Engineer プラットフォーム

Linux・サーバー

27 件の実績

システムの各部分を結び付ける

「Linux・サーバー」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。

Linuxサーバーを守る1,000件のRBACルールを作成

開発者、アプリケーションインスタンス、PostgreSQL、MS SQL、その他のLinuxサーバーに1,000件のRBACルールを実装して不正アクセスを防ぎ、データセキュリティを強化。Ansibleによる自動化として記録した。

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。

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。

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。

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。

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。

3つあったが、どれも機能していなかった防御

一度も機能していなかったSSH総当たり攻撃対策を3件発見・修正。デーモンが1986番で待ち受けるのに22番を監視する遮断jail、上位の広いルールに隠されたレート制限、バイナリを解決できず遮断を一度も実行できなかったアクションを直した。

SSHの24設定項目を検証・堅牢化

SSHの24設定項目を、候補ファイル、統合済み設定、デーモン自身の読み戻しによる3段階で検証するよう堅牢化。読み戻しによって、稼働中のサーバーが24項目のうち2項目を黙って上書きしていたことを検出した。

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。

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。

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。

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。

読み取り専用のホストレポートロールを7個構築

事実、アクセス、git、指標、トラフィック、セキュリティ、プロバイダー資産を、稼働ホストからMarkdownにまとめる7個の読み取り専用レポートロールを構築し、実行で測定していない数値は出力しない規則を設けた。

自社サーバーでコードを管理

Web管理画面を持たず、初期状態で非公開のSoft Serveによる自社gitホスティングを導入。SSHポートを踏み台ホストの背後のループバックに限定し、入口のページを手作業の複製ではなく主サイトのビルド成果物にした。

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。

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。

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。

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。

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「Linux・サーバー」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/linux/