Engineer

{ 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.

Avklarer dere fortsatt hva dere trenger?

Fortell oss hva som går galt, eller hva dere prøver å bygge. Vi sier hva det vil kreve, omtrent hva det vil koste, og om vi er de rette til oppgaven.

Fra 1 400 DKK timen, eksklusive mva. — og mindre per time ved et lengre oppdrag.

Ta kontakt Se hva det koster

Utvalgt arbeid som viser denne tjenesten.

Gjør tjenesten til et resultat

La «DevOps og CI/CD-automatisering» støtte et forretningsmål. Vi kan knytte det tekniske arbeidet til brukerne, driften og beslutningene det skal støtte. Testet, utgitt og mulig å reversere, slik at endringer når kundene en tirsdag ettermiddag i stedet for sent en fredagskveld.

Ta kontakt

Undersøk det relevante arbeidet

Bruk arbeidet på denne siden til å undersøke tilnærmingen vår. Spør om avveiningene og forutsetningene bak hvert resultat. Automatiserte GIS-SaaS-utrulling i GitHub Actions — produksjonssetting, data og rapportering, alt uten manuelle grep

Automatiserte GIS-SaaS-utrulling i GitHub Actions Ta kontakt →

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.

Reduserte driftsrisiko med Grafana og Prometheus

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

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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 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.

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.

Leter dere etter DevOps og CI/CD-automatisering?

Bruk de tilknyttede arbeidene til å vurdere relevant erfaring. Fortell oss hva dere skal bygge; vi kan diskutere omfang, fremgangsmåte og overlevering.

Ta kontakt Hele porteføljen

Avgrens oppgaven før bestilling

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 →

Gå gjennom kravene med oss

Fortell hva dere trenger innen «DevOps og CI/CD-automatisering», inkludert fristen og teamet som skal vedlikeholde løsningen. Vi kan drøfte en praktisk vei videre.

Ta kontakt

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://engineer.company/nb/services/devops-cicd/