri hadde virket: en jail for utestenging som overvåket port 22 mens daemonen lyttet på 1986, en hastighetsgrense skygget av en bredere regel over den, og en utestengingshandling der binærfilen aldri ble funnet, slik at ingen utestenging noen gang hadde trådt i kraft.

Bygde fail-closed hastighetsgrenser mot misbruk

Bygde misbruksvern som stenger ved feil — 22 Redis-baserte hastighetsbegrensere, Cloudflare Turnstile, idempotente forespørsler og en origin-lås — slik at plattformen avviser boter og flom i stedet for å stole på dem som kaller den.

Flyttet tregt arbeid over i en River-jobbkø

Flyttet tregt arbeid ut av forespørselsbanen og over i en River-jobbkø — 15 worker-moduler, 8 planlagte oppgaver og 20 pg_cron-jobber — slik at en forespørsel returnerer mens arbeidet bak den fortsetter.

Bygde infrastruktur som fortsatt kjører år senere

Planla og implementerte ny infrastrukturfunksjonalitet for interne og eksterne systemer, og bygde løsninger så solide at de fortsatt kjører år senere med minimale endringer.

Holdt IPTV-strømmesignaler i gang uten stopp

Sikret uavbrutt levering av IPTV-strømmesignaler mellom leverandører og kunder, og overvåket og vedlikeholdt strømmenettverket og IP-telefonien døgnet rundt.

Ga 24/7-støtte for ca. 1 000 IPTV/OTT-servere

Ga infrastrukturstøtte døgnet rundt, 24/7, for en IPTV/OTT-strømmeplattform og administrerte ca. 1 000 servere i tillegg til kundeeide systemer for globale kunder i Kina, USA og Tyskland.

Bygde e-post med failover og leveringslogging

Bygde e-post som en plattformfunksjon — tre leverandører med failover, leveringswebhooks, logging av sending og levering, maler og kampanjer — bak en oppstartskontroll som ikke lar plattformen starte uten minst én leverandør.

Et kartprodukt, håndtert fra ende til annen

Utformet, utviklet og støttet infrastruktur, databehandling og en kartapplikasjon over 2 år, med både programvareleveranser og driftsvedlikehold.

Administrerte 40 nettsteder på Apache og Nginx

Administrerte 40 nettsteder på Ubuntu Linux-servere for webhotell med Apache og Nginx, og sikret høy tilgjengelighet og ytelse.

Drev 10 PostgreSQL- og MS SQL-servere på Ubuntu

Designet, satte i produksjon og vedlikeholdt 10 PostgreSQL- og MS SQL-servere på Ubuntu Linux-VPS-er, og sikret optimal serverytelse og pålitelighet.

Bygde selskapets første observerbarhetsdashbord

Utviklet og lanserte selskapets første dashbord for observerbarhet, med sanntidsinnsikt i systemytelsen og datavisualisering på den store TV-skjermen på kontoret.

Tjenesten inngår i arbeid som står andre steder på siden:

Delte opp hver driftshemmelighet per vert · Fikk Ansibles check-modus til å si sannheten · Bygde vertsovervåking med dødmannsknapp · Backupautomatisering med en tydelig gjenopprettingsmetode · Bygde egen feilovervåking og sporing · Hadde ansvar for Azure-utrulling av plattformen · Reduserte driftsrisiko med Grafana og Prometheus

{ Sky, infrastruktur & DevOps }

Sikkerhetskopiering og gjenoppretting

Alt tilbake etter en dårlig dag

Sikkerhetskopier som er testet i stedet for tatt for gitt, og en skriftlig plan for morgenen da en server ikke kommer opp igjen.

Administrerte 30 Ubuntu Linux-VPS-instanser

Administrerte 30 Ubuntu Linux-VPS-instanser, innførte strategier for katastrofegjenoppretting og sikret optimale nettverkskonfigurasjoner.

Tjenesten inngår i arbeid som står andre steder på siden:

Delte opp hver driftshemmelighet per vert ·

Backupautomatisering med en tydelig gjenopprettingsmetode ·

La sikkerhetskopi og katastrofeberedskap i kode · Automatiserte 100 kritiske sikkerhetskopier

{ Sky, infrastruktur & DevOps }

Systemadministrasjon

Holde hjulene i gang

Daglig ettersyn av servere, e-post, kontoer og oppdateringer, slik at det alle er avhengige av, forblir kjedelig.

Verifiserte brannmurregler etter plassering

Verifiserte brannmurregler etter plassering i stedet for tilstedeværelse ved å lese den nummererte regellisten og den aktive pakkefilterkjeden, fordi en regel som finnes, ikke er det samme som en regel en pakke faktisk når.

Integrerte en passordbehandler i hele selskapet

Integrerte et passordhåndteringssystem for hele selskapet, styrket sikkerheten og effektiviserte tilgangskontrollen.

Administrerte 100 Bare Bone-servere

Administrerte 100 Bare Bone-servere, fysiske nettverk og IP-telefonisystemer, og sikret en robust infrastruktur for selskapets vekst.

Konfigurerte og installerte 1 000 Wi-Fi-rutere

Konfigurerte og installerte 1 000 Wi-Fi-rutere og forbedret nettverkstilgangen og ytelsen for kundene.

Administrerte nettverksinfrastrukturen

Administrerte nettverksinfrastrukturen for over 1 000 servere og sikret optimal produksjonssetting, sikkerhet og feilsøking av systemene.

Administrerte og feilsøkte 8 VPN-forbindelser

Administrerte og feilsøkte 8 VPN-forbindelser med WireGuard og IPsec, og sikret trygg kommunikasjon på tvers av Google Cloud og Linux-systemer.

Ledet tilgangsstyring på tvers av M365 og SSO

Forhindret sikkerhetsbrudd ved å lede tiltak for tilgangsstyring med M365, 1Password, Red Hat SSO og OKTA SSO.

Tjenesten inngår i arbeid som står andre steder på siden:

Publiserte et lesbart onion-speil på Tor · Fikk seks innsamlere til å si hva de hoppet over · Prøvekjørte CI-hooken i forgen og fant 2 feil · Koden deres på deres egen server · Bygde 7 skrivebeskyttede rapportroller for verter · Strammet inn systemd-sandkassen for hver enhet · Avstemte en DNS-sone med 20 poster som kode · Bygde vertsovervåking med dødmannsknapp · Backupautomatisering med en tydelig gjenopprettingsmetode · Beviste at utestengingen virker ved hver kjøring · Herdet SSH til 24 validerte direktiver · Offentlige tjenester på en liten server · Infrastruktur beskrevet i et kodelager · Bygde hele kjerneinfrastrukturen fra bunnen av · Bygde infrastruktur som fortsatt kjører år senere · Holdt IPTV-strømmesignaler i gang uten stopp · Ga 24/7-støtte for ca. 1 000 IPTV/OTT-servere · Administrerte 40 nettsteder på Apache og Nginx · Administrerte 30 Ubuntu Linux-VPS-instanser · Skrev 1 000 RBAC-regler for å sikre Linux-servere · Drev 10 PostgreSQL- og MS SQL-servere på Ubuntu

{ Sky, infrastruktur & DevOps }

Nettverk og VPN-oppsett

Koble sammen kontorene og menneskene dine

Wi-Fi, brannmurer og fjerntilgang satt opp slik at de ansatte kan jobbe hvor som helst, og bare de rette personene slipper inn.

Tjenesten inngår i arbeid som står andre steder på siden:

Klargjorde en server nummer to fra kode · Avstemte en DNS-sone med 20 poster som kode · Verifiserte brannmurregler etter plassering · Tre låser på døren, ingen av dem låst · Holdt IPTV-strømmesignaler i gang uten stopp · Klargjorde Azure-infrastruktur som kode med Bicep ·

Administrerte 100 Bare Bone-servere · Konfigurerte og installerte 1 000 Wi-Fi-rutere · Administrerte nettverksinfrastrukturen · Administrerte og feilsøkte 8 VPN-forbindelser · Administrerte 30 Ubuntu Linux-VPS-instanser

{ Sky, infrastruktur & DevOps }

Sikkerhet og tilgangsstyring

Bestemme hvem som får tilgang til hva

Kontoer, passord og tilganger ordnet slik at folk har nøyaktig den tilgangen jobben krever, og de som slutter, mister den samme dag.

Skrev en avgrensningsregel etter tilgangslekkasje

Skrev en avgrensningsregel inn i repoet etter at en omstrukturering hadde ført et annet selskaps inventar, brannmuråpninger og tekst inn i det — og holdt de karantenesatte restene under skanneren for hemmeligheter i stedet for å unnta dem.

Bygde eierskapskrav og avgjørelse for selskaper

Bygde løsningen for eierskapskrav til selskaper fra ende til annen — en bruker gjør krav på et selskap, en administrator avgjør saken, og en godkjenning skriver om autorisasjonsgrafen som bestemmer hvem som får redigere hva.

Bygde tilgangsrettigheter for Stripe og appkjøp

Bygde laget for betaling og tilgangsrettigheter — Stripe sammen med kjøp i appen hos Apple og Google — som styrer tilgangen til katalogen, søket og eksporten gjennom en tilgangsmodell på 11 tabeller som kontrolleres på serveren.

Herdet applikasjonen med nonce-basert CSP

Herdet applikasjonen med nonce-basert CSP, HSTS, SameSite-informasjonskapsler, databaseroller etter prinsippet om minste privilegium og ny kontroll av tilgangsrettigheter på serversiden.

Bygde bytte av organisasjonskontekst for brukerne

Designet et system for å bytte organisasjonskontekst med localStorage på klienten og speiling i informasjonskapsler på serversiden, slik at brukere kan opptre som organisasjoner de administrerer, med autorisasjon etter prinsippet om minste privilegium.

Forbedret kommunikasjonen og samarbeidet i teamet

Forbedret kommunikasjonen og samarbeidet i teamet ved å innføre Slack, Mattermost, 1Password og Jira, og sparte 8 000 arbeidstimer.

Tjenesten inngår i arbeid som står andre steder på siden:

Publiserte et lesbart onion-speil på Tor · Målte 13 adresser uten sporing · Kjenn lisensen til alt dere leverer · Nådde den ubrukte halvdelen av kjernen · Delte opp hver driftshemmelighet per vert · Klargjorde en server nummer to fra kode ·

Kjørte containere på Podman og Quadlet · Strammet inn systemd-sandkassen for hver enhet · Avstemte en DNS-sone med 20 poster som kode · La til ett sekunds preflight for hemmeligheter · Verifiserte brannmurregler etter plassering · Beviste at utestengingen virker ved hver kjøring · Herdet SSH til 24 validerte direktiver · Tre låser på døren, ingen av dem låst · Drev et løpende program for sikkerhetsskanning · Bygde fail-closed hastighetsgrenser mot misbruk · Integrerte en passordbehandler i hele selskapet · Automatiserte samarbeid i teamet · Klargjorde Azure-infrastruktur som kode med Bicep · Bygde e-post med failover og leveringslogging · Automatiserte SSL/TLS for 100 Docker-apper · Administrerte nettverksinfrastrukturen · Administrerte og feilsøkte 8 VPN-forbindelser · Ledet tilgangsstyring på tvers av M365 og SSO · Skrev 1 000 RBAC-regler for å sikre Linux-servere

{ Sky, infrastruktur & DevOps }

Sikkerhet for AI-agenter

Holde AI-en unna det den ikke skal røre

Hver agents tilgang kartlagt og begrenset til det nødvendige,
hver handling logget der agenten ikke kan endre loggen,
og hvert sikkerhetstiltak testet i stedet for tatt på tro.

Tjenesten inngår i arbeid som står andre steder på siden:

Delte opp hver driftshemmelighet per vert · Skrev en avgrensningsregel etter tilgangslekkasje · La til ett sekunds preflight for hemmeligheter · Beviste at utestengingen virker ved hver kjøring · Tre låser på døren, ingen av dem låst · Drev et løpende program for sikkerhetsskanning · Bygde fail-closed hastighetsgrenser mot misbruk · Ledet tilgangsstyring på tvers av M365 og SSO · Skrev 1 000 RBAC-regler for å sikre Linux-servere

Sammenlign utgangspunktet deres

Velg et sammenlignbart arbeid og identifiser forskjellene i deres egen situasjon.

Se hva det koster Ta kontakt →

Drøft en relevant tilnærming

Be oss gjøre relevant erfaring til en tilnærming dere kan vurdere.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://platform.engineer.company/nb/portfolio/