OpenSearchCon EUROPE

Distributed OpenSearch Monitoring at Scale with Apache NiFi and MiNiFi Agents

OpenSearchCon 2026
*Vincenzo Lombardo - **Seacom***

About me

Vincenzo Lombardo - Operations Manager @Seacom

- Specialized in data orchestration and ETL processes in Seacom
- Lead a team focused on the NiFi ecosystem (NiFi, MiNiFi, NiFi Registry, C2 Server, NiFiKop, etc.)
- Background in enterprise search engine technologies, such as OpenSearch, Elasticsearch, Google Search Appliance, Mindbreeze

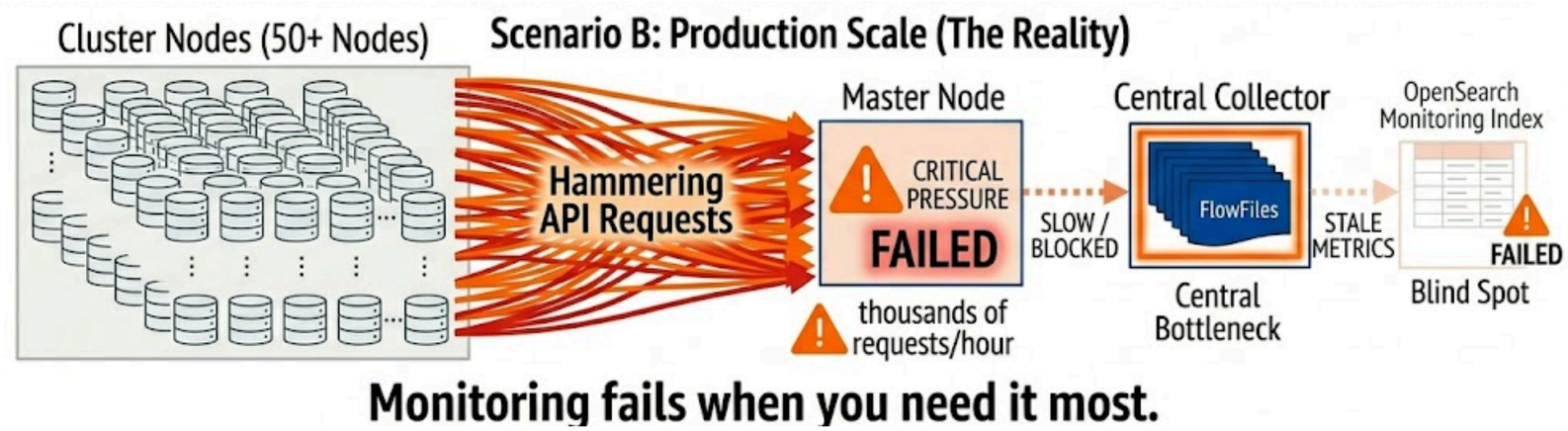
vincenzo.lombardo@seacom.it

https://shorturl.at/FwOT1

The Problem: Centralized Monitoring Doesn't Scale

- **Small Scale:** 1 collector · 5 nodes · requests every 10s → few requests, everything is fine.
- **Production Scale:** 1 collector · 50 nodes · requests every 10s → thousands of requests per hour hitting the Master node.
- **Critical Pressure:** Master node under pressure exactly when the cluster is struggling (e.g., heavy rebalancing).
- **Central Bottleneck:** the collector struggles with fan-in; processing delays lead to stale metrics.
- **The Blind Spot:** Monitoring fails when you need it most. If the API is unresponsive, you are flying blind during a disaster.

This is not a configuration but an architectural problem



Apache NiFi – The Central Orchestrator

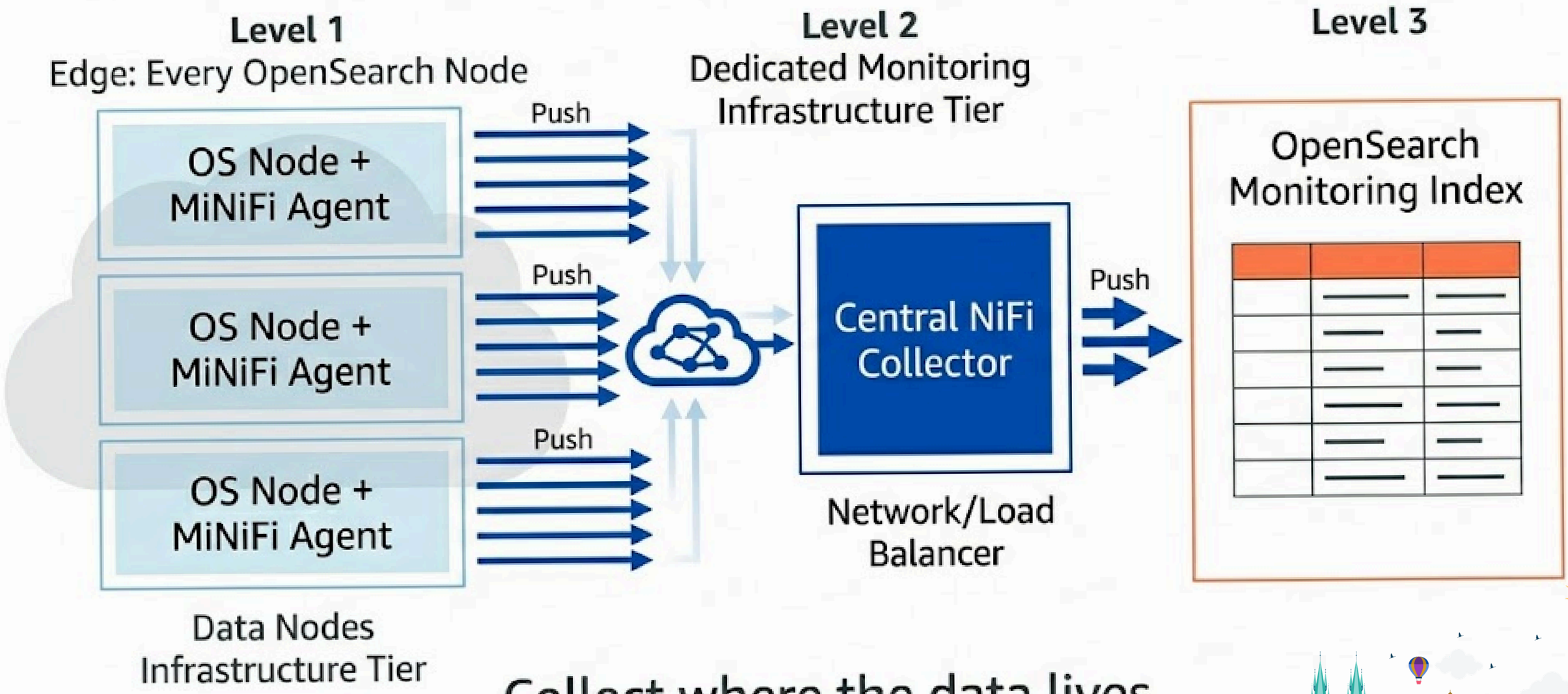
- **Visual Dataflow Engine:** A powerful platform for real-time data orchestration and distribution.
- **No-Code Interface:** Pipelines are built on a visual canvas by connecting pre-built or custom processors.
- **Central Management:** Acts as the command center for all incoming data streams from the edge and data sources.
- **Agnostic Ingestion:** Capable of routing data from and to hundreds of different destinations simultaneously.

Apache MiNiFi Java – The Edge Agent

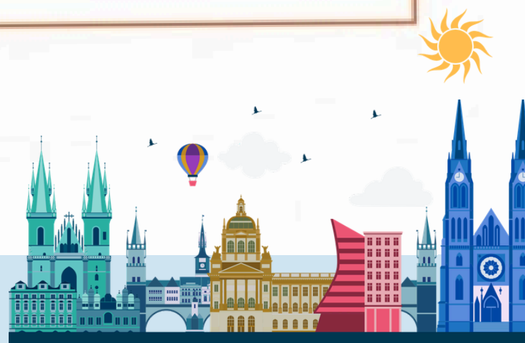
APACHE NIFI minifi

- **Lightweight Foundation:** A sub-project of Apache NiFi, specifically designed for edge computing and data collection.
- **Low Footprint:** A small-footprint Java agent that runs side-by-side with your applications.
- **Edge Processing:** Capable of filtering, transforming, and prioritizing data locally before transmission.
- **Proven Protocol:** Uses the native NiFi Site-to-Site (S2S) protocol to push data securely and reliably

The Architecture: Push, Don’t Pull



Collect where the data lives



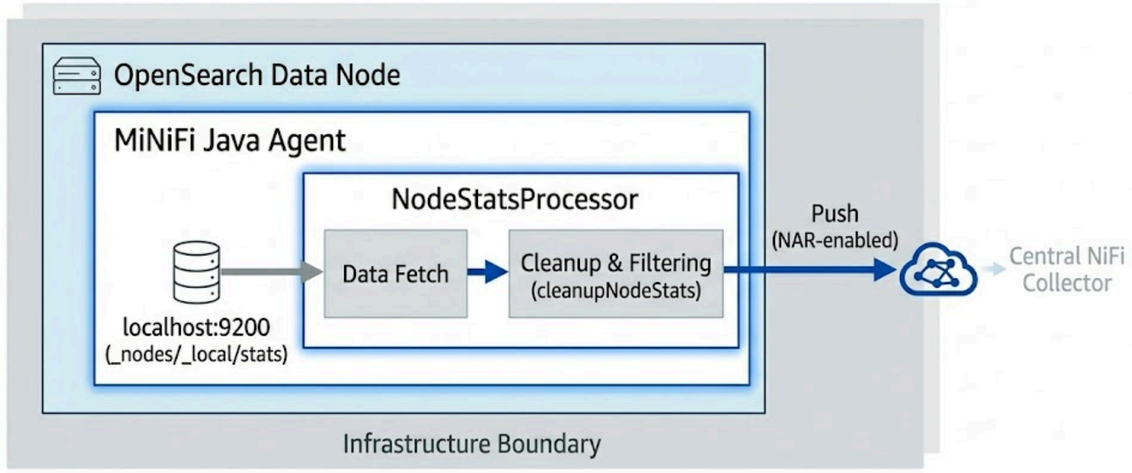
Edge Intelligence: NodeStatsProcessor

OpenSearchCon EUROPE

How it works

- Calls _nodes/_local/stats, localhost only with zero cluster API calls
- Strips redundant fields before sending
- Supports ${HOSTNAME} as automatic node label
- Runs inside MiNiFi on every node

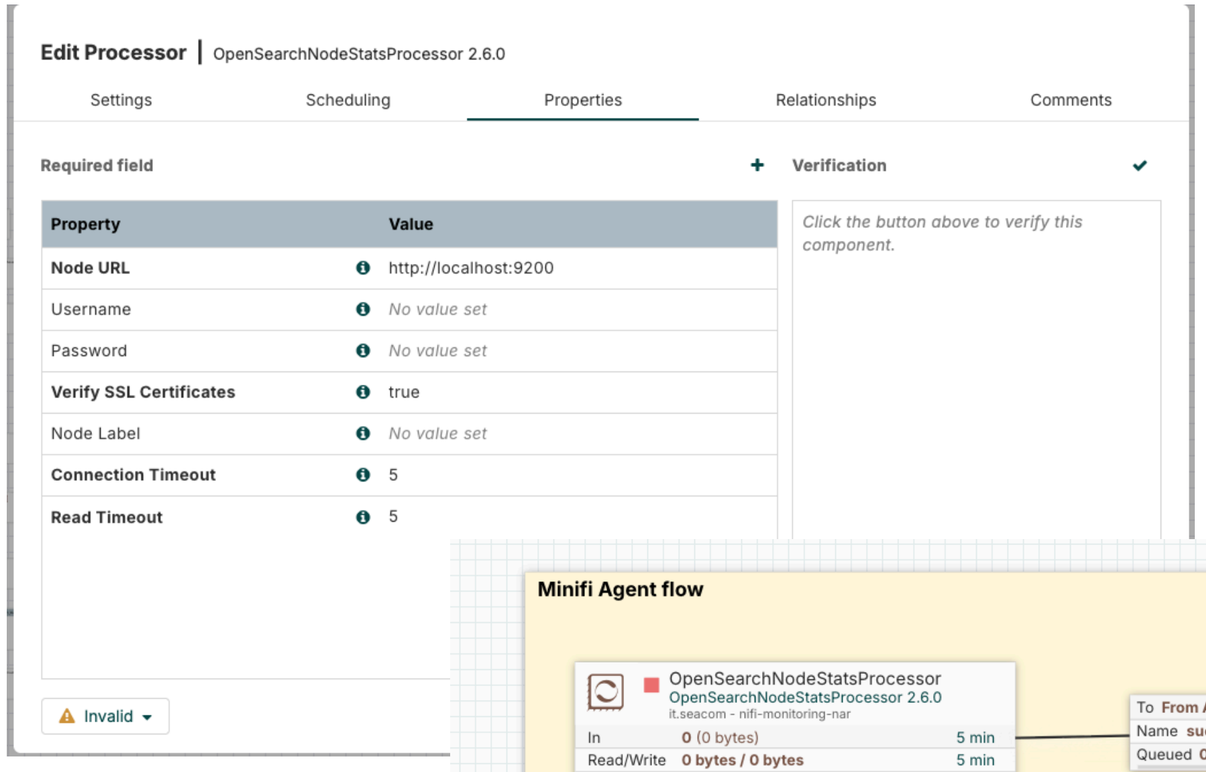
"Works even when the cluster is degraded — you get metrics precisely when you need them most"



What it collects

- indices
- os.cpu
- process.cpu
- jvm.mem
- thread_pool
- fs
- transport
- http
- breakers
- script

OpenSearchNodeStatsProcessor



Cluster-Wide Stats: One Request ONLY

OpenSearchCon EUROPE

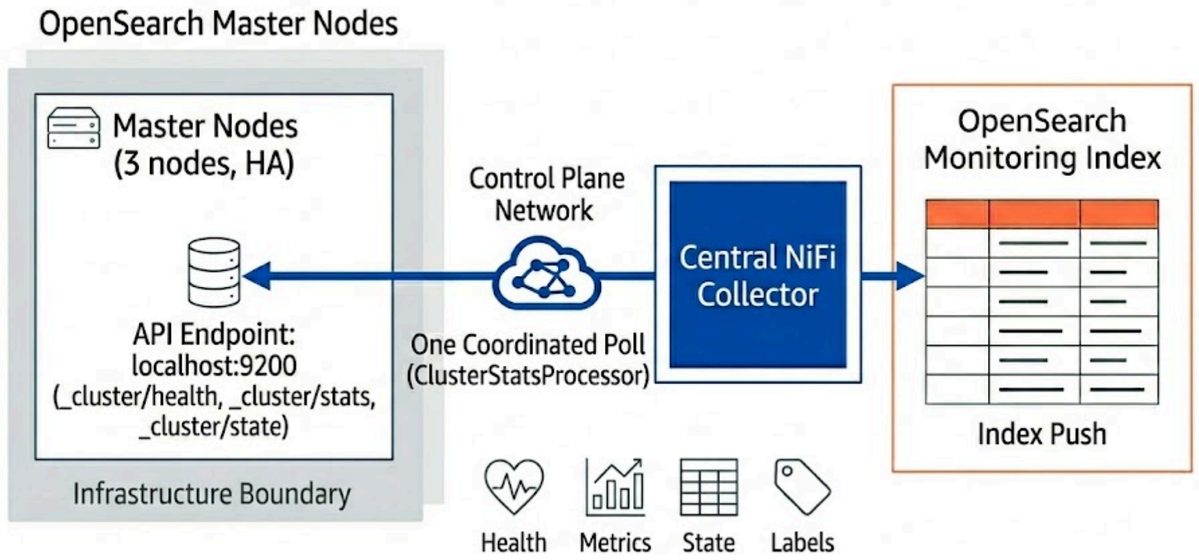
Deploy on NiFi cluster

Three API Calls

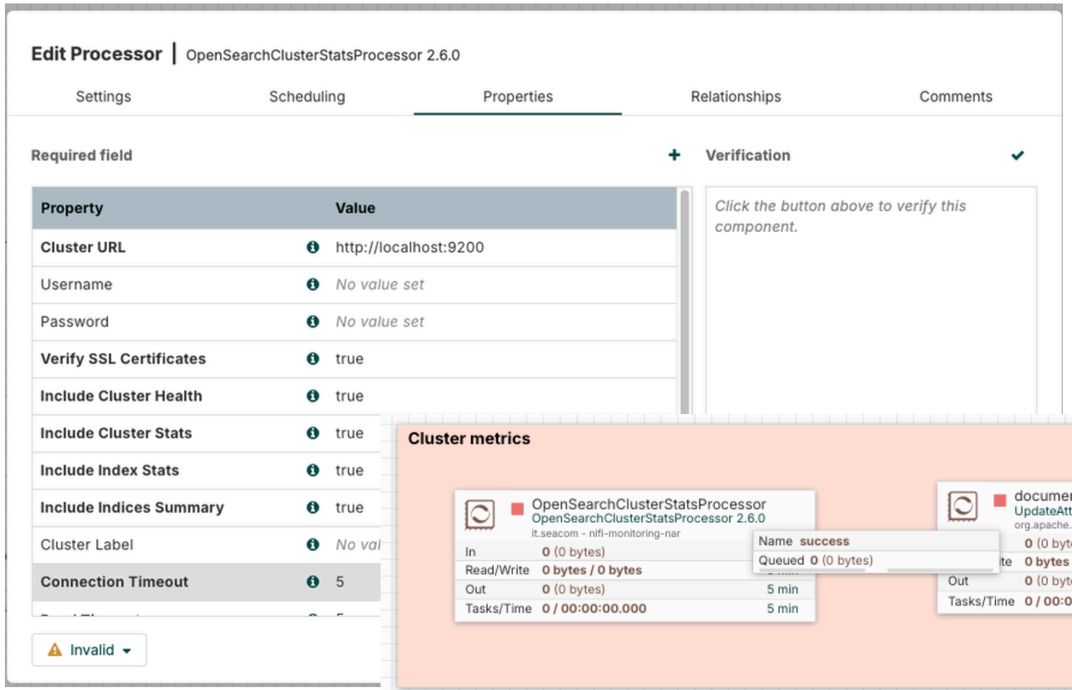
- _cluster/health => (green, yellow or red)
- _cluster/stats => aggregate cluster metrics
- _cluster/state => full shard routing table (primaries, replicas, unassigned, relocating)

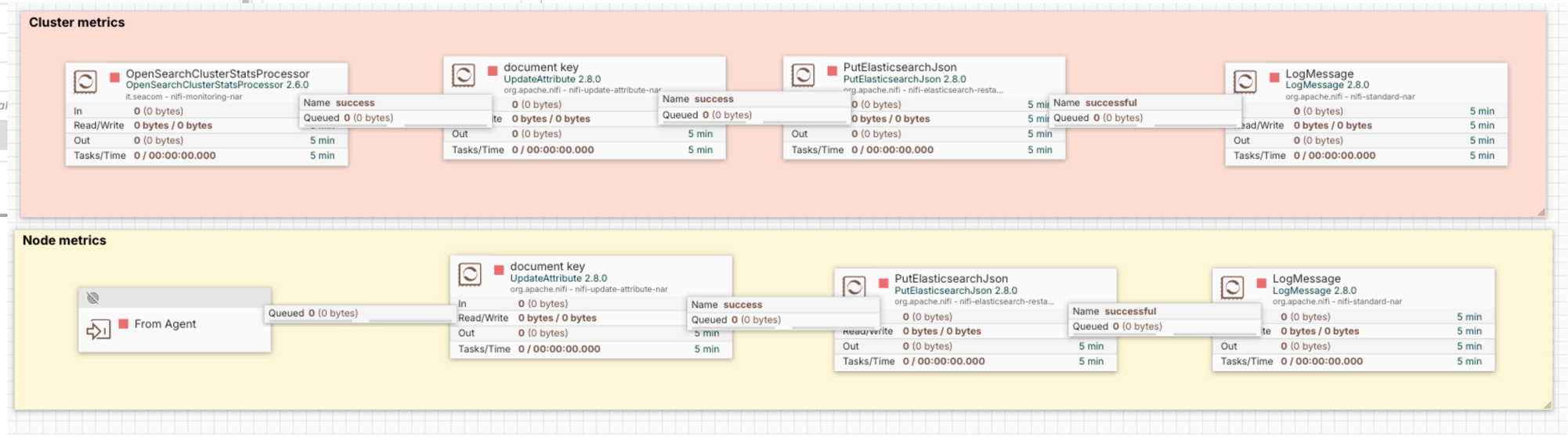
What you get for index

- active primaries & replicas shards
- unassigned shards
- inizializing/relocating shards
- cluster_label tag => multi cluster ready



OpenSearchClusterStatsProcessor





Resilience & Backpressure

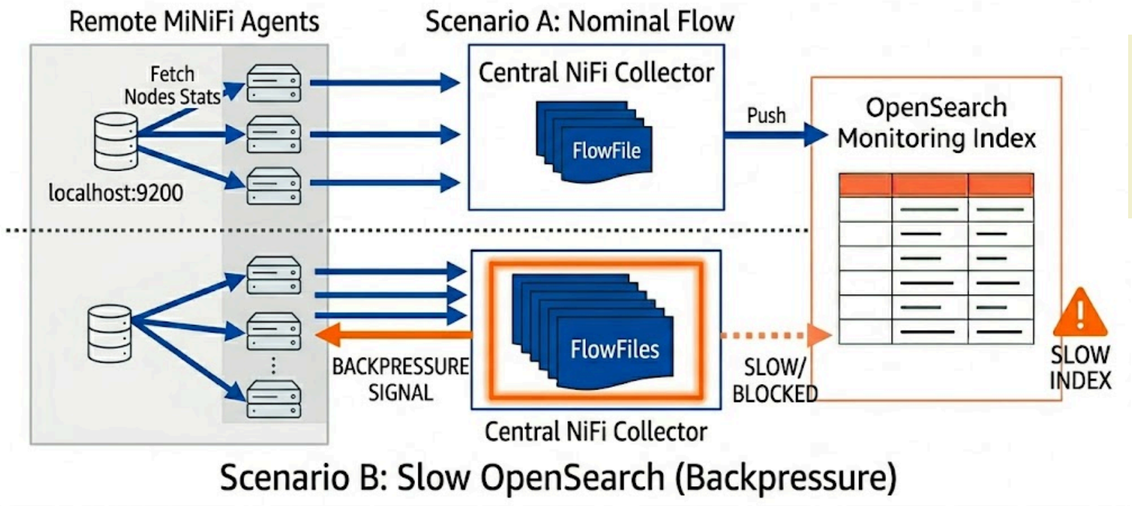
OpenSearchCon EUROPE

Normal flow

- MiNiFI collect edge metrics and sends to NiFi
- NiFi receives metrics from MiNiFi and sends to OpenSearch
- NiFi get cluster's metrics and sends to OpenSearch

Under pressure

- OpenSearch slow => NiFi signals backpressure
- MiNiFi buffers flowfiles on local disk
- Once cleared => data flows again, in order
- Result: 99,9 % delivery SLA



Traditonal setup: slow OpenSearch => data loss.

NiFi setup: slow OpenSearch => data queued

Deployment strategy*

***Currently validating on production cluster**

Bare Metal / VM

- MiNiFi as systemd service
- 256MB heap, G1GC opensearch
- Cap with cgroups

docker

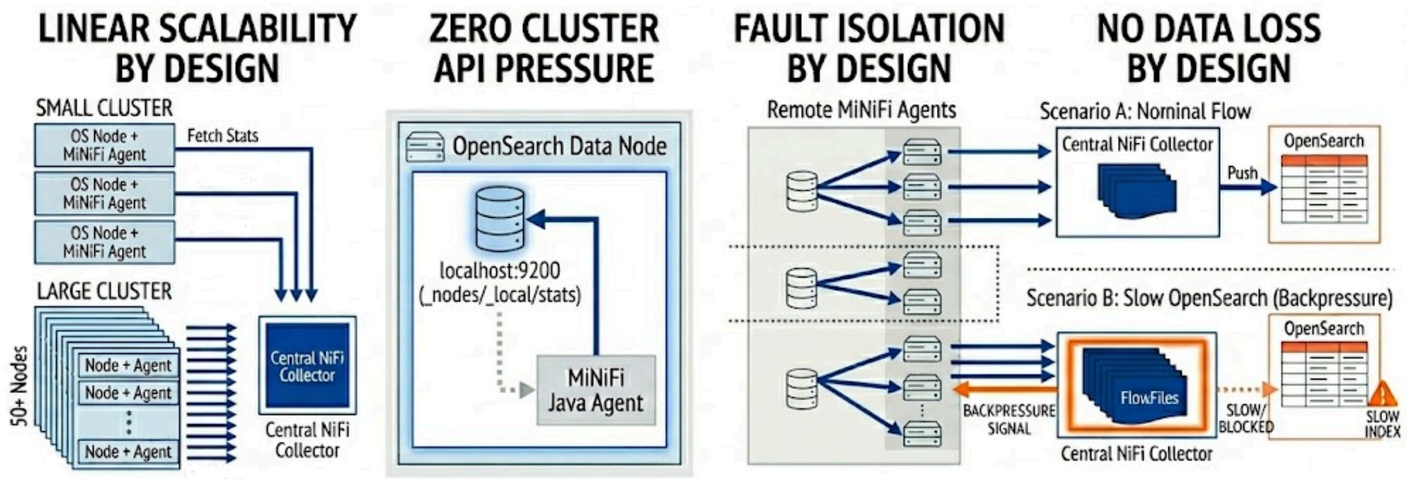
- Separate container per node
- Communicate over loopback
- Persistent volume required for backpressure buffering

kubernetes

- MiNiFi as sidecar container in OS pod
- Lifecycle coupled to OS node => no orphan agents

Why This Architecture Works

- **Linear scalability by design:** one agent per node, fully independent. Adding a node means adding one MiNiFi instance. Nothing else changes.
- **Zero cluster API pressure:**_nodes/_local/stats is a local call. The cluster API is never touched by node-level collection.
- **Fault isolation by design**: one agent failing affects only that node's metrics. No cascading failures, no monitoring blackouts.
- **No data loss by design:** NiFi backpressure queues FlowFiles on disk when downstream is slow. Data flows again when the bottleneck clears.



Deployment & Rollout Strategy

- **Central Flow Governance:** Design the ingestion logic on the NiFi cluster and publish it as a versioned "Golden Template" via NiFi Registry.
- **Local Pilot**: Deploy a single MiNiFi agent on one representative node to validate the resource impact and metrics schema.
- **Automated Fleet Distribution:** Roll out the MiNiFi configuration across all nodes using your existing Configuration Management tools
- **Shadow Ingestion Phase:** Run the new pipeline in parallel with the legacy system for some time to ensure data consistency before decommissioning

Lessons Learned

- **Treat MiNiFi flows like code:** version control via NiFi Registry. Without it you're SSH-ing to 50 nodes for every config change.
- **Idempotent writes from day one:** deterministic doc IDs: nodeId + epoch bucket. Retrofitting deduplication is painful.
- **Start small:** migrate your chattiest metrics first. Validate shape and throughput on one node before scaling.
- **Monitor your monitoring:** alert on NiFi collector queue depth via REST API. A growing queue is your early warning system for downstream problems.

3 things to remember

OpenSearchCon EUROPE

- Collect where the data lives
- Push, don't pull
- Monitor your monitoring

Q&A


OpenSearchCon
EUROPE