Engineer Plattform

Pålitelighet og sikkerhetskopier

43 resultater

Knytt sammen delene av systemet

Arbeidet under «Pålitelighet og sikkerhetskopier» 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.

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.

Administrerte 30 Ubuntu Linux-VPS-instanser

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

Reduserte driftsrisiko med Grafana og Prometheus

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

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.

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.

Konfigurerte og installerte 1 000 Wi-Fi-rutere

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

Administrerte 100 Bare Bone-servere

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

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.

Et kartprodukt, håndtert fra ende til annen

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

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 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.

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.

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.

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.

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.

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.

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.

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.

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 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.

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.

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.

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.

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.

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.

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.

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.

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.

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 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 «Pålitelighet og sikkerhetskopier» 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/reliability/