SEIDOR

IBM i Observability & Monitoring

Prometheus · VictoriaMetrics · Grafana · Manzan · Loki

A production-ready monitoring pattern for IBM i specialists

Nicolae Chirea
System Architect
Seidor Iberia
nchirea@seidor.com

9853	2 years	Loki	Grafana
metric endpoint	metrics retention	logs + events	console

#whoami

Nicolae Chirea

- Infrastructure solutions architect – design and implement solutions based on IBM Power and IBM Storage

- Covering (What I mostly do)
  - IBM Power system - PowerVM
  - IBM i, Linux, AIX
  - Block storage: Flash Systems, SVc and DS8K
  - AI on Power
  - HA, Performance
  - Open-Source on IBM i
  - Automation: Ansible, PowerVC

- 25+ years of working experience with IBM i

- Joined Seidor in 2007 as IBM Power specialist
  - IBM i – HA/DR solutions, Performance analysis, DB2 for i
  - Linux – HANA, DB2 Warehouse
  - IBM Power automation projects – infrastructure provisioning, HA/DR management, security

- CEAC member since 2025

Nicolae Chirea | LinkedIn

Power servers | SEIDOR

Agenda

From business drivers to implementation patterns, operations and roadmap.

01 Architecture and data flows

02 Prometheus + IBM i JDBC exporter

03 VictoriaMetrics and retention design

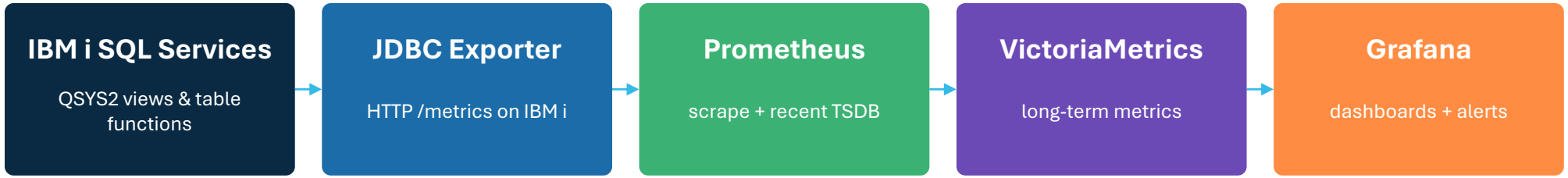
04 Grafana dashboards and alerting

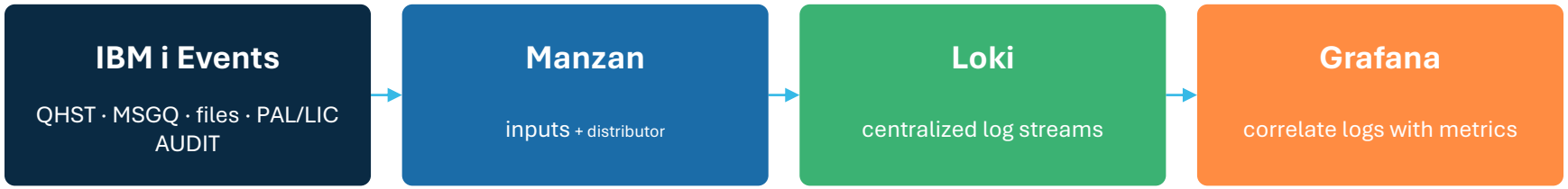
05 Manzan + Loki for events and logs

06 Container deployment, security and rollout

Target architecture at a glance

Metrics and events follow different paths but converge in Grafana for operators.

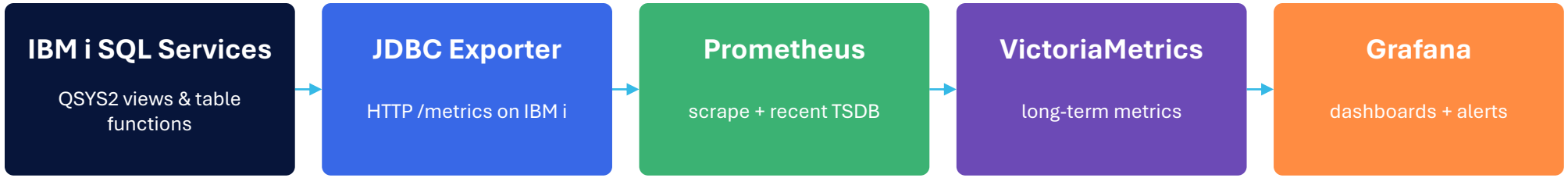


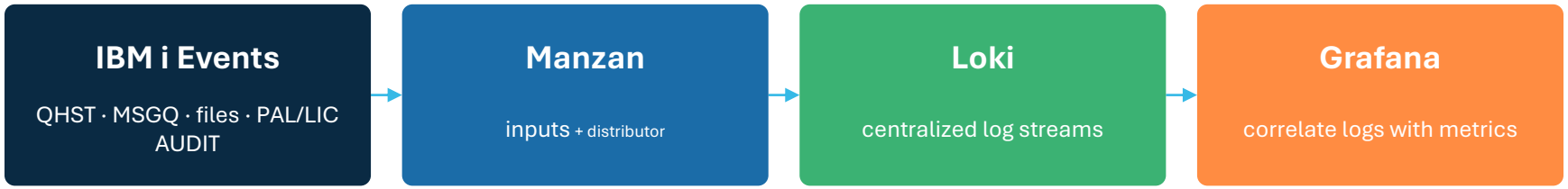


Operational loop: Alert → Dashboard → Drill-down → Logs → Corrective action

Target architecture at a glance

Metrics and events follow different paths but converge in Grafana for operators.

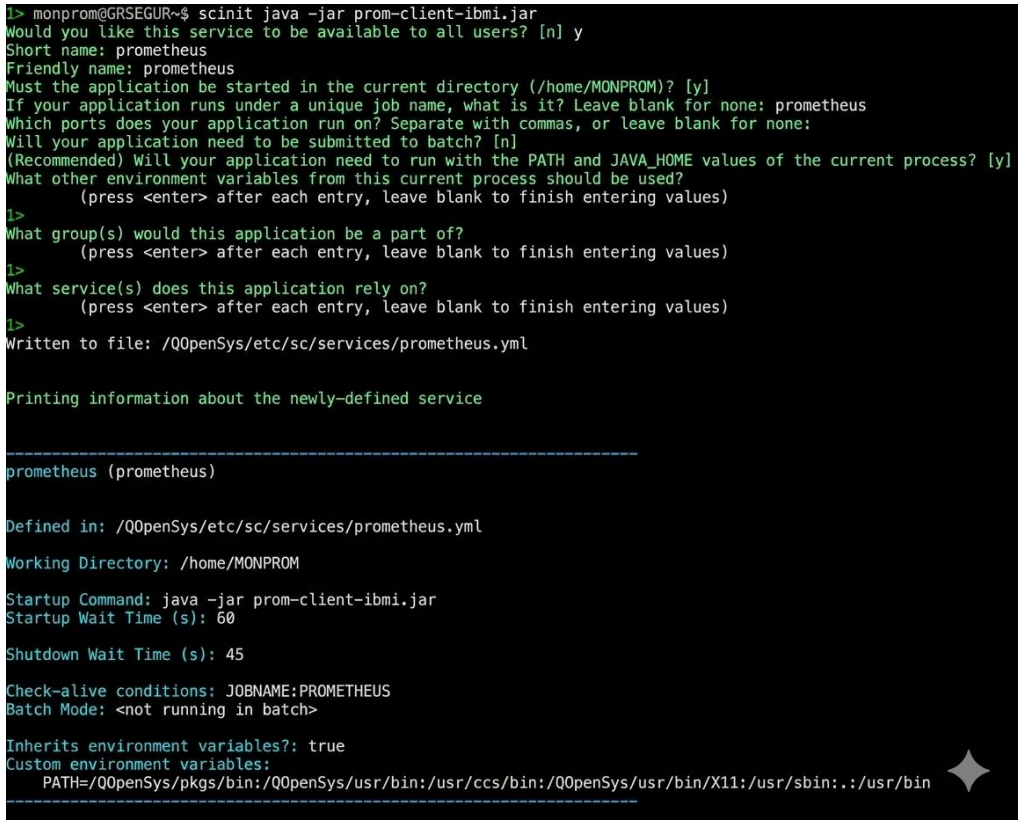




Operational loop: Alert → Dashboard → Drill-down → Logs → Corrective action

IBM i JDBC Exporter: Installation and Startup

Highly recommended – configure from bash, not from qsh or PASE



1. User profile and working directory

```
CRTUSRPRF USRPRF(MONPROM) PASSWORD() TEXT('Monitorizacion Prometheus')
CHGUSRPRF USRPRF(MONPROM) SPCAUT(*JOBCTL)
mkdir /home/MONPROM
Password: *NONE
```

2. Required tools and exporter download

```
Install: Service commander, nano, wget
wget https://github.com/ThePrez/prometheus-exporter-
jdbc/releases/download/v1.0.2/prom-client-ibmi.jar
java -jar prom-client-ibmi.jar
scinit java -jar prom-client-ibmi.jar
```

3. Job Scheduler entries for controlled start/stop

```
ADDJOBSCDE JOB(STRMONPROM) CMD(qsh CMD(('/home/MONPROM/startprom'))) FRQ(*WEEKLY)
SCDDATE(*NONE) SCDDAY(*ALL) SCDTIME('07:00:00') JOBQ(QSYSNOMAX)
ADDJOBSCDE JOB(STPMONPROM) CMD(qsh CMD(('/home/MONPROM/stopprom'))) FRQ(*WEEKLY)
SCDDATE(*NONE) SCDDAY(*ALL) SCDTIME('06:55:00')
HLDJOBSCDE JOB(STPMONPROM)
```

Operational note: in production, validate JAVA_HOME/PATH inheritance, confirm the service name, and test the startprom/stopprom wrapper scripts before enabling automatic scheduling.

IBM i metrics collection: SQL as the source of truth

The exporter turns IBM i SQL Services into Prometheus metrics without a heavy proprietary agent.

example collector SQL

```
{
  "port": 9853,
  "include_hostname": true,
  "queries": [{
     "name": "System Statistics",
     "interval": 120,
     "enabled": true,
     "prefix": "STATS",
     "sql": "SELECT * FROM TABLE(QSYS2.SYSTEM_STATUS(RESET_STATISTICS=>'YES',DETAILED_INFO=>'ALL')) X"
   },
   {
     "name": "System Activity",
     "interval": 120,
     "prefix": "SYSACT",
     "enabled": true,
     "sql": "SELECT * FROM TABLE(QSYS2.SYSTEM_ACTIVITY_INFO())"
   },
  {
```

Passive scrape model

Prometheus controls the scrape cadence. IBM i exposes a metrics endpoint rather than pushing samples.

Transparent metric definitions

Collectors are JSON entries with name, interval, prefix and SQL. Numeric SQL results become Prometheus gauges.

Extendable beyond infrastructure

Custom SQL lets teams publish application, business and service indicators with the same pattern.

Performance-aware scheduling

Intervals can differ per query; expensive collectors can run less often than critical indicators.

Recommended IBM i metric domains

Start with high-signal operational domains, then add application-specific SQL collectors.

System health

CPU, ASP usage, temporary storage, jobs, threads

Memory pools

Current size, defined size, active threads, pool pressure

Workload & jobs

Batch backlog, subsystem load, critical server jobs

Database signals

Plan cache, SQL CPU, journaling, query activity

Network & services

TCP established connections, HTTP server, Liberty/WAS

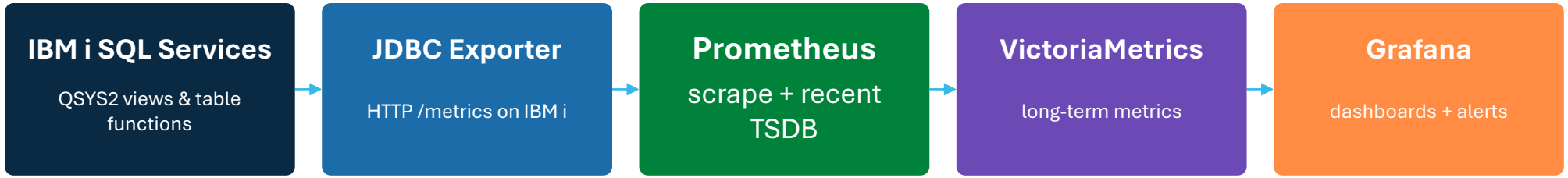
Power/HMC

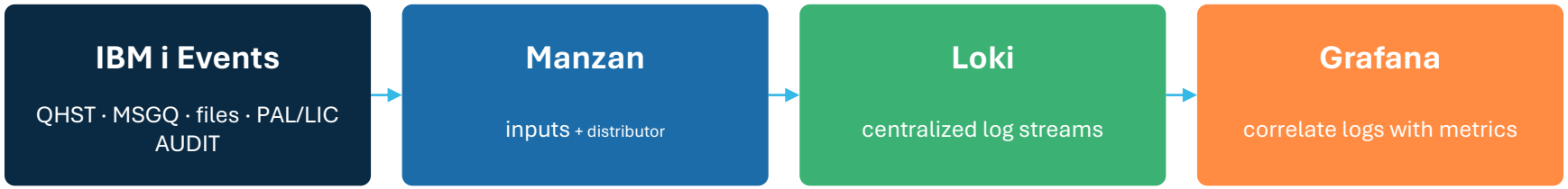
LPAR status, processor allocation, energy where available

Design rule: every metric should have an owner, a query source, a collection interval, and a decision it supports.

Target architecture at a glance

Metrics and events follow different paths but converge in Grafana for operators.



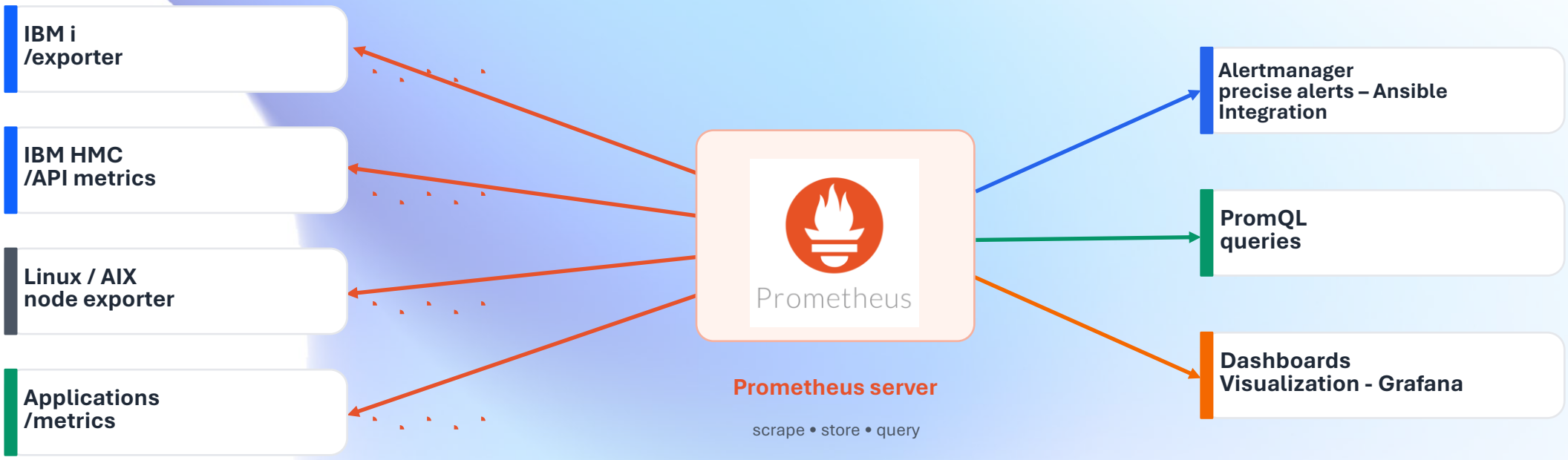


Operational loop: Alert → Dashboard → Drill-down → Logs → Corrective action

Prometheus

Open source metrics and monitoring for your systems and services

Instrument, collect, store, and query time-series metrics for alerting, dashboards, and operations.

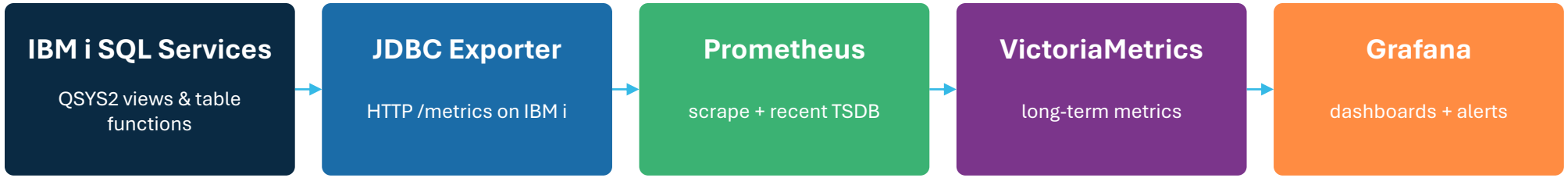


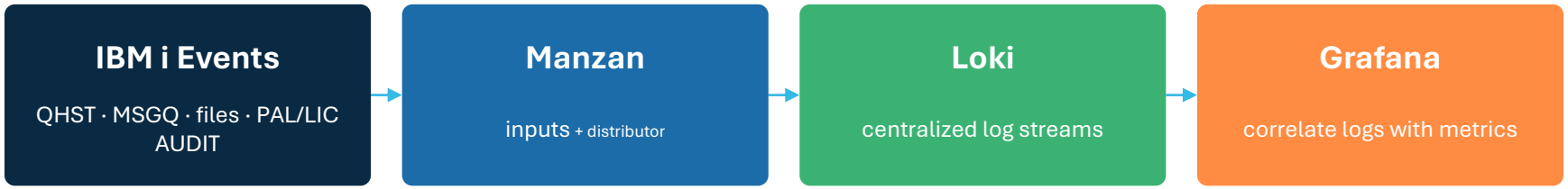
Dimensional time-series model using metric names plus key-value labels.
Open source monitoring toolkit designed for reliability and operational simplicity.
HTTP pull model scrapes targets/exporters and stores samples locally.
PromQL + alerting support dashboards, SLOs, and incident response.

Based on Prometheus concepts: time series with labels, PromQL, and scraping targets over HTTP.

Target architecture at a glance

Metrics and events follow different paths but converge in Grafana for operators.

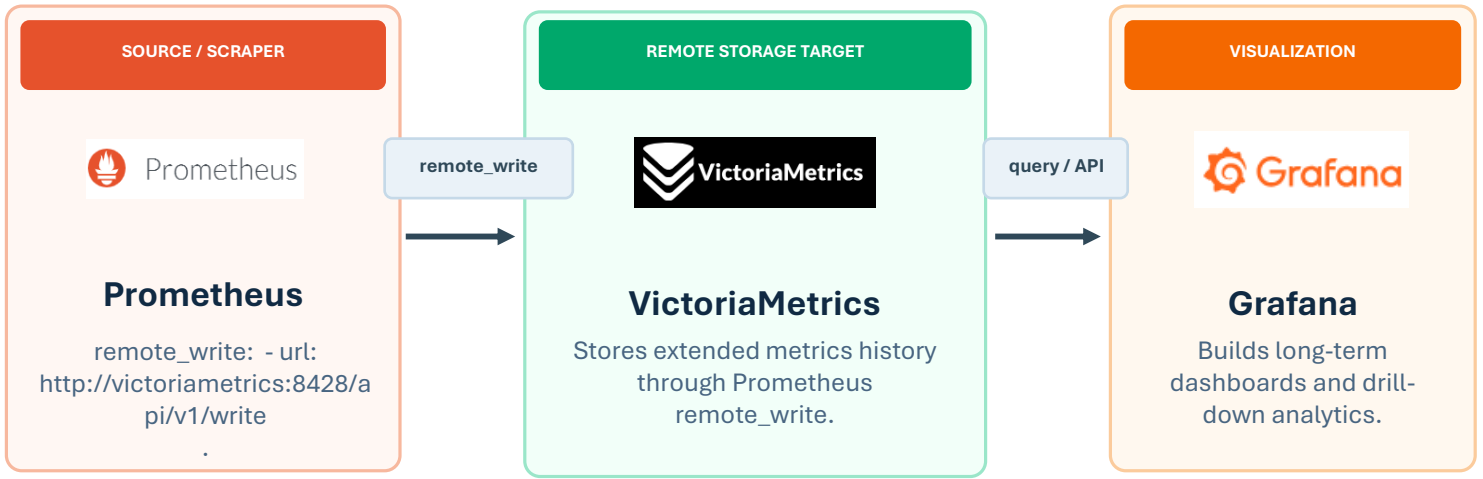




Operational loop: Alert → Dashboard → Drill-down → Logs → Corrective action

VictoriaMetrics as long-term storage for Prometheus

Keep Prometheus focused on scraping and short-term operations while extending retention for historical analytics.



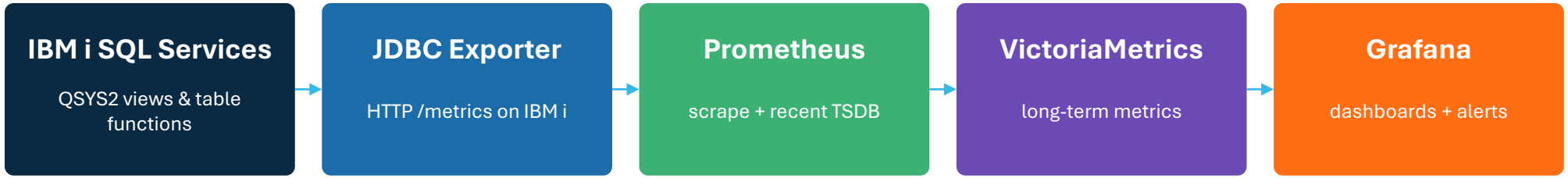
Prometheus keeps the collection model intact; VictoriaMetrics extends history without overloading Prometheus' core role.

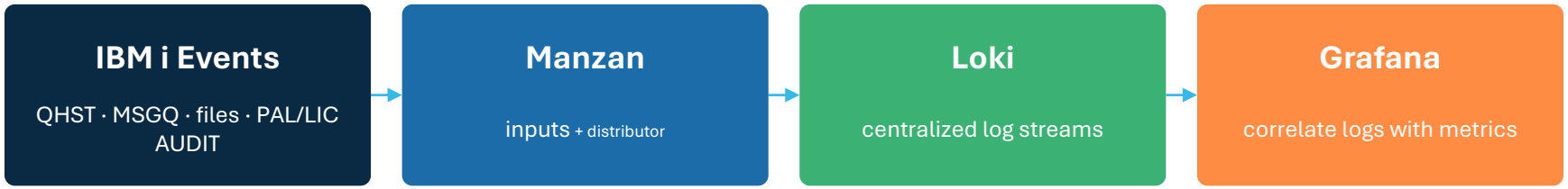
- Prometheus is excellent for **scraping and short-term operational access**, but long retention is often delegated to a specialized backend.
- VictoriaMetrics receives metrics as **remote storage writes**, preserving the Prometheus collection model.
- Longer history enables capacity trending, longer lookback windows, and historical comparisons.
- Architecture stays simple: the main Prometheus change is a **remote_write** configuration entry.

Result: long-term metrics history for Grafana dashboards, drill-down analysis, capacity trending, and historical comparison.

Target architecture at a glance

Metrics and events follow different paths but converge in Grafana for operators.



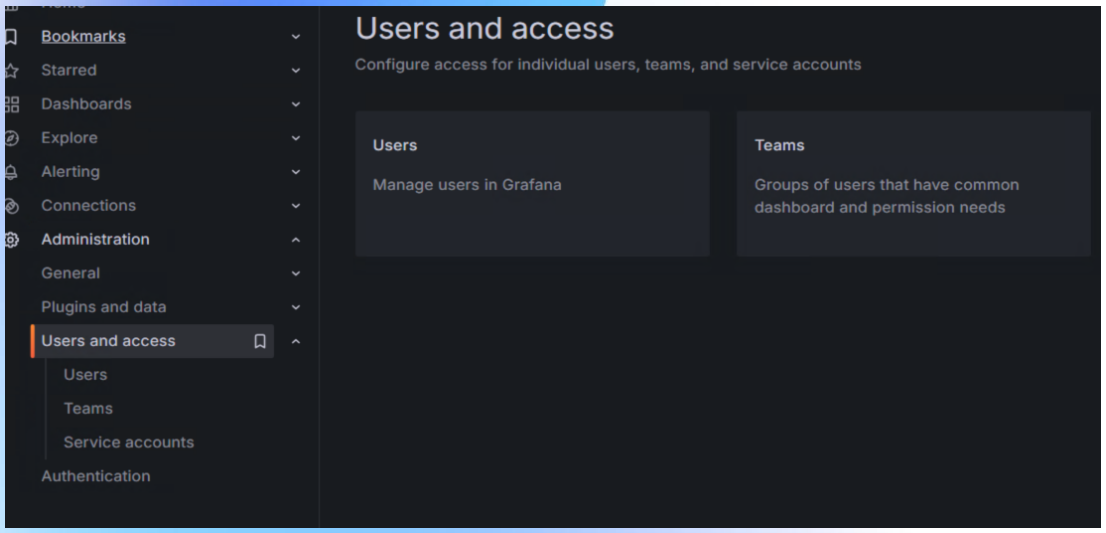


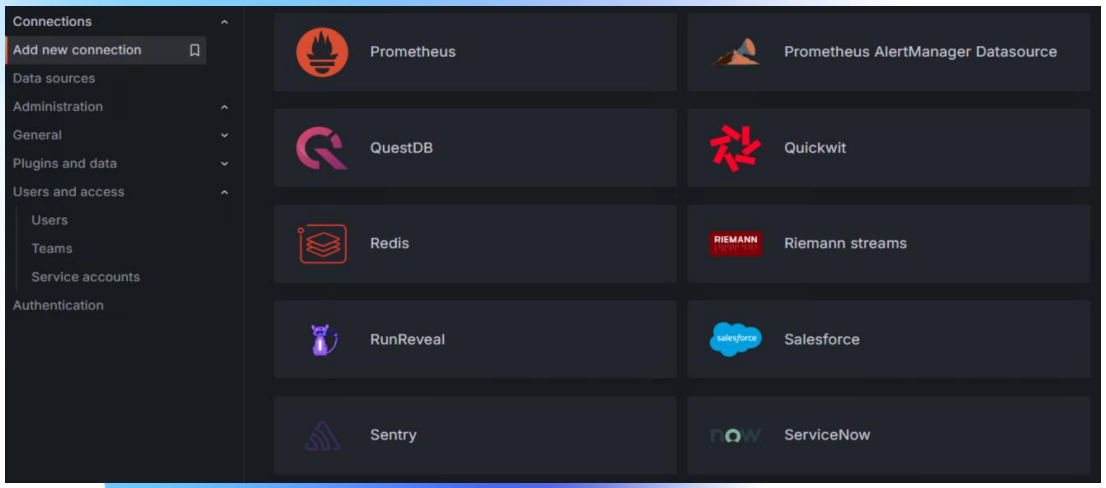
Operational loop: Alert → Dashboard → Drill-down → Logs → Corrective action

Grafana as the IBM i Operations Console

Presentation layer for cross-domain observability

- Turns IBM i metrics and logs into dashboards, drill-down views and operational portals.
- Presents Prometheus metrics and Loki logs through the same Grafana user interface.
- Can be deployed as an open-source container and accessed from a standard browser.
- Gives operators one common UI instead of separate tools for charts, log searches and alert review.



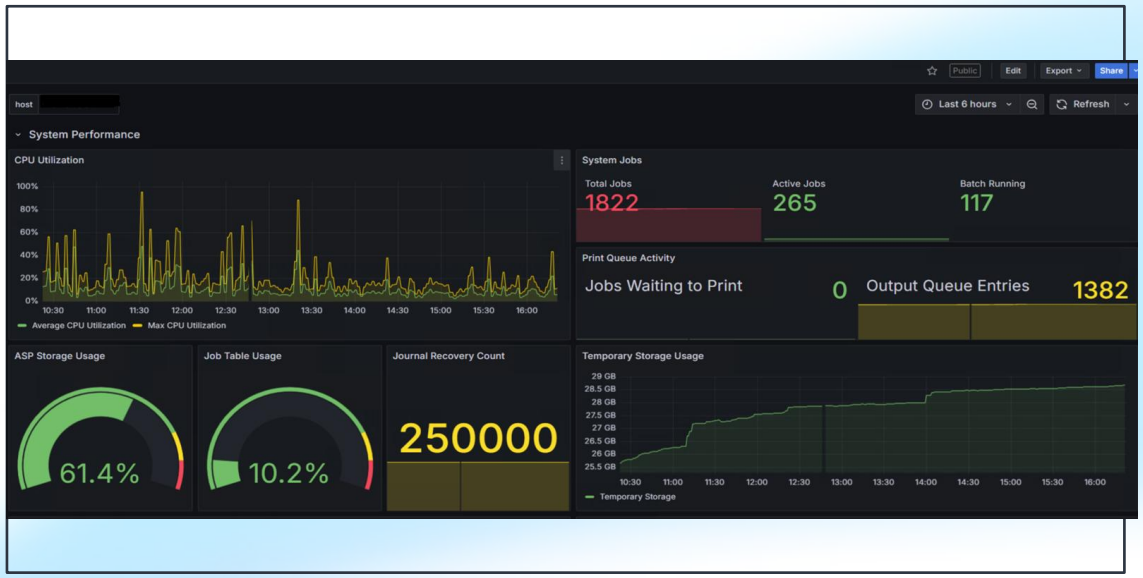


Operational outcome

Faster triage, shared context, and fewer tool switches for IBM i administrators, application teams and support.

Grafana Dashboard Domains

Start with platform health, then move toward workload and service views



1. Platform health

CPU, memory pools, ASP/storage, active jobs, temporary storage, communication activity.

2. Workload behavior

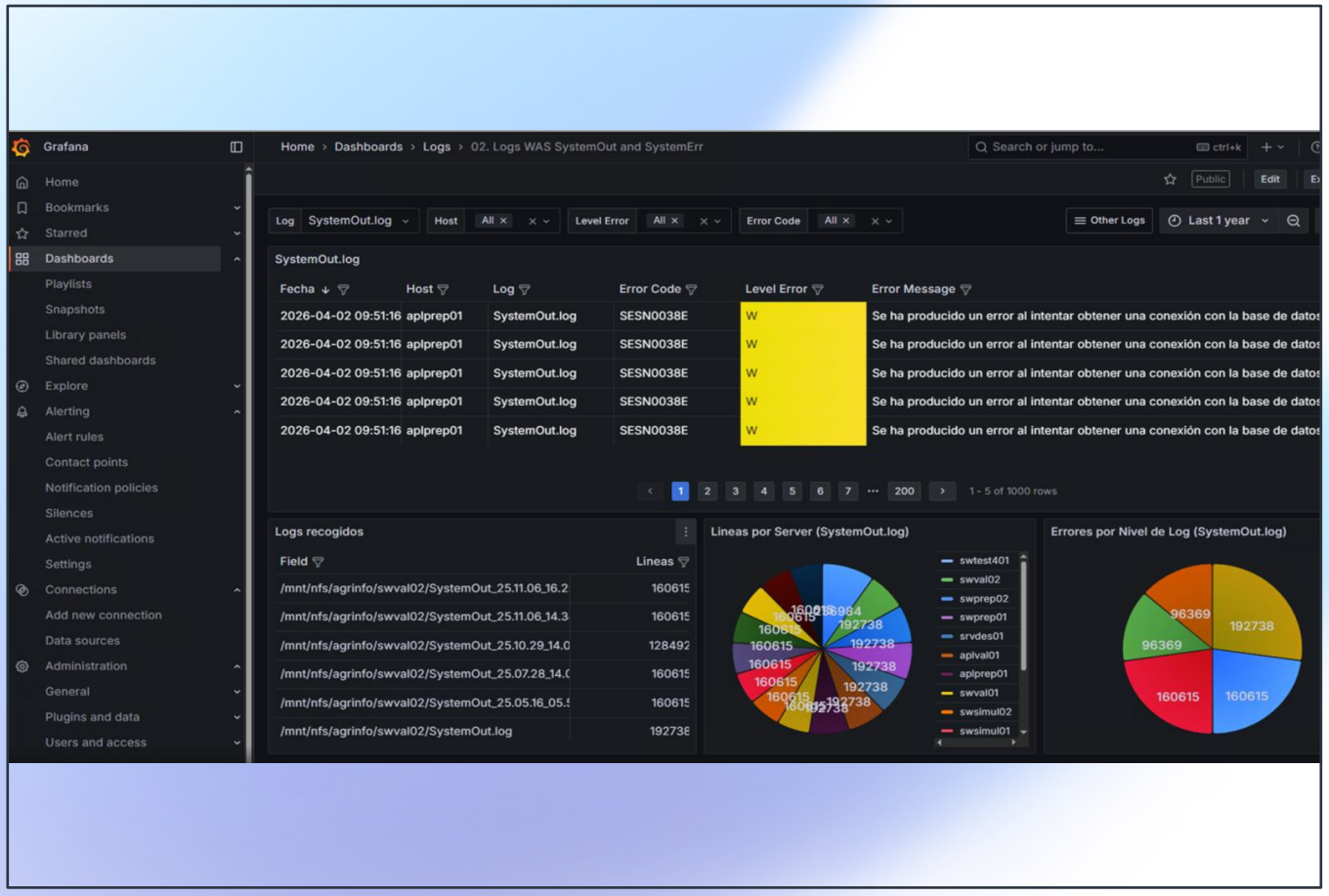
Subsystems, batch, application KPIs and business service metrics from SQL queries.

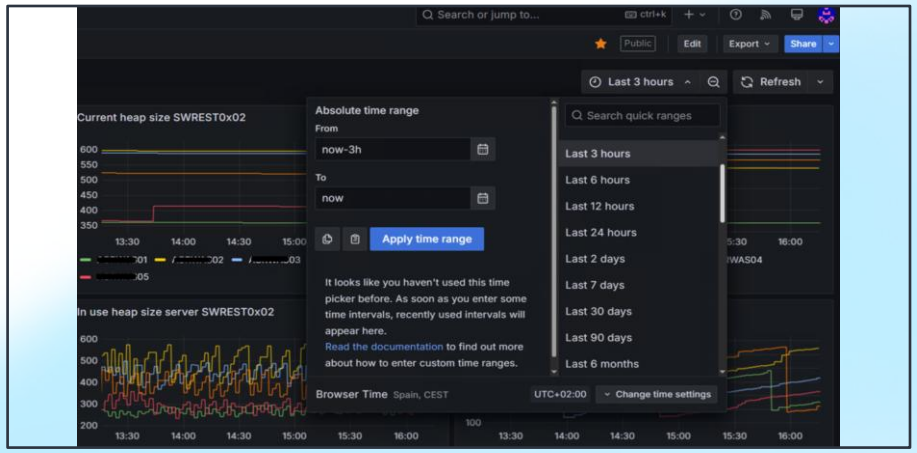
3. Service views

Executive summaries for management plus drill-down pages for administrators and support.

Grafana Correlation and Drill-Down

Use the same time range to move from symptom to evidence



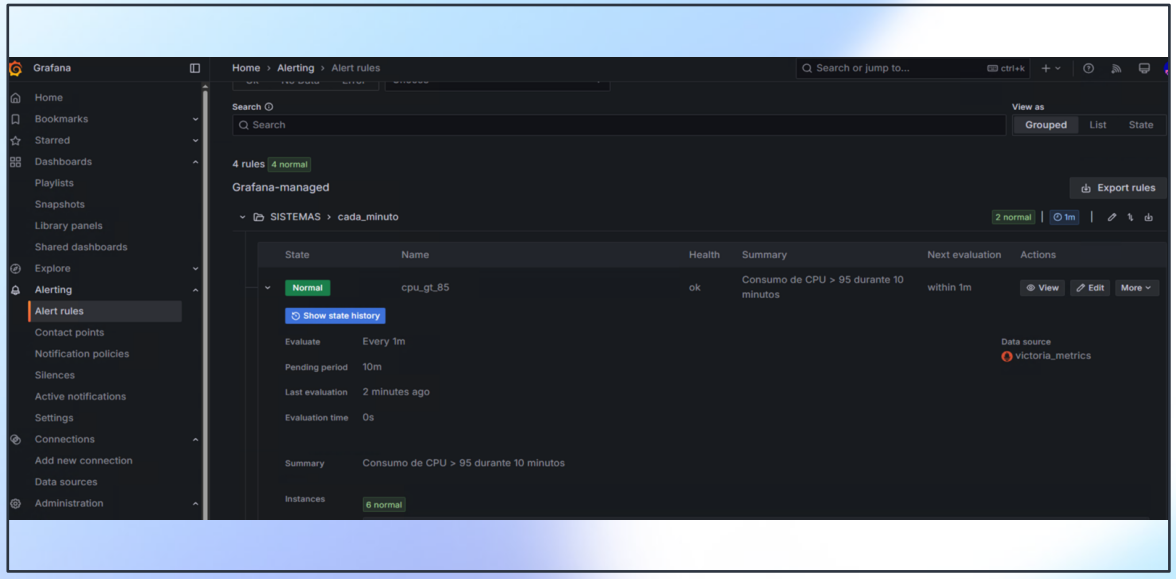


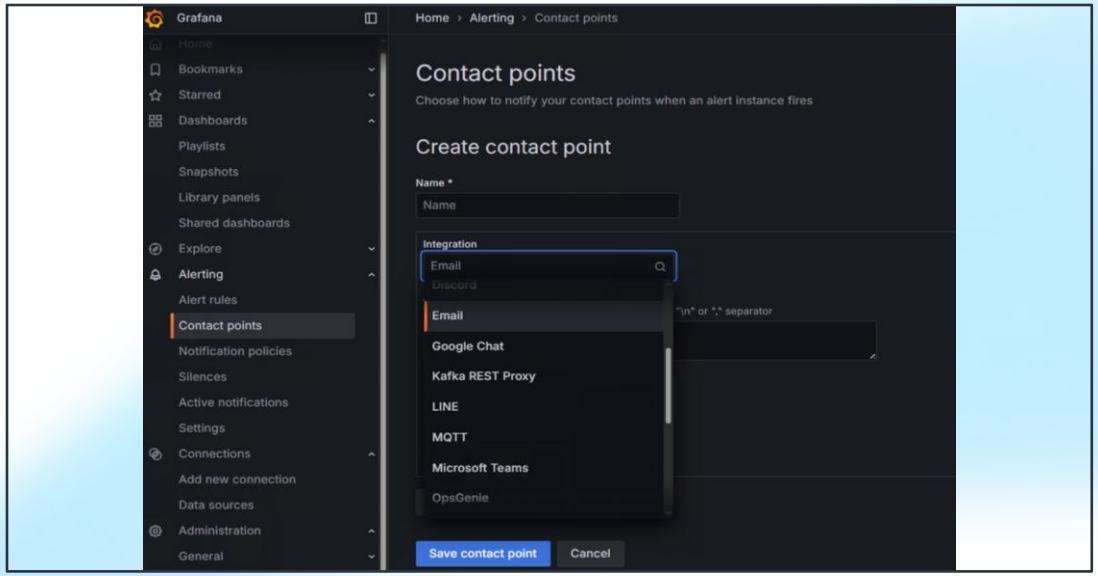
Example troubleshooting flow

- Detect an abnormal CPU, jobs or storage pattern.
- Keep the same absolute time range for all panels.
- Open logs and filter by host, log type, error level or error code.
- Confirm whether the event is infrastructure, application or database related.

Grafana Alerting and Operational Response

High-value alerts should be tied to service impact





Recommended alert design

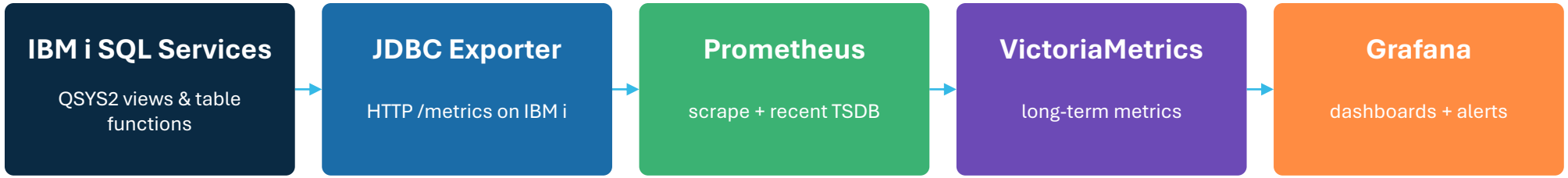
- Define a small number of alerts for conditions that require action.
- Use pending periods to avoid noise from transient spikes.
- Route notifications to email or collaboration tools such as Teams.
- Review alert health and history so the monitoring system itself remains trusted.

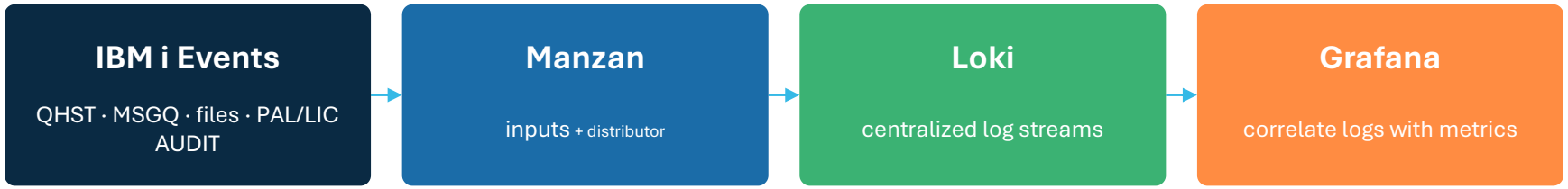
Implementation sequence

- Deploy Grafana container and connect Prometheus/Loki data sources.
- Build platform health dashboards first.
- Add SQL-based workload KPIs and service summaries.
- Tune alerts with operators before expanding coverage.

Target architecture at a glance

Metrics and events follow different paths but converge in Grafana for operators.





Operational loop: Alert → Dashboard → Drill-down → Logs → Corrective action

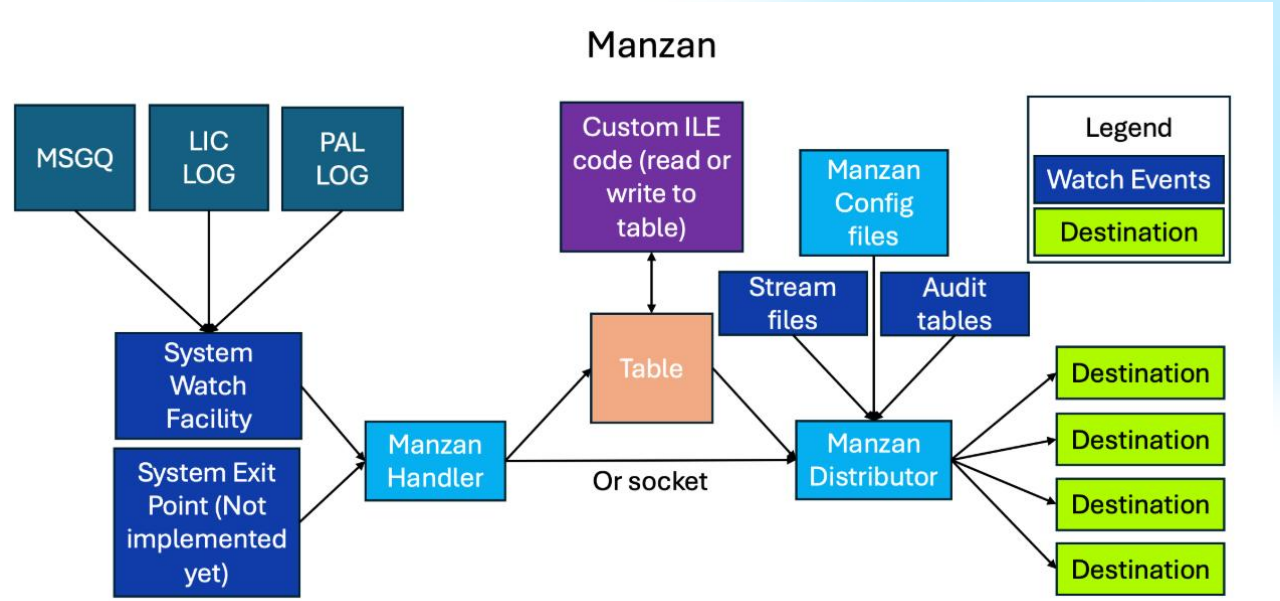
Manzan

IBM i event monitoring for the observability stack

Complements Prometheus metrics with message queues, history log, audit-related events, stream files, and routing to Loki/Grafana.

EVENTS + LOGS | IBM i | LOKI / GRAFANA

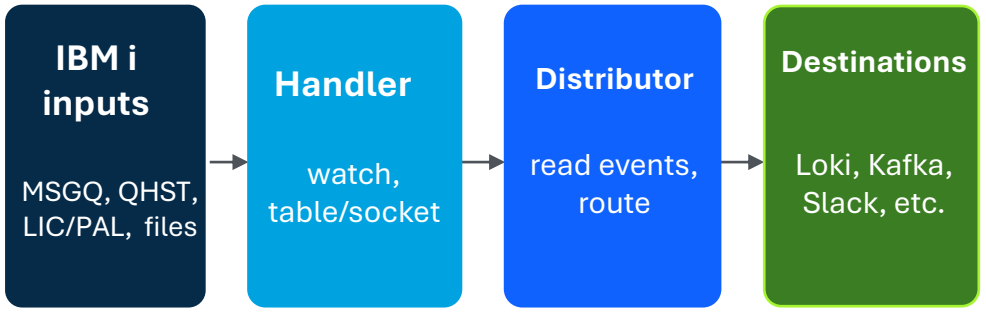
```
mkdir -p /opt/download
cd /opt/download
wget
https://github.com/ThePrez/Manzan/releases/download/v0.0.14/manzan-installer-v0.0.14.jar
 java -jar manzan-installer-v0.0.14.jar
```



Working model: Manzan Handler collects IBM i events; Manzan Distributor routes structured events to downstream destinations.

IBM i inputs

data.ini dests.ini




```
/QOpenSys/etc/manzan/data.ini

[audit1]
type=audit
destinations=loki_out
auditType=PASSWORD
fallbackStartTime=100
interval=30000
enabled=true
injections.HOSTNAME=ANSIBLE

[watcher1]
type=watch
id=ans_OPR1
destinations=loki_out
strwch=WCHMSG((*ALL)) WCHMSGQ((QSYS/QSYSOPR))
interval=3000
numToProcess=100
enabled=true
injections.HOSTNAME=ANSIBLE
```

History log / QHST LIC / PAL logs	Operational evidence and job lifecycle messages. Internal events. Everything supported by STRWCH
MSGQ	QSYSOPR and application queues
Stream files, table, SQL	Application and open-source logs.
http	Fetches data from an http endpoint
Audit tables / journals	Security and compliance event streams

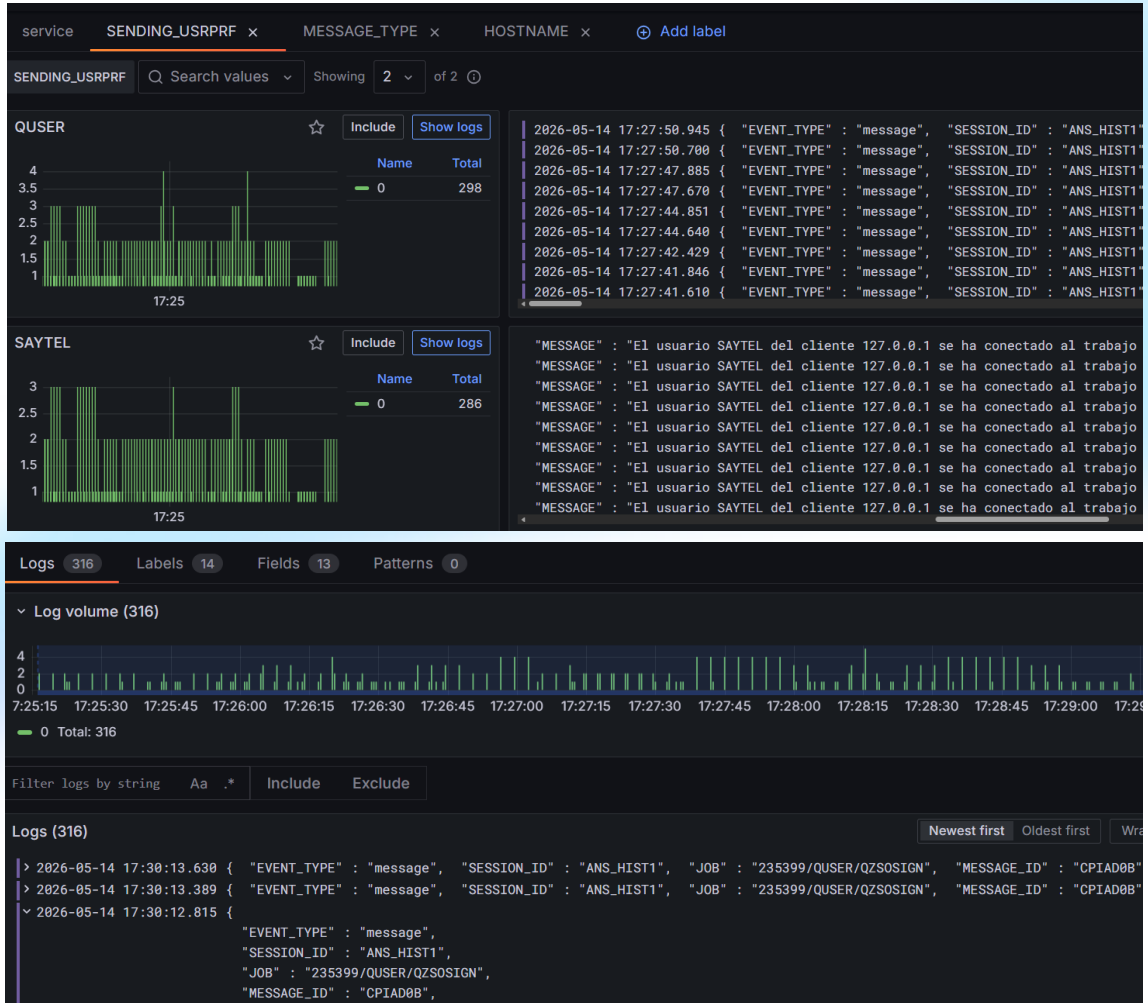
This gives operators the evidence behind metric changes: sign-ons, job messages, errors, application warnings, and security-relevant events.

Manzan destinations

Available destination families

HTTP/HTTPS endpoints	Email / SMTP(S)
SMS via Twilio	Slack
FluentD	Elasticsearch
Sentry	**Grafana Loki**
Google Pub/Sub	ActiveMQ
Kafka	Splunk, PagerDuty, Mezmo

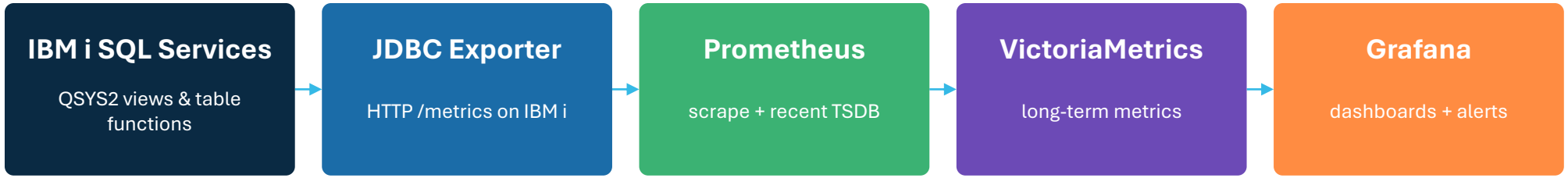
```
/QOpenSys/etc/manzan/dests.ini
[loki_out]
type=loki
url=<loki_url>
username=<loki_username>
password=<loki_password
```

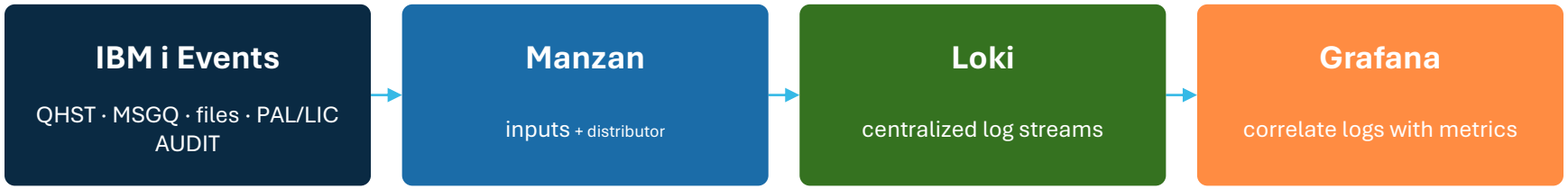


Loki drilldown makes Manzan events filterable by service, hostname, user, message type , etc

Target architecture at a glance

Metrics and events follow different paths but converge in Grafana for operators.





Operational loop: Alert → Dashboard → Drill-down → Logs → Corrective action

Why Loki is the chosen log platform

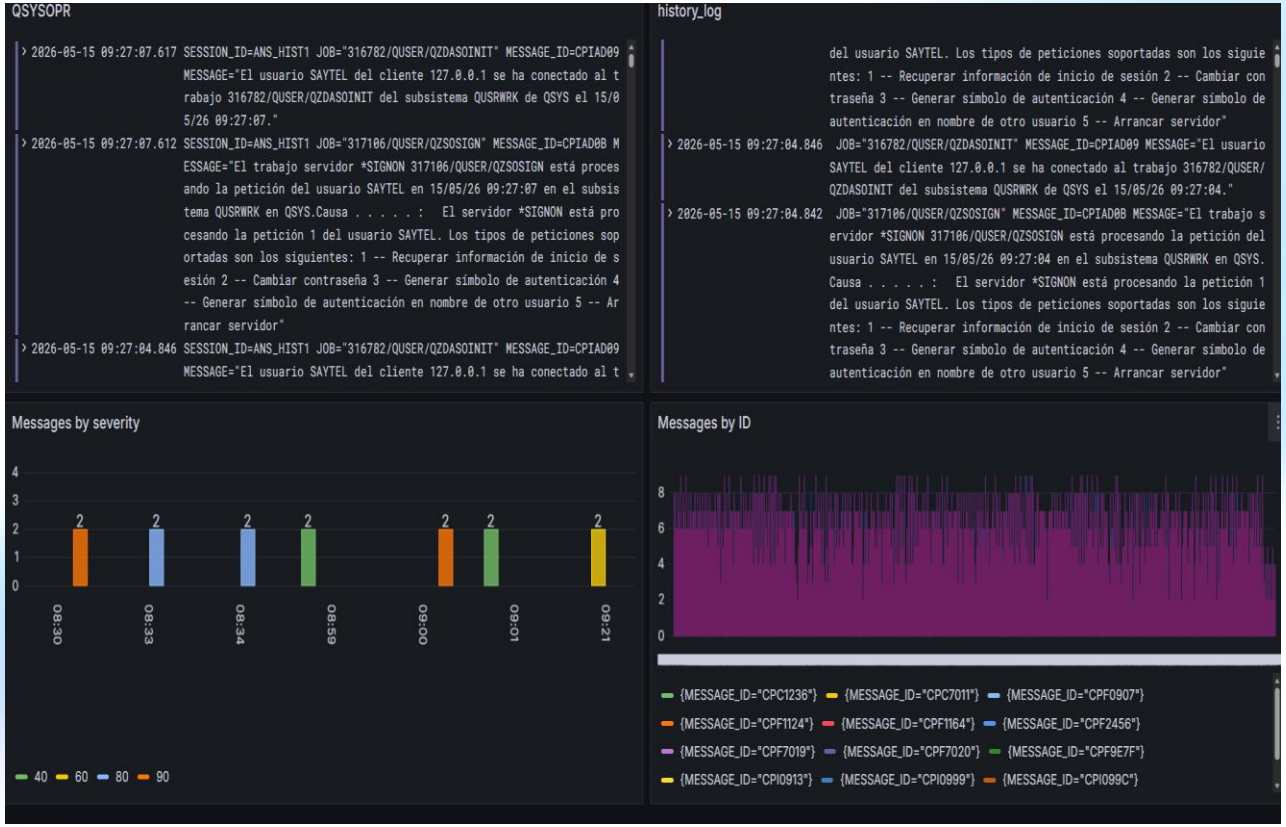
IBM i messages and operational events become searchable, dashboard-ready logs next to your metrics.

Core idea

Loki gives Manzan a central log destination for IBM i message text, QHST events, selected queues and operational logs. Grafana then turns those logs into dashboards, drill-downs and alerting views.

- Log aggregation complements Prometheus metrics.
- Useful for message frequency and critical patterns.
- One Grafana UI for charts and log investigation.

Manzan → Loki → Grafana

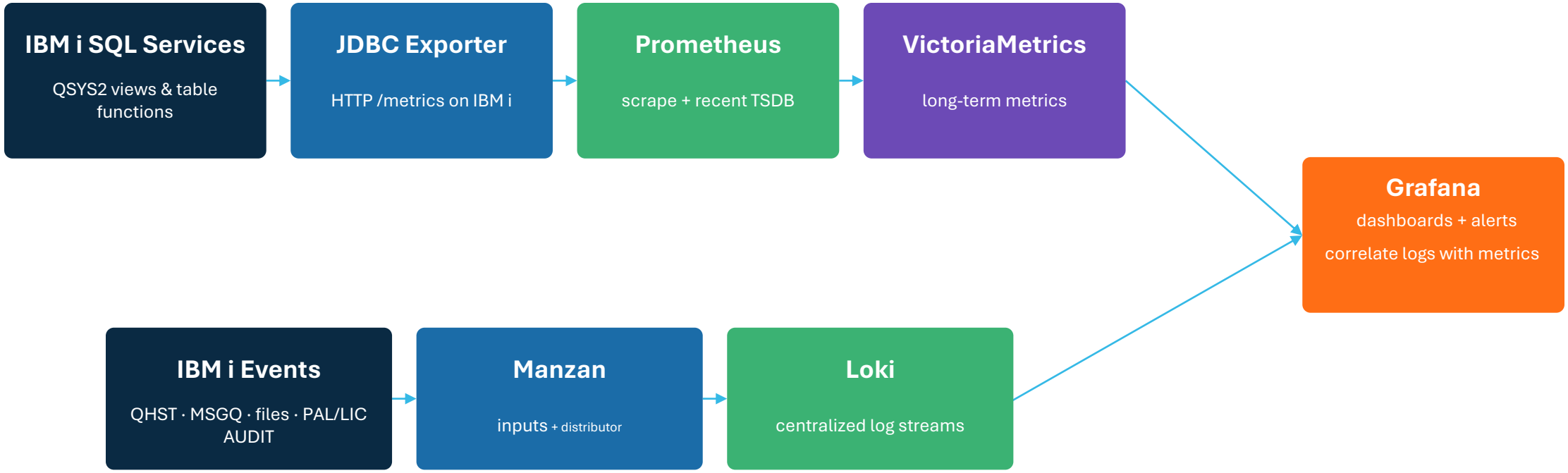


Live dashboard outcome: IBM i event text + message panels in Grafana

Sources: Grafana Loki docs; TechChannel Manzan announcement; user-provided Grafana/Manzan dashboard screenshot.

Target architecture at a glance

Metrics and events follow different paths but converge in Grafana for operators.

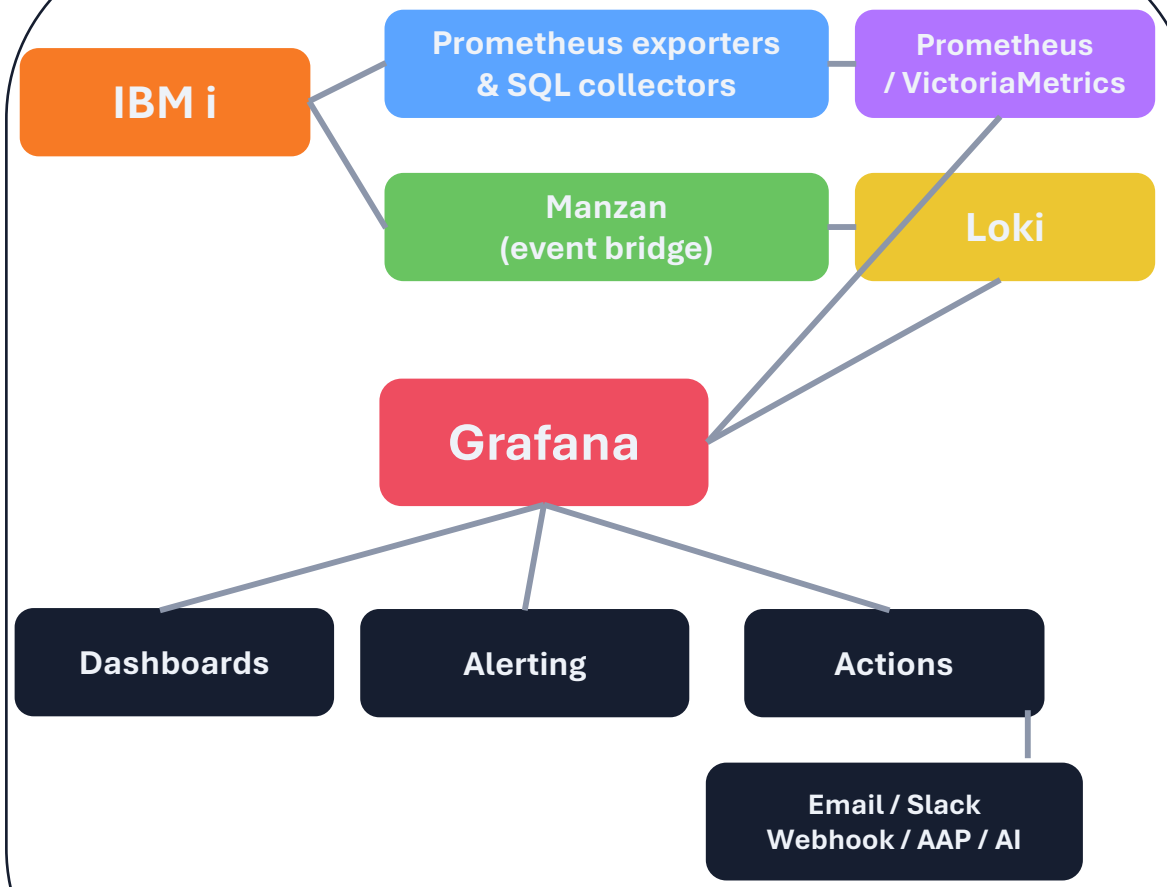


Operational loop: Alert → Dashboard → Drill-down → Logs → Corrective action

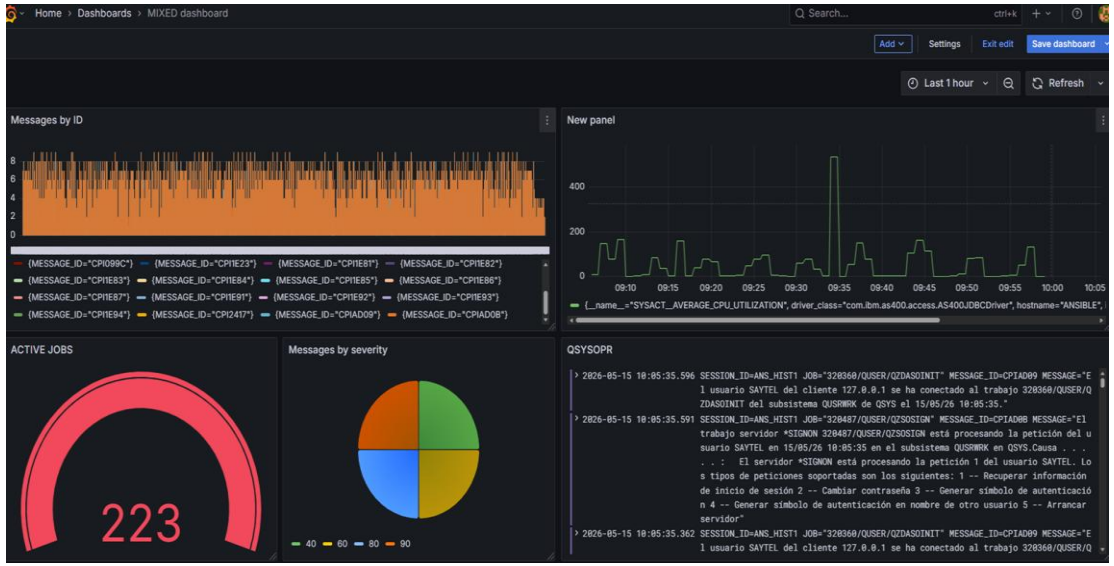
IBM i Observability with Grafana, Prometheus, Loki, and Manzan

SEIDOR

Reference architecture



Unified mixed dashboard example

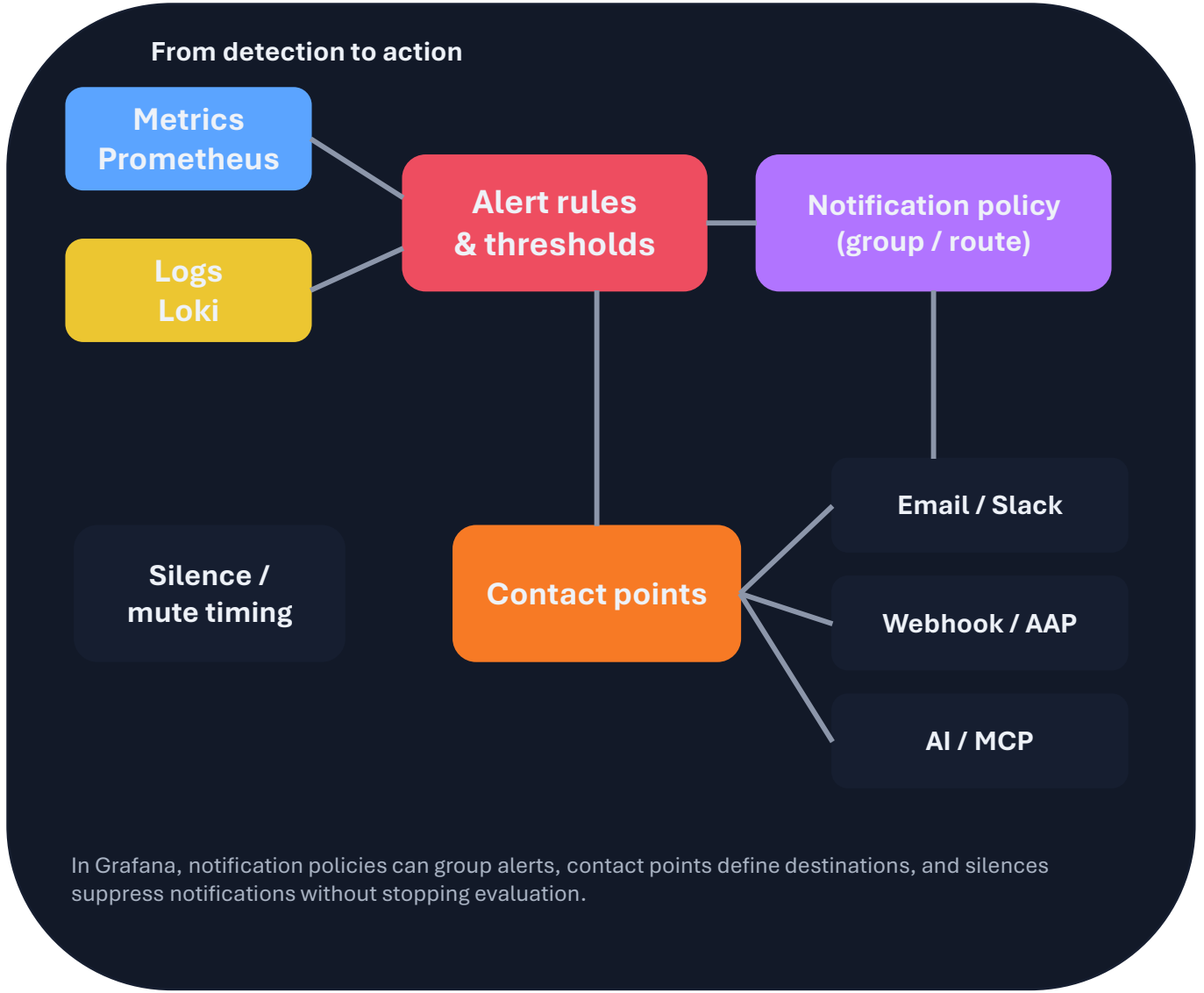


Thresholds, alarms, and advanced alert management

Grafana Alerting turns metrics and logs into notifications, routes them to the right teams, and can trigger downstream automation.

Grafana - first line of defense against incidents

- Alert rules can watch threshold breaches, error patterns, missing signals, unusual trends, or business KPI deviations.
- Notification policies help reduce noise by routing alerts, grouping related instances, and controlling notification timing.
- Silences and mute timings support planned maintenance windows, while inhibition can suppress secondary noise during root-cause incidents.
- Contact points support destinations such as email, Slack, Teams, PagerDuty, IRM, and webhooks.
- Webhooks make it practical to call Ansible Automation Platform jobs or forward context to AI workflows through MCP-enabled

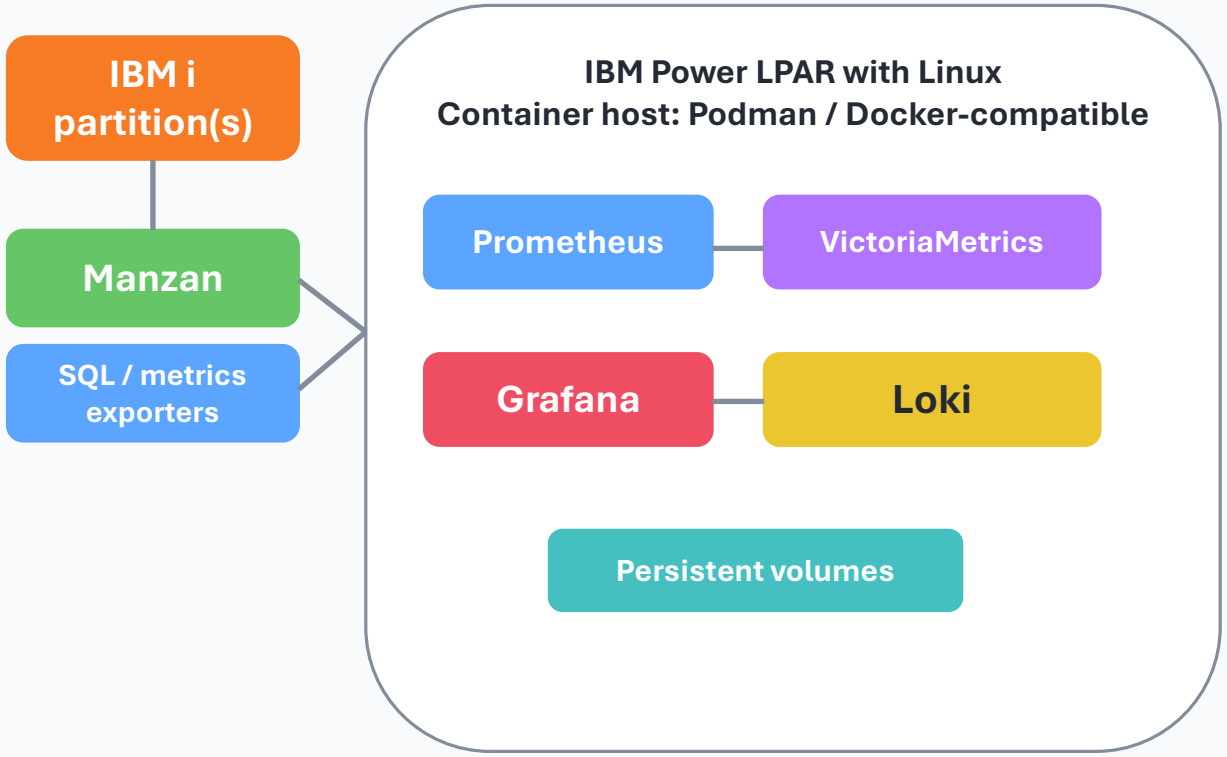


In Grafana, notification policies can group alerts, contact points define destinations, and silences suppress notifications without stopping evaluation.

Containerized deployment on one IBM Power Linux platform

SEIDOR

Reference architecture



Name	Last Pushed ↑
nchirea/**loki-alpine-ppc64le**	19 days ago
nchirea/**grafana-alpine-ppc64le**	19 days ago

https://hub.docker.com/repositories/nchirea

Why start this way

- Simpler rollout
- Cleaner lifecycle
- Lower dependency risk
- Easy updates
- Simple networking, DNS, storage, and security boundaries
- Same logical design can later move to multiple hosts

Container deployment is the design choice for fast installation, easier upgrades, and avoiding classic software dependency hell.

Persistent data must be designed in from day one

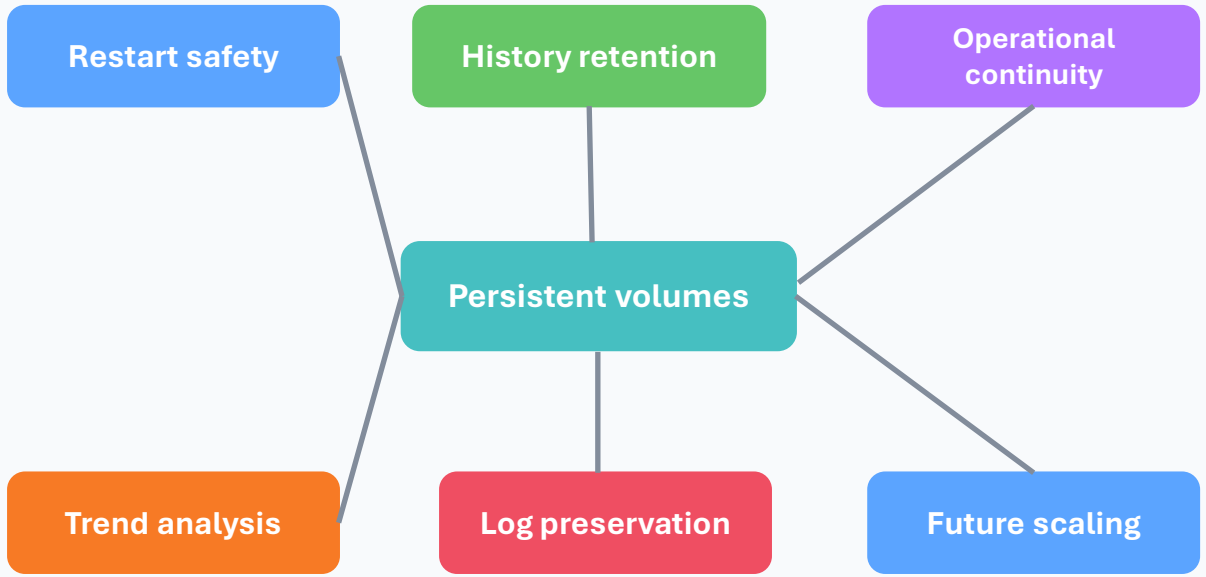
Containers make deployment easier, but persistent volumes are what preserve dashboards, metric history, and log retention across restarts.

SEIDOR

Persistent volume map

Grafana	Stores users, dashboards, alert rules, and data source configuration
VictoriaMetrics	Stores retained time-series data and indexes for historical analysis
Loki	Stores log chunks, indexes, and storage state for QHST and job/event visibility

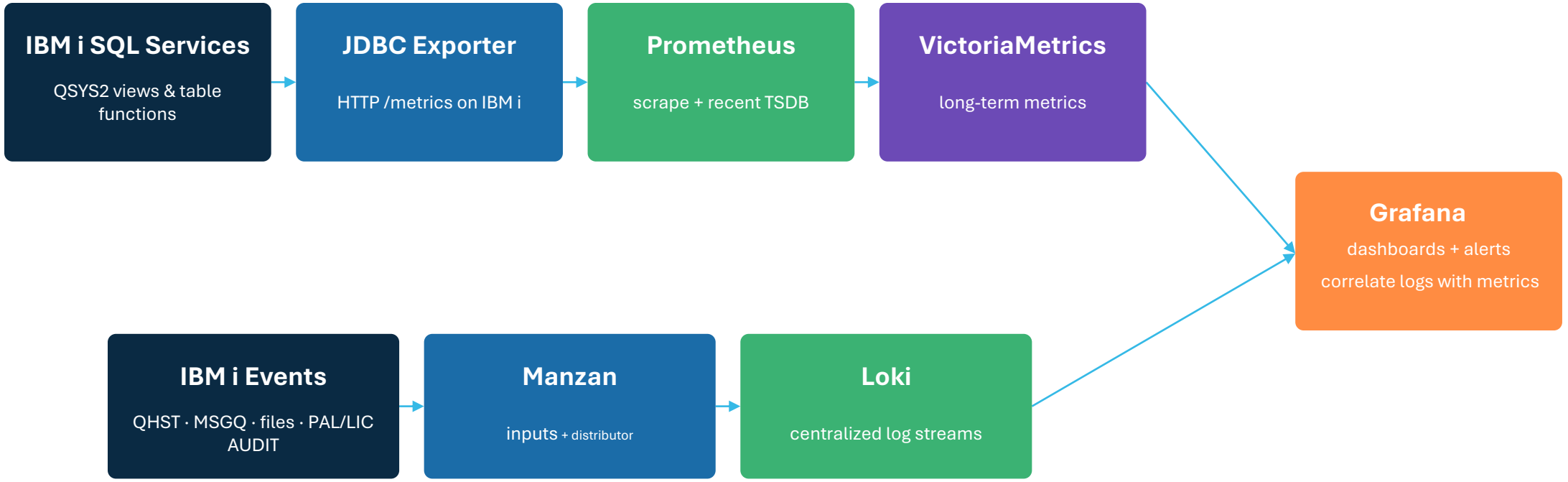
Design principles



Without persistent storage, an otherwise successful container rollout still loses context, trend data, and log history after failure or maintenance events.

Target architecture at a glance

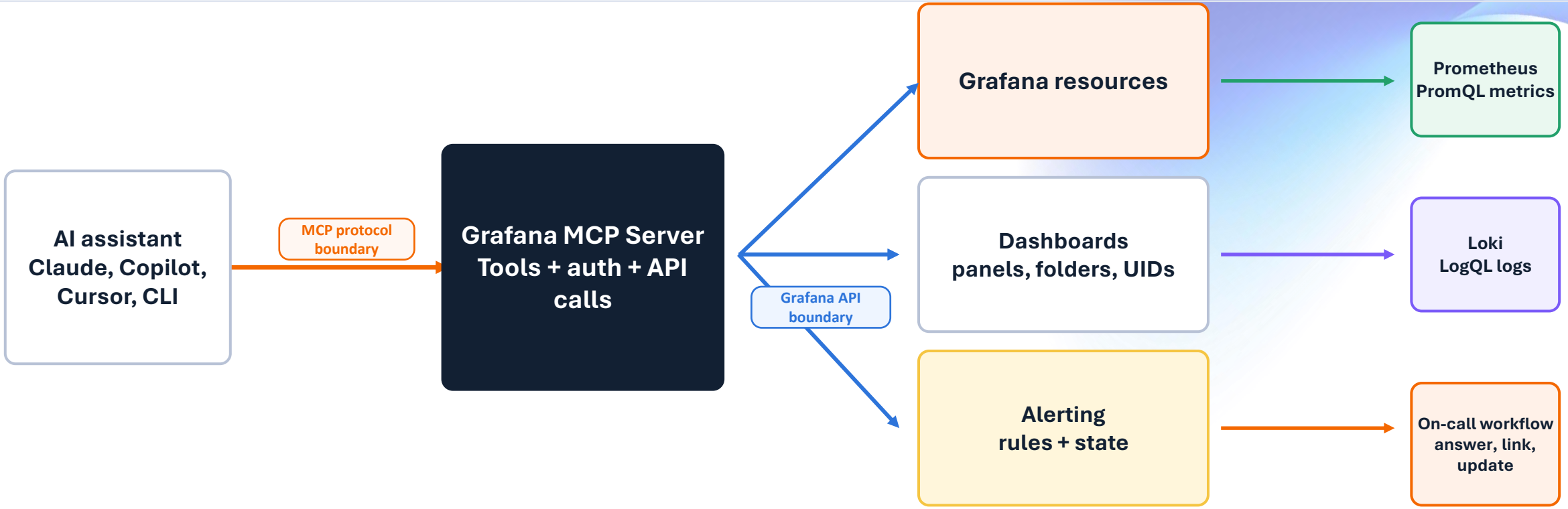
Metrics and events follow different paths but converge in Grafana for operators.



Operational loop: Alert → Dashboard → Drill-down → Logs → Corrective action

MCP makes Grafana conversational

The MCP server acts as a governed bridge between AI assistants and observability resources.

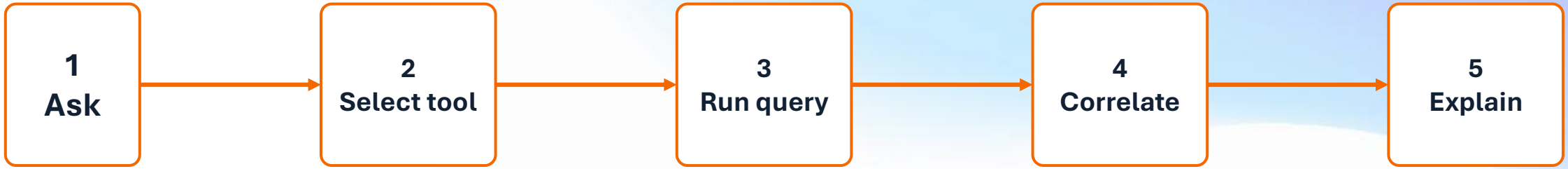


Governance built in

Service-account permissions, datasource scopes, enabled tool sets, and audit trails define what an assistant can read or change.

From prompt to observability answer

MCP exposes Grafana actions as tools: discover, query, inspect, and return evidence with links.



Example operator questions

"Why did CPU spike on this IBM i LPAR?"
"Show errors around the last alert."
"Find the dashboard for storage latency."

Tools used by the assistant

- Prometheus: instant/range PromQL
- Loki: LogQL over time windows
- Dashboards: search + inspect panels

Result delivered to the engineer

A compact answer with the PromQL/LogQL used, summarized evidence, affected time range, and a Grafana deep link for validation.

Key idea: the AI does not replace dashboards - it accelerates finding the right query, panel, or alert context.

Executive message

IBM i can be monitored with an open stack while preserving IBM i-native operational concepts.

Why this matters

Classic monitoring often separates performance, system messages and application events. A unified stack gives operators one path from symptoms to evidence.

How the solution works

IBM i SQL Services expose numeric metrics through a JDBC exporter; Manzan captures operational events; Grafana unifies metrics and logs.

Congress takeaway

This is not a demo-only architecture: it has clear roles, containerized deployment, extensible SQL collectors and a phased rollout model.

Open source stack | **IBM i native SQL** | **Event correlation** | **Long retention** | **Actionable alerts**

SEIDOR

Tu Socio Tecnológico

Red Hat

Premier

Business Partner

IBM

Platinum Partner