

Android Cast — Graph Visualization Pivot

From GRAFANA_vs_others_graphvis_pivot.md

Field	Value
Author	Anton Afanasyeu
Revision	R1
Creation date	2026-06-02
Last modification date	2026-06-02
Co-authored	
Severity	medium
State	pending review
Document type	DR

Table of contents

[Context and constraints](#)
[Executive summary](#)
[Option comparison](#)
[Role-based graph packs \(suggested\)](#)
 [User graphs](#)
 [Slug admin graphs](#)
 [Platform admin graphs](#)
[Estimates \(engineering\)](#)
[Recommendation under your constraints](#)
[Implementation slice \(proposed\)](#)
[Source linkage](#)

Source: GRAFANA_vs_others_graphvis_pivot.md — section links work in PDF viewers that support internal anchors.

Context and constraints

- Stack: minimalistic LAMP with nginx + php-fpm + MariaDB.
- Main constraint: **no data duplication/ETL pipeline** (no cron-based mirroring or parallel stores).
- Need role-hierarchical analytics: user → slug admin → platform admin.

Executive summary

- **Grafana is viable** if it can query MariaDB directly and enforce tenant/role boundaries safely.
- **Custom PHP dashboards are safer initially** for auth/session reuse and strict slug scoping.
- Recommended approach: **hybrid**.
- Define a canonical SQL metrics contract once.
- Ship custom BE dashboards first.
- Add Grafana later only if direct DB + RBAC integration is clean.

Option comparison

Option	Pros	Cons / Risks	Stack impact
Grafana (direct DB)	Fast polished charts, panel ecosystem, alerting	Auth/RBAC integration effort; slug filtering discipline required; one more service to operate	Adds Grafana runtime (no ETL required)
Custom PHP + nginx	Native shared login/session, direct BE internals, deterministic tenant logic	More UI/chart development effort	No new runtime dependencies
Hybrid (recommended)	Fast practical delivery + future flexibility	Needs shared metrics contract discipline	Minimal immediate risk

Role-based graph packs (suggested)

User graphs

- Unique devices per day (registrations/device IDs).
- Avg cast duration by day/week.
- Avg receive sessions by day/week.
- Success ratio (session start vs completed).
- App version distribution.

Slug admin graphs

- Everything from user level, aggregated by slug.
- Active vs passive users/devices.
- Avg app bandwidth (send/rcv) in 1d/7d windows.
- Install source pie: Play Market vs OTA vs custom.
- NTP/time-source usage split and correction savings metric.

Platform admin graphs

- Everything from slug admin level, cross-slug view.
- Crashes per slug/device/user over time.
- Crash trend by app version / Android API / fingerprint.
- Crash-to-ticket linkage counts with issue tracker links.
- Top unreliable slugs/devices (rate-based).

Estimates (engineering)

Track	Initial delivery	Hardening	Total
Grafana direct DB	4-7 dev days	5-8 dev days	9-15 dev days
Custom PHP dashboards	5-9 dev days	3-6 dev days	8-15 dev days
Hybrid phase-1 PHP then optional Grafana	6-10 dev days	+4-7 dev days (optional)	10-17 dev days

Assumptions: current MariaDB crash/ticket/session structures are available; no major schema rewrite.

Recommendation under your constraints

1. Keep **single source of truth** in MariaDB (no ETL copy pipeline).
2. Build **SQL metrics views/contracts** first.
3. Implement **custom PHP dashboards** first for guaranteed auth and tenant correctness.
4. Add Grafana later only if direct DB + RBAC mapping is validated without complexity growth.

Implementation slice (proposed)

1. Define metrics dictionary + SQL views by role scope (1-2 days).
2. Extend heartbeat with `time_source` + NTP correction savings fields (0.5-1 day).
3. Build user + slug-admin dashboard pages with 1d/7d selectors (3-5 days).
4. Build platform admin reliability board with crash/ticket links (2-3 days).
5. Optional Grafana POC on same views (2-3 days).

Source linkage

- This markdown file is the editable source for:
- `docs/GRAFANA_vs_others_graphvis_pivot.pdf`