Engineer Plateforme

Portfolio

Des prestations d'ingénierie illustrées par des travaux réalisés.

Engineer ApS a été immatriculée en 2025. La plupart des réalisations présentées ici ont été livrées auparavant par notre fondatrice au sein d'équipes clientes ; certaines ont été réalisées par l'entreprise elle-même, et d'autres depuis dans le cadre d'un emploi salarié. Les mêmes mains et les mêmes exigences — nous précisons chaque contexte plutôt que de les confondre.

Trouvez une expérience utile à mettre en pratique

Utilisez ce parcours pour trouver une expérience pertinente pour votre projet.

Découvrir nos prestations Contactez-nous →

Lisez les décisions derrière les réalisations

Les entrées décrivent des travaux livrés dans différentes prestations. Lisez une analyse pour découvrir la situation et les décisions qui les sous-tendent.

Études de cas Contactez-nous →

Parcourir par catégorie

INGÉNIERIE LOGICIELLE

Ingénierie backend · Ingénierie frontend · Développement full-stack · API et intégration · Développement web · Tests et assurance qualité · Optimisation des performances · Python · Internationalisation

DONNÉES ET BASES DE DONNÉES

Bases de données · PostgreSQL · SQL · Ingénierie des données · Pipelines de données (ETL/ELT) · Analyse de données · Gouvernance des données · GIS et données géospatiales

CLOUD, INFRASTRUCTURE ET DEVOPS

Cloud · Conteneurs (Docker/Kubernetes) · DevOps

Automatisation et CI/CD
Infrastructure
Linux et serveurs
Administration système
Réseaux et VPN
Supervision et observabilité
Fiabilité et sauvegardes
Sécurité
Architecture de plateformes
Migrations et modernisation

CONSEIL ET DIRECTION

Direction technique
Encadrement d'équipes
Méthodes agiles et Scrum
Gestion de projets
Produit et besoins métier
Parties prenantes et reporting
Documentation
Architecture de solutions
Marque et marketing

Tout ce qui suit **a été réalisé** — pas au stade de la proposition ou de la planification.

Chaque entrée décrit ce que la réalisation a changé et renvoie aux prestations qu'elle illustre.

Ce que nous proposons Nos tarifs →

{ Cloud, infrastructure et DevOps }

Infrastructure cloud et migration

L'endroit où fonctionne votre logiciel

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.

Une infrastructure définie dans un dépôt

Conçu l'infrastructure de l'entreprise avec 19 playbooks Ansible et 34 rôles totalisant 12 065 lignes de YAML, pour faire converger un hôte en service vers un état déclaré, chaque play étant idempotent.

Infrastructure centrale construite de zéro

Parmi les premières personnes recrutées, conçu et construit de zéro toute l'infrastructure centrale et les processus associés d'une startup SaaS d'énergie verte, posant les bases d'une croissance rapide.

Déploiements Azure pris en charge de bout en bout

Pris en charge les déploiements de bout en bout sur Azure, en gérant les livraisons dans les environnements de développement, de préproduction et de production.

Infrastructure Azure sous forme de code

Provisionné l'infrastructure Azure sous forme de code avec Bicep — Container Apps, PostgreSQL Flexible Server, Front Door/WAF et réseau — dans les environnements de développement, de préproduction et de production.

Infrastructure simplifiée au-delà d'AWS

Refondu l'infrastructure d'un client international en supprimant les services inutilisés, en ajustant la capacité et en déplaçant les charges de travail hors d'AWS. La fondatrice fait état de coûts cloud divisés par 10, sans perte de productivité.

20 applications Docker sur Google Cloud et AWS

Déployé et maintenu 20 applications conteneurisées avec Docker, résolu des incidents avec Podman et Kubernetes et géré des applications R sur Google Cloud et AWS.



{ Cloud, infrastructure et DevOps }

Infrastructure as Code

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

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.

Votre code sur votre propre serveur

Déployé la forge git de l'entreprise avec Soft Serve, privée par défaut, sans interface web et avec un port SSH lié à l'interface de bouclage derrière un hôte relais, puis fait de sa page d'accueil un artefact de compilation du site principal plutôt qu'une copie maintenue à la main.

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.

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.

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.

Mode de vérification Ansible fiabilisé

Fiabilisé le mode de vérification sur toute la plateforme après avoir trouvé six sondes prenant des décisions sur une valeur que l'hôte n'avait jamais fournie : le module command d'Ansible annonce une réussite sous --check tout en sautant entièrement la commande.

Supervision par absence de signal de vie

Conçu une supervision par absence de signal de vie qui n'émet un signal que si mémoire et disque sont sains, pour qu'un hôte dégradé déclenche une alerte en se taisant, et corrigé six noms de variables indiquant « free » alors que le contrôle mesurait bien « available », des valeurs d'un ordre de grandeur différent sur une machine de 464 MB.

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.

SSH durci avec 24 directives validées

Durci SSH avec 24 directives vérifiées et une validation en trois étapes — fichier candidat, configuration assemblée, puis relecture par le démon — après que cette dernière a révélé que le serveur en service remplaçait silencieusement deux des vingt-quatre directives.

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.

1 000 règles RBAC pour sécuriser Linux

Renforcé la sécurité des données en mettant en place 1 000 règles RBAC pour les développeurs, les instances applicatives, PostgreSQL, MS SQL et d'autres serveurs Linux afin de prévenir les accès non autorisés, avec une documentation et une automatisation Ansible.

Cette prestation intervient aussi dans des réalisations répertoriées ailleurs sur cette page :

Second serveur provisionné à partir du code · Services publics sur un petit serveur · Une infrastructure définie dans un dépôt · Infrastructure Azure sous forme de code

{ Cloud, infrastructure et DevOps }

DevOps et automatisation CI/CD

Livrer des changements sans agitation

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.

Suite de tests ramenée à 8,9 secondes

Fait passer la suite de six tests dépassant soixante secondes à 135 tests réussis en 8,9 secondes, en profilant le thread principal et en supprimant les deux appels où il restait bloqué pendant 3 989 des 4 017 échantillons.

Parseur vérifiant chaque appel FFI

Écrit un parseur lisant le véritable en-tête C de 7 308 lignes et vérifiant chaque site d'appel, chaque constante d'énumération et le caractère final de toute classe propriétaire de pointeurs, après qu'un en-tête de substitution manuscrit a laissé compiler et lier des appels à trois fonctions supprimées, puis provoqué un plantage.

Divergences entre hooks et contrôle qualité

Découvert que les hooks de commit et le contrôle qualité exécutaient des vérifications différentes alors qu'un document les disait identiques, en comparant les deux listes dans un test : les cinq contrôles seulement lancés à la main étaient ceux qui lisaient les textes du CV.

Tests des contrôleurs eux-mêmes

Écrit des tests pour les contrôleurs après avoir établi qu'un contrôleur alimenté uniquement en entrées correctes peut finir par déclarer une réussite sans rien lire : introduction d'une faute pour vérifier sa détection et lecture d'une plage d'identifiants dans la base plutôt que dans une constante du test.

Chaque règle du linter traitée comme une erreur

Activé comme erreurs toutes les règles du linter Python, corrigé 1 815 constats pour atteindre zéro et documenté la justification de chaque rare exemption, dont deux soutenues par un contrôle plutôt que par un commentaire.

981 tests et 92 % de couverture

Vérifié le générateur avec 981 cas de test et un seuil de couverture des branches de 92 %, en traitant les avertissements comme des échecs, et vérifié l'idempotence en exécutant deux fois tout le pipeline depuis un fichier vide, sans changement autorisé au second passage.

Contrôle qualité : 10 242 lignes vérifiées

Soumis 10 242 lignes de JavaScript des contrôles qualité à un formateur et à un linter, après avoir établi qu'il s'agissait du plus gros ensemble de code du dépôt et du seul que rien ne contrôlait, en corrigeant 13 constats sans en masquer aucun.

Dates identiques du sitemap corrigées

Corrigé un sitemap dont 172 des 176 URL partageaient la même date de modification, en utilisant l'historique git après avoir établi que l'export réécrivait chaque fichier à chaque exécution.

Miroirs du site sur Gemini et Gopher

Construit les sorties Gemini et Gopher depuis la même source web ; la réalisation consigne 777 documents par format à ce stade et zéro octet HTML modifié lors de leur ajout.

Des vérifications qui ne prennent plus une demi-heure

Réduit les vérifications qualité du site pilotées par navigateur de 1 636 à 615 secondes, en ordonnançant les contrôles du plus long au plus court dans un pool de quatre processus, après avoir mesuré un coût de 320 secondes en ordre alphabétique contre 224.

Contrôle avant commit avec 22 linters

Conçu un contrôle obligatoire avant commit composé de 22 linters tenant sur une ligne et de cinq nécessitant un paragraphe, sans niveau d'avertissement ni suppression locale autorisée, couvrant HTML, CSS, JavaScript, Python, YAML, Markdown, shell, liens, orthographe, secrets et typographie.

Programme continu d'analyse de sécurité

Mis en place un programme continu de sécurité — analyse du code, DAST, contrôles des dépendances et des vulnérabilités, génération de SBOM, détection de secrets et actions épinglées par SHA — accompagné de 21 audits de sécurité écrits.

1 022 migrations contrôlées contre les écarts

Maintenu la cohérence du schéma au fil de 1 022 migrations avec un contrôle CI construisant la base de deux façons — installation neuve, puis installation suivie de toutes les migrations — et échouant en cas de différence.

Surveillance des erreurs et traçage internes

Conçu en interne la surveillance des erreurs et le traçage OpenTelemetry plutôt que de les acheter — assainissement des charges utiles, détection des pics et régressions, symbolication et signal de vie synthétique — accessibles dans 11 vues d'exploitation.

Moteur de 268 contrôles pour le dépôt

Conçu le moteur de garde-fous du dépôt — 268 contrôles de commit enregistrés, 277 règles de linting et 15 règles ESLint personnalisées — ainsi que 146 tests de ces garde-fous, pour que les exigences soient appliquées par la compilation et non par la seule revue.

Suite de tests Go, SQL et navigateur

Conçu une suite de tests automatisés en couches — 981 tests Go, 543 spécifications frontend et navigateur, 494 tests de comportement SQL — avec tests par mutation, tests fondés sur les propriétés et contrôle obligatoire de l'accessibilité.

Collaboration d'équipe automatisée

Automatisé la collaboration, la gestion des mots de passe, des tâches et du temps, et construit un système semi-automatique de présentation des projets pour accroître la productivité de l'équipe.

Zéro avertissement dans six langages

Fixé un niveau d'exigence de zéro avertissement dans six langages — Go, TypeScript, SQL, Python, Shell et Markdown — appliqué par des hooks pre-commit.

578 cibles d'automatisation go-task

Créé 578 cibles d'automatisation go-task couvrant les modes de développement natif, Docker et HTTPS, les contrôles de code, les tests, la base de données et le déploiement.

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.

Contrat d'API généré à partir de la base

Généré le contrat d'API à partir de la base de données — OpenAPI, client TypeScript typé de 44 076 lignes, 61 gestionnaires de simulation et limites appliquées par l'interface — avec un contrôle à chaque étape qui échoue en cas de divergence.

Traitement automatisé : Python et PL/pgSQL

Automatisé les traitements de données avec des scripts Shell, PL/pgSQL, Python et Transact-SQL pour accroître la productivité et l'efficacité.

4 000 h gagnées avec GitHub et GitLab CI/CD

Simplifié les processus d'analyse de données et de développement logiciel et économisé 4 000 heures en introduisant des pratiques CI/CD avec GitHub, GitLab, Bash et Python.

CI/CD simplifiée : 4 000 heures gagnées

Simplifié les processus CI/CD et économisé 4 000 heures grâce à l'automatisation des pipelines de développement logiciel.

SSL/TLS automatisé pour 100 applications Docker

Automatisé la création de certificats SSL/TLS pour 100 applications Docker afin de sécuriser les connexions sur des hôtes Ubuntu Linux.

Processus repensés : 8 000 heures gagnées

Refondu les processus internes et économisé 8 000 heures en améliorant l'architecture logicielle, les systèmes et la planification.

Risques réduits avec Grafana et Prometheus

Réduit les risques d'exploitation en mettant en place un tableau de bord de supervision avec Grafana et Prometheus pour améliorer la fiabilité du système.

100 sauvegardes critiques automatisées : Barman

Automatisé 100 sauvegardes critiques avec Barman, Google Cloud, Bash et Python pour préserver l'intégrité des données dans les bases.

Livraison de 20 pipelines GIS automatisée

Automatisé la livraison de 20 pipelines de données GIS et processus ETL de données applicatives, en simplifiant l'automatisation de l'infrastructure et le reporting.

Déploiement SaaS GIS automatisé : GitHub Actions

Automatisé le déploiement d'une application SaaS GIS, le traitement des données et le système de reporting avec GitHub Actions CI/CD, Python, Bash et SQL.

Cette prestation intervient aussi dans des réalisations répertoriées ailleurs sur cette page :

Hook CI de forge répété : 2 défauts détectés · Conteneurs exploités avec Podman et Quadlet · Votre code sur votre propre serveur · Vérification des secrets en une seconde · Mode de vérification Ansible fiabilisé · Une infrastructure définie dans un dépôt · Infrastructure centrale construite de zéro · Déploiements Azure pris en charge de bout en bout · Infrastructure simplifiée au-delà d'AWS · 20 applications Docker sur Google Cloud et AWS

{ Cloud, infrastructure et DevOps }

Conteneurisation et orchestration

Un logiciel qui fonctionne partout de la même façon

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.

Cette prestation intervient aussi dans des réalisations répertoriées ailleurs sur cette page :

Conteneurs exploités avec Podman et Quadlet · Exposition des bacs à sable systemd réduite · Pipelines CI/CD GitHub Actions · SSL/TLS automatisé pour 100 applications Docker · 20 applications Docker sur Google Cloud et AWS

{ Cloud, infrastructure et DevOps }

Fiabilité des services et supervision

Être informé avant vos clients

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.

7 rôles de reporting de l'hôte en lecture seule

Conçu sept rôles de reporting en lecture seule décrivant un hôte en service en Markdown — faits, accès, git, métriques, trafic, sécurité et inventaire fournisseur — selon une règle interdisant d'afficher un chiffre non mesuré pendant l'exécution.

Bannissement testé à chaque exécution

Vérifié de bout en bout le blocage des intrusions à chaque exécution du durcissement, en bannissant une adresse réservée aux tests, en lisant la règle noyau obtenue et en levant le bannissement dans un bloc de nettoyage garanti, pour qu'une protection défaillante fasse échouer l'exécution au lieu de se déclarer saine.

Trois verrous sur la porte, aucun fermé

Détecté et corrigé trois protections SSH contre la force brute qui n'avaient jamais fonctionné : une prison de bannissement surveillant le port 22 alors que le démon écoutait sur 1986, une limitation de débit masquée par une règle plus large placée avant elle et une action de bannissement dont le binaire était introuvable, empêchant tout bannissement.

Limitation de débit et protection contre les abus

Conçu des protections contre les abus qui refusent l'accès en cas de défaillance — 22 limiteurs de débit Redis, Cloudflare Turnstile, idempotence des requêtes et verrouillage de l'origine — pour rejeter robots et inondations de requêtes sans faire confiance aux appelants.

Traitements lents confiés à une file River

Sorti les traitements lents du chemin des requêtes pour les confier à une file de tâches River — 15 modules de traitement, 8 tâches planifiées et 20 tâches pg_cron — afin de répondre aux requêtes pendant la poursuite du travail en arrière-plan.

Infrastructure durable, encore en service

Planifié et mis en œuvre de nouvelles fonctionnalités d'infrastructure pour des systèmes internes et externes, avec des solutions assez durables pour fonctionner encore des années plus tard avec peu de changements.

Continuité des flux de diffusion IPTV

Assuré la continuité de l'acheminement des flux IPTV entre fournisseurs et clients, en supervisant et maintenant le réseau de diffusion et la téléphonie IP jour et nuit.

Assistance 24/7 : environ 1 000 serveurs IPTV/OTT

Assuré une assistance infrastructure 24/7 pour une plateforme de diffusion IPTV/OTT, en administrant environ 1 000 serveurs et des systèmes appartenant aux clients pour une clientèle mondiale en Chine, aux États-Unis et en Allemagne.

E-mails avec basculement et journal de livraison

Conçu l'envoi d'e-mails comme une capacité de plateforme — trois fournisseurs avec basculement, webhooks de livraison, journaux d'envoi et de livraison, modèles et campagnes — avec un contrôle au démarrage qui refuse de lancer le service sans fournisseur.

Un produit cartographique pris en charge de bout en bout

Conçu l'architecture, développé et maintenu l'infrastructure, le traitement des données et une application cartographique pendant 2 ans, en couvrant la livraison logicielle et la maintenance d'exploitation.

40 sites administrés sur Apache et Nginx

Administré 40 sites web sur des serveurs d'hébergement Ubuntu Linux avec Apache et Nginx, en veillant à leur disponibilité et à leurs performances.

10 serveurs PostgreSQL et MS SQL sur Ubuntu

Conçu, déployé et maintenu 10 serveurs PostgreSQL et MS SQL sur des VPS Ubuntu Linux, en veillant à leurs performances et à leur fiabilité.

Premier tableau de bord d'observabilité

Développé et lancé le premier tableau de bord d'observabilité de l'entreprise, affichant en temps réel les performances du système et des visualisations de données sur le grand téléviseur des bureaux.

Cette prestation intervient aussi dans des réalisations répertoriées ailleurs sur cette page :

Secrets d'exploitation séparés par hôte · Mode de vérification Ansible fiabilisé ·
Supervision par absence de signal de vie ·
Sauvegardes automatisées et méthode de restauration explicite ·
Surveillance des erreurs et traçage internes · Déploiements Azure pris en charge de bout en bout ·
Risques réduits avec Grafana et Prometheus

{ Cloud, infrastructure et DevOps }

Sauvegarde et reprise après sinistre

Tout retrouver après une mauvaise journée

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

30 instances VPS Ubuntu Linux administrées

Géré 30 instances VPS Ubuntu Linux, mis en place des stratégies de reprise après sinistre et adapté les configurations réseau.

Cette prestation intervient aussi dans des réalisations répertoriées ailleurs sur cette page :

Secrets d'exploitation séparés par hôte ·
Sauvegardes automatisées et méthode de restauration explicite ·
Sauvegardes et reprise sous forme de code · 100 sauvegardes critiques automatisées : Barman

{ Cloud, infrastructure et DevOps }

Administration système

Faire fonctionner l'essentiel

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

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.

Gestionnaire de mots de passe commun

Intégré un système de gestion des mots de passe à l'échelle de l'entreprise pour renforcer la sécurité et simplifier le contrôle des accès.

100 serveurs Bare Bone administrés

Administré 100 serveurs Bare Bone, des réseaux physiques et des systèmes de téléphonie IP pour soutenir la croissance de l'entreprise.

1 000 routeurs Wi-Fi configurés et déployés

Configuré et déployé 1 000 routeurs Wi-Fi pour améliorer l'accès au réseau et ses performances pour les clients.

Infrastructure réseau administrée

Administré l'infrastructure réseau de plus de 1 000 serveurs, en veillant à leur déploiement, à leur sécurité et à la résolution des incidents.

8 connexions VPN WireGuard et IPsec gérées

Géré et dépanné 8 connexions VPN WireGuard et IPsec, en sécurisant les communications entre Google Cloud et les systèmes Linux.

Gestion des accès avec M365 et SSO

Prévenu des incidents de sécurité en pilotant la gestion des accès avec M365, 1Password, Red Hat SSO et OKTA SSO.

Cette prestation intervient aussi dans des réalisations répertoriées ailleurs sur cette page :

Miroir Tor onion à une adresse lisible · Six collecteurs indiquent les entrées ignorées · Hook CI de forge répété : 2 défauts détectés · Votre code sur votre propre serveur · 7 rôles de reporting de l'hôte en lecture seule · Exposition des bacs à sable systemd réduite · Zone DNS de 20 enregistrements en code · Supervision par absence de signal de vie · Sauvegardes automatisées et méthode de restauration explicite · Bannissement testé à chaque exécution · SSH durci avec 24 directives validées · Services publics sur un petit serveur · Une infrastructure définie dans un dépôt · Infrastructure centrale construite de zéro · Infrastructure durable, encore en service · Continuité des flux de diffusion IPTV · Assistance 24/7 : environ 1 000 serveurs IPTV/OTT · 40 sites administrés sur Apache et Nginx · 30 instances VPS Ubuntu Linux administrées · 1 000 règles RBAC pour sécuriser Linux · 10 serveurs PostgreSQL et MS SQL sur Ubuntu

{ Cloud, infrastructure et DevOps }

Réseaux et configuration de VPN

Relier vos bureaux et vos équipes

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

Cette prestation intervient aussi dans des réalisations répertoriées ailleurs sur cette page :

Second serveur provisionné à partir du code · Zone DNS de 20 enregistrements en code · Règles de pare-feu vérifiées par position · Trois verrous sur la porte, aucun fermé · Continuité des flux de diffusion IPTV · Infrastructure Azure sous forme de code · 100 serveurs Bare Bone administrés · 1 000 routeurs Wi-Fi configurés et déployés · Infrastructure réseau administrée · 8 connexions VPN WireGuard et IPsec gérées · 30 instances VPS Ubuntu Linux administrées

{ Cloud, infrastructure et DevOps }

Sécurité et gestion des accès

Décider qui peut accéder à quoi

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

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.

Revendications de propriété et arbitrage

Conçu de bout en bout les revendications de propriété d'entreprise : un utilisateur revendique une entreprise, un administrateur statue, puis l'approbation réécrit le graphe d'autorisation déterminant qui peut modifier quoi.

Droits d'accès Stripe et achats intégrés

Conçu la couche de paiement et de droits d'accès — Stripe et achats intégrés Apple et Google — en contrôlant l'accès à l'annuaire, à la recherche et à l'export au moyen d'un modèle de 11 tables vérifié côté serveur.

Application durcie avec une CSP à nonces

Durci l'application avec une CSP fondée sur des nonces, HSTS, des cookies SameSite, des rôles de base de données au moindre privilège et une nouvelle vérification des droits côté serveur.

Changement de contexte d'organisation

Conçu un système de changement de contexte d'organisation avec localStorage côté client et réplication dans un cookie côté serveur, permettant d'agir au nom des organisations administrées tout en appliquant le moindre privilège.

Communication et collaboration améliorées

Amélioré la communication et la collaboration de l'équipe avec Slack, Mattermost, 1Password et Jira, économisant 8 000 heures de travail.

Cette prestation intervient aussi dans des réalisations répertoriées ailleurs sur cette page :

Miroir Tor onion à une adresse lisible · 13 adresses mesurées sans traçage · Connaître la licence de tout ce que vous livrez · Moitié jusque-là inutilisée du cœur exploitée · Secrets d'exploitation séparés par hôte · Second serveur provisionné à partir du code · Conteneurs exploités avec Podman et Quadlet · Exposition des bacs à sable systemd réduite · Zone DNS de 20 enregistrements en code · Vérification des secrets en une seconde · Règles de pare-feu vérifiées par position · Bannissement testé à chaque exécution · SSH durci avec 24 directives validées · Trois verrous sur la porte, aucun fermé · Programme continu d'analyse de sécurité · Limitation de débit et protection contre les abus · Gestionnaire de mots de passe commun · Collaboration d'équipe automatisée · Infrastructure Azure sous forme de code · E-mails avec basculement et journal de livraison · SSL/TLS automatisé pour 100 applications Docker · Infrastructure réseau administrée · 8 connexions VPN WireGuard et IPsec gérées · Gestion des accès avec M365 et SSO · 1 000 règles RBAC pour sécuriser Linux

{ Cloud, infrastructure et DevOps }

Sécurité des agents d'IA

Limiter votre IA à ce qu'elle doit faire

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.

Cette prestation intervient aussi dans des réalisations répertoriées ailleurs sur cette page :

Secrets d'exploitation séparés par hôte · Règle de périmètre après une fuite de secrets · Vérification des secrets en une seconde · Bannissement testé à chaque exécution · Trois verrous sur la porte, aucun fermé · Programme continu d'analyse de sécurité ·

Limitation de débit et protection contre les abus · Gestion des accès avec M365 et SSO · 1 000 règles RBAC pour sécuriser Linux

Comparez votre point de départ

Choisissez une réalisation comparable et identifiez ce qui diffère dans votre propre situation.

Consulter les tarifs Contactez-nous →

Discutez d'une démarche pertinente

Demandez-nous de transformer l'expérience pertinente en une démarche que vous pouvez évaluer.

Découvrir nos prestations

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

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/portfolio/