職務経歴書

清水 直也

リードエンジニア／Platform・SRE

基準日 2026.08.30

GitHub https://github.com/43z708

経験ポジション Lead Engineer／Engineering Management　Platform／SRE・Cloud Infrastructure　Kubernetes／IaC・GitOps　Blockchain Validator Infrastructure　Web Application Engineering

職務要約

Webアプリケーション開発からAWS／Kubernetesを中心とするPlatform／SREへ専門性を広げてきました。現職ではシニアエンジニア5名を率い、アップタイムが収益へ直結するバリデータ事業を運用。脱属人化によって組織上の単一障害点を減らすとともに、Kubernetes／ベアメタル基盤の実装、AIによる業務自動化を進めています。並行する業務委託では生成AI基盤のEKS標準化と突発負荷対策を担っています。

代表実績

Kubernetesを中核に3種類のバリデータ基盤を実装

3種類の高可用性ワークロードに対し、AWS EKS／Kubernetesとベアメタルを要件に応じて選択しました。Terraform、Argo CD、Ansibleで構成と変更をコード化し、監視、冗長化、更新、バックアップ、復旧まで一体で設計・実装。約200のバリデータ、約6,400 ETH相当を支える署名基盤も構築しています。

数十チェーンの更新運用を仕組み化し、対応漏れをほぼ解消

5名のチームを率い、複数の連絡経路へ分散していた更新告知をAIエージェントで自動収集・通知する基盤を立案しました。ダブルチェックやGitHub Releases監視も残した多層防御により、情報収集に起因する更新対応の抜け漏れをほぼ解消。外部監査の指摘64件も63件を是正し、残る1件をリスク受容として収束させました。

約1週間を要したTVL集計・報告を毎日の自動投稿へ移行

預かり資産総額（TVL）に必要なデータを決定論的に収集・集計し、結果を毎日自動投稿する仕組みを本番導入しました。これにより、経理担当が1回あたり約1週間を要していた集計・報告業務を定常的な自動処理へ置き換えました。

活かせる経験・強み

技術統制・運用マネジメント

更新告知の自動収集と多層監視で数十チェーンの更新漏れを防止。手順と判断基準を共有し、担当者不在でも継続できる運用へ変えています。

Platform／SRE

EKS／Kubernetesとベアメタルを選び分け、Terraform／Argo CD／AnsibleでIaC・GitOps化。Prometheus／Grafanaの監視から冗長化、更新・復旧まで設計・実装できます。

AI活用・業務自動化

静的サイト運営、記事生成、日報、TVL集計をAIで本番運用へ組み込み、反復作業と品質課題を解消しています。

職務経歴

株式会社Omakase

2023.12 — 現在

契約形態 正社員　　**役割** リードエンジニア

チーム規模 エンジニア5名

シニアエンジニア5名を率い、数十種類のブロックチェーンを扱うバリデータ事業の運用統制、Kubernetes／ベアメタル基盤の設計・実装、AIによる業務自動化を担当しています。事業継続に直結する運用判断から自ら手を動かす基盤構築、業務フローの改善までを一貫して担っています。

バリデータ事業の運用統制・セキュリティ推進

2023.12 — 現在

概要

クライアント更新の遅延が停止と収益減へ直結する環境で、アップタイムを事業の生命線に据え、運用判断とセキュリティ対応を横断して担っています。

背景・課題

バリデータはブロックチェーンネットワークの運用に参加し、安定稼働に応じて報酬を得る仕組みです。クライアントを規定日までに更新できないと停止し、アップタイムの低下がそのまま収益減少につながります。しかし更新・設定変更の告知はDiscord、Telegram、Slack等へ分散しており、人手確認、ダブルチェック、GitHub Releases監視を重ねても、数十チェーンを扱う規模では認知限界による対応漏れが重大な事業リスクとして残っていました。

本人の判断・行動

運用体制

個人の注意力や特定担当者だけが持つ知識を組織上の単一障害点と捉え、手順、監視対象、判断基準、対応履歴を共有・標準化しました。担当者へ調査・実装の裁量を委ねる一方、優先順位、技術判断、レビュー、最終判断はリードが担当。担当者が不在でも業務を継続できるようにし、稼働状況・作業残量とチェーン別採算も要員配置と技術構成の判断へ反映しています。

仕組みによる統制

2026年初頭からAIエージェントによる更新情報の自動収集・通知基盤を立案し、要件整理、担当者への実装指示、成果物レビューを担当しました。ダブルチェックとGitHub Releases監視を残した多層防御とし、Cosmos系では再利用可能な構築テンプレートを整備。監視対象と通知経路の棚卸しを主導し、SOC 2／Vantaの指摘も是正・適用除外・代替統制へ分類して収束させました。

結果

導入後は、更新情報の見落としに起因する対応漏れがほぼ発生しなくなりました。外部監査の指摘64件も63件を是正し、残る1件はリスク受容として承認され、監査対応を収束させました。

技術環境

運用・統制環境

- Prometheus／Grafana／CloudWatch／オンコール
- Discord／Telegram／Slack／GitHub Releases／AIエージェント
- Cosmos validator template／SOC 2 Type II／Vanta／ISO 27001

Kubernetes／ベアメタルによるバリデータ基盤設計・運用

2024.12 — 現在

概要

3種類の高可用性ワークロードについて、要件整理、アーキテクチャ設計、IaC、監視、セキュリティ、更新・復旧までを一貫して担当しました。AWS EKS／Kubernetesを共通基盤としながら、高い処理性能が必要な領域ではベアメタルを採用し、可用性・性能・運用性から実行基盤を選び分けています。

背景・課題

ワークロードごとに、機密データを保護した計算処理、約200のバリデータを支える署名、コンテナを使えない専用サーバー運用という異なる制約がありました。一方、安定稼働には構成変更、監視、機密情報と設定の同

期、更新、復旧を、手作業に依存しない再現可能な運用へ揃える必要がありました。

本人の判断・行動

EKS／Kubernetes標準化

機密データを保護して処理するワークロードをAWS EKS上へ標準化し、周辺リソースもTerraformでIaC化してArgo CDによるGitOpsへ統一しました。変更をPull Requestでレビューできる状態にしたうえで、Prometheusのメトリクスを AMP／Grafanaへ集約し、CloudWatchも含めて監視を標準化。段階的な更新、バックアップ、障害切り分け、復旧までをDay 2運用として実装しました。

高可用・セキュアな署名基盤

約200のバリデータ、約6,400 ETH相当を支える署名基盤をEKS上へ設計しました。機密鍵をアプリケーションから分離して集中管理し、AuroraとS3を正本とする多層防御、設定の自動同期、ローリング更新を実装。多数の署名処理、設定変更、障害時の切り替えを検証してから本番へ移行しました。

ベアメタル基盤のIaC

コンテナを使わず専用サーバーの性能を直接引き出す要件に対し、サーバーリソースをTerraform、OS・ディスク・ミドルウェア・systemd・監視・更新処理をAnsibleでIaC化しました。事前に同期した次期サーバーへ処理を順序立てて切り替える更新手順も実装し、構成の再現性と停止時間の抑制を両立しました。

結果

GitOps・統合監視・バックアップ復旧を備えたEKS基盤を成立させ、運用中に発見したTerraform moduleの修正は開発元へすべて採用されました。ベアメタル基盤では事前に同期した次期サーバーへ切り替えることで、実運用の更新時に再同期による停止をほぼ発生させませんでした。

技術環境

- AWS EKS／Kubernetes／Aurora／S3／ベアメタル
- Terraform／Argo CD／Ansible／systemd
- Prometheus／AMP／Grafana／CloudWatch
- GitOps／Blue-Green更新／バックアップ・復旧

AI駆動の業務自動化・企業サイト刷新

2023.12 — 現在

概要

企業サイト、記事制作、日報、TVL集計に残る反復作業と品質課題を見つけ、AIを単発利用で終わらせず、継続的に動く本番の業務フローとして設計・実装しました。

背景・課題

WordPressはAI支援で記事を継続生成する運用と相性が悪く、日報は手入力のため記録忘れや内容の薄さが発生していました。TVLの集計・報告にも長時間の手作業が残っており、個人の記憶と作業時間へ依存する業務フローそのものを見直す必要がありました。

本人の判断・行動

サイトと記事制作

WordPressを廃止してAstro／TypeScriptの静的サイトへ刷新し、日本語・英語のコンテンツをMarkdownとしてGit管理へ移行しました。記事制作では調査、構成、執筆、レビューを分担するAIエージェントを組み合わせ、最終確認は人が行う生成工程を構築しました。

日報と集計

GitHub、Slack、Notion等の業務履歴から日報下書きを生成・記録するBotを実装し、各自は確認・加筆へ集中する運用に変更しました。TVLは必要データを決定論的に収集・集計し、結果を毎日自動投稿する仕組みを本番導入。人手による長時間の集計・報告を定常的な自動処理へ置き換えました。

結果

サイトと記事をMarkdown／Git中心で更新できる基盤へ移し、AI生成と人のレビューを両立しました。日報は業務履歴から下書きされるため、チームの日常的な入力工数をほぼ不要にしています。TVLでは、経理担当が1回あたり約1週間を要していた集計・報告を、毎日の自動収集・投稿へ置き換えました。

技術環境

技術・業務環境

- Astro／TypeScript／Markdown／Git
- Codex／Claude／AIエージェントのオーケストレーション
- GitHub／Slack／Notion／Python

本業と並行する業務委託

生成AI基盤案件（社名非公開）

2025.12 — 現在

契約形態 業務委託　　**役割** Infrastructure／SRE領域の設計・実装

PMから提示された抽象的な依頼を技術要件へ具体化し、生成AI基盤の調査、設計、実装、検証を一貫して担当しています。PMは各案件の要件・受入の最終判断とレビューを担います。

生成AIアプリケーション向けEKS標準基盤の再設計・横展開

2025.12 — 現在

概要

依頼内容は『複数のDify Enterprise案件で使えるKubernetes環境を自社テンプレートとして作る』というものでした。

背景・課題

引き継ぎ時点は既存CDKによるPoC実装でした。複数案件でDay 2運用を続けるには、案件ごとの差分と変更影響をレビュー可能な形で管理する必要がありました。

本人の判断・行動

IaCとsecret管理

長期運用と複数案件への展開では、宣言差分、remote state／locking、PRレビューを重視してTerraformを採用しました。secretは導入当初からAWS Secrets Managerを正本とし、External Secrets OperatorでKubernetesへ同期することで、Helm valuesへ秘匿情報を残さない構成にしました。

GitOpsと運用設計

自社管理ChartならArgo CDを優先できますが、Difyは外部ベンダー管理Chartを継続更新するため、将来追加されるHookのライフサイクルを制御できません。過去にpre-install Jobがupgrade時にもPreSyncとして実行され、finalizer滞留でApplication全体が止まった経験から、Helm本来のinstall／upgrade semanticsを維持するFlux HelmReleaseを採用しました。自社固有のDB初期化は外部Chartから分離し、独立したKustomizationと冪等Jobとして管理しています。

可観測性と変更統制

Prometheus stack／Grafanaとdashboard JSON／PromQLをGit管理し、毎朝9時のTerraform planでdriftを検知。staging適用の承認とE2E automationも組み込み、安全に更新できる運用経路を整えました。

結果

3環境への展開

社内環境へ適用し、別の受注案件2件でも基盤として活用されています。前提が揃う1件では設定値と手順書により約0.5人月相当の構築を1日へ短縮し、もう1件ではCDK実運用案件のTerraform移行基盤として利用されています。

技術環境

- AWS EKS／Terraform

- Flux HelmRelease／Kustomization／External Secrets Operator
- Prometheus／Grafana／GitHub Actions

突発的な1,000人負荷に備える生成AI基盤の性能・コスト最適化

2025.12 — 現在

概要

『普段はコストを抑えた構成とし、災害時などの突発利用にはAuto Scalingで追従する。最大1,000人の同時負荷を測定し、成立性をレポートする』という要求を試験仕様へ具体化し、ボトルネックの調査から改善実装、再検証まで担当しました。

背景・課題

LLMの生成待ちを含む長時間処理では、短時間に多数の要求が入ると処理待ちが先に積み上がり、既存のインフラ指標から必要容量を判断するまでに遅れが生じました。

DB接続

同時に、ワーカーが長時間処理の間も接続を保持し、PostgreSQL backend接続数が負荷上限の一つになっていました。

本人の判断・行動

Auto Scaling

CPU target trackingは需要の先行指標にならず、ALB RequestCountもCloudWatch反映とタスク起動を待つため棄却しました。代わりにCeleryのactive／reservedとRedisのqueuedを読むサイドカーを実装し、需要を10秒ごとにcustom metric化。需要低下時だけ直近300秒の最大値を保持するpeak-holdをExactCapacityへ反映し、処理待ちから必要容量を直接決める方式としました。

DB接続

RDS Proxyを実機検証したところ、16KB超SQLによるsession pinningと多数のidle backendが残り、Difyの大容量insertでは接続多重化が効きにくいと判断しました。そこでSQLサイズに依存せずtransaction終了時に接続を再利用するPgBouncer transaction modeを選定し、API／workerのlocalhost sidecarとして実装しました。

結果

1,000人規模の突発負荷

外部LLM／APIを実測した応答特性のmockへ置き換え、1,000人・10分rampの総合負荷試験を実施しました。主要APIが目標水準へ収束するまでの時間を11〜12分から約6分へ短縮し、負荷中の早すぎるscale-downも防げることを確認しました。

DB接続数を約65%圧縮

PgBouncer導入前後を同じ負荷条件で比較し、PostgreSQL backend接続数を5,473から1,928へ約65%圧縮しました。

平時コストと拡張性の両立

Aurora min 16 ACU×2を含む高負荷構成を平時から固定すると月額5,000ドル超（検証時の構成値とAWS単価による試算）となるのに対し、通常時はシステム全体を月額約1,450〜1,750ドルに抑えながら1,000人規模へ拡張できる構成としました。

技術環境

- AWS ECS／ALB／CloudWatch／Aurora PostgreSQL
- Celery／Redis／custom metrics／Step Scaling
- RDS Proxy／PgBouncer transaction mode／負荷試験

過去案件（案件単位）

2020.03 — 2024.04

契約形態 案件単位　　**役割** Webアプリケーションエンジニア

過去案件

生成AI議事録ツール

2023.06 — 2023.10

PoC／個人開発（単独担当）。企画から実装・運用まで担い、実際の利用では生成後の確認・修正作業を約60〜90分から約10分へ短縮しました。

コミュニティ向けBot

2023.02 — 2025.11

コミュニティ運営ツールの設計・開発・運用保守を担当しました。

大規模BtoB人事サービス

2022.06 — 2024.04

BtoB SaaS／業務委託（フロントエンド担当）。CTOの段階移行方針に対し、Vue 2／Reactを共存できるqiankunを選定しました。

ナビゲーションのReact置換まで実装し、契約終了時に後任へ引き継ぎました。

URL共有サービス

2021.08 — 2022.04

Webサービス開発（フロントエンド・バックエンド担当）。URL共有サービスの設計から相互レビューまで担当しました。

業務SaaSのリニューアル

2021.03 — 2021.07

SaaSリニューアル（フロントエンド・バックエンド担当）。機能開発とテストを担当しました。

業務Webアプリ

2020.11 — 2021.07

業務システム開発（API・管理画面担当）。詳細設計、API・管理画面開発、テスト、環境構築を担当しました。

オンラインイベント基盤

2020.03 — 2021.02

個人学習PoC／個人開発（単独担当）。WordPress中心だった時期にWebアプリケーション開発を学ぶため、設計・開発・デプロイ・試用まで行いました。

技術スタック

Cloud／Platform

- AWS（EKS、ECS、Aurora、AMP、S3、Secrets Manager）
- Terraform／Ansible
- Argo CD／Flux／Helm／Kustomize

Observability／Reliability

- Prometheus／Grafana／CloudWatch
- 負荷試験／SLO設計／Auto Scaling
- GitHub Actions／drift検知／review gate

Application／AI

- TypeScript／React／Vue
- Python
- 生成AIを用いた業務自動化

自己PR

私の強みは、事業上の優先順位を見極め、技術課題をチームが継続運用できる仕組みまで整えることです。現職ではシニアエンジニア5名のリードとして、担当者へ調査・実装の裁量を委ねつつ、難易度や事業影響を踏まえた判断とレビューを担当。判断だけでなく、必要に応じて自ら実装まで担います。手順・監視・判断基準を共有し、担当者不在にも耐える運用へ変えてきました。

技術面では、AWS EKS／Kubernetesとベアメタルを要件に応じて選び、IaC／GitOps、監視、冗長化、更新、バックアップ、復旧までを一体で設計・実装できます。並行する業務委託では、生成AI向けEKS標準基盤の再設計と展開、最大1,000人の突発利用を想定したAuto ScalingやDB接続最適化も担当しています。

また、組織内の反復作業や品質低下を発見し、AIと自動化で業務フローそのものを再設計することを得意としています。静的サイトへの刷新、AIエージェントによる記事生成、日報、更新情報・TVLの収集投稿を本番運用へ組み込みました。要件が曖昧な段階から運用定着までをつなぎ、自ら実装しながらチームの意思決定を前へ進められる点が、私の提供できる価値です。