Nous exploitons les plateformes **sur lesquelles vos logiciels sont déployés** — reconstruites à partir du code, surveillées et documentées.

Cloud, infrastructure sous forme de code, CI/CD, conteneurs, supervision, sauvegardes, réseaux et accès — depuis Copenhague.



company
Engineer
Plateforme

Une infrastructure que vous pouvez reconstruire depuis un dépôt

+45 50 10 61 84 · Copenhague, Danemark

Créé avec soin et amour par D.

Infrastructure cloud sous forme de code, déploiement automatisé et préparation de la reprise pour les systèmes dont votre entreprise dépend. Les comptes rendus décrivent la situation initiale, le travail effectué et le résultat.

Commencez par un parcours de déploiement ou un risque d'exploitation. Nous pouvons convenir d'une revue ou d'une mise en œuvre avec un plan documenté de déploiement, de retour arrière et de responsabilités, pour que votre équipe sache quoi exploiter et comment.

Travaux d'infrastructure · Tarifs et modalités de collaboration · Parlons de votre projet

Partez du résultat recherché

Dites-nous ce que votre entreprise doit accomplir. Nous relions logiciels, données et infrastructure autour des personnes qui les utilisent.

Découvrir nos prestations Contactez-nous →

Découvrez notre façon de travailler

Trouvez la prestation qui correspond à votre objectif et découvrez, à travers nos réalisations, ce qui a été livré et comment.

Études de cas Contactez-nous →

Découvrez les réalisations **dans leur contexte** — les décisions, la réalisation et les résultats rapportés.

Voir l'ensemble du catalogue

Un seul partenaire pour **10 prestations** du premier serveur à l'alerte de 3 heures du matin.

Parcourir le catalogue

{ Cloud, infrastructure et DevOps }

L'endroit où fonctionne votre logiciel

Infrastructure cloud et migration

Des serveurs dimensionnés selon vos usages réels, migrés vers le cloud ou hors du cloud, avec une estimation préalable de la facture mensuelle.

Second serveur provisionné à partir du code

Automatisé le provisionnement d'un second serveur chez un second fournisseur cloud, en créant le pare-feu avant la machine pour qu'elle soit protégée dès sa création, et configuré les pare-feu des deux fournisseurs directement par leurs API REST pour éviter une collection tierce.

Conteneurs exploités avec Podman et Quadlet

Déployé une couche de conteneurs Podman et Quadlet sous systemd plutôt que Docker, car la publication de ports sur le réseau bridge par défaut de Docker peut contourner les règles UFW, et confié les déploiements à un utilisateur non privilégié autorisé à une seule commande fixe plutôt qu'à root.

Services publics sur un petit serveur

Hébergé les services publics de l'entreprise sur un hôte monocœur de 512 MB — forge git, serveur web pour sept domaines, Tor, deux serveurs de protocoles alternatifs et blocage des intrusions — avec 464 MB de mémoire utilisable comme contrainte de conception ; l'automatisation des sauvegardes constitue une mise en œuvre distincte.

Voir les 9 réalisations

Notre façon de travailler

Comprendre d'abord

Nous étudions le problème avant de proposer une solution. Le cahier des charges que vous apportez est le début de cette conversation, pas sa conclusion.

Construire pour durer

La durabilité plutôt que la rapidité, la clarté plutôt que les astuces. Le système doit rester exploitable pour les personnes qui en hériteront.

Assumer la responsabilité

Toute exploitation ou assistance continue fait l'objet d'un accord dans le cadre de la mission. La transmission précise les responsabilités et la voie à suivre pour tout contact ultérieur.

{ Cloud, infrastructure et DevOps }

Des serveurs que vous pouvez reconstruire à partir d'une configuration versionnée

Infrastructure as Code

Toute la configuration décrite dans des fichiers, pour la recréer à l'identique sans qu'elle ne dépende de la mémoire d'une seule personne.

Miroir Tor onion à une adresse lisible

Publié un miroir Tor onion du site à une adresse lisible commençant par engineer, avec une clé générée hors de l'hôte et jamais présente dans ce dépôt ni sur un portable, en conservant ce quatrième point d'entrée comme le seul des quatre à ne jamais être compté.

Hook CI de forge répété : 2 défauts détectés

Répété l'exécution du hook CI côté forge et trouvé deux défauts impossibles à atteindre par la seule lecture : un mécanisme de repli fournissant un argument impossible à résoudre à l'entrée du hook et une variable d'environnement git suivant le contrôle dans le checkout, faisant échouer 19 tests.

Secrets d'exploitation séparés par hôte

Séparé les quatre secrets d'exploitation par hôte après avoir établi que deux hôtes partageant un seul signal de vie produisent moins d'alertes que deux signaux distincts, et qu'une phrase secrète de sauvegarde commune peut exposer les sauvegardes des deux hôtes, même dans des dépôts distincts, si une personne l'obtient et peut lire les données chiffrées et les fichiers de clés des deux dépôts.

Voir les 17 réalisations

{ Cloud, infrastructure et DevOps }

Livrer des changements sans agitation

DevOps et automatisation CI/CD

Des changements testés, publiés et réversibles, pour livrer aux clients un mardi après-midi plutôt qu'un vendredi soir tard.

Une norme d'historique pour 4 dépôts

Appliqué à quatre dépôts une même norme d'historique — commits conventionnels, sans emojis ni mention d'attribution, imposés par un hook de message de commit — et 48 documents d'instructions régissant le travail.

Connaître la licence de tout ce que vous livrez

Audité à chaque compilation la compatibilité des licences de 644 crates Rust, puis vérifié le déclenchement du contrôle en modifiant la licence d'une crate et en déplaçant la révision épinglée du cœur sans régénération.

Moitié jusque-là inutilisée du cœur exploitée

Exploité la moitié du cœur de messagerie jusque-là inutilisée — transfert des sauvegardes, messages éphémères, modification et renvoi de messages, invitations vérifiées, proxys et politique de chiffrement — en exécutant chaque test contre la vraie bibliothèque, sans simulations.

Voir les 43 réalisations

{ Cloud, infrastructure et DevOps }

Un logiciel qui fonctionne partout de la même façon

Conteneurisation et orchestration

Un logiciel conditionné pour se comporter à l'identique sur un portable, un serveur de test et en production, et monter en charge lors des semaines chargées.

Conteneurs exploités avec Podman et Quadlet

Déployé une couche de conteneurs Podman et Quadlet sous systemd plutôt que Docker, car la publication de ports sur le réseau bridge par défaut de Docker peut contourner les règles UFW, et confié les déploiements à un utilisateur non privilégié autorisé à une seule commande fixe plutôt qu'à root.

Exposition des bacs à sable systemd réduite

Réduit l'exposition des bacs à sable systemd de chaque unité installée par la plateforme — service de signal de vie de 9,6 UNSAFE à 1,5, forge git exposée à Internet de 8,3 EXPOSED à 1,5 — et ajouté un contrôle du parseur lors de la convergence après avoir découvert une directive mal orthographiée silencieusement ignorée dans trois modèles d'unités.

Pipelines CI/CD GitHub Actions

Conçu des pipelines GitHub Actions CI/CD avec une image frontend de production distroless et une promotion entre plusieurs environnements.

Voir les 5 réalisations

{ Cloud, infrastructure et DevOps }

Être informé avant vos clients

Fiabilité des services et supervision

Une supervision et des alertes qui signalent les incidents tant qu'ils restent mineurs, avec les chiffres qui montrent à quelle fréquence ils le restent.

Six collecteurs indiquent les entrées ignorées

Fait expliciter à six collecteurs de reporting la partie de leurs entrées jamais lue — fichiers non ouverts, sondes non installées, lignes du journal non reconnues — et ajusté les seuils sur 116 échantillons mesurés, après avoir établi qu'une collecte aveugle et un hôte sain produisaient la même page.

13 adresses mesurées sans traçage

Mesuré treize adresses uniquement à partir du journal d'accès du serveur web — sans cookie, traqueur ni service tiers — en masquant le réseau du client à l'écriture de chaque ligne, puis en intégrant les résultats dans une archive permanente de 21 comptes quotidiens et 20 facettes mensuelles.

Un pipeline d'événements borné pour une application macOS

Borné les tampons d'événements et la journalisation après un incident macOS enregistré : croissance de 41 MB par seconde et 111 GB de pages compressées sur une machine de 36 GB ; une exécution complète des tests consignée a réduit les journaux du cœur de 610 996 à 1 411 lignes.

Voir les 23 réalisations

{ Cloud, infrastructure et DevOps }

Tout retrouver après une mauvaise journée

Sauvegarde et reprise après sinistre

Des sauvegardes testées plutôt que supposées fiables, et un plan écrit pour le matin où un serveur ne redémarre pas.

Secrets d'exploitation séparés par hôte

Séparé les quatre secrets d'exploitation par hôte après avoir établi que deux hôtes partageant un seul signal de vie produisent moins d'alertes que deux signaux distincts, et qu'une phrase secrète de sauvegarde commune peut exposer les sauvegardes des deux hôtes, même dans des dépôts distincts, si une personne l'obtient et peut lire les données chiffrées et les fichiers de clés des deux dépôts.

Sauvegardes automatisées et méthode de restauration explicite

Mis en place l'automatisation des sauvegardes restic avec conservation, contrôles d'intégrité et procédure mensuelle de test de restauration ; documenté l'activation du stockage et la vérification d'une restauration réelle comme des étapes d'exploitation distinctes.

Sauvegardes et reprise sous forme de code

Configuré la conservation des sauvegardes sous forme d'Infrastructure as Code, puis évalué les possibilités de récupération et documenté la restauration, en explicitant les lacunes restantes avant qu'un incident ne les révèle.

Voir les 5 réalisations

{ Cloud, infrastructure et DevOps }

Faire fonctionner l'essentiel

Administration système

L'entretien quotidien des serveurs, de la messagerie, des comptes et des mises à jour, pour que les outils indispensables restent sans histoire.

Miroir Tor onion à une adresse lisible

Publié un miroir Tor onion du site à une adresse lisible commençant par engineer, avec une clé générée hors de l'hôte et jamais présente dans ce dépôt ni sur un portable, en conservant ce quatrième point d'entrée comme le seul des quatre à ne jamais être compté.

Six collecteurs indiquent les entrées ignorées

Fait expliciter à six collecteurs de reporting la partie de leurs entrées jamais lue — fichiers non ouverts, sondes non installées, lignes du journal non reconnues — et ajusté les seuils sur 116 échantillons mesurés, après avoir établi qu'une collecte aveugle et un hôte sain produisaient la même page.

Hook CI de forge répété : 2 défauts détectés

Répété l'exécution du hook CI côté forge et trouvé deux défauts impossibles à atteindre par la seule lecture : un mécanisme de repli fournissant un argument impossible à résoudre à l'entrée du hook et une variable d'environnement git suivant le contrôle dans le checkout, faisant échouer 19 tests.

Voir les 28 réalisations

{ Cloud, infrastructure et DevOps }

Relier vos bureaux et vos équipes

Réseaux et configuration de VPN

Du Wi-Fi, des pare-feu et des accès à distance configurés pour travailler de partout, en laissant entrer uniquement les personnes autorisées.

Second serveur provisionné à partir du code

Automatisé le provisionnement d'un second serveur chez un second fournisseur cloud, en créant le pare-feu avant la machine pour qu'elle soit protégée dès sa création, et configuré les pare-feu des deux fournisseurs directement par leurs API REST pour éviter une collection tierce.

Zone DNS de 20 enregistrements en code

Réconcilié déclarativement une zone DNS de 20 enregistrements avec l'API Cloudflare, avec des points d'entrée distincts pour l'audit et l'export BIND, puis désactivé le proxy CDN après sa mise en place pour des raisons de confidentialité.

Règles de pare-feu vérifiées par position

Vérifié les règles du pare-feu selon leur position plutôt que leur seule présence, en lisant la liste numérotée et la chaîne active de filtrage des paquets : une règle existante n'est pas forcément atteinte par le moindre paquet.

Voir les 11 réalisations

{ Cloud, infrastructure et DevOps }

Décider qui peut accéder à quoi

Sécurité et gestion des accès

Des comptes, mots de passe et droits attribués selon les besoins exacts de chaque poste, puis retirés le jour du départ.

Miroir Tor onion à une adresse lisible

Publié un miroir Tor onion du site à une adresse lisible commençant par engineer, avec une clé générée hors de l'hôte et jamais présente dans ce dépôt ni sur un portable, en conservant ce quatrième point d'entrée comme le seul des quatre à ne jamais être compté.

13 adresses mesurées sans traçage

Mesuré treize adresses uniquement à partir du journal d'accès du serveur web — sans cookie, traqueur ni service tiers — en masquant le réseau du client à l'écriture de chaque ligne, puis en intégrant les résultats dans une archive permanente de 21 comptes quotidiens et 20 facettes mensuelles.

Connaître la licence de tout ce que vous livrez

Audité à chaque compilation la compatibilité des licences de 644 crates Rust, puis vérifié le déclenchement du contrôle en modifiant la licence d'une crate et en déplaçant la révision épinglée du cœur sans régénération.

Voir les 31 réalisations

{ Cloud, infrastructure et DevOps }

Limiter votre IA à ce qu'elle doit faire

Sécurité des agents d'IA

Les accès de chaque agent recensés et limités au nécessaire, chaque action consignée hors de sa portée et chaque protection testée plutôt que supposée fiable.

Secrets d'exploitation séparés par hôte

Séparé les quatre secrets d'exploitation par hôte après avoir établi que deux hôtes partageant un seul signal de vie produisent moins d'alertes que deux signaux distincts, et qu'une phrase secrète de sauvegarde commune peut exposer les sauvegardes des deux hôtes, même dans des dépôts distincts, si une personne l'obtient et peut lire les données chiffrées et les fichiers de clés des deux dépôts.

Règle de périmètre après une fuite de secrets

Écrit une règle de périmètre dans le dépôt après qu'une restructuration y a introduit l'inventaire, les autorisations de pare-feu et les textes d'une autre entreprise, en conservant les éléments mis en quarantaine dans le champ de la détection des secrets.

Vérification des secrets en une seconde

Ajouté un play de vérification préalable exécutant en une seconde environ le même code que la convergence, avec les secrets locaux de l'opérateur, après qu'une exécution de production partiellement appliquée s'est arrêtée à la neuvième tâche, les paramètres de swap étant déjà écrits sur l'hôte en service.

Voir les 9 réalisations

Nous travaillons avec **les plateformes** sur lesquelles votre entreprise s'appuie déjà.



Apple Microsoft DigitalOcean AWS Google

Voir l'ensemble des technologies

Audit de sécurité des agents IA dans le périmètre convenu

Cartographiés

Un inventaire convenu des agents, des identifiants d'accès et des accès aux données compris dans le périmètre, conservé avec votre configuration.

Verrouillés

Chaque agent dispose de sa propre identité et du minimum de droits nécessaire ; le texte qu'il lit est traité comme des données, jamais comme des ordres.

Consignés

Chaque action est journalisée en indiquant la personne pour laquelle elle est effectuée, à un endroit que l'agent ne peut pas modifier, avec des alertes en cas d'anomalie.

Voir comment les agents sont sécurisés Parlons de vos agents →

Dites-nous ce que vous construisez.

Pas de formulaire, pas de tunnel de conversion, pas de séquence commerciale — simplement un courriel à une personne qui le lira.

Engager la conversation

Cette adresse ne présente que les travaux d'infrastructure. L'entreprise développe aussi des logiciels, travaille sur les données et conseille des équipes : ces activités sont sur engineer.company.
Voir tout ce que fait Engineer ApS →

Commencez par une amélioration ciblée

Un processus de travail, un flux de données ou un système à améliorer peut constituer une première mission utile. Nous pouvons en définir le périmètre ensemble.

Consulter les tarifs Contactez-nous →

Préparez le premier échange

Indiquez-nous le résultat attendu, ce qui est déjà en place et le calendrier souhaité. Nous pourrons alors discuter de la prochaine étape.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Côté serveur, nous utilisons **Linux en logiciel libre** — sans licence à payer ni rien de caché.

Ubuntu Debian

Comment nous exploitons les serveurs

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://platform.engineer.company/fr/