Engineer Plattform

Automatisering og CI/CD

61 resultater

Knytt sammen delene av systemet

Arbeidet under «Automatisering og CI/CD» viser én del av et fungerende system. Vi kan hjelpe med å knytte den delen til resten av driften deres. Overvåking og varsler som sier fra mens en feil fortsatt er liten, og tallene som viser hvor ofte den holder seg liten.

Se tjenestene våre Ta kontakt →

Finn en sammenlignbar utfordring

Åpne et eksempel som ligner situasjonen deres. Gjennomgangen forklarer beslutningene bak resultatet, utover navnet på teknologien. Bygde selskapets første observerbarhetsdashbord — selskapets første innsyn i egne systemer

Bygde selskapets første observerbarhetsdashbord Ta kontakt →

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.

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.

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.

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.

Automatiserte leveransen av 20 GIS-datapipelines

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

Automatiserte 100 kritiske sikkerhetskopier

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

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.

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.

Reduserte driftsrisiko med Grafana og Prometheus

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

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.

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.

Administrerte nettverksinfrastrukturen

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

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.

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.

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.

Automatiserte databehandling i Python og PL/pgSQL

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

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.

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.

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.

Skrev 578 automatiseringsmål i go-task

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

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.

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.

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.

Integrerte en passordbehandler i hele selskapet

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

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

Fikk testsuiten ned i 8,9 sekunder

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

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.

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.

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.

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.

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.

Fokuser den tekniske innsatsen

Ta med utrullingsløpet, driftsrisikoene og dem som har vakt. Vi kan avgrense en endring med kontroller, en vei tilbake og tydelig ansvar.

Se hva det koster Ta kontakt →

Sett emnet i deres sammenheng

Fortell hvordan «Automatisering og CI/CD» inngår i systemet deres. Vi kan hjelpe med å velge riktig tjeneste og avgrense arbeidet.

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/categories/automation/