Operational Report: Live Monorepo Project Status Dashboard

Author: Manus AI **Date:** Oct 14, 2025 PDT

1. Introduction

This report outlines a strategy for implementing a live dashboard to visualize the idea-to-production journey within a monorepo environment. The goal is to provide a graphical representation and a clear timeline of project status, enabling better oversight and decision-making for a solo developer managing multiple projects.

2. Research and Best Practices for Monorepo Dashboards

Effective monorepo dashboards require a focus on aggregating data from various sources to provide a holistic view of project health and progress. Key considerations include [1, 2]:

- **Centralized Visibility**: A single pane of glass for all projects within the monorepo.
- **Granular Insights**: Ability to drill down into specific project metrics, CI/CD pipeline statuses, and deployment details.
- **Real-time Updates**: Essential for a

live dashboard to reflect the current state of the system.

- **Customizability**: Dashboards should be adaptable to display the most relevant metrics for your specific workflow.

3. Key Metrics and Stages for Visualization

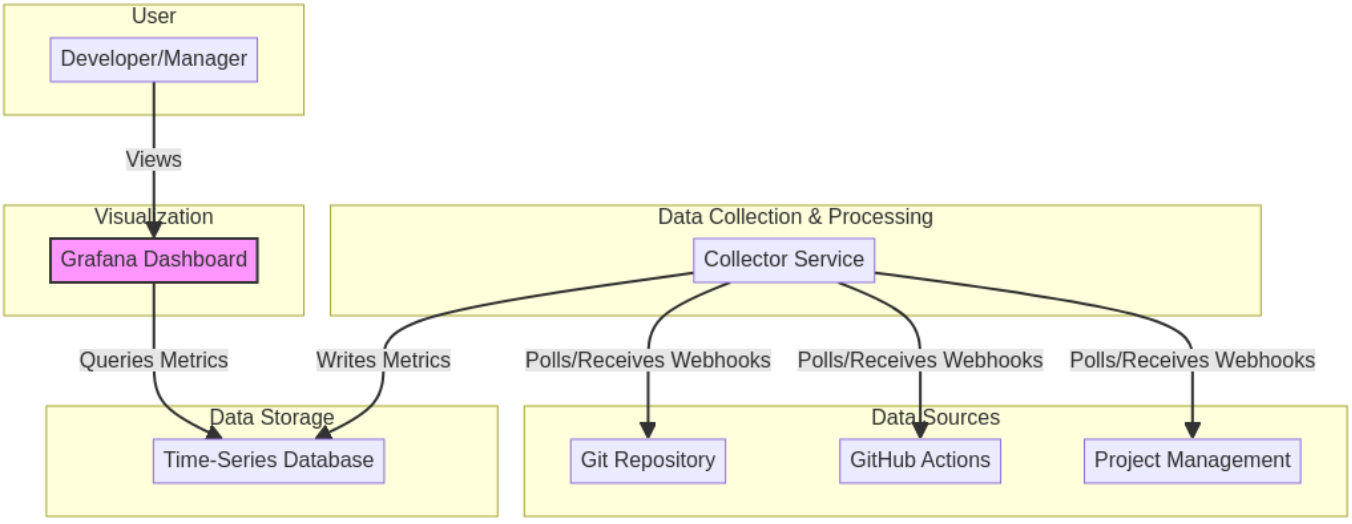
To visualize the idea-to-production journey, we need to track projects through several key stages and measure relevant metrics at each stage [3, 4].

Stage	Key Metrics	Data Sources
Idea/Planning	Number of open issues/tasks, task progress	Project Management Tool
Development	Commit frequency, branch activity, pull requests	Git Repository (e.g., GitHu
CI/CD	Build/test duration, success/failure rate	CI/CD Platform (e.g., GitH
Staging	Deployment frequency, uptime, performance metrics	Monitoring Tools (e.g., Pro
Production	Deployment frequency, uptime, error rate	Monitoring Tools (e.g., Pro

4. Conceptual Architecture and Recommended Tools

The proposed architecture for the live dashboard consists of four main components:

1. **Data Sources**: The raw data from your development and operational tools.
2. **Data Collection & Processing**: A service to collect, aggregate, and process data from the various sources.
3. **Data Storage**: A time-series database to store the collected metrics.
4. **Visualization**: A dashboarding tool to visualize the data.



Recommended Tools:

- **Data Collector**: A custom service written in **Python** or **Node.js** that uses APIs and webhooks to collect data from GitHub (commits, pull requests, Actions) and your project management tool.

- **Time-Series Database**: **Prometheus** is a powerful open-source monitoring system with a time-series database that is well-suited for this purpose [5]. **InfluxDB** is another excellent alternative.
- **Visualization**: **Grafana** is the industry standard for creating beautiful, interactive, and real-time dashboards [6]. It integrates seamlessly with Prometheus and other data sources.

5. High-Level Implementation Plan

Here is a high-level plan for implementing the live project dashboard:

1. **Set up Monitoring Infrastructure**:
   - Deploy Prometheus and Grafana using Docker Compose.
   - Configure Prometheus to scrape metrics from your applications and the data collector.
   - Configure Grafana to use Prometheus as a data source.
2. **Develop the Data Collector Service**:
   - Create a new service (e.g., in Python) within your monorepo under `apps/`.
   - Implement logic to fetch data from the GitHub API (commits, pull requests, Actions status).
   - Expose the collected data as Prometheus metrics.
3. **Instrument CI/CD Pipelines**:
   - Modify your GitHub Actions workflows to send metrics to the data collector or expose them for Prometheus to scrape.
   - Track key metrics like build duration, test coverage, and deployment status.
4. **Build the Grafana Dashboard**:
   - Create a new dashboard in Grafana.
   - Add panels to visualize the metrics collected from Prometheus.
   - Create a timeline view to represent the idea-to-production journey, using stages and metrics defined in Section 3.
5. **Integrate with MAMOS (Optional but Recommended)**:
   - The live dashboard can be integrated into the MAMOS dashboard as a new page or an embedded iframe, providing a single point of access for all your operational insights.

6. Conclusion

By implementing this live project dashboard, you will gain a powerful tool for visualizing the entire development lifecycle within your monorepo. This will enable you to identify bottlenecks, track progress, and make data-driven decisions to improve your development workflow. The recommended tools (Prometheus and Grafana) are open-source, highly scalable, and widely used in the industry, ensuring a robust and future-proof solution.

References

[1] Datadog. (2024, January 8). *Best practices for CI/CD monitoring*. Datadog. https://www.datadoghq.com/blog/best-practices-for-ci-cd-monitoring/

[2] GitLab. (2025, September 29). *How to keep up with CI/CD best practices*. GitLab. https://about.gitlab.com/blog/how-to-keep-up-with-ci-cd-best-practices/

[3] Edge Delta. (2024, July 19). *Mastering CI/CD Monitoring: Essential Tools and Best Practices*. Edge Delta. https://edgedelta.com/company/blog/mastering-ci-cd-monitoring

[4] LaunchDarkly. (2024, July 9). *Ultimate Guide to CI/CD Best Practices to Streamline DevOps*. LaunchDarkly. https://launchdarkly.com/blog/cicd-best-practices-devops/

[5] Prometheus. (n.d.). *Prometheus - Monitoring system & time series database*. https://prometheus.io/

[6] Grafana. (n.d.). *Grafana - The open and composable observability platform*. https://grafana.com/