Engineer Platform

Automatisering & CI/CD

61 resultater

Forbind delene af jeres system

Arbejdet under »Automatisering & CI/CD« viser en del af et fungerende system. Vi kan hjælpe med at forbinde den del med resten af jeres drift. Overvågning og alarmer, der siger til, mens fejlen stadig er lille, og tallene, der viser, hvor tit den bliver ved med at være det.

Se vores ydelser Kontakt os →

Find en sammenlignelig udfordring

Åbn et eksempel, der ligner jeres situation. Gennemgangen beskriver beslutningerne bag resultatet, ud over teknologiens navn. Har bygget det første observerbarhedsdashboard — virksomhedens første overblik over sine egne systemer

Har bygget det første observerbarhedsdashboard Kontakt os →

Har bygget det første observerbarhedsdashboard

Har udviklet og lanceret virksomhedens første dashboard til observerbarhed, der viste systemets ydeevne i realtid og visualiserede data på den store tv-skærm på kontoret.

Har drevet 10 PostgreSQL- og MS SQL-servere

Har designet, idriftsat og vedligeholdt 10 PostgreSQL- og MS SQL-servere på Ubuntu Linux VPS og sikret optimal ydeevne og driftssikkerhed på serverne.

Har skrevet 1.000 RBAC-regler til Linux-servere

Har forbedret datasikkerheden ved at implementere 1.000 RBAC-regler for udviklere, applikationsinstanser, PostgreSQL, MS SQL og andre Linux-servere for at forhindre uautoriseret adgang; dokumenteret med Ansible-automatisering.

Har automatiseret GIS-SaaS-udrulning med CI/CD

Har automatiseret udrulningen af en GIS SaaS-applikation, databehandlingen og rapporteringssystemet med GitHub Actions CI/CD, Python, Bash og SQL.

Har automatiseret leveringen af 20 GIS-pipelines

Har automatiseret levering af 20 GIS-datapipelines og ETL-processer til appdata og strømlinet infrastrukturautomatisering og rapportering.

Har automatiseret 100 kritiske databackups

Har automatiseret 100 kritiske databackups ved hjælp af Barman, Google Cloud, Bash og Python og sikret dataintegriteten i alle databaser.

Har drevet 20 Docker-apps på Google Cloud og AWS

Har udrullet og vedligeholdt 20 containeriserede Docker-applikationer, fejlsøgt med Podman og Kubernetes og administreret R-baserede apps på Google Cloud og AWS.

Har ledet adgangsstyring med M365 og SSO

Har forhindret sikkerhedsbrud ved at lede initiativer inden for adgangsstyring med M365, 1Password, Red Hat SSO og OKTA SSO.

Har reduceret driftsrisici med Grafana

Har reduceret driftsrisici ved at implementere et overvågningsdashboard med Grafana og Prometheus og forbedret systemets driftssikkerhed.

Har forbedret teamets kommunikation og samarbejde

Har forbedret teamets kommunikation og samarbejde ved at implementere Slack, Mattermost, 1Password og Jira og sparet 8.000 arbejdstimer.

Har fornyet processer og sparet 8.000 timer

Har gennemgribende fornyet interne processer og sparet 8.000 timer ved at forbedre softwarearkitektur, systemer og planlægningseffektivitet.

Har administreret netværksinfrastruktur

Har administreret netværksinfrastruktur for over 1.000 servere og sikret optimal systemudrulning, sikkerhed og fejlfinding.

Har automatiseret SSL/TLS til 100 Docker-apps

Har automatiseret oprettelsen af SSL/TLS-certifikater til 100 Docker-applikationer og dermed sikret forbindelserne på alle Ubuntu Linux-værter.

Har strømlinet CI/CD og sparet 4.000 timer

Har strømlinet CI/CD-processer og sparet 4.000 timer ved at indføre automatisering i softwareudviklingspipelines.

Har sparet 4.000 timer med CI/CD i GitHub

Har strømlinet processer for dataanalyse og softwareudvikling og sparet 4.000 timer ved at indføre CI/CD-praksis med GitHub, GitLab, Bash og Python.

Har automatiseret databehandling i Python og SQL

Har automatiseret databehandlingsopgaver med shell-scripting, PL/pgSQL, Python og Transact-SQL og øget produktiviteten og effektiviteten.

Har genereret API-kontrakten ud fra databasen

Har genereret API-kontrakten udad fra databasen — OpenAPI, en typet TypeScript-klient på 44.076 linjer, 61 mock handlers og de grænser, UI'et håndhæver — med en kontrol i hvert led, der fejler ved afvigelser.

Har provisioneret Azure-infrastruktur som kode

Har provisioneret Azure-infrastruktur som kode med Bicep — Container Apps, PostgreSQL Flexible Server, Front Door/WAF og netværk — i udviklings-, staging- og produktionsmiljøet.

Har bygget GitHub Actions CI/CD-pipelines

Har bygget GitHub Actions CI/CD-pipelines med et distroless frontend-image til produktion og promovering gennem flere miljøer.

Har skrevet 578 go-task-automatiseringsmål

Har skrevet 578 go-task-automatiseringsmål, der dækker udviklingstilstandene native, Docker og HTTPS, linting, test, database og udrulning.

Har haft ansvaret for alle udrulninger til Azure

Har haft ansvaret for udrulningen af platformen til Azure fra ende til anden og styret releases i udviklings-, staging- og produktionsmiljøet.

Har indført nul advarsler som krav i seks sprog

Har sat en kvalitetsstandard med nul advarsler i seks sprog — Go, TypeScript, SQL, Python, Shell og Markdown — håndhævet af pre-commit-hooks.

Har automatiseret teamsamarbejde

Har automatiseret teamsamarbejde, adgangskodehåndtering, opgave- og tidsstyring og bygget et halvautomatisk system til fremvisning af projekter og dermed øget teamets produktivitet.

Har integreret en fælles adgangskodeløsning

Har integreret et virksomhedsdækkende adgangskodehåndteringssystem, der styrkede sikkerheden og strømlinede adgangskontrollen.

Har bygget en testsuite til Go, SQL og browseren

Har bygget en lagdelt automatiseret testsuite — 981 Go-tests, 543 frontend- og browserspecs, 494 SQL-adfærdstests — med mutationstest, property-based tests og en tilgængelighedsgate.

Har bygget en kontrolmotor med 268 tjek

Har bygget repositoryets kontrolmotor — 268 registrerede commit-tjek, 277 lint-regler og 15 egne ESLint-regler — plus 146 tests af selve tjekkene, så det er builden, der holder standarden, ikke reviewet.

Har flyttet langsomme opgaver til en River-jobkø

Har flyttet langsomt arbejde væk fra request-stien og over på en River-jobkø — 15 worker-moduler, 8 planlagte opgaver og 20 pg_cron-jobs — så en request returnerer, mens arbejdet bag den kører videre.

Har kørt et løbende sikkerhedsscanningsprogram

Har etableret et løbende sikkerhedsprogram — code scanning, DAST, afhængigheds- og sårbarhedstjek, SBOM-generering, secret scanning og SHA-pinnede actions — sammen med 21 skriftlige sikkerhedsgennemgange.

Infrastruktur beskrevet i et repository

Har bygget virksomhedens egen infrastruktur som 19 Ansible-playbooks og 34 roller fordelt på 12.065 linjer YAML, der konvergerer den kørende vært til en deklareret tilstand, og hvor hvert play er idempotent.

Har hærdet SSH til 24 validerede direktiver

Har hærdet SSH til 24 direktiver, der alle kontrolleres med assertions, med validering i tre trin — kandidatfilen, den samlede konfiguration og til sidst dæmonens egen tilbagelæsning — efter at tilbagelæsningen afslørede, at den kørende server ubemærket tilsidesatte to af de fireogtyve.

Har bevist, at blokeringen virker ved hver kørsel

Har eftervist blokeringsvejen fra ende til anden ved hver hærdningskørsel ved at blokere en reserveret testadresse, læse den resulterende kerneregel og ophæve blokeringen i en oprydningsblok, der altid kører, så en jail, der holder op med at virke, får kørslen til at fejle i stedet for at melde alt i orden.

Har verificeret firewallregler efter position

Har verificeret firewallregler efter position snarere end tilstedeværelse ved at læse den nummererede regelliste og den kørende pakkefilterkæde, fordi det, at en regel findes, ikke betyder, at nogen pakke når frem til den.

Backupautomatik med en tydelig gendannelsesmetode

Har implementeret restic-backupautomatik med opbevaringspolitik, integritetskontrol og et forløb til månedlige gendannelsestest og dokumenteret aktivering af lager og kontrol af en virkelig gendannelse som særskilte driftsopgaver.

Har bygget overvågning med dead man's switch

Har bygget overvågning med dead man's switch, der kun pinger, mens hukommelse og disk er i orden, så en svækket vært udløser en alarm ved at gå tavs — og opdaget seks variabelnavne, der sagde "free", hvor tjekket korrekt målte "available", en størrelsesorden til forskel på en maskine med 464 MB.

Har fået check-tilstanden til at vise sandheden

Har fået check-tilstanden til at vise sandheden på hele platformen efter at have fundet seks sonder, der traf beslutning ud fra en værdi, værten aldrig havde leveret, fordi Ansibles command-modul melder succes under --check, mens det springer kommandoen helt over.

Har tilføjet et preflight-tjek af hemmeligheder

Har tilføjet et preflight-play, der kører den samme kode som konvergensen mod operatørens lokale hemmeligheder på omkring et sekund, efter at en produktionskørsel var gået ned halvvejs, ved sin niende opgave, med swap-indstillingerne allerede skrevet til den kørende vært.

Har afstemt en DNS-zone med 20 poster deklarativt

Har afstemt en DNS-zone med 20 poster deklarativt mod Cloudflares API med separate indgange til gennemgang og BIND-eksport og slået CDN-proxyen fra igen af hensyn til privatlivet, efter at den var bygget.

Har strammet systemd-sandboxen på hver unit

Har reduceret eksponeringen i systemd-sandkassen for hver unit, platformen installerer — en dead man's switch-tjeneste fra 9,6 UNSAFE til 1,5, en git-forge med adgang fra internettet fra 8,3 EXPOSED til 1,5 — og tilføjet et parsertjek ved konvergens efter at have fundet et fejlstavet direktiv, der ubemærket blev ignoreret i tre unit-skabeloner.

Har bygget 7 skrivebeskyttede rapporteringsroller

Har bygget syv skrivebeskyttede rapporteringsroller, der gengiver den kørende vært som Markdown — fakta, adgang, git, metrikker, trafik, sikkerhed og udbyderinventar — efter en regel om, at der ikke udskrives et tal, kørslen ikke selv har målt.

Jeres kode på jeres egen server

Har udrullet virksomhedens egen git-forge på Soft Serve, privat som udgangspunkt og uden webpanel, med SSH-porten bundet til loopback bag en jump-vært, og gjort landingssiden foran den til et build-artefakt fra hovedsitet i stedet for en manuelt vedligeholdt kopi.

Har kørt containere på Podman og Quadlet

Har udrullet et containerlag på Podman og Quadlet under systemd i stedet for Docker, fordi Dockers standardpublicering af porte på bridge-netværk kan omgå UFW-regler — og givet udrulninger en uprivilegeret bruger med én fast kommando i stedet for root.

Har provisioneret en ekstra server ud fra kode

Har automatiseret provisioneringen af en ekstra server hos en anden cloududbyder og oprettet firewallen før maskinen, så den står bag en firewall fra første øjeblik, med begge udbyderes firewalls skrevet direkte mod deres REST-API'er for at undgå en collection fra tredjepart.

Har skrevet en afgrænsningsregel efter et læk

Har skrevet en afgrænsningsregel ind i repositoryet, efter at en omstrukturering havde ført en anden virksomheds inventar, firewalltilladelser og tekst med ind i det — og holdt de karantænesatte rester under hemmelighedsscanneren i stedet for at udelukke dem.

Har adskilt alle driftshemmeligheder pr. vært

Har gjort de fire driftshemmeligheder individuelle pr. vært efter at have fastslået, at to værter, der deler én dead man's switch, giver færre alarmer end to switches, ikke flere, og at en fælles adgangssætning til backup kan blotlægge begge værters kopier, selv i separate lagre, hvis nogen får fat i den og kan læse begge lagres krypterede data og nøglefiler.

Har bygget en commit-gate med 22 lintere

Har bygget en commit-gate med 22 lintere, der hver kan beskrives på én linje, plus fem, der fortjener et afsnit, uden advarselsniveau og uden tilladte inline-undertrykkelser, og som dækker HTML, CSS, JavaScript, Python, YAML, Markdown, shell, links, stavning, hemmeligheder og typografi.

Et sikkerhedsnet, der ikke længere koster en halv time

Har skåret sitets browserdrevne kvalitetsgate fra 1.636 sekunder til 615 ved at afvikle dens tjek med de længste først gennem en worker-pulje begrænset til fire spor, efter at have målt, at alfabetisk rækkefølge kostede 320 sekunder mod 224.

Har spejlet sitet til Gemini og Gopher

Har bygget Gemini- og Gopher-output fra samme webkilde; resultatbeskrivelsen angiver 777 dokumenter i hvert format på det pågældende trin og nul ændrede HTML-bytes ved tilføjelsen af formaterne.

Har rettet et sitemap med ét fælles tidsstempel

Har rettet et sitemap, hvor 172 af 176 URL'er delte ét ændringstidsstempel, ved at tage datoen fra git-historikken efter at have fastslået, at eksporten omskriver hver fil ved hver kørsel.

Har lintet selve kvalitetsgaten på 10.242 linjer

Har bragt 10.242 linjer kvalitetsgate-JavaScript under en formatter og en linter efter at have fastslået, at det var den største kodemængde i repositoryet og den eneste, som intet værktøj læste, og rettet 13 fund uden at undertrykke nogen.

Har givet generatoren 981 tests og 92 % dækning

Har holdt generatoren på 981 testtilfælde og en nedre grænse på 92 % grendækning med advarsler behandlet som fejl og eftervist idempotens ved at køre hele build-pipelinen to gange fra en tom fil og kræve, at det andet gennemløb intet ændrer.

Har slået alle linterregler til som fejl

Har slået hver eneste regel i Python-linteren til som fejl og arbejdet 1.815 fund ned til nul, hvor hver af de få undtagelser har en skriftlig begrundelse, og to af dem er sikret af et tjek i stedet for en kommentar.

Har skrevet tests af selve tjekkene

Har skrevet tests af selve tjekkene efter at have fastslået, at et tjek, der kun fodres med rent input, en dag melder fejlfrit, fordi det intet har læst — ved at plante en stavefejl for at bekræfte, at stavekontrollen finder den, og ved at tage et id-interval fra databasen i stedet for fra et tal i testen.

Har fundet uenighed mellem hooks og kvalitetsgate

Har opdaget, at commit-hooks og kvalitetsgaten kørte forskellige tjek, selvom et dokument lovede, at de var de samme, ved at sammenligne de to lister i en test — de fem, der kun nogensinde blev kørt manuelt, var dem, der læste CV-prosaen.

Har prøvekørt forgens CI-hook og fundet 2 fejl

Har prøvekørt CI-hooken på forge-siden og fundet to fejl, som ikke kunne findes ved at læse filen: en fallback, der sendte et argument, som ikke kunne opløses, ind i hookens input, og gits egen miljøvariabel, der fulgte gaten ind i checkouten og gjorde 19 tests røde.

Har skrevet en parser, der tjekker hvert FFI-kald

Har skrevet en parser, der læser den rigtige C-header på 7.308 linjer og verificerer hvert kaldssted, hver enum-konstant, og at hver klasse, der ejer en pointer, er final, efter at en håndskrevet pladsholder-header lod kald til tre fjernede funktioner kompilere, linke og crashe.

Har fået testsuiten ned på 8,9 sekunder

Har taget testsuiten fra seks tests over en grænse på tres sekunder til 135 bestående på 8,9 sekunder ved at profilere hovedtråden og fjerne de to kald, den hang i ved 3.989 ud af 4.017 stikprøver.

Har taget den ubrugte halvdel af kernen i brug

Har taget den halvdel af beskedkernen i brug, som applikationen aldrig havde brugt — backupoverførsel, forsvindende beskeder, redigering og gensendelse af beskeder, verificerede invitationer, proxyer og krypteringspolitik — og kørt hver test mod det rigtige bibliotek uden mocks.

Kend licensen på alt, hvad I sender ud

Har kontrolleret 644 Rust-crates for licenskompatibilitet ved hvert build og bevist, at tjekket slår til, ved at ændre licensen på én crate og ved at flytte den fastlåste kernerevision uden at regenerere.

Har holdt 4 repositories på én historikstandard

Har holdt fire repositories på én standard for historikken — konventionel, uden emojis, uden attributionslinjer, håndhævet af en commit-message-hook — sammen med 48 instruktionsdokumenter, der styrer, hvordan arbejdet udføres.

Har målt 13 adresser uden tracking

Har målt tretten adresser alene ud fra webserverens egen adgangslog — ingen cookie, ingen tracker og ingen tredjepartstjeneste — hvor klientens netværk maskeres i samme øjeblik linjen skrives, og resultatet samles i et permanent arkiv med 21 daglige tællinger og 20 månedlige facetter.

Har fået seks indsamlere til at melde det ulæste

Har fået seks rapporteringsindsamlere til at oplyse den del af deres input, de aldrig læste — uåbnede filer, ikke-installerede sonder og ugenkendte journallinjer — og tilpasset tærskler til 116 målte stikprøver efter at have fastslået, at en blind indsamling og en rask vært giver den samme side.

Sæt fokus på den tekniske indsats

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Sæt emnet ind i jeres sammenhæng

Fortæl os, hvordan »Automatisering & CI/CD« indgår i jeres system. Vi kan hjælpe med at vælge den relevante ydelse og afgrænse arbejdet.

Se vores ydelser

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

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://platform.engineer.company/da/categories/automation/