AHMAD ZUSHAN

Delhi, India · **email** | **phone** | **LinkedIn** | **GitHub**

Focus: Efficiency-first AI/ML & Platform Engineering · eBPF/Kernel Profiling · Cloud/MLOps · Security Hardening

---

1) EXECUTIVE SUMMARY

I design and ship production systems that favor **stream-first**, **zero/low-copy**, and **non-blocking** execution. The work spans: - **Kernel/eBPF observability:** purpose-built probes that extract only actionable signals (especially **outbound third-party API calls**), minimizing user-space cost. - **Security & hardening at scale:** vulnerability discovery and remediation across dozens of VMs and Kubernetes clusters; measurable reductions in attack surface and brute-force noise. - **AI pipelines:** OCR + LLM/VLM extraction for Indian invoices; retail display scoring; GPU-hosted VLM/OCR services with OpenAI-compatible APIs. - **Cloud & MLOps:** AWS/Azure/OCI, Kubernetes/EKS, Docker, Nginx/PM2/systemd, Redis/Kafka streaming, Postgres/Mongo data layers.

Guiding principles: **selective tracing**, **no TLS decryption** (capture at the application boundary only when necessary), **deterministic JSON contracts**, and **resource-budgeted** deployments (GPU/CPU/memory/concurrency defined up front).

---

2) SECURITY & INFRASTRUCTURE HARDENING (MMTC CASE STUDY)

Scope: Secured ~**48 virtual machines** in a heterogeneous estate while maintaining developer velocity.

Outcomes - Prevented **brute-force** attempts via layered controls (cloud SGs + host firewalls + log-aware bans).
- Reduced noisy auth failures and lateral-movement avenues; tightened remote access with VPN/bastion and IP allow-lists. - Documented a **rapid triage playbook** for infrastructure incidents and developer escalations.

Key Activities - Network policy tightening (cloud security groups, host-level UFW/iptables) with country/IP block lists and rate limits. - **Auth hygiene:** per-role accounts, key-preferred auth, MFA where supported, session timeouts, audit logging. - SSH protection (ban rules for repeated failures, non-standard ports where viable, banner/legal notices, restricted shells as needed). - Service exposure review (Nginx, DBs, Redis, admin consoles); public endpoints minimized; WAF-style rules for common CVEs. - Backups & recovery: snapshot cadence, config/state versioning, restoration drills. - Monitoring: central auth-log ingestion, brute-force detectors, alert thresholds; clear owner/runbook for each alert.

---

3) AGENTSIGHT — STREAM-FIRST eBPF/SSL OBSERVABILITY

An end-to-end platform that captures minimal yet rich telemetry **without decrypting TLS** and **without stalling user-space**.

3.1 Design Goals

- Capture **only** what matters: outbound third-party API calls and agent actions.
- Keep **hot loops** in kernel; emit compact events; **no long user-space buffers**.
- Route via **Redis Streams**/**Kafka** for backpressure; **do not block** producers.

3.2 Kernel → Userspace Pipeline

- BPF programs (kprobes/uprobes) write to **per-CPU ring buffers**.
- Userspace reader performs **non-blocking** drains, converts to compact structs, and immediately pushes to **Redis Streams** (node-dedicated) or **Kafka**.
- Achieved sustained export of **millions of events** with host CPU ~**0.7%** under steady load.

3.3 Application-Boundary Interception (No TLS Decryption)

- **Java (JDK 8–11):** ByteBuddy/ASM instrumentation around encrypt/wrap paths (e.g., `SSLCipher` T13 GCM write generator, `OutputRecord`, `SSLEngine`) to access plaintext **before** encryption for **outbound** calls.
- **Node.js/Python:** targeted SSL wrappers and `requests` hooks to tag traffic and headers; attribution occurs **before** payload leaves the process.
- **Filter policy:** exclude service-mesh/Nginx and intra-cluster responses to avoid noise.

3.4 Kubernetes/Containers Coverage

- Deployed as a **privileged DaemonSet** with **hostPID** access and necessary BPF capabilities.
- Mapped host/containers into the agent pod via shared volumes (e.g., `/proc`, container runtime sockets, `/lib/modules`), enabling process discovery and attach.
- Implemented a container mapping layer so a single node-resident agent **observes all containers** on that node.

3.5 Kernel Compatibility Strategy

- Containerized build with **kernel access + shared volumes** for headers/BTF.
- **Dynamic compilation** of probes against the host kernel (`/lib/modules/$(uname -r)/build`), falling back to bundled headers where possible.
- **Python BCC** was compiled from source in the controlled container when required, despite upstream friction, to preserve compatibility across diverse kernels.
- Preference for **libbpf/CO-RE** paths where viable; verifier-friendly helpers and bounded loops.

3.6 Control Plane & Data Contracts

- A Python **controller** distributes **Redis ACLs** and endpoint configs to Java apps at startup and refreshes every **5 minutes** (heartbeat).

- **JSON contracts** with validator-first schema: deterministic field types, amounts/dates normalized (Indian formats), and idempotent **MD5** signatures.
- **Bitmasking** in the nucleus: dense representation of categories, flags, severity, and source to cut memory and improve cache locality; supports O(1) checks.

3.7 Classification & Severity

- **Categories:** API Gateway, Auth, DB, Cache, CDN, Data Pipeline, Elasticsearch, External API, etc.
- **Severity:** Credit card → **critical**; PII → **medium**; Email → **low**.

---

4) AI/ML & CV PIPELINES

4.1 Invoice Vision (India-first)

- OCR stack (PaddleOCR/TrOCR/LayoutLMv3) fused with **Qwen2.5-VL** prompts/validators.
- Robust **PDF→image** rendering and **auto-cropping** that avoids text truncation; strict **JSON** outputs.
- Normalization: `DD-MM-YYYY → YYYY-MM-DD`, tax/amount unification, HSN handling, stamp/annotation heuristics, synonym dictionaries.
- Served via **FastAPI** with background tasks + webhooks; OpenAI-style responses for drop-in adoption.

4.2 Retail Display Scoring (Haier)

- Detection of stands, empty slots, cartons blocking, competitor products, and brand logos; compliance and visibility scoring pipelines.
- Dataset generation and auto-labeling flows with S3/Blob conventions; YOLO fine-tuning readiness.

4.3 GPU Model Serving

- **AWS g5.xlarge (A10G 24GB):** NVIDIA drivers, **vLLM/Ollama** for Qwen2.5-VL and OCR (e.g., **allenai/olmOCR-2-7B-1025**).
- Tuned `--gpu-memory-utilization`, CPU offload, swap, and context length; supervised with Nginx + PM2/systemd; exposed **OpenAI-compatible** APIs.

4.4 Forecasting & Analytics

- Gold retail **Sales Forecasting**: dataset contracts (Sales/Product/Store/Inventory/Prices/Events), baseline demand (LightGBM quantile, TFT/N-HiTS), promo uplift/elasticity, constraint layers.

---

5) CLOUD, DATA, AND DELIVERY

- **AWS:** EC2/EKS, S3 (credential migrations and CloudCube transitions), Amplify, IAM. Reverse proxying with Nginx; `/api` routing.
- **Azure/OCI:** Azure VM crash RCA; Oracle **boot-volume** detach/reattach workflows; Azure MySQL Flexible → Oracle migration planning.

- **Databases:** PostgreSQL admin (e.g., created `haier_dashboard_dbase_2` with owner `app_user`), MongoDB replication without downtime.
- **CI/CD:** Bitbucket Pipelines, GitHub Actions; preference for **non-Docker CI** when simpler; multi-stage Dockerfiles otherwise; artifact promotion.
- **Runtime supervision:** PM2/systemd autostart, health checks, log rotation; hardened Nginx on Ubuntu 22.04.

---

6) TROUBLESHOOTING & DEVELOPER ENABLEMENT

- First-responder for **stuck developers** across Python/Node/infra; created reproducible scripts and one-command runbooks.
- SendGrid deliverability diagnosis (keys, suppression, DNS alignment) when dashboards showed **sent/opened** but inboxes were empty.
- PM2 God Daemon duplication fixes; systemd unit scoping for user services.
- Kaggle/TensorFlow CUDA plugin registration and import issues; stabilized vLLM/OCR containers.
- Auth/VPN support and **bastion** patterns for office/home access while preserving client allow-lists.

---

7) PERFORMANCE, METRICS & BENCHMARKS

- eBPF event export sustained at **millions of records** with host CPU ~**0.7%**.
- GPU model servers kept predictable latency under concurrent requests via context/memory throttles and queue depth limits.
- Nucleus memory footprint reduced using **bitmasking** and compact struct layouts; improved cache hit rates on signature lookups.

---

8) OPERATIONS PLAYBOOKS (EXCERPTS)

- **Brute-force Triage:** identify source / geo; block at SG + host; ban policy; audit keys; rotate creds; notify stakeholders; post-incident review.
- **SSL/Intercept Safety:** intercept only for **outbound** calls; never store secrets beyond minimum; redact PII by policy; retention caps.
- **Kernel Probe Rollout:** staged deploy, verifier dry-run, feature flags, kernel-header availability check, rollback plan.
- **Cluster Coverage:** DaemonSet privileged rollout, hostPID, mount points, capability audit; runtime attach logs; per-node health.

---

9) TOOLS & UTILITIES SHIPPED

- **Fast .txt Finder:** scans Downloads/Documents; matches **all** fuzzy substrings (order-agnostic), returns file list + context; read-only.

- **Agent Backend API:** OpenAI-backed agent with endpoints for prompt execution and safe file/folder listing for UI integration.

---

10) TECH STACK

Python · TypeScript/JavaScript · FastAPI · Django · NestJS · Express · Next.js · PM2 · Nginx · PyTorch · Transformers · vLLM · Ollama · Qwen2.5-VL · Llama-3.x/4 · PaddleOCR · TrOCR · LayoutLMv3 · YOLO · LightGBM · TFT/N-HiTS · Kafka · Redis/Streams · PostgreSQL · MongoDB · Docker · Kubernetes/EKS · AWS · Azure · Oracle Cloud · Linux (Ubuntu/Amazon Linux) · eBPF (BCC/libbpf) · ByteBuddy/ASM · Git · Bitbucket/GitHub Actions · Make · GPU servers (A10G/RTX)

---

11) EDUCATION & CREDENTIALS

(Add degree/institute/year and any certifications: AWS/Azure/K8s/Deep Learning)

---

12) APPENDICES (ADD ON DEMAND)

- Architecture diagrams (Agent→Nucleus→Kafka/Redis→Backend)
- JSON Schemas (payloads, signatures, bitmask maps)
- K8s DaemonSet manifest (privileged/hostPID, mounts)
- BPF attach-point list and verifier notes
- Incident runbooks and checklists

Status: Updated November 2025. Ready to export as PDF/DOCX on request.