CV

Serhii Ostrikov, Senior DevOps Engineer | Cloud Engineer | SRE (Remote Only)

Linkedin | YouTube | Email: serhii@ostrikov.org | Time Zone: GMT+2 (Kyiv) | Portfolio

Work Experience

- **SRE | DevSecOps** @ *Temabit, Fozzy Group* *Jul 2025 / Present, Full-Time, Remote*
  • **Designed and managed AWS enterprise cloud infrastructure** for a regulated environment, ensuring **auditability, cost transparency, and compliance** by applying the **Well-Architected Framework** and leveraging **AWS Config, CloudTrail, IAM policies, and custom tagging**
  • Built **secure, multi-account identity architecture** with **IAM best practices**, including **OIDC federation**, **Access Analyzer**, **SSO (Control Tower)**, **temporary tokens**, and **least privilege**.
  • Delivered **scalable solutions** for **real-time data processing** and **business automation** using **ECS**, **Lambda**, **MSK (Kafka)**, **EventBridge**, **RDS**, **EC2**, and **Transit Gateway**.
  • **Developed production-grade Terraform modules** in GitLab private registry, with **unit/E2E tests**, **CI/CD pipelines**, **linting**, and **security scanning** (TFLint, tfsec, Checkov, pre-commit hooks), with **environment-specific** and **component-specific** state separation.
  • Architected **secure CI/CD pipelines** for **.NET apps** using **Docker (without DinD)**, **Skopeo**, **Buildah**, and **SonarQube** for quality gates with Established **release governance** enabling **verified, decoupled CI/CD workflows**
  • Implemented an **observability stack** combining **AWS CloudWatch**, **Prometheus**, and **Grafana**
- **DevOps | Cloud Engineer @** *CPCS (Custom PC Software)* *June 2024 / Jul 2025, Full-Time, Remote*
  • **Managed multiple AWS environments** for different needs, configured permissions, users, services, wrote documentation, disaster recovery plans and billing reports on Confluence for a foreign client to understand costs
  • **Collaborated with different teams** to fit in the tight deadlines - Frontend, Backend, DevOps, IOS and Android
  • **Automated AWS infrastructure provisioning** on multiple accounts and regions with Terraform IaC, wrote custom modules for various different purposes, such as - Setting up VPC, Route53, S3, SNS and SQS queues, SES for Email, OpenVPN server, EKS, ECS, ensuring best practices - Remote and decoupled state, versioning, IaC pipelines;
  • **Did a K8S clusters management** - multiple EKS clusters, Node Groups, Karpenter for autoscaling, Metrics Server, Secrets, ConfigMaps, Nginx and ALB Ingress controllers, CSI drivers, AWS Container Insights, Prometheus and Grafana with Helm Charts and many more
  • **Created and streamlined CI/CD pipelines for Microservices (ECS, EKS), AWS Lambdas and Mobile Apps** via GitLab CI on both hosted EC2 runners with autoscaling and GitLab's owned MacOS instances for IOS automated builds Docker Images Caching, Merge Trains, automated release notes, Integration Testing with Docker, Sonarqube Code smell testing, Custom Branch rules, Slack, Jira, Allure integrations, Deployments of a custom written helm chart that contained our K8S deployment code, separated repository for our CI/CD code
  • **Managed an ECS and EKS microservices,** ensured that deployments were safe, with rollbacks, notifications to Slack, made a dashboards with logs and common metrics, such as latency, 5XX on a load balancers and Java errors with Prometheus, Grafana, CloudWatch (Dashboards, Log Insights, Alarms), FluentD for log exports, Istio, Kiali
- **DevOps Engineer @** *Games On Cloud* *May 2024 / June 2024 (Contract), Full-Time, Remote*
  • Optimized AWS infrastructure (5%-15% cost reduction), managed cloud services with proprietary software
  • Developed Discord/Telegram bots using Python, and managed databases on RDS (MySQL) with CI/CD pipelines
- **DevOps and Developer @** *Trementum Research* *Apr 2024 / May 2024 (Contract), Full-Time, Remote*
  • Wrote and maintained existing **Node.JS**, **Python** code with unit testing coverage, ensuring proper behavior
  • **Automated scheduled jobs**, **code deployments from GitHub**, managed existing jobs with Jenkins Server
- **DevOps Engineer @** *NDA* *Jun 2022 / Apr 2024, Full-Time, Remote*
  • Configured AWS Control Tower for multi-account governance, enforcing baseline security across all accounts.
  • **Cut EKS cluster costs by ~60%** by automating node scaling with Karpenter + spot instances
  • Centralized EKS + app logs with FluentBit + CloudWatch (custom alerts, dashboards, Slack notifications)
  • Automated deployments with GitLab CI, reduced technical debt with templates and shareable pipelines

Skills

• **Languages** - English (Professional Fluency), Ukrainian (Native), Russian (Native)
• **Linux and Windows Server Administration** - VPNs - WireGuard, OpenVPN, Nginx, Apache2 (LAMP stack), WordPress, IPtables, UFW, vsFTPd, bind9, automation with Ansible, Docker, Docker Compose, KVM, K8S
• **Programming / Scripting** – Bash, Python, Node.js, C/C++ also basics of PHP
• **Version Control Systems (VSC or SCM), CI/CD** - Git, GitLab CI, GitHub Actions, Jenkins, Cloud Build
• **AWS** - EC2 (VPC, Load Balancers, Security Groups, Spot Instances, etc), EKS (Autoscaling, Monitoring), ECS, S3, IAM, RDS, DynamoDB, Route 53, Elastic Beanstalk, Lambda, SQS, SNS, EventBridge, CloudWatch, etc.
• **Kubernetes** - Autoscaling with Node Groups and Karpenter, Nginx and ALB ingresses, EFS CSI, Monitoring with Prometheus, Grafana, FluentD, FluentBit and CloudWatch, Writing and using Helm Charts, Istio, Kiali
• **GCP Basics** - Compute, VPC, IAM, Cloud SQL, App Engine, Cloud Functions, Cloud Build
• **IaC (Infrastructure as Code) –** Terraform, Terragrunt, Ansible, Helm, K8S Manifest files also basics of Packer
• **SQL and NoSQL Databases** - MySQL, PostgreSQL, DynamoDB, basics of Redis, Elastic Search, MongoDB
• **Monitoring and Logging** - Prometheus, Grafana, CloudWatch (Alarms, Logs, Events), FluentD, Elastic | ELK Basics

Certifications and others

- AWS - Cloud Practitioner, SysOps Administrator Associate, Solutions Architect Associate, Developer Associate
- Bachelor's degree in **Computer Engineering** - *Kharkiv National University of Radio Electronics (2022-2026)*

Portfolio

Introduction

Hi! My name is Serhii Ostrikov

This is a **portfolio** of mine that I've crafted over the years of work as a **Software Engineer**, **DevOps engineer** and **SRE**. I also have a CV, in case it is what you are looking for.

Also check-out my Website and YouTube channel, where I teach people how to build stuff 🙂

If I caught your eye, you can reach out to me through Linkedin or email - serhii@ostrikov.org

---

Event-Driven Enterprise Service Bus (ESB) using Kafka

Client Story

In this project the client required an ESB capable of handling **thousands of real-time messages per second** across fully isolated environments. Because the data included sensitive and potentially PII-level information, they needed **strict, granular access controls** and end-to-end security, **including encryption at-rest and in-transit**.

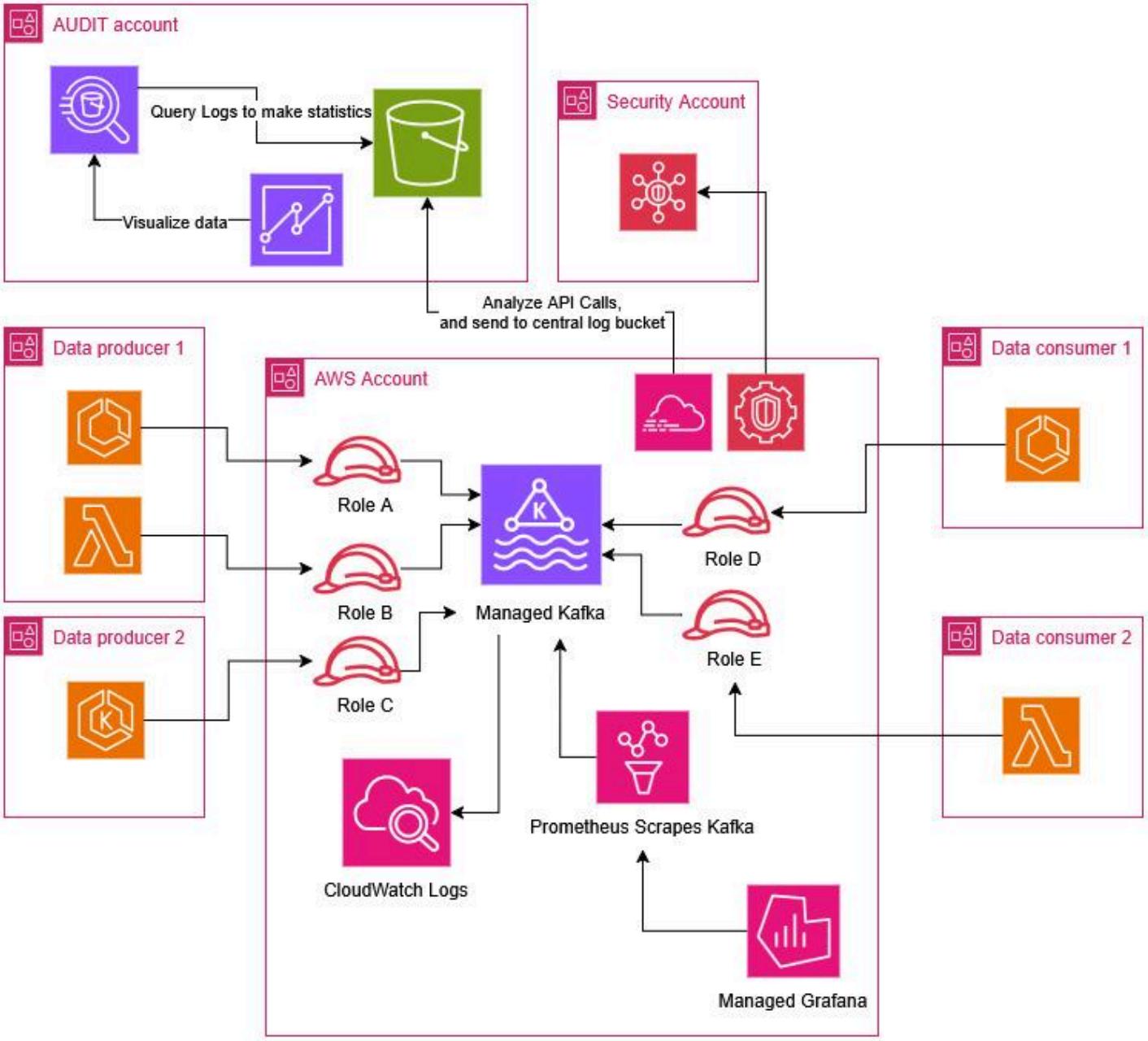
They also demanded a platform with **minimal manual upkeep**, automated incident detection, and alerting robust enough to maintain strict SLAs. The system had to be **highly scalable, fault-tolerant, auditable**, and supported by compliant, isolated build and access workflows.

Solution used

- **AWS MSK (Kafka)** - based ESB connecting **isolated AWS accounts** via **cross-account IAM roles**.
- **Terraform** for infrastructure and granular per-topic **IAM access control**.
- **Jira Service Desk + Automation** for self-service access management.
- **CloudTrail**, **S3**, **GuardDuty**, **Athena** for centralized audit and security logging.
- **Kafka-UI** for topic inspection, message tracing, and debugging.
- **CloudWatch**, **Prometheus**, **Grafana** for monitoring and alerting (integrated with **Jira Alerts**) with SRE shifts to reach desired SLA level.
- **GitLab CI** pipelines with **self-hosted runners** for compliance and isolated execution.
- End-to-end encryption and AWS-managed services for **scalability**, **resilience**, and **low maintenance**.
- **Glue schema registry** for storing **Avro schemas**

High Level Overview

Infrastructure



Code Organization

The code is hosted on GitLab with GitLab CI pipelines. The main repo with infrastructure is written with **terraform** and protected with **pre-commit** (trivy, tfsec, tf-lint, etc), **SonarQube** to manage the code quality

Social Media Platform (Facebook-like)

Client story

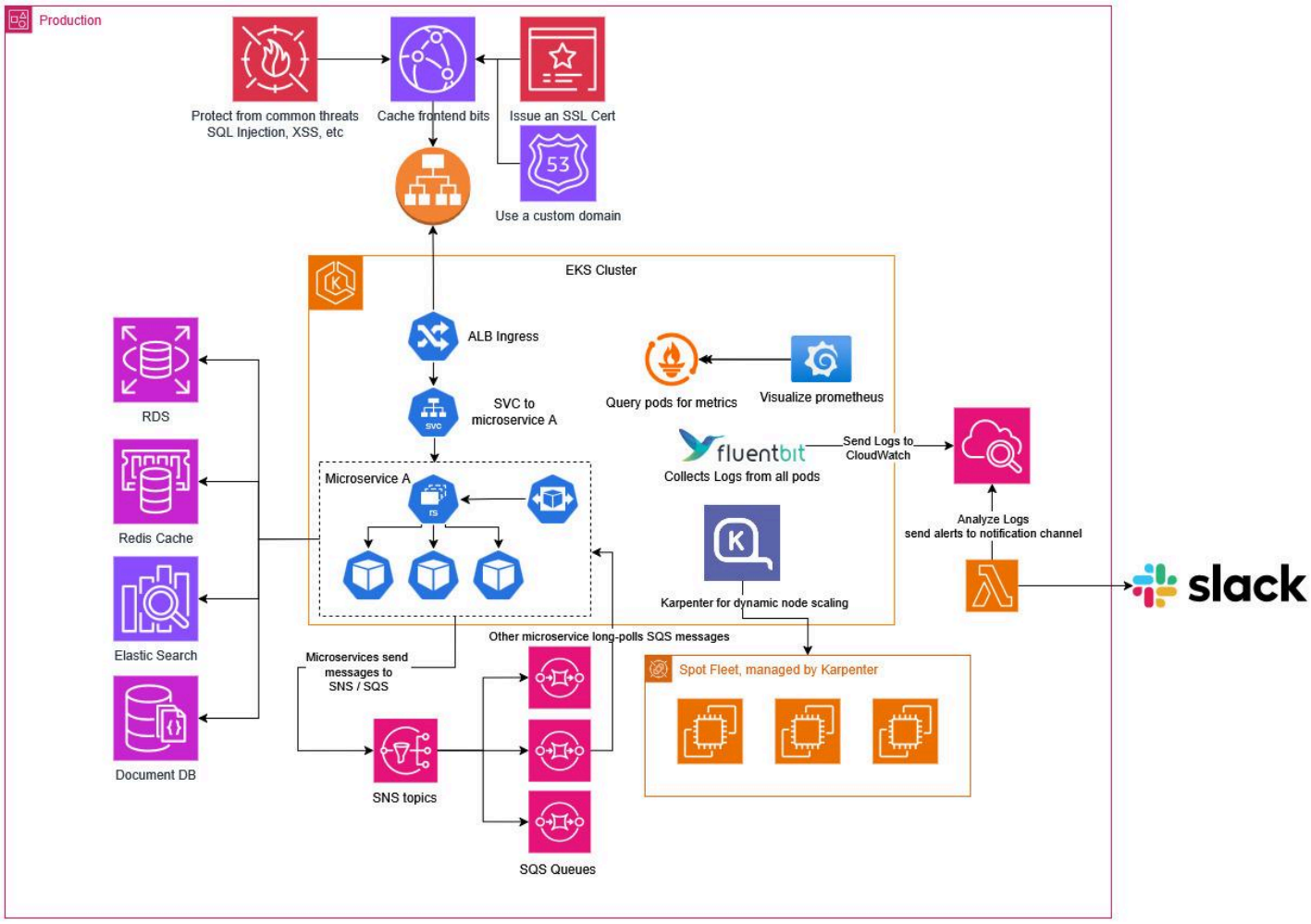
The client needed a **secure, multi-environment social platform** managed across regions with **granular access**, clear documentation, DR readiness, and **cost visibility**. They required a **fully automated, scalable application platform** with standardized environments, centralized observability, and **end-to-end CI/CD** for backend, serverless, and mobile apps to support rapid releases. Finally, they demanded **high production reliability** with safe deployments, automatic rollbacks, real-time alerts, and actionable insights to keep a global social app stable.

Solution Used

- Multi-environment **AWS** platform management with granular **IAM**, cross-team access policies, documentation, DR procedures, and cost reporting for a distributed international team.
- Automated provisioning across several AWS accounts and regions using **Terraform IaC** with custom modules (**VPC**, **Route53**, **S3**, **SNS/SQS**, **SES**, **OpenVPN**, **EKS**, **ECS**) following best practices: remote state, versioning, modular design, and pipeline-driven deployments.
- Operated and optimized multiple **EKS** clusters: **Node Groups**, **Karpenter** autoscaling, **Metrics Server**, **Secrets/ConfigMaps**, **NGINX Ingress**, **ALB Ingress**, **CSI drivers**, **AWS Container Insights**, and full observability via **Prometheus**, **Grafana** with **Helm** charts.
- Built end-to-end **CI/CD** pipelines for microservices (**ECS/EKS**), **AWS Lambda**, and mobile apps using **GitLab CI** with autoscaled **EC2 runners** and GitLab-managed **macOS** runners: caching, merge trains, release automation, integration testing with **Docker**, **SonarQube**, **Slack**, **Jira**, **Allure** integrations, and Helm-based deployments (centralized deployment repo).
- Maintained operational reliability of **ECS/EKS** microservices with safe deployments, automated rollbacks, **Slack** alerts, and comprehensive monitoring dashboards (latency, 5XX, JVM metrics) using **Prometheus**, **Grafana**, **CloudWatch** (Dashboards, Log Insights, Alarms), **FluentD**, **Istio**, and **Kiali**.

High Level Overview

Infrastructure



On the following chart, for simplicity sake, only production is shown

Code Organization

As I mentioned, there were multiple environments, following the gitflow convention, for automating things, the **Gitlab CI** was used for CI/CD pipelines. The application and microservices followed the gitflow branching and release strategy

For code quality check the **SonarQube** was integrated with the pipeline

The infrastructure was entirely written with **terraform** using a custom modules

Gradual on-prem migration to AWS

Client Story

The client needed a **phased migration** from an aging on-premise environment to AWS without disrupting day-to-day operations, while keep growing the existing product.

The goal was to move the **database**, **application services**, and **scheduled processing tasks** gradually, ensuring *compatibility*, *security*, and *minimal downtime* throughout the transition.

Solution Used

- Migrated the on-prem **MS SQL** database to **Amazon Aurora** using **AWS DMS**.
- Utilized write-through cache with **Redis** (**ElastiCache**) to speed up queries
- Containerized and deployed application services with **Docker** on **ECS Fargate**.
- Rebuilt supporting logic as **AWS Lambda** functions where appropriate.
- Exposed external functionality through **HTTP API Gateway** with **AD JWT authorizers**.
- Secured all entry points using **CloudFront**, **AWS WAF**, and edge protections (CF Functions, Lambda@Edge).
- Implemented scheduled data-processing tasks with **AWS Glue Jobs**.
- Published WAL (Write Ahead Log) changes to a **Kafka** using **logical replication**

High Level Overview

Code Organization

The main stack was written with **Python**, infrastructure mainly in **Terraform** and bits of **CDK (CloudFormation)**. I later moved the CDK bits into terraform so that our stack is entirely managed with it, instead of depending on two different technologies.

Regarding a CI/CD pipeline, two business critical requirements were issued:

1. No untested code must be shipped to a production
2. Developments must be agile and quick, with a trunk-based development approach

The main part of the application was located in the monorepository so the CI/CD pipeline was built with those requirements in mind.