Engineer Plattform

Zuverlässigkeit und Sicherungen

43 Leistungen

Die Teile Ihres Systems verbinden

Die unter „Zuverlässigkeit und Sicherungen“ gesammelten Arbeiten zeigen einen Teil eines funktionierenden Systems. Wir können helfen, diesen Teil mit dem Rest Ihres Betriebs zu verbinden. Monitoring und Alarme melden sich, solange ein Fehler noch klein ist; Kennzahlen zeigen, wie oft das gelingt.

Unsere Leistungen entdecken Kontakt aufnehmen →

Eine vergleichbare Herausforderung finden

Folgen Sie einem Beispiel, das Ihrer Situation ähnelt. Die Rückschau beschreibt die Entscheidungen hinter dem Ergebnis und geht über die bloße Technologiebezeichnung hinaus. Das erste Observability-Dashboard aufgebaut — der erste Einblick des Unternehmens in die eigenen Systeme

Das erste Observability-Dashboard aufgebaut Kontakt aufnehmen →

Das erste Observability-Dashboard aufgebaut

Das erste Observability-Dashboard des Unternehmens entwickelt und eingeführt, das Systemperformance und Datenvisualisierungen in Echtzeit auf dem großen Bürofernseher anzeigt.

10 PostgreSQL-/MS-SQL-Server auf Ubuntu

10 PostgreSQL- und MS-SQL-Server auf Ubuntu-Linux-VPS entworfen, bereitgestellt und gewartet und dabei Serverperformance und Zuverlässigkeit sichergestellt.

GIS-SaaS-Deployments mit GitHub automatisiert

Bereitstellung, Datenverarbeitung und Berichtswesen einer GIS-SaaS-Anwendung mit GitHub Actions CI/CD, Python, Bash und SQL automatisiert.

20 GIS-Datenpipelines automatisiert ausgeliefert

Die Bereitstellung von 20 GIS-Datenpipelines und ETL-Prozessen für Anwendungsdaten automatisiert und dadurch Infrastrukturautomatisierung und Berichterstattung vereinfacht.

100 wichtige Sicherungen mit Barman automatisiert

100 geschäftskritische Datensicherungen mit Barman, Google Cloud, Bash und Python automatisiert und die Datenintegrität über die Datenbanken hinweg abgesichert.

20 Docker-Apps auf Google Cloud und AWS betrieben

20 in Docker containerisierte Anwendungen bereitgestellt und gewartet, Fehler mit Podman und Kubernetes behoben und R-basierte Anwendungen auf Google Cloud und AWS verwaltet.

30 Ubuntu-Linux-VPS-Instanzen verwaltet

30 Ubuntu-Linux-VPS-Instanzen verwaltet, Strategien zur Notfallwiederherstellung umgesetzt und geeignete Netzwerkkonfigurationen sichergestellt.

Betriebsrisiken mit Grafana/Prometheus gesenkt

Betriebsrisiken durch ein Monitoring-Dashboard mit Grafana und Prometheus verringert und die Systemzuverlässigkeit verbessert.

8 WireGuard-/IPsec-VPN-Verbindungen verwaltet

8 WireGuard-VPN- und IPsec-VPN-Verbindungen verwaltet und entstört und eine sichere Kommunikation zwischen Google-Cloud- und Linux-Systemen sichergestellt.

Netzwerkinfrastruktur administriert

Die Netzwerkinfrastruktur für über 1.000 Server administriert und eine geeignete Systembereitstellung, Sicherheit und Fehlerdiagnose sichergestellt.

SSL/TLS für 100 Docker-Apps automatisiert

Die Erstellung von SSL/TLS-Zertifikaten für 100 Docker-Anwendungen automatisiert und sichere Verbindungen auf Ubuntu-Linux-Hosts sichergestellt.

CI/CD vereinfacht und 4.000 Stunden gespart

CI/CD-Prozesse verbessert und durch die Automatisierung von Softwareentwicklungspipelines 4.000 Stunden eingespart.

1.000 WLAN-Router konfiguriert und bereitgestellt

1.000 WLAN-Router konfiguriert und bereitgestellt und die Netzwerkverfügbarkeit und Performance für Kunden verbessert.

100 Bare-Bone-Server administriert

100 Bare-Bone-Server, physische Netzwerke und IP-Telefoniesysteme administriert und eine tragfähige Infrastruktur für das Unternehmenswachstum bereitgestellt.

40 Websites mit Apache und Nginx administriert

40 Websites auf Ubuntu-Linux-Hostingservern mit Apache und Nginx administriert und hohe Verfügbarkeit und Performance sichergestellt.

Ein Kartenprodukt durchgängig verantwortet

Infrastruktur, Datenverarbeitung und eine Kartenanwendung über 2 Jahre konzipiert, entwickelt und betreut, einschließlich Softwareauslieferung und betrieblicher Wartung.

Den API-Vertrag aus der Datenbank generiert

Den API-Vertrag von der Datenbank aus nach außen generiert — OpenAPI, einen typisierten TypeScript-Client mit 44.076 Zeilen, 61 Mock-Handler und die von der UI durchgesetzten Grenzen — mit einer Prüfung an jedem Übergang, die Abweichungen zurückweist.

E-Mail mit Failover und Zustellprotokoll gebaut

E-Mail als Plattformfunktion entwickelt — drei Anbieter mit Ausfallsicherung, Zustell-Webhooks, Versand- und Zustellprotokollen, Vorlagen und Kampagnen — hinter einer Startprüfung, die ohne eingerichteten Anbieter den Systemstart verhindert.

Azure-Infrastruktur als Code provisioniert

Azure-Infrastruktur mit Bicep als Code provisioniert — Container Apps, PostgreSQL Flexible Server, Front Door/WAF und Netzwerke — für Entwicklungs-, Staging- und Produktivumgebung.

GitHub-Actions-CI/CD-Pipelines aufgebaut

GitHub-Actions-CI/CD-Pipelines mit einem Distroless-Frontend-Image für den Produktivbetrieb und der Übernahme von Releases über mehrere Umgebungen entwickelt.

Azure-Plattformbereitstellungen verantwortet

Die durchgängige Bereitstellung der Plattform auf Azure verantwortet und Releases über Entwicklungs-, Staging- und Produktivumgebungen hinweg gesteuert.

Sicherungen und Notfallwiederherstellung als Code

Die Aufbewahrung von Datenbanksicherungen als Infrastructure as Code konfiguriert, anschließend die Wiederherstellbarkeit geprüft und das Wiederherstellungsverfahren dokumentiert — mit ausdrücklich benannten Lücken, statt sie erst im Ernstfall entdecken zu lassen.

24/7-Support für rund 1.000 IPTV/OTT-Server

Rund um die Uhr 24/7-Infrastruktursupport für eine IPTV/OTT-Streaming-Plattform geleistet und rund 1.000 Server sowie kundeneigene Systeme für internationale Kunden in China, den USA und Deutschland administriert.

IPTV-Signale unterbrechungsfrei übertragen

Die unterbrechungsfreie Übertragung von IPTV-Streaming-Signalen zwischen Anbietern und Kunden sichergestellt und Streaming-Netzwerk sowie IP-Telefonie rund um die Uhr überwacht und gewartet.

Infrastruktur entwickelt, die Jahre später läuft

Neue Infrastrukturfunktionen für interne und externe Systeme geplant und umgesetzt und dabei langlebige Lösungen geschaffen, die Jahre später mit minimalen Änderungen weiterlaufen.

Testsuite für Go, SQL und Browser aufgebaut

Eine mehrschichtige automatisierte Testsuite aufgebaut — 981 Go-Tests, 543 Frontend- und Browserspezifikationen, 494 verhaltensprüfende SQL-Tests — mit Mutationstests, eigenschaftsbasierten Tests und einer verbindlichen Barrierefreiheitsprüfung.

Langsame Aufgaben in River-Jobqueue verlagert

Langsame Verarbeitung aus dem Anfragepfad in eine River-Jobqueue verlagert — 15 Worker-Module, 8 geplante Aufgaben und 20 pg_cron-Jobs — sodass Anfragen zurückkehren, während die zugehörige Arbeit weiterläuft.

Eigene Fehlerüberwachung und Tracing entwickelt

Eigene Fehlerüberwachung und OpenTelemetry-Tracing entwickelt statt eingekauft — Bereinigung von Nutzdaten, Erkennung von Fehlerspitzen und Regressionen, Symbolauflösung und synthetisches Lebenszeichen — mit 11 Betriebsansichten.

1.022 Migrationen mit Schemaabweichungsprüfung

Die Konsistenz des Schemas über 1.022 Migrationen mit einer CI-Prüfung abgesichert, die die Datenbank auf zwei Wegen aufbaut — Neuinstallation sowie Installation mit sämtlichen Migrationen — und bei Unterschieden fehlschlägt.

Bei Fehlern sperrenden Missbrauchsschutz gebaut

Bei Fehlern sperrende Missbrauchsschutzmechanismen aufgebaut — 22 Redis-gestützte Ratenbegrenzer, Cloudflare Turnstile, Anfrage-Idempotenz und eine Ursprungssperre — damit die Plattform Bots und Überlastungsangriffe abweist, statt ihren Aufrufern zu vertrauen.

Kontinuierliche Sicherheitsprüfungen etabliert

Ein kontinuierliches Sicherheitsprogramm aufgebaut — Codeanalyse, DAST, Abhängigkeits- und Schwachstellenprüfungen, SBOM-Erzeugung, Geheimnissuche und per SHA festgelegte Actions — ergänzt durch 21 schriftliche Sicherheitsaudits.

Infrastruktur in einem Repository definiert

Die eigene Unternehmensinfrastruktur als 19 Ansible-Playbooks und 34 Rollen mit 12.065 YAML-Zeilen aufgebaut, die einen laufenden Host mit ausschließlich idempotenten Plays in den deklarierten Sollzustand überführen.

Öffentliche Dienste auf einem kleinen Server

Die öffentlichen Unternehmensdienste auf einem Single-Core-Host mit 512 MB gehostet — Git-Forge, Webserver für sieben Domains, Tor, zwei Server für alternative Protokolle und Angriffssperren — mit 464 MB nutzbarem Arbeitsspeicher als Entwurfsgrenze; Sicherungsautomatisierung ist eine getrennte Implementierung.

Angriffssperren bei jedem Lauf nachgewiesen

Den Pfad zur Angriffssperre bei jedem Härtungslauf durchgängig nachgewiesen: eine reservierte Testadresse gesperrt, die entstandene Kernelregel ausgelesen und die Sperre in einem garantierten Aufräumblock aufgehoben, sodass eine ausgefallene Jail den Lauf scheitern lässt, statt einen gesunden Zustand zu melden.

Sicherungsautomatisierung mit klarem Wiederherstellungsweg

restic-Sicherungsautomatisierung mit Aufbewahrung, Integritätsprüfungen und monatlichem Wiederherstellungstest umgesetzt; Speicheraktivierung und Nachweis einer tatsächlichen Wiederherstellung als getrennte Betriebsschritte dokumentiert.

Totmannüberwachung für Hosts entwickelt

Eine Totmannüberwachung entwickelt, die nur bei gesundem Speicher- und Plattenzustand Signale sendet, sodass ein beeinträchtigter Host durch sein Verstummen alarmiert — und sechs Variablennamen mit "free" gefunden, deren Prüfungen korrekt "available" maßen, auf einem Rechner mit 464 MB um eine Größenordnung verschieden.

Ansibles Prüfmodus aussagekräftig gemacht

Den Prüfmodus auf der gesamten Plattform aussagekräftig gemacht, nachdem sechs Sonden anhand eines vom Host nie gelieferten Werts entschieden hatten: Ansibles command-Modul meldet unter --check Erfolg, obwohl es den Befehl vollständig überspringt.

Sekunden-Preflight für Geheimnisse ergänzt

Ein Preflight-Play ergänzt, das denselben Code wie der Konvergenzlauf mit den lokal beim Betreiber liegenden Geheimnissen in etwa einer Sekunde ausführt, nachdem ein teilweise durchgelaufener Produktivlauf bei Aufgabe neun abbrach, während die Swap-Einstellungen bereits auf dem Produktivhost geschrieben waren.

Jedes Betriebsgeheimnis je Host getrennt

Die vier betrieblichen Geheimnisse je Host getrennt, nachdem sich gezeigt hatte, dass zwei Hosts mit einer gemeinsamen Totmannschaltung weniger zuverlässig alarmieren als zwei getrennte Schaltungen und eine gemeinsame Sicherungspassphrase die Sicherungen beider Hosts auch in getrennten Repositorys offenlegen kann, wenn jemand die Passphrase erlangt und die verschlüsselten Daten und Schlüsseldateien beider Repositorys lesen kann.

Generator mit 981 Tests und 92 % Abdeckung

Den Generator mit 981 Testfällen und mindestens 92 % Zweigabdeckung geprüft, Warnungen als Fehler behandelt und Idempotenz durch zwei vollständige Pipeline-Durchläufe aus einer leeren Datei nachgewiesen, wobei der zweite nichts ändern darf.

Eine begrenzte Ereignispipeline für eine macOS-App

Ereignispuffer und Protokollierung nach einem dokumentierten macOS-Vorfall mit 41 MB Wachstum pro Sekunde und 111 GB komprimierten Speicherseiten auf einem Rechner mit 36 GB begrenzt; ein dokumentierter vollständiger Testlauf reduzierte Kernprotokollzeilen von 610.996 auf 1.411.

Die ungenutzte Hälfte des Kerns erschlossen

Die zuvor ungenutzte Hälfte des Messaging-Kerns erschlossen — Sicherungsübertragung, verschwindende Nachrichten, Nachrichtenbearbeitung und erneutes Senden, verifizierte Einladungen, Proxys und Verschlüsselungsrichtlinien — und jeden Test ohne Mocks gegen die echte Bibliothek ausgeführt.

Sechs Sammler weisen übersprungene Eingaben aus

Sechs Berichtssammler um Angaben zu ungelesenen Eingaben ergänzt — ungeöffnete Dateien, nicht installierte Sonden, unbekannte Journalzeilen — und Schwellenwerte anhand von 116 gemessenen Stichproben bestimmt, nachdem eine blinde Erfassung und ein gesunder Host dieselbe Seite ergeben hatten.

Die technische Arbeit gezielt einsetzen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Das Thema in Ihren Kontext einordnen

Erzählen Sie uns, welche Rolle „Zuverlässigkeit und Sicherungen“ in Ihrem eigenen System spielt. Wir können helfen, die passende Leistung auszuwählen und die damit verbundene Arbeit zu definieren.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://platform.engineer.company/de/categories/reliability/