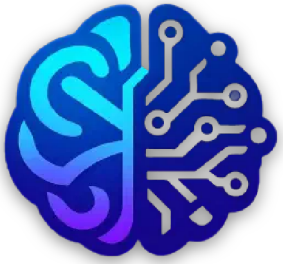
ソフトウェアを届ける **プラットフォームを運用し、** コードから再構築し、監視し、文書化します。

クラウド、コードで管理するインフラ、CI/CD、コンテナ、監視、バックアップ、ネットワーク、アクセス管理を、コペンハーゲンから。



company
Engineer
プラットフォーム

リポジトリから再構築できるインフラ

+45 50 10 61 84 · デンマーク、コペンハーゲン

心を込めて丁寧に制作： D.

事業を支えるシステムのために、クラウドインフラのコード化、デプロイの自動化、復旧計画の策定を行います。実績の記録では、出発点となった状況、実施した内容、その結果を説明しています。

まず、デプロイの経路か運用上のリスクを一つ取り上げます。レビューまたは実装の範囲を合意し、展開、ロールバック、責任分担を文書化することで、何をどのように運用するかをチームが把握できるようにします。

インフラの実績 · 料金と依頼の形 · プロジェクトを相談する

必要な成果から考える

事業として実現したいことをお聞かせください。利用する人々を中心に、ソフトウェア、データ、インフラを結び付けます。

サービスを見る お問い合わせ →

仕事の進め方を見る

目標に合うサービスを探し、何をどのように実施したかを事例でご確認ください。

事例 お問い合わせ →

実績を読む **その背景とともに** — 判断、実施内容、報告された成果をご覧ください。

実績一覧を見る

一社で支える **10 件のサービス** 最初のサーバーから午前3時の呼び出しまで。

サービス一覧を見る

{ クラウド・インフラ・DevOps }

ソフトウェアを動かす場所を整える

クラウドインフラ・移行

実際の使用量に合わせてサーバーを構成し、月額費用を事前に把握したうえで、クラウドへの移行もクラウドからの移転も進めます。

2台目のサーバーをコードで構築

別のクラウド事業者で2台目のサーバーの構築を自動化。先にファイアウォールを作成し、起動時から保護されるようにし、両事業者のファイアウォールをREST APIから直接操作して、第三者のcollectionへの依存を避けた。

PodmanとQuadletでコンテナを運用

Dockerの既定のブリッジネットワークでのポート公開がUFWルールを迂回する場合があるため、systemd配下のPodman・Quadletでコンテナ基盤を構築。デプロイにはrootではなく、固定コマンド1つを使う非特権ユーザーを用いた。

小型サーバー上の公開サービス

利用可能メモリー464 MBを設計制約として、512 MB・1コアのホストで自社の公開サービスを運用。gitホスティング、7ドメインのWebサーバー、Tor、2つの代替プロトコルサーバー、不正接続元の遮断を備え、バックアップ自動化は別の実装として扱った。

9件の実績をすべて見る

仕事の進め方

まず理解する

解決策を提案する前に、課題を理解します。最初にいただくご要望は、対話の結論ではなく出発点です。

長く使えるものをつくる

速さより持続性、技巧より分かりやすさを大切にします。引き継いだ人が扱えるシステムであることが必要です。

仕事に責任を持つ

継続的な運用やサポートも、契約の一部として合意します。引き継ぎでは、責任の所在と、その後の連絡先を明確にします。

{ クラウド・インフラ・DevOps }

バージョン管理された設定から再構築できるサーバー

コードによるインフラ管理

構成全体をファイルに記述し、特定の人の記憶だけに頼らず、まったく同じ環境を再現できるようにします。

読みやすいTor onionミラーを公開

engineerで始まる読みやすいTor onionアドレスでサイトのミラーを公開。鍵はホスト外で生成し、このリポジトリにもノートパソコンにも置かず、4つの入口のうち計測しない唯一の入口にした。

gitホスティングCIを試し2不具合を発見

gitホスティング側のCIフックを予行実行し、ファイルを読むだけでは到達できない2件の不具合を発見。解決不能な引数を入力へ渡すフォールバックと、検査先に引き継がれ19テストを失敗させるgit自身の環境変数を修正した。

すべての運用秘密情報をホストごとに分離

2ホストで1つのデッドマンズスイッチを共有すると2つのスイッチより監視能力が下がり、バックアップのパスフレーズ共有では、リポジトリが別々でも、そのパスフレーズを入手し、両リポジトリの暗号化データと鍵ファイルを読める相手が2ホストのバックアップを復号できることを確認し、ホストごとの運用秘密情報4つを分離した。

17件の実績をすべて見る

{ クラウド・インフラ・DevOps }

落ち着いて変更を届ける

DevOps・CI/CD自動化

変更を検証し、リリースし、元に戻せる仕組みを整え、金曜の深夜ではなく火曜の午後にお客様へ届けられるようにします。

4リポジトリで共通の履歴基準を維持

4つのリポジトリに対し、Conventional Commits、絵文字禁止、帰属トレーラー禁止という共通の履歴基準をコミットメッセージフックで適用し、作業方法を定める48件の指示文書とともに運用した。

公開するすべての部品のライセンスを把握

毎回のビルドで644個のRust crateのライセンス互換性を監査し、1つのcrateのライセンスを書き換えた場合と、再生成せずコアの固定リビジョンを変えた場合に検査が働くことを実証した。

未使用だったコアの半分の機能を実装

バックアップ転送、消えるメッセージ、編集・再送、確認済み招待、プロキシ、暗号化方針など、アプリケーションが未使用だったメッセージングコアの半分の機能に対応し、モックなしで実ライブラリに対して全テストを実行した。

43件の実績をすべて見る

{ クラウド・インフラ・DevOps }

どこでも同じように動くソフトウェア

コンテナ化・オーケストレーション

ノートパソコン、テストサーバー、本番環境で同じ動作になるようにまとめ、忙しい週には処理能力を増やせるようにします。

PodmanとQuadletでコンテナを運用

Dockerの既定のブリッジネットワークでのポート公開がUFWルールを迂回する場合があるため、systemd配下のPodman・Quadletでコンテナ基盤を構築。デプロイにはrootではなく、固定コマンド1つを使う非特権ユーザーを用いた。

全ユニットのsystemdサンドボックス露出を低減

この基盤が導入するすべてのユニットのsystemdサンドボックスの露出度を低減。デッドマンズスイッチのサービスを9.6 UNSAFEから1.5へ、インターネット公開のgitホスティングを8.3 EXPOSEDから1.5へ改善し、3つのユニットテンプレートで黙って無視されていた設定名の誤記を受け、収束時の構文検査を加えた。

GitHub Actions CI/CDパイプラインを構築

本番フロントエンド用のdistrolessイメージと複数環境間の昇格を備えた、GitHub ActionsのCI/CDパイプラインを構築した。

5件の実績をすべて見る

{ クラウド・インフラ・DevOps }

お客様より先に異変に気づく

サイト信頼性・監視

障害が小さいうちに知らせる監視とアラートを整え、安定した状態をどれだけ維持できているかを数値で示します。

6収集処理が省略した入力を明示

6つのレポート収集処理が、未読ファイル、未導入の検査、認識できなかったjournal行など、読み取れなかった入力範囲を明示するよう変更。収集不能と正常なホストが同じページになると確認し、116件の実測サンプルから閾値を定めた。

追跡せずに13アドレスを測定

Cookie、トラッカー、第三者サービスを使わず、Webサーバーのアクセスログのみから13アドレスを測定。書き込み時に接続元ネットワークをマスクし、21の日別集計と20の月別分析軸からなる永続アーカイブへまとめた。

macOSアプリのイベントパイプラインに上限を設定

毎秒41 MBの増加と、36 GB端末で111 GBの圧縮ページを記録したmacOSの障害を受け、イベントバッファとログ出力に上限を設定。記録された全テスト実行では、コアのログ行が610,996から1,411へ減った。

23件の実績をすべて見る

{ クラウド・インフラ・DevOps }

大きな障害のあとにすべてを取り戻す

バックアップ・災害復旧

あると信じるだけでなく、復元を試したバックアップと、
サーバーが起動しない朝に使える手順書を用意します。

すべての運用秘密情報をホストごとに分離

2ホストで1つのデッドマンズスイッチを共有すると2つのスイッチより監視能力が下がり、バックアップのパスフレーズ共有では、リポジトリが別々でも、そのパスフレーズを入手し、両リポジトリの暗号化データと鍵ファイルを読める相手が2ホストのバックアップを復号できることを確認し、ホストごとの運用秘密情報4つを分離した。

復元方法を明示したバックアップ自動化

resticのバックアップを、保持期間、整合性検査、毎月の復元テストとともに自動化し、ストレージの有効化と実環境での復元検証を別の運用手順として記録した。

バックアップと災害復旧をコードで実装

データベースのバックアップ保持をコードで設定し、復旧可能な状態を監査して復元手順を文書化。残る不足を障害時に初めて知ることがないよう明記した。

5件の実績をすべて見る

{ クラウド・インフラ・DevOps }

日々の稼働を支える

システム管理

サーバー、メール、アカウント、更新を日々管理
し、誰もが頼る仕組みを平穏に動かし続けます。

読みやすいTor onionミラーを公開

engineerで始まる読みやすいTor onionアドレスでサイトのミラーを公開。鍵はホスト外で生成し、このリポジトリにもノートパソコンにも置かず、4つの入口のうち計測しない唯一の入口にした。

6収集処理が省略した入力を明示

6つのレポート収集処理が、未読ファイル、未導入の検査、認識できなかったjournal行など、読み取れなかった入力範囲を明示するよう変更。収集不能と正常なホストが同じページになると確認し、116件の実測サンプルから閾値を定めた。

gitホスティングCIを試し2不具合を発見

gitホスティング側のCIフックを予行実行し、ファイルを読むだけでは到達できない2件の不具合を発見。解決不能な引数を入力へ渡すフォールバックと、検査先に引き継がれ19テストを失敗させるgit自身の環境変数を修正した。

28件の実績をすべて見る

{ クラウド・インフラ・DevOps }

拠点と働く人をつなぐ

ネットワーク・VPN構築

Wi-Fi、ファイアウォール、リモートアクセスを整え、どこからでも働け、権限のある人だけが接続できるようにします。

2台目のサーバーをコードで構築

別のクラウド事業者で2台目のサーバーの構築を自動化。先にファイアウォールを作成し、起動時から保護されるようにし、両事業者のファイアウォールをREST APIから直接操作して、第三者のcollectionへの依存を避けた。

20レコードのDNSゾーンをコードで照合

20レコードのDNSゾーンをCloudflare APIと宣言的に照合し、監査とBIND形式の書き出しを別の実行入口として提供。CDNプロキシは構築後、プライバシーの観点から再び無効にした。

ファイアウォールルールを位置で検証

存在するルールが実際のパケットに到達されるとは限らないため、番号付きルール一覧と稼働中のパケットフィルターチェーンを読み、ファイアウォールルールを存在だけでなく位置で検証した。

11件の実績をすべて見る

{ クラウド・インフラ・DevOps }

誰が何にアクセスできるかを決める

セキュリティ・アクセス管理

アカウント、パスワード、権限を整理し、業務に必要なアクセスだけを付与し、退職者の権限は当日に解除します。

読みやすいTor onionミラーを公開

engineerで始まる読みやすいTor onionアドレスでサイトのミラーを公開。鍵はホスト外で生成し、このリポジトリにもノートパソコンにも置かず、4つの入口のうち計測しない唯一の入口にした。

追跡せずに13アドレスを測定

Cookie、トラッカー、第三者サービスを使わず、Webサーバーのアクセスログのみから13アドレスを測定。書き込み時に接続元ネットワークをマスクし、21の日別集計と20の月別分析軸からなる永続アーカイブへまとめた。

公開するすべての部品のライセンスを把握

毎回のビルドで644個のRust crateのライセンス互換性を監査し、1つのcrateのライセンスを書き換えた場合と、再生成せずコアの固定リビジョンを変えた場合に検査が働くことを実証した。

31件の実績をすべて見る

{ クラウド・インフラ・DevOps }

AIが扱ってよいものだけを扱わせる

AIエージェントのセキュリティ

各エージェントのアクセスを必要な範囲に限定し、エージェント自身が改変できない場所に操作を記録し、安全策を信じるだけでなく検証します。

すべての運用秘密情報をホストごとに分離

2ホストで1つのデッドマンズスイッチを共有すると2つのスイッチより監視能力が下がり、バックアップのパスフレーズ共有では、リポジトリが別々でも、そのパスフレーズを入手し、両リポジトリの暗号化データと鍵ファイルを読める相手が2ホストのバックアップを復号できることを確認し、ホストごとの運用秘密情報4つを分離した。

認証情報漏出後に対象範囲の規則を記述

構成変更によって別会社の資産一覧、ファイアウォール許可、文章が混入したことを受け、リポジトリの対象範囲を定める規則を記述し、隔離した残存物も除外せず秘密情報検査の対象に残した。

秘密情報の1秒の事前検査を追加

本番実行が9番目のタスクで停止し、すでにスワップ設定が稼働ホストへ書き込まれた事態を受け、運用担当者のローカルな秘密情報に対し、収束処理と同じコードを約1秒で実行する事前検査プレイを追加した。

9件の実績をすべて見る

お客様の事業で **すでに使われている基盤を** 私たちも扱っています。

Apple Microsoft DigitalOcean AWS Google

技術構成をすべて見る

対象のAIエージェントのセキュリティレビュー

把握する

対象とするエージェント、認証情報、データへのアクセスを一覧にして合意し、設定とともに管理します。

制限する

各エージェントに固有のIDと最小限の権限を与え、読み込む文章は命令ではなくデータとして扱います。

記録する

誰のために行ったかを含め、すべての操作を、エージェントが変更できない場所に記録し、通常と異なる動作には通知を出します。

エージェントの保護方法を見る 自社のエージェントについて相談する →

つくりたいものをお聞かせください。

きちんと読む担当者へのメール一通から始まり、フォームや営業への誘導、自動配信の営業メールは挟みません。

相談を始める

このサイトではインフラ分野の仕事のみを紹介しています。ソフトウェア開発、データ関連の仕事、チームへの助言については、engineer.companyをご覧ください。
Engineer ApSの事業をすべて見る →

具体的な改善から始める

改善が必要な作業手順、データの流れ、システムなど、一つの課題から始められます。その範囲を一緒に決めましょう。

料金を見る お問い合わせ →

最初の相談に向けて

必要な成果、現在の仕組み、ご希望の時期をお聞かせください。それを基に、次の進め方をご相談できます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

サーバーには **オープンソースのLinuxを採用** — ライセンス料がかからず、中身も公開されています。



Ubuntu



Debian

サーバーの運用方法



このページを持ち歩く



連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://platform.engineer.company/ja/