Engineer Plattform

{ Infrastructure as Code }

Server, die Sie aus versionierter Konfiguration neu aufbauen können

Die gesamte Einrichtung wird in Dateien festgehalten, damit sie sich exakt wiederherstellen lässt und nicht nur im Kopf einer einzelnen Person existiert.

Sie klären noch, was Sie benötigen?

Erzählen Sie uns, was nicht funktioniert oder was Sie entwickeln möchten. Wir sagen Ihnen, was dafür nötig wäre, was es ungefähr kosten würde und ob wir die Richtigen dafür sind.

Ab 1.400 DKK pro Stunde, zuzüglich Mehrwertsteuer — und weniger pro Stunde bei längerer Buchung.

Kontakt aufnehmen Preise ansehen

Ausgewählte Arbeiten, die diese Leistung zeigen.

Die Leistung in ein Ergebnis überführen

Machen Sie „Infrastructure as Code“ für ein Geschäftsergebnis nutzbar. Wir können die technische Arbeit mit den Nutzern, Betriebsabläufen und Entscheidungen verbinden, die sie unterstützen soll. Die gesamte Einrichtung wird in Dateien festgehalten, damit sie sich exakt wiederherstellen lässt und nicht nur im Kopf einer einzelnen Person existiert.

Kontakt aufnehmen

Die passenden Arbeiten prüfen

Prüfen Sie unser Vorgehen anhand der zugehörigen Arbeiten auf dieser Seite. Fragen Sie nach den Abwägungen und den Bedingungen hinter jedem Ergebnis. 1.000 RBAC-Regeln für sichere Linux-Server — 1,000 Zugriffsregeln zum Schließen einer Sicherheitslücke

1.000 RBAC-Regeln für sichere Linux-Server Kontakt aufnehmen →

1.000 RBAC-Regeln für sichere Linux-Server

Die Datensicherheit durch 1.000 RBAC-Regeln für Entwickler, Anwendungsinstanzen, PostgreSQL, MS SQL und weitere Linux-Server erhöht, unberechtigte Zugriffe verhindert und die Regeln mit Ansible-Automatisierung dokumentiert.

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.

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.

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.

SSH mit 24 geprüften Direktiven gehärtet

SSH auf 24 geprüfte Direktiven gehärtet und dreistufig validiert — Kandidatendatei, zusammengesetzte Konfiguration und anschließend die Rückmeldung des Daemons selbst — nachdem diese gezeigt hatte, dass der laufende Server zwei der vierundzwanzig Direktiven stillschweigend überschrieb.

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.

DNS-Zone mit 20 Einträgen als Code abgeglichen

Eine DNS-Zone mit 20 Einträgen deklarativ über die Cloudflare-API abgeglichen, mit separaten Einstiegspunkten für Audit und BIND-Export, und den zuvor eingerichteten CDN-Proxy aus Datenschutzgründen wieder abgeschaltet.

Alle systemd-Units stärker isoliert

Die Angriffsfläche der systemd-Sandboxen für jede von der Plattform installierte Unit verringert — einen Totmannschaltungsdienst von 9,6 UNSAFE auf 1,5, eine internetseitig erreichbare Git-Forge von 8,3 EXPOSED auf 1,5 — und eine Parserprüfung beim Konvergenzlauf ergänzt, nachdem eine falsch geschriebene Direktive in drei Unit-Vorlagen stillschweigend ignoriert worden war.

Ihr Code auf Ihrem eigenen Server

Die eigene Git-Forge auf Soft Serve bereitgestellt, standardmäßig privat, ohne Weboberfläche und mit an Loopback gebundenem SSH-Port hinter einem Sprunghost; die vorgeschaltete Startseite als Build-Artefakt der Hauptwebsite statt als manuell gepflegte Kopie umgesetzt.

Einen zweiten Server aus Code provisioniert

Die Provisionierung eines zweiten Servers bei einem zweiten Cloud-Anbieter automatisiert und die Firewall vor dem Rechner angelegt, damit er von Anfang an geschützt ist; die Firewalls beider Anbieter direkt über deren REST-APIs gesteuert, um eine Drittanbieter-Collection zu vermeiden.

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.

Forge-CI-Hook geprobt und 2 Fehler gefunden

Den CI-Hook auf der Forge probeweise ausgeführt und zwei beim bloßen Lesen unerreichbare Fehler gefunden: einen Fallback mit einem nicht auflösbaren Eingabeargument und eine Git-Umgebungsvariable, die der Qualitätsprüfung in den Checkout folgte und 19 Tests scheitern ließ.

Lesbaren Tor-Onion-Spiegel veröffentlicht

Die Website als Tor-Onion-Spiegel unter einer lesbaren, mit engineer beginnenden Adresse veröffentlicht, den Schlüssel außerhalb des Hosts erzeugt, sodass er weder dieses Repository noch einen Laptop erreichte, und diesen einen von vier Zugängen von der Zählung ausgenommen.

Sie suchen Infrastructure as Code?

Beurteilen Sie anhand der verlinkten Arbeiten die einschlägige Erfahrung. Erzählen Sie uns, was Sie entwickeln möchten; wir können Umfang, Vorgehen bei der Umsetzung und Übergabe besprechen.

Kontakt aufnehmen Gesamtes Portfolio

Den Umfang vor der Beauftragung festlegen

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 →

Ihre Anforderungen besprechen

Erzählen Sie uns, was Sie von „Infrastructure as Code“ benötigen, einschließlich des Termins und des Teams, das die Wartung übernehmen wird. Wir können einen praktikablen Weg besprechen.

Kontakt aufnehmen

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/services/infrastructure-as-code/