Engineer プラットフォーム

信頼性・バックアップ

43 件の実績

システムの各部分を結び付ける

「信頼性・バックアップ」に分類した実績は、稼働するシステムの一部分を示しています。その部分を他の業務につなげるお手伝いをします。 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。

サービスを見る お問い合わせ →

比較できる課題を探す

ご自身の状況に近い事例をご覧ください。振り返りでは、技術名だけでは分からない、結果を導いた判断を説明しています。 社内初のオブザーバビリティダッシュボードを構築 — 社内システムの状態を初めて可視化

社内初のオブザーバビリティダッシュボードを構築 お問い合わせ →

社内初のオブザーバビリティダッシュボードを構築

社内初のオブザーバビリティダッシュボードを開発・公開し、オフィスの大型テレビでシステム性能とデータをリアルタイムに把握できるようにした。

UbuntuでPostgreSQL・MS SQLサーバー10台を運用

Ubuntu Linux VPS上にPostgreSQL・MS SQLサーバー10台を設計・デプロイ・保守し、性能と信頼性を維持した。

GitHub ActionsでGIS SaaSのデプロイを自動化

GitHub Actions CI/CD、Python、Bash、SQLで、GIS SaaSアプリケーションのデプロイ、データ処理、レポート作成を自動化した。

GISデータパイプライン20本の提供を自動化

20本のGISデータパイプラインとアプリケーションデータのETL処理の提供を自動化し、インフラ自動化とレポート作成を効率化した。

Barmanで重要データのバックアップ100件を自動化

Barman、Google Cloud、Bash、Pythonで重要データのバックアップ100件を自動化し、データベース間のデータ整合性を確保した。

Google Cloud・AWSでDockerアプリ20個を運用

Dockerでコンテナ化した20個のアプリケーションをデプロイ・保守。PodmanとKubernetesで問題を調査し、Google Cloud・AWS上のRアプリケーションも管理した。

Ubuntu Linux VPSインスタンス30台を管理

Ubuntu Linux VPSインスタンス30台を管理し、災害復旧方針の実装とネットワーク構成の最適化を行った。

GrafanaとPrometheusで運用リスクを低減

GrafanaとPrometheusの監視ダッシュボードを導入して運用リスクを低減し、システムの信頼性を高めた。

WireGuard・IPsec VPN接続8本を管理

WireGuard VPN・IPsec VPNの8接続を管理して問題に対処し、Google CloudとLinuxシステム間の安全な通信を確保した。

ネットワークインフラを管理

1,000台を超えるサーバーのネットワークインフラを管理し、適切なシステム構築、セキュリティ、障害対応を行った。

Dockerアプリ100個のSSL/TLS証明書を自動化

Dockerアプリケーション100個のSSL/TLS証明書の作成を自動化し、Ubuntu Linuxホスト上での安全な接続を確保した。

CI/CDの効率化で4,000時間を削減

ソフトウェア開発パイプラインを自動化してCI/CD工程を効率化し、4,000時間を削減した。

Wi-Fiルーター1,000台を設定・導入

1,000台のWi-Fiルーターを設定・導入し、顧客のネットワーク接続性と性能を改善した。

ベアボーンサーバー100台を管理

ベアボーンサーバー100台、物理ネットワーク、IP電話システムを管理し、会社の成長を支えるインフラを維持した。

Apache・NginxでWebサイト40件を管理

Apache・Nginxを使用するUbuntu Linuxホスティングサーバー上で40件のWebサイトを管理し、高い可用性と性能を維持した。

地図製品を最初から運用まで支える

2年間にわたり、インフラ、データ処理、地図アプリケーションを設計・開発・支援し、ソフトウェアの提供と運用保守の両方を担当した。

データベースを起点にAPI契約を生成

データベースを起点に、OpenAPI、44,076行の型付きTypeScriptクライアント、61個のモックハンドラー、UIが適用する制限を生成し、各段階に不一致で失敗する検査を設けた。

切り替えと配信ログを備えるメール機能を構築

メール送信をプラットフォームの共通機能として構築。3つのプロバイダーによるフェイルオーバー、配信Webhook、送信・配信ログ、テンプレート、キャンペーンを備え、送信手段がなければ起動を拒否する検査を設けた。

Azureインフラをコードとして構築

Bicepで、開発・ステージング・本番の各環境に、Container Apps、PostgreSQL Flexible Server、Front Door/WAF、ネットワークを含むAzureインフラをコードとして構築した。

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。

Azureへのプラットフォーム導入を一貫して担当

プラットフォームのAzureへのデプロイを最初から最後まで担当し、開発・ステージング・本番環境のリリースを管理した。

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。

約1,000台のIPTV/OTTサーバーを24/7で支援

中国、米国、ドイツの世界各地の顧客に向け、IPTV/OTT配信プラットフォームのインフラを24/7体制で支援し、約1,000台のサーバーと顧客所有のシステムを管理した。

IPTV配信信号の連続稼働を維持

配信ネットワークとIP電話を昼夜問わず監視・保守し、供給元と顧客の間でIPTV信号の途切れない配信を維持した。

数年後も稼働するインフラを構築

社内外のシステム向けに新たなインフラ機能を計画・実装し、数年後も最小限の変更で稼働し続ける仕組みを構築した。

Go・SQL・ブラウザーのテスト群を構築

981件のGoテスト、543件のフロントエンド・ブラウザーテスト、494件のSQL動作テストを持つ多層の自動テスト群を構築し、ミューテーションテスト、プロパティベーステスト、アクセシビリティの品質ゲートを加えた。

遅い処理をRiverジョブキューへ移行

時間のかかる処理をリクエスト経路からRiverジョブキューへ移し、15個のワーカーモジュール、8件の定期タスク、20件のpg_cronジョブにより、背後の作業が続く間も応答を返せるようにした。

独自のエラー監視・トレースを構築

ペイロードの無害化、急増・再発の検知、シンボル解決、擬似ハートビートを備える独自のエラー監視とOpenTelemetryトレースを外部購入せず構築し、11画面の運用ビューから扱えるようにした。

1,022件のマイグレーションを差分検査で保護

新規インストールと、インストール後に全マイグレーションを適用する2通りでデータベースを構築し、不一致で失敗するCIゲートを設け、1,022件のマイグレーションにわたるスキーマの整合性を保った。

異常時拒否のレート制限・不正利用対策を構築

Redisによる22個のレートリミッター、Cloudflare Turnstile、リクエストの冪等性、オリジン制限を備え、異常時は拒否する不正利用対策を構築し、呼び出し元を無条件に信頼せずボットや大量アクセスを排除した。

継続的なセキュリティ検査を運用

コード検査、DAST、依存関係・脆弱性検査、SBOM生成、秘密情報検査、SHA固定のActionsと、21件の文書によるセキュリティ監査を組み合わせた継続的なセキュリティ活動を確立した。

リポジトリで定義するインフラ

12,065行のYAMLにわたる19本のAnsible プレイブックと34個のロールで自社インフラを構築し、すべてのプレイを冪等にして、稼働中のホストを宣言した状態へ収束させた。

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。

実行ごとに侵入元の遮断を実証

堅牢化の実行ごとに、予約済みテストアドレスを遮断し、生成されたカーネルルールを読み、必ず実行する後処理で遮断を解除する一連の検証を設け、jailの故障が正常報告ではなく実行失敗になるようにした。

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。

デッドマンズスイッチによるホスト監視を構築

メモリーとディスクが正常な間だけ通知を送るデッドマンズスイッチ監視を構築し、ホストの悪化を通知の停止で知らせるようにした。また、正しく測っていた「available」を「free」と記す変数名を6件発見。464 MBのホストでは両者に1桁の差があった。

Ansibleチェックモードを実態に即した検証に

Ansibleのcommandモジュールが--checkでコマンドを省略しながら成功を返すため、ホストから得ていない値で判断していた6個の検査を修正し、プラットフォーム全体のチェックモードが実態を伝えるようにした。

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。

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。

生成器に981テストと92%カバレッジを適用

警告を失敗扱いにし、分岐カバレッジ92%を下限として生成器に981件のテストを適用。空のファイルから全構築処理を2回実行し、2回目に何も変わらないことを要求して冪等性を検証した。

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。

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。

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。

技術的な取り組みを絞り込む

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

自社の状況に当てはめる

お使いのシステムで「信頼性・バックアップ」がどのように関わっているかをお聞かせください。適したサービスを選び、その業務範囲を定めるお手伝いをします。

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

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/categories/reliability/