IDERA® | SQL Server Whitepaper

7 STEPS TO ALERT EFFECTIVELY

TABLE OF CONTENTS

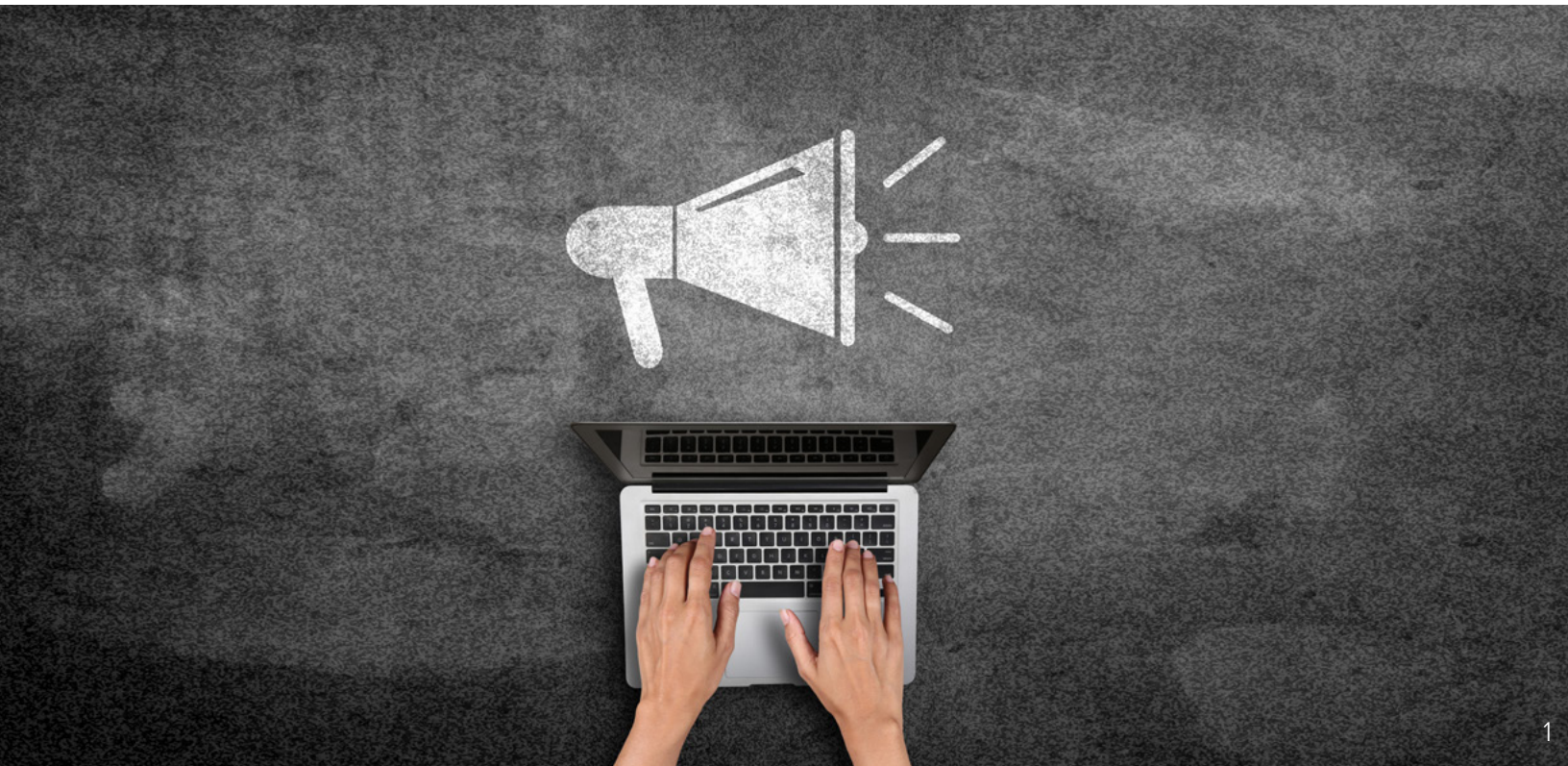
INTRODUCTION

Monitoring systems increase visibility into infrastructures and applications. Such monitoring systems also define acceptable ranges of performance and reliability. Begin to plan monitoring strategies that cover all critical parts of systems. Start by understanding what components and metrics to measure for different cases.

Graphs and dashboards are go-to tools for understanding data in systems. However, these graphs and dashboards are only useful in contexts where the staff is viewing pages. One of the most critical responsibilities of monitoring systems is to relieve staff from actively watching systems. That way staff can pursue more valuable tasks. Systems must ask for attention when necessary to ensure awareness of significant changes to make this feasible. Monitoring systems use end-user-defined metric thresholds and alert systems to accomplish this.

Automated alerts are essential to performance monitoring. Automated alerts enable spotting issues in infrastructures, identifying their causes, and rapidly minimizing degradation and disruption of services. Alerts draw human attention to particular systems that require observation, inspection, and intervention when metrics and other measurements facilitate observability. Automated alerts save time by removing regular manual checks of metrics. Automate alerts across as many of systems as possible to respond quickly to issues and to provide better service.

However, alerts are not always as useful as they can be. In particular, noisy alarms often mask real issues. Alerts sent to the wrong staff delay responses. A lack of helpful context in alert messages may cause redundant manual efforts. Moreover, a lack of automated remediation causes unnecessarily high workloads and distractions for staff. To improve the effectiveness of alerts, follow the seven steps that this whitepaper presents.



DETERMINE TYPES OF ISSUES

Notifications, and pages first determine the types of issues and how to handle them when considering setting alerts. Issues can be real and not real. Real issues can be urgent and not urgent.

Real Issues

Symptoms require attention when they are real. Issues should trigger alerts when they demand attention. These alerts should notify someone who can investigate and fix issues. Send notifications via email, chat, and issue tracking systems so that recipients can prioritize their response. Automate responses to issues when that is feasible.

Non-real Issues

Issues should not trigger alerts when they are not symptomatic of practical issues. Such issues include issues that are fully expected to occur, issues that are caused by intentional human intervention, and issues in test environments.

Urgent Issues

Staff should intervene immediately when critical systems stop performing useful work at acceptable rates. Issues should trigger pages when they are compelling.

Non-urgent Issues

Not all issues are emergencies that require immediate attention. Alerting, notifications, and especially paging on issues that are not urgent contributes to alert fatigue and can cause staff to ignore more pressing issues. Also, avoid calling someone away from work, sleep, and personal time. Record triggered alerts in monitoring systems for later analysis and correlation when these alerts do not link to notifications.

DISTINGUISH BETWEEN EXTERNAL AND INTERNAL MONITORING

Define thresholds and alerts via external monitoring and internal monitoring. External and internal monitoring describe different models for monitoring. These two models are not mutually exclusive. Often systems use mixtures of each model to take advantage of their unique strengths.

External Monitoring

External monitoring describes alert definitions based only on externally visible factors. This model of monitoring takes outside perspectives to maintain focus on the behavior of systems that face end-users. External monitoring provides data about functionalities of systems from perspectives of end-users with no specialized knowledge of the health of underlying components. This model may seem restrictive. However, this data map closely to issues that are actively affecting end-users. As such, these issues are good candidates for alert triggers.

Internal Monitoring

Internal monitoring describes monitoring based on privileged, inside data about infrastructures of systems. There exists far more internal data than external data because the amount of internal processes vastly exceeds externally visible behavior. Internal monitoring can be predictive since it operates with more comprehensive data about systems.

Combined External and Internal Monitoring

External and internal are merely two methods to categorize different types of views into systems. With internal data the internals of systems is visible. Use internal data to investigate issues, assess causes, and find correlated factors when issues are known and for regular administration purposes. At the same time, use external monitoring to detect severe issues quickly by immediately demonstrating the impact on end-users.

CLASSIFY BY LEVELS OF SEVERITY

Alerts and notifications are some of the most critical parts of monitoring systems. Staff is not aware of events impacting systems and need to continuously monitor dashboards to stay informed without notifications about significant changes. In contrast, overly aggressive messaging with high fractions of false positives, non-urgent events, and ambiguous messaging does more harm than good.

Different alerts have different degrees of importance. Some alerts require immediate human intervention. Other alerts need future human intervention. Moreover, some alerts point to areas where attention should focus on in the future. At least log all alerts to central locations for correlation with other metrics and events.

High Severity

Focus on the alerts that are most urgent. Such high-severity alerts should escalate to pages via pagers to request human attention urgently. Response times facing end-users should have internal service-level agreements that are at least as aggressive as the strictest service-level agreement that faces end-users. Any response times exceeding internal service-level agreements warrants immediate attention.

Pages

Pages are notifications that attempt to call attention to critical issues with systems urgently. Pages start with alerts of the highest priority. Use alerts of highest priority for cases that demand immediate resolution due to their severity. Paging systems require reliable, aggressive ways of reaching out to staff with responsibilities and power to work on resolving issues.

Reserve pages for critical issues with systems. Systems send pages as essential alerts. Good paging systems are reliable, persistent, and sufficiently aggressive that ignoring them is not reasonable. Paging systems should include options to notify secondary staff when initial pages are not acknowledged within certain amounts of time to ensure timely responses.

Use pages only when strictly needed because they are incredibly disruptive. Just use pages when it is clear that there are operationally unacceptable issues. As such, pages are tied to observed symptoms in systems using external techniques.

Highest priority notifications are intended for critical issues currently affecting production.

Page on Symptoms

Pages are appropriate kinds of alert for when systems stop performing useful work with acceptable throughput, latency, and error rates. It is essential to find out about such issues immediately. Pages deliver crucial data concisely. However, pages disrupt staff when overused and when they are linked to poorly designed alerts. When systems stop performing useful work these are symptoms of underlying issues. Symptoms refer to manifestations of issues that may have any number of different causes. Whenever possible, implement pages around signs rather than reasons. That is, use metrics that separate symptoms from causes. Paging on symptoms revels real issues that often face end-users rather than theoretical and internal issues.

Set up alerts based on cases with real end-user impact. Such alerts mean analyzing failures and performance-degrading cases and understanding how and when they appear to end-users. Understand infrastructures, relationships between different components, and goals of the organization for availability and performance. Discover symptomatic metrics that can reliably indicate present and impending end-user-impacting issues.

Also, page on symptoms because alerts triggered by symptoms are often durable. Durability refers to the fact that it is not necessary to update alert definitions when underlying architectures of systems change.

Page on Early Warnings

The exception to the rule are metrics for early warnings. Call attention to small sets of metrics for early warnings even when systems perform adequately. Metrics for early warnings refer to unacceptably high probabilities that severe symptoms may soon develop. These severe symptoms require immediate intervention. There is no need to wake anyone at night time when it is possible to notify with plenty of lead time. Better yet, anticipate problematic cases and implement automated remediation.

A typical example is disk space: Systems can recover after running out of free memory and computational processing power. However, when running out of disk space, systems do not likely recover by themselves. After short times systems probably come to hard stops without disk space. Automatically remediate low disk space by removing erasable data (such as logs and redundant data that also exists at another location).

Moderate Severity

Moderately severe alerts require intervention, but not immediately. Notify by sending emails and posting notifications in chat rooms of service owners for such moderate-severity alerts. Both message types are highly visible. Both message types do not wake anyone at night time and disrupt workflows of staff.

Secondary Notifications

Stepping down in severity are secondary notifications. Examples of secondary notifications are emails and tickets. Secondary notifications are designed to leave persistent reminders that staff should investigate developing cases when possible. Notification alerts do not require immediate action.

Consequently, notifications are usually alert working staff rather than on-call staff. Align notifications to cases that can wait until subsequent next work days when organizations do not have staff working at all times.

Secondary notifications generated by monitoring systems enable staff to understand the work that they should focus on when they are available again. Such non-critical alerts are frequently internal indicators that can predict and identify evolving issues that need a resolution soon. Alternatively, notification alerts monitor the same behavior as paging alerts but respond to less critical thresholds.

Notifications require responses but do not pose immediate threats to the stabilities of systems. In these cases, bring awareness to issues so that staff can investigate and mitigate before it impacts end-users and transforms into more significant issues.

Low Severity

Many alerts do not associate with service issues. Consequently, humans may never need to be aware of such alerts. Low-severity issues should trigger low-urgency alerts that monitoring systems recorded for future reference and investigation. However, such low-urgency alerts should not interrupt the work of anyone. After all, transient issues may be the cause, and they often dissipate on their own. Nonetheless, alert-based data provides invaluable context for subsequent investigation when monitoring systems start returning large numbers of problematic metrics.

Logging

Staff may want to note observed behavior in places that are easily accessible later without bringing it to the attention of anyone else immediately. For this purpose, set up thresholds that log data. Write logs to files and use logs to increment counters on dashboards within monitoring systems. Provide readily compiled data for investigative purposes to reduce the number of queries staff must construct to gather data.

Logging of data makes sense for triggers that have low priority and do not need responses on their own. Use logging of data to correlate related factors and summarize point-in-time data to reference later as supplemental sources at later times. Use such triggers to look up the same data each time issues surface. Limit the number of triggers of this type. Such triggers for logging of data are technically not alerts.

Alternatives that provide similar benefits are custom investigative dashboards and saved queries.

SET UP PROPER ALERT DEFINITIONS

Alert systems notify staff when data indicates necessary changes reliably and to leave them alone otherwise. Define alerting criteria so that systems recognize significant events. Alert definitions are composed of notification methods and metric thresholds that systems continuously evaluate based on incoming data. Limits usually define maximum and minimum average data for metrics over specified time frames. In contrast, notification methods describe how to send out alerts.

One of the most challenging parts of alerting is finding the balance that enables responsiveness to issues while not over alerting. Accomplish this by understanding what metrics are the best indications of real issues, what issues require attention immediately, and what methods of notification to use for different cases. Threshold definition languages must be sufficiently robust to describe criteria adequately to support this. Similarly, the component for notification must offer methods of communicating appropriately for various levels of severity.

Determine Thresholds

After identifying symptomatic metrics, the subsequent challenge is determining appropriate data to implement as thresholds with graduated severity. Consider using trial and error to discover appropriate limits for some parameters. Check available historical data to determine what cases required remediation in the past. For each metric, define emergency thresholds that trigger pages. Also, define warning thresholds for messages of lower priority.

Fine Tune with Feedback

After establishing new alerts, ask for feedback on whether limits were overly aggressive and not sensitive. Use such input to fine-tune systems to best align to expectations of staff.

ALERT RELEVANT STAFF

Alerts are only useful when they are actionable. Whether alerts are actionable depends on levels of knowledge, experience, and permissions of responding individuals. Deciding on what staff to alert can be either straightforward or ambiguous. Developing on-call rotations for different staff and designing escalation plans removes some of the ambiguity in these decisions.

On-call Rotations

On-call rotations include sufficiently capable individuals to avoid burnout and alert fatigue. Alerting systems should comprise mechanisms for scheduling on-call shifts. Without such scheduling mechanisms, develop procedures to rotate manually alert contacts based on schedules. Owners of specific parts of systems can populate multiple on-call rotations.

Escalation Plans

Escalation plans are another tool to ensure that incidents route to appropriate staff. Send alerts generated by monitoring systems to on-shift staff rather than on-call rotations when staff cover systems 24 hours each day. Responders can then perform mitigation themselves and decide to manually page on-call staff when they need additional assistance and expertise. Set up plans that outline when and how to escalate issues to minimize unnecessary alerts and to maintain the sense of urgency that pages are to convey.

SHARE CONTEXT

Reducing the time it takes for responders to begin investigating issues helps to recover from incidents faster. For this purpose, try to provide a context within alert texts so that staff can understand cases quickly and start working on appropriate next steps.

Alerts should indicate affected components and systems, metric thresholds that were triggered, and times that incidents began. Alerts should also provide links to additional data. For example, include links to specific dashboards associated with triggered metrics, to issue tracking systems when generating automated tickets, and to alerts pages of monitoring systems where more detailed context is available.

Provide staff with sufficient data to formulate their initial response and to enable them to focus on incidents. Providing every piece of data concerning event is neither required nor recommended. However, giving necessary details with several options for where to navigate next shortens initial discovery phases of responses.

AUTOMATE REMEDIATION

Clarify what alerts different staff share. Each alert should signify that issues are occurring that requires manual human action and input on decisions. With such focus, determine opportunities to automate reactions when considering metrics to alert on.

Design automated remediation when recognizable signatures reliably identify issues, responses are always the same, and responses do not require any human input and decision making. Some reactions are simpler to automate than others. However, it is generally possible to script away any case that fits the criteria mentioned above.

Automated responses are still tied to alert thresholds. Nevertheless, rather than sending messages to staff, triggers initiate scripted remediations to solve issues. Logging each alert provides valuable data about the health of systems and the effectiveness of metric thresholds and automated measures.

Automated processes can experience issues as well. Consequently, add extra alerting to scripted responses to notify staff when automation fails. As such, hands-off reactions handle the majority of cases. Moreover, systems inform staff of incidents that require intervention.

SUMMARY

This whitepaper presented seven steps to alert effectively. First, determine the types of issues regarding real issues, non-real issues, urgent issues, and non-urgent issues. Second, distinguish between external and internal monitoring. Third, classify by levels of severity and select relevant methods of notification. Fourth, set up proper alert definitions, determine meaningful thresholds, and fine-tune with feedback. Fifth, alert relevant staff and set up escalation plans. Sixth, share context in alert messages. And finally, automate remediation when feasible.

SQL DIAGNOSTIC MANAGER

With SQL Diagnostic Manager, configure alerts to inform and warn about approaching issues with SQL Server instances. View these alerts using the desktop, web, and mobile console, and the newsfeed.When reaching alert thresholds, send email notifications, pop up alert messages in the Windows taskbar, write events to the Windows Event log, generate events on the timeline, and send alert messages to the newsfeed action provider. After correcting problems triggering alerts, alert again when situations recur.

Choose from more than 100 pre-defined and configurable alert settings based on industry best practices with advanced configuration settings to allow for greater flexibility. Configure general alert settings as templates to apply to other servers and groups of servers. Set different alert thresholds for each database or disk within each monitored instance. Capture baselines of past performance of monitored instances to determine whether alert thresholds are excessively noisy or cause false positive. When metrics are continually alerting, view flags to indicate that changes may be needed and see recommendations for new limits.

Easily snooze alerts and groups of alerts for a specified time to prevent alerts from recurring while resolving problems. Determine the length of time that an issue must occur before sending first alerts for problematic metrics that occasionally spike for very short durations. Disable data collection and alerting during maintenance periods to avoid false positives. Maintenance mode can be on-demand, one-time, and or weekly scheduled maintenance periods. Enable maintenance mode via PowerShell to integrate with outside job scheduling.

Start for FREE