## Observability Architecture Proposals (AKS + Azure)

**Author:** S  
**Date:** 13 Aug 2025  
**Scope:** AKS workloads plus Azure PaaS (Logic Apps, Function Apps, App Services) using OSS tools: OpenTelemetry, OTel Collector, Prometheus, Loki, Jaeger, Grafana.

## Executive Summary

We have two viable paths to centralize monitoring and visualization after path-rewrite issues through APIM impacted Grafana access:

* **Option A — Centralized VM Stack (current draft):** Move the full OSS stack (OTel Collector, Prometheus, Loki, Jaeger, Grafana) to an Azure VM gateway tier. All apps and clusters push to the VM’s collector and backends.

**NOTE:** Simple topology but introduces a single point of failure unless you build HA around the VM and storage.

* **Option B — Distributed Data Planes + Central Grafana (recommended):** Keep data-plane components (collectors and storage backends) in each cluster, **securely expose them via TLS/mTLS Ingress** and **authenticate with Entra ID**. Run **Grafana on an external VM** only as a query/alert control plane that connects to each datasource. Improves resilience (no central SPOF), keeps data at source, and scales horizontally.

*Note: I recommend* ***Option B****. It minimizes blast radius, avoids data loss if the VM/Grafana is down, and reduces cross-tenant/network chokepoints.*

## Non‑Functional Requirements (NFRs)

* **Availability:** Target 99.9% for visualization plane, 99.5%+ for data-plane per cluster, no single regional SPOF.
* **Security:** mTLS for telemetry ingestion, TLS 1.2+ everywhere; OIDC/OAuth2 (Azure Entra ID) SSO; least-privilege; private networking/peering; per-tenant tokens in Loki/Prometheus where applicable.
* **Performance:** p50 < 200ms query latency intra‑VNet; p95 < 800ms cross‑cluster queries; ingestion up to N events/sec (size by environment below).
* **Retention:** Metrics 7-15 days hot; Logs 7-14 days hot; Traces 3-7 days hot (sampling 5-10% default). Long-term is optional via archive/export.
* **Operability:** GitOps configs (Helm/Kustomize), IaC for VM + networking; dashboards & alerts provisioned as code; health probes & SLOs.
* **Cost:** Optimize egress by local writes; use SSD for WAL/TSDB and blob for chunks/indices where feasible.

## Current Constraints & Context

* APIM path rewrite and layering broke Grafana UI and some plugin endpoints.
* Need to visualize multi-source data: AKS clusters, Logic Apps, Function Apps, App Services, etc.
* Initial all-in-one cluster approach worked functionally but suffered from upstream routing transformations.

## Option A — Centralized VM Observability Stack (Current Proposal)

### High-Level Diagram

**Apps & PaaS → OTel SDK/exporters → AKS OTel Collector (DaemonSet/Sidecar) → mTLS/OTLP → VM OTel Collector (gateway)**

**→ Prometheus (metrics) → Grafana**

**→ Loki (logs) → Grafana**

**→ Jaeger (traces) → Grafana**

### Components

* **Azure VM (Obs Gateway & Backends):**
* OTel Collector (gateway)
* Prometheus (scrape & remote\_write endpoints)
* Loki (log ingestion)
* Jaeger (collector + query + UI)
* Grafana (dashboards/alerts)
* **In Cluster:** OTel Collector (DaemonSet or sidecar) for node/app scraping/export.

## Sizing & Ports (initial)

* **VM sizes:** Medium (D8s\_v3) or Large (D16s\_v3) depending on environment.
* **Ports (inbound):** 4317/4318 (OTLP), 9090 (Prometheus), 3100 (Loki), 14268 & 16686 (Jaeger), 3000 (Grafana).
* **Storage layout:**
  + /data/loki: 200-500 GB SSD, 7-14d
  + /data/prom: 50-150 GB SSD, 7-15d
  + /data/jaeger: 20-50 GB SSD, 3-7d
  + /data/grafana: 20-50 GB

## Network & Security

* Same VNET or peered; dedicated subnet and NSG; private DNS.
* mTLS end‑to‑end for OTLP; Grafana OIDC with Entra ID.
* Admin access via VPN/JIT and hardened OS baseline.

## Pros

1. Single, simple landing zone for ingestion and querying.
2. Minimal change to app/cluster agents (single OTLP target).
3. Easier to grant external teams read‑only Grafana access.

## Cons / Risks

1. **SPOF risk:** If the VM stack fails, all ingestion and query can be impacted.
2. Horizontal scale requires sharding and multiple VMs (more work).
3. Higher cross‑network dependency and potential backpressure under spikes.
4. Local TSDB/indices can be a bottleneck without careful I/O tuning and backups.
5. Grafana downtime removes visibility **and** alert evaluation if run on same host.

## HA Enhancements (if you proceed with A)

1. **Active/Active VMs** behind an **Azure Load Balancer** for OTLP 4317/4318.
2. **Split roles:** Move Grafana off the data VM or run 2+ Grafana instances with **PostgreSQL** backend (not SQLite) for HA.
3. **Stateful backends:**
   1. Prometheus HA pair + federation; scheduled snapshots to Azure Files/Blob.
   2. Loki with chunk store on Azure Blob + SSD for WAL; run 2+ ingesters/queriers (containerized) and a simple distributor.
   3. Jaeger: use Kafka (optional) or persistent storage for collectors; run query/UI separately.
4. **Back-pressure control:** OTel Collector **queue\_retry** + **memory\_limiter** processors; rate limits per tenant.
5. **Backups:** Daily snapshots of /data/\* and Grafana DB; IaC for rebuild in < 2 hours.

## Option B — Distributed Data Planes + Central Grafana (Recommended)

### High-Level Diagram

**Cluster A Cluster B PaaS Apps**

**│ │ │**

**[OTel] [OTel] [OTel]**

**│ │ │**

**[Prom] [Prom] (OTLP → cluster collector)**

**[Loki ] [Loki ]**

**[Jaeger] [Jaeger]**

**│ │**

**└── TLS/mTLS Ingress (private) ──► Grafana (VM, HA) ◄── other data sources (e.g., Azure Monitor, CI/CD)**

## Key Principles

* **Keep data where it is produced.** Each cluster runs its own **OTel Collector, Prometheus, Loki, and Jaeger** with appropriate retention.
* **Expose only query endpoints** (Prometheus /api/v1, Loki /loki/api, Jaeger query) via **Ingress** with **TLS** and **mTLS** from Grafana to the data sources.
* **Centralize only the control plane (Grafana)** on a VM (or small VM set) for dashboards, alert rules, and SSO.

## Secure Exposure Pattern

1. **Ingress Controller:** NGINX Ingress Controller or AGIC without path rewrites for data sources.
2. **Certificates:** cert‑manager issuing per‑service certs; client certs for Grafana; TLS 1.2+.
3. **Auth:**

* Grafana SSO with **Azure Entra ID** (Teams/Groups for RBAC).
* **Data source protection:** mTLS + IP allowlist (VNet peering) + optional basic/OAuth2 via oauth2‑proxy at the ingress layer.

1. **Networking:**

* **Private IPs** only for backends; Grafana VM in same VNET/peered VNets; Private DNS A records per service.
* Bypass APIM for Grafana and data-source paths to avoid rewrite issues.

## Implementation Details

1. **Grafana VM:**

* 2 small/medium VMs or a VMSS (HA) with **PostgreSQL** as the Grafana database; dashboards & datasources provisioned as code.

1. **Prometheus per cluster:**

* Scrape in cluster; 7-15d retention.
* Optional: **federation** or **remote\_write** to a long‑term store later (kept out of scope for now to stay OSS‑pure).

1. **Loki per cluster:**

* Ingest via Promtail/OTel; 7-14d retention.
* Optional: use Azure Blob for chunks to reduce node SSD pressure.

1. **Jaeger per cluster:**

* OTel→Jaeger collector; 3-7d hot storage with sampling 5-10%.
* Query endpoint exposed via Ingress for Grafana plugin.

1. **OTel Collector:**

* In-cluster **DaemonSet** (preferred) for hosts + app sidecars where necessary.
* **Processors:** batch, memory\_limiter, resource detection; **exporters:** prometheusremotewrite, loki, jaeger.

## Pros

1. **Resilience:** Loss of Grafana does not impact ingestion or data retention.
2. **No central SPOF:** Each cluster continues to collect and store locally.
3. **Scalability:** Add clusters without re‑architecting the center; per‑cluster SLOs.
4. **Security:** Minimal surface; mTLS and private endpoints; avoids APIM rewrite issues entirely.
5. **Cost control:** Limits cross‑network egress; right‑size per cluster.

## Cons / Trade‑offs

1. Multiple datasources to manage in Grafana (solved by provisioning-as-code).
2. Cross‑cluster panels can have higher query fan‑out latency.
3. Need consistent schema/labels across clusters for clean dashboards.

## Access & AuthZ Strategy

1. **Grafana:** Entra ID OIDC; map groups to folders/teams; disable local sign‑ups; enforce MFA via Entra policy.
2. **Data sources:**
   1. mTLS client certs (Grafana→datasource).
   2. Per‑cluster API tokens (Loki/Prometheus) with least‑privilege scopes.
   3. Network ACLs restrict to Grafana VM subnet.

## Storage & Retention

1. Keep hot retention small (7-15 days) per cluster; push long‑term to archive later if needed.
2. Filesystems with node‑local SSDs for WAL; consider blob/object storage for chunks/indices to reduce risk on node failover.

## HA, DR, and Backups

1. **Grafana:**
   1. 2 instances behind NLB; PostgreSQL single‑AZ with PITR or Flexible Server HA.
   2. Nightly backup of dashboards (JSON) + provisioning repos.
2. **Per‑cluster backends:**
   1. **Prometheus:** double‑replica; PVC with fast SSD; snapshot to Azure Files weekly.
   2. **Loki:** min 3 replicas for querier/ingester in prod; WAL on SSD; chunks to blob (optional).
   3. **Jaeger:** at least 2 collectors; consider external storage if trace volume grows.
3. **Disaster recovery:** Rehydrate Grafana from Git + DB backup; clusters continue operating independently.

## Networking Reference

1. **Avoid APIM** for Grafana and data-source paths. Use dedicated DNS if possible
2. Private DNS zone linked to VNets; no public exposure.
3. NSG rules: allow only required ports from Grafana subnet to cluster nodepools.

## Rollout Plan

1. **Foundations:** Create VNET peerings, subnets, NSGs, Private DNS zones.
2. **Grafana VM:** Provision HA pair; attach PostgreSQL; configure Entra ID SSO; enable backup.
3. **Cluster Baseline:** Ensure OTel Collector, Prometheus, Loki, Jaeger running with desired retention; standardize labels.
4. **Ingress Security:** Issue certs; apply Ingress for Prom/Loki/Jaeger with mTLS; validate from Grafana VM.
5. **Datasource as Code:** Commit Grafana datasources.yml and folder/permissions; import curated dashboards.
6. **Cutover:** Point users to Grafana VM URL; decommission APIM path to old Grafana; monitor SLOs.
7. **Hardening & Scale:** Add rate limits, alert noise reduction, and per‑env RBAC.

## Cost Considerations

1. **Option A:** Larger VM(s), higher SSD, and ops for HA; simpler per‑cluster footprint.
2. **Option B:** Modest Grafana VM(s) + storage per cluster; reduced egress; scales linearly with number of clusters.

## Risk Register & Mitigations

* **Path rewrite regression:** Bypass APIM, test SSO and callbacks explicitly.
* **Query fan‑out latency:** Use dashboard variables to filter scope, set sensible panel timeouts.
* **Label cardinality explosion:** Enforce label hygiene in OTel pipelines and scrape configs.
* **Trace volume spikes:** Apply tail‑based sampling in OTel Collector.
* **Secrets handling:** KMS/Key Vault integration, short‑lived tokens, cert rotation policy.

## Final Recommendation

Adopt **Option B (Distributed Data Planes + Central Grafana)**. It prevents a central SPOF, keeps ingestion resilient, and aligns with security best practices (mTLS, private networking, SSO). Reserve **Option A** for constrained environments or interim phases, and only with explicit HA additions (ALB, storage externalization, and split roles).