

Android Cast — Graph Visualization Pivot

Scope: LAMP + nginx backend, no data duplication pipeline, role-based analytics.

Field	Value
Date	2026-06-02
Context	apps.f0xx.org + BE Alpine (nginx + php-fpm + MariaDB)
Constraint	No cron ETL/data copy/pipeline complexity
Output	Decision memo: Grafana vs custom PHP dashboards
Markdown source	docs/GRAFANA_vs_others_graphvis_pivot.md

1. Executive summary

- Grafana is viable only if it can query MariaDB directly (or via native SQL datasource) with strict RBAC mapping and SSO/session bridging.
- If direct DB + auth integration becomes complex, custom PHP dashboards are lower risk and fit your stack with zero infra additions.
- Recommended path: **hybrid** — build canonical SQL views + metrics contract once; start with custom PHP pages, optionally add Grafana later reusing same SQL.

2. Option comparison

Option	Pros	Cons / Risks	Stack impact
Grafana (direct DB)	Fast charting, polished visuals, alerting, flexible panels	Session/RBAC integration effort; slug scoping must be enforced in queries; additional service to operate	Adds Grafana service only (no ETL required)
Custom PHP + nginx	Native app auth/session reuse; direct BE logic access; deterministic tenant filtering	More frontend/dev work for visual polish; chart UX must be implemented	No new runtime dependencies
Hybrid (recommended)	Ship fast with PHP now, preserve Grafana option later; one metrics SQL contract	Needs discipline in shared metrics layer	Minimal immediate changes

3. Role-based graph pack (suggested)

Role	Core dashboards	Example charts
User	Own slug, own devices/sessions	DAU devices, avg cast duration, avg recv sessions/day, success rate, app version split
Slug admin	All slug analytics + health	Active/passive users, bandwidth send/recv 1d/7d, install source pie (Play/OTA/custom), NTP source usage and correction savings
Platform admin	Cross-slug global + reliability	Crashes per slug/device/user, trend by app version/SDK, links to ticket IDs/issues, top regressions by fingerprint

4. Effort and cost estimate (engineering)

Track	Initial delivery	Hardening	Total estimate
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 Grafana	6-10 dev days	optional +4-7 days	10-17 dev days

Assumptions: existing MariaDB schema with crash/ticket/session tables, role data available; excludes major schema migration.

5. Recommendation under your constraints

- Do not introduce ETL/duplication. Keep one source of truth: MariaDB + live SQL views.
- Implement a metrics SQL layer now (views/materialized semantics via SQL only, no copied store).
- Build custom PHP dashboard pages first for guaranteed auth/session + slug scoping correctness.
- Re-evaluate Grafana after metrics contract stabilizes; adopt only if direct DB + RBAC mapping is clean.

6. Proposed next implementation slice

Step	Deliverable	ETA
1	Define metrics dictionary + SQL views for user/slug/admin scopes	1-2 days
2	Add heartbeat metric: time source in use + ntp correction savings counters	0.5-1 day
3	Create PHP dashboards: user and slug-admin pages with 1d/7d selectors	3-5 days
4	Add admin reliability board + crash/ticket links	2-3 days
5	Optional Grafana POC over same SQL views	2-3 days