FireHydrant

Building Mission-Critical Reliability:

A Technical Guide to FireHydrant Signals' High-Availability Alert Systems

Executive Summary

When critical systems fail, alerting infrastructure becomes the lifeline for engineering teams. This white paper examines the comprehensive architectural strategies and redundancy mechanisms that enable FireHydrant Signals to achieve 99.99% uptime, ensuring that when everything else fails, your alerts still get through.

Through rigorous engineering and proven architectural patterns, Signals provides enterprise-grade alerting with:

- **99.99% uptime SLOs** backed by multi-region active-active deployments that eliminate single points of failure
- **Zero-downtime operations** through automated failover mechanisms that seamlessly redirect traffic during outages
- **True service independence** where alert delivery continues even when primary systems are completely offline
- **Unlimited scalability** via stateless architecture that scales linearly with demand
- **Comprehensive redundancy** across every layer, from notification channels to data centers

TABLE OF CONTENTS

This document provides technical and strategic insights for organizations evaluating or implementing FireHydrant Signals as their mission-critical alerting infrastructure, demonstrating why Signals is the foundation of reliable incident response.

THE IMPERATIVE FOR ABSOLUTE RELIABILITY

When Failure Is Not an Option

In the modern digital economy, system outages directly translate to revenue loss, damaged reputation, and eroded customer trust. The irony of incident management is stark: the very moment when alerting systems are most critical (during widespread failures) is when they face their greatest challenges.

Consider scenarios in which major cloud providers experience regional outages, affecting thousands of services simultaneously. In these moments, alert volumes can spike 100x normal levels while traditional communication channels become unreliable.

The True Cost of Alerting Failures

When alerting systems fail, the cascade effects are severe:

- **Extended Time to Detection:** Without functioning alerts, teams may not discover critical issues until customers report them, adding precious minutes or hours to resolution times.
- **Delayed Incident Response:** Even when issues are detected, the inability to page on-call engineers means mobilization delays that compound customer impact.
- **Communication Breakdown:** During major incidents, coordinated response depends on reliable communication. Without it, teams work in silos, duplicating effort and missing critical dependencies.
- **Compliance Violations:** Many industries require demonstrable incident notification capabilities; alerting failures can result in regulatory penalties.

This reality demands a fundamental rethinking of how alerting systems are architected. With FireHydrant Signals, we moved beyond traditional high-availability approaches to construct a true fault-tolerant design.

FOUNDATIONAL ARCHITECTURE DECISIONS

Technology Selection for Uncompromising Reliability

The foundation of any reliable system lies in its technology choices. Each component in the stack must be selected not just for its features, but for its proven ability to operate under extreme conditions.

Container Orchestration with Kubernetes

Kubernetes has emerged as the de facto standard for reliable service deployment, which is why there was no question around why we built Signals on it. The platform's self-healing capabilities mean that failed containers are automatically detected and replaced without human intervention. Kubernetes' pod distribution algorithms ensure that services are spread across available infrastructure, preventing localized hardware failures from causing service outages. The rolling update mechanisms allow for zero-downtime deployments, critical for maintaining availability during frequent updates that security and feature improvements demand.

High-Performance Processing with Go

Go's lightweight concurrency model enables a single service instance to handle thousands of simultaneous alert deliveries without the overhead of traditional thread-based systems. Its memory-safe design eliminates entire classes of bugs that could cause service crashes, while its fast compilation and startup times mean services can be rapidly deployed or recovered. The language's simplicity also reduces the likelihood of subtle bugs that could compromise reliability. Signals' use of Go ensures alert delivery remains performant and reliable even under massive load spikes.

Durable Workflow Execution with Temporal

Perhaps no decision impacts reliability more than the choice of workflow orchestration. In Signals, Temporal provides a revolutionary approach to handling long-running operations like escalation policies. Unlike traditional queue-based systems, Temporal maintains a complete event history for every workflow, enabling recovery from any point of failure.

If a service crashes mid-escalation, Temporal automatically replays the workflow history to reconstruct the exact state, then continues execution as if nothing happened. This durability even extends across data center boundaries: workflows can be resumed in entirely different regions without data loss.

STATELESS ARCHITECTURE

The Key to Infinite Scale

Traditional stateful services create inherent limitations on scalability and reliability. By embracing stateless design principles, Signals achieves remarkable flexibility. Each service instance becomes interchangeable — any request can be handled by any instance, eliminating complex session affinity requirements. This enables true horizontal scaling where capacity can be added simply by deploying more instances, with no architectural limits on growth.

The benefits extend beyond scalability. Stateless services can be killed and restarted at will without losing in-flight operations. During deployments, traffic seamlessly shifts to healthy instances with no user impact. When entire data centers fail, services in other regions immediately take over the load without complex state synchronization.

Data Layer Redundancy Without Compromise

While services can be stateless, data must be preserved with absolute reliability. Signals implements multiple layers of data redundancy:

- **Synchronous In-Region Replication** ensures that every write is immediately copied to multiple database instances within the same data center. This protects against individual server failures while maintaining the strong consistency required for critical configuration data.
- **Asynchronous Cross-Region Replication** provides disaster recovery capabilities by continuously copying data to geographically distant locations. This ensures that even catastrophic regional failures cannot cause data loss.
- **Intelligent Read Distribution** leverages read replicas to distribute query load, preventing database bottlenecks during high-volume incidents. The system automatically routes read operations to the nearest available replica, optimizing both performance and reliability.

Service Independence and Graceful Degradation

Microservice Architecture That Actually Delivers

While microservices have become an industry buzzword, their implementation in mission-critical systems requires exceptional discipline. Signals' microservice design emphasizes true independence and isolation, meaning each service operates in its own failure domain with no shared dependencies that could create cascading failures.

The architecture achieves this isolation through several key design principles:

Dedicated Infrastructure Per Service: Each core component within Signals runs on isolated compute resources, with distinct memory, CPU, and network quotas. Each service is deployed and managed separately, preventing resource contention even during sudden workload spikes.

The Signals alert delivery service is federated across two distinct, physically-separated clusters. This provides for physical isolation of networking resources, ensuring they aren't reliant on the overall cluster health, as well as a redundant layer of compute resources. If one cluster, or even the entire region it exists in, experiences degraded or no service, the other region will continue to serve requests and dispatch alerts normally.

Independent Data Stores: Services never share databases or caches. The alert routing service maintains its own optimized data store containing only the routing rules it needs. This local data is periodically synchronized but operates autonomously. If the central configuration service fails, routing continues uninterrupted using the last known good configuration.

Isolated Network Paths: Network segmentation ensures that services communicate through well-defined APIs with no backdoor dependencies. FireHydrant's alert delivery pipeline has its own dedicated network path to external providers (SMS, voice, email, etc.), isolated from general application traffic. This prevents scenarios where high web traffic could delay critical alert delivery. Note that Slack is the exception here. While fully supported, it doesn't offer the same network isolation. For critical alerts, SMS, voice, and email remain the most reliable channels.

Autonomous Operation: Services are designed to operate with zero dependencies on other internal services. The SMS delivery service, for instance, needs only an alert payload and recipient information to function. It doesn't call back to check user preferences, validate permissions, or log analytics —

all necessary data is included in the initial request or cached locally. Similar to above, Slack doesn't provide isolated network paths. We recommend treating SMS, voice, and email as the most reliable channels for urgent alerts.

Degradation That Users Never Notice

Should partial failures occur, Signals' isolation architecture enables selective degradation that preserves critical functionality. Non-essential services can fail completely without impacting alert delivery. When impacted by an incident, the system automatically sheds non-critical load, such as:

- Analytics and reporting pipelines continue collecting data but may delay processing
- User preference updates queue for later processing
- Historical data queries return cached results
- UI dashboards show slightly stale data

Meanwhile, the critical path — alert ingestion, routing, and delivery — continues at full speed with full functionality. Users receive their alerts with the same reliability and speed, even as other parts of the system gracefully degrade. This selective degradation is possible only because of the complete isolation between services. There are no hidden dependencies that could cause unexpected impacts on critical paths.

Operational Excellence and Continuous Reliability

Beyond Building: Operating for Reliability

Creating reliable systems is only the beginning. Maintaining that reliability requires operational excellence that matches the architectural sophistication. At FireHydrant, this starts with comprehensive monitoring that goes beyond simple uptime metrics.

End-to-End Synthetic Monitoring continuously exercises the complete alert path. Synthetic alerts flow through the entire system — from initial ingestion through final delivery — validating that every component functions correctly. These tests run from multiple global locations, ensuring that regional issues are detected immediately.

Component-Level Health Tracking provides deep visibility into system internals. Every service exposes detailed metrics about its operations: queue depths, processing latencies, error rates, and resource utilization. These metrics feed into sophisticated anomaly detection that identifies problems before they impact users.

Proactive Capacity Management ensures that systems never approach their limits. Automated scaling responds to load changes, while capacity planning models predict future needs based on growth trends. Regular load testing validates that systems can handle 10x current volumes, providing confidence during unexpected spikes.

Building Customer Trust Through Transparency

Proven Track Record of Reliability

Trust in FireHydrant Signals comes from demonstrated reliability and openness. Here's how we accomplish that:

Published SLO Performance: Historical achievement of uptime targets, including 99.99% uptime for Signals.

Detailed Architecture Documentation: Transparent explanation of how reliability is achieved.

Third-Party Audits and Certifications: SOC 2 Type II reports, ISO certifications, and penetration testing provide independent validation.

Customer References and Case Studies: Customers that successfully run their on-call and alerting on Signals, including Qlik, Backblaze, and AuditBoard.

CONCLUSION

A New Standard for Reliability

FireHydrant Signals represents a fundamental shift: from simple notifications to sophisticated reliability infrastructure. By combining proven architectural patterns, operational excellence, and transparent communication, Signals has been able to achieve reliability levels that were previously impossible.

The 99.99% uptime achieved through these approaches isn't just a metric — it represents our commitment to being available when customers need it most. Through multi-region deployments, service independence, comprehensive redundancy, and operational excellence, Signals has become the bedrock upon which reliable operations are built.

For organizations evaluating alerting solutions, the message is clear: reliability can no longer be treated as a nice-to-have feature. With Signals, reliability is built into every architectural decision, setting a new standard for what alerting infrastructure should deliver.

For more information about implementing enterprise-grade alerting infrastructure that your organization can depend on, ***contact the FireHydrant team****.*