Zakaria EL MOUMNAOUI

Casablanca, Maroc | zakariaelmoumnaoui96@gmail.com | +212 622906171
LinkedIn: zakaria el moumnaoui | GitHub: zakaria-statistics

Ingénieur DevOps et Cloud

Résumé

Ingénieur DevOps et Cloud, passionné par l'automatisation, l'orchestration et l'intégration continue. Conception et exploitation d'infrastructures cloud (Azure/AWS) et on-prem, IaC avec Terraform, conteneurisation Docker/Compose, orchestration Kubernetes. Mise en œuvre de CI/CD sécurisés et observables (tests, monitoring, alerting, logs).

Compétences Techniques

CI/CD et Automatisation : Terraform, GitLab CI/CD, Ansible, Jenkins, GitHub Action, ArgoCD, Vagrant
Conteneurisation et Orchestration : Docker, Containerd, Kubernetes, Helm, Docker swarm
Cloud et Infrastructure : AWS, Azure, VMware ESXi, VirtualBox
Systèmes et Sécurité : Linux, KMS Vault, SSL/TLS, Firewalls, Nginx, GitGuardian
Monitoring et logging : Prometheus, Grafana, Loki
Langages et Scripts : Bash, Java, Python, TypeScript, PowerShell
Bases de Données : MySQL, PostgreSQL, MongoDB, Firebase
FrameWorks et Libreries : Spring Boot, Angular
AI et LLMOps : Ollama, LangChain, llms, Agents/Tools, Prompt engineering

Expériences

Ingénieur DevOps Cloud chez We Are Beebay: client Marjane Holding (2024 - Présent)

Python Automation & Reporting – Agrégation Multi-Outils de Sécurité 2025

Développement d'un module Python complet pour centraliser et analyser les rapports CI :

- Parsing avancé de **rapports XML/JSON** : SpotBugs, PMD, Checkstyle, Dependency-Check, Gitleaks.
- Construction dynamique de **schémas XML** (streaming, `iterparse`, namespaces) pour gérer de très gros fichiers.
- Normalisation des findings (file, line, severity, rule, CVSS, CWE) et **calcul des KPIs globaux** par outil.
- Intégration complète de l'API SonarQube : `/api/issues/search`, `/api/hotspots/search`, `/api/qualitygates/project_status`, `/api/measures/component`.
- Agrégation multi-sources : **fusion des vulnérabilités, hotspots, code smells, secrets, SCA** dans une structure uniforme.
- Production d'un **rapport exécutif automatisé** : résumé, statut (HEALTHY/ATTENTION/BLOCKER), recommandations.
- Appel d'un modèle LLM via API (OpenAI compatible) pour enrichir le rapport avec des insights et recommandations.
- Création et envoi d'un **rapport PDF** par email (PDF + texte), consolidant toutes les analyses.
- Logging structuré, architecture modulaire (*utils*, *details*, *sonar*, *mailer*, *llm_client*).

Plateforme Kubernetes auto-gérée sur Azure sécurisée avec Vault (Terraform + Ansible) 2025

Mise en place pas à pas d'une base solide pour applis web avec bases séparées :

- **Réseau** : VNet unique avec 5 sous-réseaux dédiés — Sub1 (bastion kube), Sub2 (control-plane), Sub3 (workers), Sub4 (DB), Sub5 (bastion DB).
- **Sécurité d'accès** : règles NSG minimales (SSH uniquement depuis notre IP publique vers les bastions ; chemins internes minimal ouverts entre bastions et VMs nécessaires).
- **Provisionnement** : infra déployée via Terraform (réseau, IPs publiques des bastions, VMs kube/DB). Sorties vérifiées (IPs publiques/privées).

- **Coffre-fort** : intégration Vault (AppRole) opérationnelle pour récupérer les secrets Azure et piloter Terraform via un wrapper (bash script).
- **Durcissement de base** : cloud-init appliqué (utilisateurs/SSH clés, mises à jour, services utilitaires) sur bastions, nœuds kube et VMs DB.
- **Bootstrap cluster K8s** : prérequis posés via **cloud-init**, orchestration via **Ansible** (init du control-plane, jonction des workers, CNI Calico) ; accès `kubectl` depuis le bastion.
- **Ansible — Caching** : mise en cache des faits partagés (kubeconfig, `join_cmd`) via backend `jsonfile`, réutilisables entre plays/runs.
- **Ansible — PostgreSQL (Sub4)** : install + écoute, ACL CIDR (`pg_hba.conf`), création DB/utilisateur, sauvegarde logique (script+cron), smoke test (`SELECT 1`) — le tout idempotent.
- **Intégration DB/K8s** : secrets fournis par Vault (AppRole), Services headless + Endpoints sur IPs privées, variables d'environnement pour applis.
- **Vault & AppRole (secure delivery)** : implémentation d'un flux sécurisé de distribution de *SecretID* via response-wrapping et consommation par Vault Agent (init/sidecar) pour générer db creds.
- **Identifiants dynamiques DB** : émission de credentials temporaires via `database/creds/<role>` (gestion des leases, rotation/renewal et smoke-tests DB).

Sauvegarde & Restauration sur Azure avec Terraform & PostgreSQL 2025

Conception et mise en œuvre d'une stratégie complète de sauvegarde et de restauration sur **Azure**, combinant des instantanés au niveau infrastructure avec la restauration à un instant précis (**PITR**) au niveau base de données :

- Provisionné des **VM Linux Azure** et des **disques managés** avec **Terraform**, incluant l'attachement explicite des disques (LUN0) pour le stockage persistant.
- Mis en place un **cycle de vie des instantanés** de disque : création d'instantanés incrémentiels, restauration sous forme de nouveaux disques managés, et ré-attachement aux VM pour tester la reprise après sinistre.
- Installé **PostgreSQL 17 (PGDG)** et préparé le cluster de base de données sur le disque persistant **/mnt/appdata**.
- Configuré l'**archivage des WAL** et effectué des **sauvegardes complètes** afin de permettre une restauration précise.
- Simulé une panne en supprimant le répertoire de données et réalisé une **restauration à un instant précis (PITR)** grâce aux segments WAL archivés et aux sauvegardes complètes.
- Validé la reprise à deux niveaux : infrastructure (remplacement de disque) et applicatif (PITR PostgreSQL), démontrant la résilience de bout en bout.
- Appliqué les bonnes pratiques d'**Infrastructure as Code (IaC)**, de maîtrise des coûts liés aux instantanés, et de gestion propre du cycle de vie des ressources avec **Terraform**.

Provisionnement d'Infrastructure Azure avec Terraform 2025

Conception et gestion d'un environnement cloud modulaire sur **Azure** à l'aide de **Terraform** :

- Conception et implémentation de deux modules réutilisables : **network** (VNet, Subnet, NSG, NIC, IP publique) et **compute** (VM Linux).
- **cloud-init** : installation de **Docker** et déploiement d'une application conteneurisée au premier démarrage, avec **injection dynamique des variables** via `templatefile` ; service `systemd` pour les redémarrages automatiques.
- **DNS & TLS/HTTPS** : enregistrement **A** du domaine vers l'**IP publique statique** ; ouverture NSG **80/443** ; **Nginx** en reverse proxy (application sur `127.0.0.1:8080`), redirection `HTTP vers HTTPS`, certificats **Let's Encrypt** (webroot) et **renouvellement automatique** avec *reload* Nginx.
- Organisation des ressources au sein de **Resource Group** dédié afin d'assurer isolation et gestion du cycle de vie.
- Mise en place de configurations spécifiques par environnement (**dev.tfvars**, **prod.tfvars**) pour paramétrer la taille des VM, les CIDR réseau, la taille des disques et les règles de pare-feu.
- Implémentation de **Network Security Groups** avec règles adaptées par environnement (HTTP + SSH en dev, SSH + HTTPS en prod).
- Adoption des pratiques **Infrastructure as Code (IaC)** pour des déploiements cohérents, reproductibles et un basculement simplifié entre environnements.
- Intégration d'une approche **optimisation des coûts** : choix des types de VM, suivi des disques/IP et planification budgétaire.

- Gestion sécurisée des identifiants sensibles via **variables d'environnement** et fichiers **tfvars**.

Pipelines CI/CD avec GitLab CI/CD, Spring Boot, Next.js et PostgreSQL 2024

Mise en place d'un pipeline CI/CD automatisé avec **GitLab CI/CD** :

- Phases de build et de tests automatisées du backend (**Spring Boot**) avec **Maven**.
- Construction et optimisation des images Docker pour le backend et le frontend (**Next.js**).
- Déploiement automatisé des applications et de la base **PostgreSQL** sur une VM dédiée via **Docker Compose**.

Infrastructure as Code Cloud (Terraform et AWS) 2024

Conception d'environnements cloud reproductibles :

- Modules Terraform : **réseau** (VPC, sous-réseaux, SG, routage) et **compute** (EC2, cloud-init).
- Séparation env **dev**/**prod** via `*.tfvars`, gestion des secrets (variables d'env./fichiers chiffrés).
- Politique d'ouverture minimale (80/443, SSH restreint).

Ingénieur DevOps Cloud chez MTS System (2022–2024)

Déploiement automatisé avec GitHub Actions 2024 - 2023

Mise en place d'un pipeline **GitHub Actions** pour **Spring Boot**, **Next.js** et **PostgreSQL**, avec construction d'images Docker multi-étapes optimisées et déploiement via Docker Compose derrière un proxy **NGINX**.

Mise en place de pipelines CI/CD et de déploiement avec Jenkins et Kubernetes 2024-2023

Conception et mise en œuvre de deux pipelines automatisés intégrés à **Jenkins** et **Kubernetes** :

- Configuration de machines virtuelles dédiées pour chaque outil de l'écosystème (**Jenkins**, **SonarQube**, **ArgoCD**).
- Mise en place de **NGINX** comme proxy inverse pour sécuriser et centraliser les accès.
- Implémentation du chiffrement **SSL**/**TLS** pour sécuriser le trafic web et protéger les données sensibles en transit.
- Pipeline CI/CD : Automatisation des builds et tests via **Maven**, analyse de code avec **SonarQube** et génération d'images Docker.
- Pipeline de déploiement : Mise à jour des manifests Kubernetes et déploiement via **ArgoCD**.

Chaîne CI/CD multi-projets (Jenkins, Docker, Docker compose) 2023 - 2022

Mise en place de pipelines CI/CD end-to-end pour applis Java (Maven) et Node.js :

- Pipelines Jenkins (build, tests, analyse **SonarQube**), génération d'images Docker, push vers registre privé.
- Déploiement via **Docker Compose** sur VMs Linux ;
- **NGINX** reverse proxy avec **TLS Let's Encrypt** (renouvellement auto), redirection HTTP→HTTPS.
- Traçabilité : tags/versions, conservation des artefacts pour audit.

Automatisation Systèmes (Ansible, Vagrant, Linux) 2022

Industrialisation d'environnements de test :

- Provisionnement VMs **Ubuntu** avec **Vagrant**, rôles **Ansible** (NGINX, PostgreSQL/MySQL, Docker).
- Templates idempotents (vars group/host dynamique), durcissement SSH, pare-feu **UFW**.

Projets internes / POC industrialisés

Infrastructure LLM sur Azure avec Terraform 2025

Conception et exploitation d'une application LLM sur **Azure** avec **Terraform** et des VM GPU :

- **RBAC** : création d'un **principal de service** et attribution de rôles à moindre privilège (`Contributor` au scope du RG, `Reader` au niveau abonnement pour les images) ; stockage sécurisé des identifiants (tenant, subscription, client, secret) pour le provider.
- **Réseau & calcul** via modules : **VNet**, sous-réseau, **NSG**, IP publique, NIC, **Key Pair** et **VM Linux** ; tous les composants dans un **Resource Group** dédié pour l'isolation et le cycle de vie.
- **cloud-init** : installation de **Docker** et déploiement de **Open WebUI** (conteneur) au premier démarrage, avec **injection dynamique des variables** via `templatefile` ; service `systemd` pour les redémarrages automatiques.

- **DNS & TLS/HTTPS** : enregistrement **A** (p.ex. `llm.infra-ia.com`) vers l'**IP publique statique** ; ouverture NSG **80**/**443** ; **Nginx** en reverse proxy (app sur `127.0.0.1:8080`), redirection `HTTP→HTTPS`, certificats **Let's Encrypt** (webroot) et **renouvellement automatique** avec *reload* Nginx.
- **GPU** : activation sur des SKU supportés (p.ex. `Standard_NC4as_T4_v3`), installation de nvidia driver et de `nvidia-container-toolkit`.
- **Quotas GPU** : interrogation et augmentation des limites par famille (p.ex. *Standard NCASv3_T4 Family*) via `az rest` et **ticket Support**.
- **Paramétrage multi-environnements** avec **tfvars** (`dev.tfvars`, `prod.tfvars`) pour tailles de VM, CIDR, disques et règles NSG ; conformité **IaC** pour des déploiements reproductibles.
- **Intégration OpenAI** via `OPENAI_API_KEY` et `OPENAI_BASE_URL` ; **LLM local** (**Ollama**) en repli ou pour réduire les coûts d'API externes.
- **Coûts & sécurité** : choix de SKU adaptés, suivi IP publique/disques, budgets/alertes ; renforcement NSG (SSH restreint en dev, HTTP/HTTPS en prod).

Infrastructure LLM sur AWS avec Terraform 2025

Mise en place d'une application LLM sur **AWS** avec **Terraform** et en supportant l'option GPU :

- **IAM** : création d'un **utilisateur**/**role** Terraform à privilèges minimaux ; variables d'environnement pour l'authentification (sans clés en dur).
- **Réseau & calcul** via **modules** : **VPC**, sous-réseau public, **Internet Gateway**, **Route Table**, **Security Group** (SSH/HTTP), **Key Pair**, **EC2** et volume **EBS** additionnel ; **sonde HTTP** Terraform pour la disponibilité.
- **cloud-init** : installation de **Docker** et déploiement de **Open WebUI** au premier démarrage, avec **injection dynamique des variables** via `templatefile` ; service `systemd` pour l'orchestration.
- **DNS & TLS/HTTPS** : enregistrement **A** (p.ex. `llm.infra-ia.com`) vers une **Elastic IP** ; ouverture SG **80**/**443** ; **Nginx** en reverse proxy (app sur `127.0.0.1:8080`), redirection `HTTP vers HTTPS`, certificats **Let's Encrypt** (webroot) et **renouvellement automatique** avec *reload* Nginx.
- **Paramétrage multi-environnements** : **tfvars** pour tailles d'instances, CIDR, disques et règles ; déploiements **IaC** reproductibles.
- **GPU (optionnel)** : familles `g4dn.xlarge`/`g5.xlarge` selon besoins ; vérification **Service Quotas** et disponibilité régionale ; installation des pilotes NVIDIA et de `nvidia-container-toolkit`.
- **Quotas** : demandes d'augmentation vCPU/GPU via **Service Quotas** et tickets Support ; démarrage possible en **CPU** pour valider la chaîne, puis montée en GPU après approbation.
- **Intégration OpenAI** via `OPENAI_API_KEY` et `OPENAI_BASE_URL` ; **LLM local** (**Ollama**) en repli pour limiter les coûts externes.
- **Coûts & gouvernance** : dimensionnement des budgets et **procédures de destroy** propres et documentées.

Kubernetes & GitHub Actions : déploiement local d'un agent LLM avec Ollama 2025

Conception et orchestration d'une plateforme d'agent IA en conditions proches de la production :

- Pile technique : backend **FastAPI** avec streaming **SSE**, interface **React**, **LangChain** + **Ollama (Mistral)** pour l'inférence locale.
- **Kubernetes** : gestion des déploiements et services (backend et frontend) avec ressources limitées. Résultat : exécution stable et supervisable.
- **CI/CD vers Kubernetes** : workflow GitHub automatisé déclenché à chaque modification, construisant et empaquetant l'application, la publiant, mettant à jour l'environnement d'exécution et vérifiant le déploiement. Résultat : mises en production prévisibles et à faible risque.
- **Accélération GPU** : inférence optimisée GPU via le **plugin NVIDIA** pour Kubernetes.

Certifications

AWS Fundamentals - AWS en cours

Compréhension des services et ressources cloud, modèles de déploiement et sécurité sur **AWS**.

Kubernetes - KodeKloud en cours

Préparation aux certifications CKAD, CKA et CKS.

Kubernetes et OpenShift - IBM 2024

Maîtrise des concepts Kubernetes : objets, networking, sécurité, déploiement et introduction de l'ecosystème OpenShift.

Docker Essentials - IBM 2023
Création et gestion d'images Docker, orchestration de services avec Docker Swarm, scalabilité et sauvegarde des images sur Docker Hub.

NDG Linux Unhatched - Cisco 2023
Introduction aux commandes shell, gestion des utilisateurs, manipulation des systèmes de fichiers, gestion des processus et scripts Bash.

Notions de Réseaux - Cisco 2023
Principes fondamentaux des réseaux : connectivité, adressage IP, sous-réseaux, routage et protocoles de communication.

SQL et Bases de Données - IBM 2023
Conception et requêtage de bases de données relationnelles.

Formation

Master en Informatique Big Data et Cloud Computing 2022-2024
École Normale Supérieure de l'Enseignement Technique Mohammadia.

Licence en Mathématiques et Statistiques 2014-2020
Faculté des Sciences Semlalia Marrakech.

Langues

Français: Courant **Anglais**: Courant