Engineer Plattform

DevOps

42 resultat

Koppla samman systemets delar

Arbetet under ”DevOps” visar en del av ett fungerande system. Vi kan hjälpa er att koppla den delen till resten av verksamheten. Övervakning och larm som säger till medan ett fel fortfarande är litet, och siffrorna som visar hur ofta det förblir så.

Se våra tjänster Hör av er →

Hitta en jämförbar utmaning

Öppna ett exempel som liknar er situation. Genomgången förklarar besluten bakom resultatet, utöver teknikens namn. Byggde företagets första observerbarhetsdashboard — företagets första överblick över de egna systemen

Byggde företagets första observerbarhetsdashboard Hör av er →

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.

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.

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.

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.

Automatiserade leveransen av 20 GIS-datapipelines

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

Automatiserade 100 kritiska säkerhetskopior

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

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.

Förvaltade 30 VPS-instanser med Ubuntu Linux

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

Minskade driftrisken med Grafana och Prometheus

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

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.

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.

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.

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.

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.

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.

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.

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.

Ansvarade för plattformens Azure-driftsättningar

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

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.

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.

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.

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.

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.

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 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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

Låt 4 repon följa en och samma historikstandard

Låt 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.

Fokusera den tekniska insatsen

Ta med driftsättningsvägen, driftsriskerna och dem som har jour. Vi kan avgränsa en förändring med kontroller, en väg tillbaka och tydligt ansvar.

Se vad det kostar Hör av er →

Sätt ämnet i ert sammanhang

Berätta hur ”DevOps” ingår i ert system. Vi kan hjälpa er att välja rätt tjänst och avgränsa arbetet.

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/categories/devops/