An Introduction to Metrics, Monitoring, and Alerting

Posted December 5, 2017 32.7k MONITORING CONCEPTUAL

By: Justin Ellingwood

Introduction

Understanding the state of your infrastructure and systems is essential for ensuring the reliability and stability of your services. Information about the health and performance of your deployments not only helps your team react to issues, it also gives them the security to make changes with confidence. One of the best ways to gain this insight is with a robust monitoring system that gathers metrics, visualizes data, and alerts operators when things appear to be broken.

In this guide, we will discuss what metrics, monitoring, and alerting are. We will talk about why they are important, what types of opportunities they provide, and the type of data you may wish to track. We will be introducing some key terminology along the way and will end with a short glossary of some other terms you might come across while exploring this space.

What Are Metrics, Monitoring and Alerting?

Metrics, monitoring, and alerting are all interrelated concepts that together form the basis of a monitoring system. They have the ability to provide visibility into the health of your systems, help you understand trends in usage or behavior, and to understand the impact of changes you make. If the metrics fall outside of your expected ranges, these systems can send notifications to prompt an operator to take a look, and can then assist in surfacing information to help identify the possible causes.

In this section, we'll take a look at these individual concepts and how they fit together.

What Are Metrics and Why Do We Collect Them?

Metrics represent the raw measurements of resource usage or behavior that can be observed and collected throughout your systems. These might be low-level usage summaries provided by the operating system, or they can be higher-level types of data tied to the specific functionality or work of a component,

like requests served per second or membership in a pool of web servers. Some metrics are presented in relation to a total capacity, while others are represented as a rate that indicates the "busyness" of a component.

Often, the easiest metrics to begin with are those already exposed by your operating system to represent the usage of underlying physical resources. Data about disk space, CPU load, swap usage, etc. are already available, provide value immediately, and can be forwarded to a monitoring system without much additional work. Many web servers, database servers, and other software also provide their own metrics which can be passed forward as well.

For other components, especially your own applications, you may have to add code or interfaces to expose the metrics you care about. Collecting and exposing metrics is sometimes known as adding **instrumentation** to your services.

Metrics are useful because they provide insight into the behavior and health of your systems, especially when analyzed in aggregate. They represent the raw material used by your monitoring system to build a holistic view of your environment, automate responses to changes, and alert human beings when required. Metrics are the basic values used to understand historic trends, correlate diverse factors, and measure changes in your performance, consumption, or error rates.

What is Monitoring?

While metrics represent the data in your system, monitoring is the process of collecting, aggregating, and analyzing those values to improve awareness of your components' characteristics and behavior. The data from various parts of your environment are collected into a **monitoring system** that is responsible for storage, aggregation, visualization, and initiating automated responses when the values meet specific requirements.

In general, the difference between metrics and monitoring mirrors the difference between data and information. Data is composed of raw, unprocessed facts, while information is produced by analyzing and organizing data to build context that provides value. Monitoring takes metrics data, aggregates it, and presents it in various ways that allow humans to extract insights from the collection of individual pieces.

Monitoring systems fulfill many related functions. Their first responsibility is to accept and store incoming and historical data. While values representing the current point in time are useful, it is almost always more helpful to view those numbers in relation to past values to provide context around changes and trends. This means that a monitoring system should be capable of managing data over periods of time, which may involve sampling or aggregating older data.

Secondly, monitoring systems typically provide visualizations of data. While metrics can be displayed and understood as individual values or tables, humans are much better at recognizing trends and understanding how components fit together when information is organized in a visually meaningful way. Monitoring systems usually represent the components they measure with configurable graphs and dashboards. This makes it possible to understand the interaction of complex variables or changes within a system by glancing at a display.

An additional function that monitoring systems provide is organizing and correlating data from various inputs. For the metrics to be useful, administrators need to be able to recognize patterns between different resources and across groups of servers. For example, if an application experiences a spike in error rates, an administrator should be able to use the monitoring system to discover if that event coincides with the capacity exhaustion of a related resource.

Finally, monitoring systems are typically used as a platform for defining and activating alerts, which we will talk about next.

What is Alerting?

Alerting is the responsive component of a monitoring system that performs actions based on changes in metric values. Alerts definitions are composed of two components: a metrics-based condition or threshold, and an action to perform when the values fall outside of the acceptable conditions.

While monitoring systems are incredibly useful for active interpretation and investigation, one of the primary benefits of a complete monitoring system is letting administrators disengage from the system. Alerts allow you to define situations that make sense to actively manage, while relying on the passive monitoring of the software to watch for changing conditions.

While notifying responsible parties is the most common action for alerting, some programmatic responses can be triggered based on threshold violations as well. For instance, an alert that indicates that you need more CPU to process the current load can be responded to with a script that auto-scales that layer of your application. While this isn't strictly an alert since it doesn't result in a notification, the same monitoring system mechanism can often be used to kick off these processes as well.

However, the main purpose of alerting is still to bring human attention to bear on the current status of your systems. Automating responses is an important mechanism for ensuring that notifications are only triggered for situations that require consideration from a knowledgeable human being. The alert itself should contain information about what is wrong and where to go to find additional information. The individual responding to the alert can then use the monitoring system and associated tooling like log files to investigate the cause of the problem and implementing a mitigation strategy.

Infrastructure of even moderate complexity requires distinctions in alert severity so that the responsible teams or individuals can be notified using methods appropriate to the scale of the problem. For instance, rising utilization of storage might warrant a work ticket or email, while an increase in client-facing error rates or unresponsiveness might require sending a page to on-call staff.

What Type of Information Is Important to Track?

The types of values you monitor and the information you track will probably change as your infrastructure evolves. Since systems usually function hierarchically, with more complex layers building on top of more primitive infrastructure, it can be useful to think about the metrics available at these different levels when planning your monitoring strategy.

Host-Based Metrics

Towards the bottom of the hierarchy of primitive metrics are host-based indicators. These would be anything involved in evaluating the health or performance of an individual machine, disregarding for the moment its application stacks and services. These are mainly comprised of usage or performance of the operating system or hardware, like:

- CPU
- Memory
- Disk space
- Processes

These can give you a sense of factors that may impact a single computer's ability to remain stable or perform work.

Application Metrics

The next category of metrics you may want to look at are application metrics. These are metrics concerned with units of processing or work that depend on the host-level resources, like services or applications. The specific types of metrics to look at depends on what the service is providing, what dependencies it has, and what other components it interacts with. Metrics at this level are indicators of the health, performance, or load of an application:

- Error and success rates
- Service failures and restarts
- Performance and latency of responses
- Resource usage

These indicators help determine whether an application is functioning correctly and with efficiency.

Network and Connectivity Metrics

For most types of infrastructure, network and connectivity indicators will be another dataset worth exploring. These are important gauges of outward-facing availability, but are also essential in ensuring that services are accessible to other machines for any systems that span more than one machine. Like the other metrics we've discussed so far, networks should be checked for their overall functional correctness and their ability to deliver necessary performance by looking at:

- Connectivity
- Error rates and packet loss
- Latency
- Bandwidth utilization

Monitoring your networking layer can help you improve the availability and responsiveness of both your internal and external services.

Server Pool Metrics

When dealing with horizontally scaled infrastructure, another layer of infrastructure you will need to add metrics for is pools of servers. While metrics about individual servers are useful, at scale a service is better represented as the ability of a collection of machines to perform work and respond adequately to requests. This type of metric is in many ways just a higher level extrapolation of application and server metrics, but the resources in this case are homogeneous servers instead of machine-level components. Some data you might want to track are:

- Pooled resource usage
- Scaling adjustment indicators
- Degraded instances

Collecting data that summarizes the health of collections of servers is important for understanding the actual capabilities of your system to handle load and respond to changes.

External Dependency Metrics

Other metrics you may wish to add to your system are those related to external dependencies. Often, services provide status pages or an API to discover service outages, but tracking these within your own systems—as well as your actual interactions with the service—can help you identify problems with your providers that may affect your operations. Some items that might be applicable to track at this level are:

- Service status and availability
- Success and error rates
- Run rate and operational costs
- Resource exhaustion

There are many other types of metrics that can be helpful to collect. Conceptualizing the most important information at varying levels of focus can help you identify indicators that are most useful for predicting or identifying problems. Keep in mind that the most valuable metrics on higher levels are likely to be resources provided by lower layers.

Factors That Affect What You Choose to Monitor

For peace of mind, in an ideal world you would track everything related to your systems from the beginning in case an item may one day be relevant to you. However, there are many reasons why this might not be possible or even desirable.

A few factors that can affect what you choose to collect and act on are:

- **Resources available for tracking**: Depending on your human resources, infrastructure, and budget, you will have to limit the scope of what you keep track of to what you can afford to implement and reasonably manage.
- **The complexity and purpose of your application**: The complexity of your application or systems can have a large impact on what you choose to track. Items that might be mission critical for some software might not be important at all in others.
- **The deployment environment**: While robust monitoring is most important for production systems, staging and testing systems also benefit from monitoring, though there may be differences in severity, granularity, and the overall metrics measured.
- **The likelihood of the metric being useful**: One of the most important factors affecting whether something is measured is its potential to help in the future. Each additional metric tracked increases the complexity of the system and takes up resources. The necessity of data can change over time as well, requiring reevaluation at regular intervals.
- **How essential stability is**: Simply put, stability and uptime might not be priorities for certain types of personal or early stage projects.

The factors that influence your decisions will depend on your available resources, the maturity of your project, and the level of service you require.

Important Qualities of a Metrics, Monitoring, and Alerting System

While each monitoring application or service will have its strengths and weaknesses, the best options often share some important qualities. A few of the more important characteristics to look for when evaluating monitoring systems are below.

Independent from Most Other Infrastructure

One of the most basic requirements of an adequate monitoring system is to be external to other services. While it's sometimes useful to group services together, a monitoring system's core responsibilities, its helpfulness in diagnosing problems, and its relationship to the watched systems means that it's important for your monitoring system to be independently accessible. Your monitoring system will inevitably have some effect on the systems it monitors, but you should aim to keep this minimal to reduce the impact your tracking has on performance and to increase the reliability of your monitoring in the event of other system problems.

Reliable and Trustworthy

Another basic requirement is reliability. As a monitoring system is responsible for gathering, storing, and providing access to high value information, it is important that you can trust it to operate correctly on a daily basis. Dropped metrics, service outages, and unreliable alerting can all have an immediate harmful impact on your ability to manage your infrastructure effectively. This applies not only to the core software reliability, but also to the configuration you enable, since mistakes like inaccurate alerting can lead to a loss of trust in the system.

Easy to Use Summary and Detail Views

The ability to display high-level summaries and ask for greater detail on-demand is an important feature to ensure that the metrics data is useful and consumable to human operators. Designing dashboards that present the most commonly viewed data in an immediately intelligible manner can help users understand system state at a glance. Many different dashboard views can be created for different job functions or areas of interest.

Equally important is the ability to drill down from within summary displays to surface the information most pertinent to the current task. Dynamically adjusting the scale of graphs, toggling off unnecessary metrics, and overlaying information from multiple systems is essential to make the tool useful interactively for investigations or root cause analysis.

Effective Strategy for Maintaining Historical Data

A monitoring system is most useful when it has a rich history of data that can help establish trends, patterns, and consistencies over long timelines. While ideally, all information would be retained indefinitely in its original granularity, cost and resource constraints can sometimes make it necessary to store older data at a reduced resolution. Monitoring systems with the flexibility to work with data both at full granularity and in a sampled format provide a wider range of options for how to handle an ever increasing amount of data.

A related feature that is helpful is the ability to easily import existing data sets. If reducing the information density of your historic metrics is not an attractive option, offloading older data to a long-term storage solution might be a better alternative. In this case, you don't need to maintain older data within the system, but you need to be able to reload it in bulk when you wish to analyze or use it.

Able to Correlate Factors from Different Sources

The monitoring system is responsible for providing a holistic view of your entire infrastructure, so it needs to be able to display related information, even if it comes from different systems or has different characteristics. Administrators should be able to glue together information from disparate parts of their systems at will to understand potential interactions and overall status across the entire infrastructure. Ensuring that time synchronization is configured across your systems is a prerequisite to being able to correlate data from different systems reliably.

Easy to Start Tracking New Metrics or Infrastructure

In order for your monitoring system to be an accurate representation of your systems, you need to be able to make adjustments as the machines and infrastructure change. A minimal amount of friction when adding additional machines will help you do so. Equally important is the ability to easily remove decommissioned machines without destroying the collected data associated with them. The system should make these operations as simple as possible to encourage setting up monitoring as part of the instance provisioning or retirement process.

A related ability that is important is the ease in which the monitoring system can be set up to track entirely new metrics. This depends on the way that metrics are defined in the core monitoring configuration as well as the variety and quality of mechanisms available to send metric data to the system. Defining new metrics

is usually more complex than adding additional machines, but reducing the complexity of adding or adjusting metrics will help your team respond to changing requirements in an appropriate time frame.

Flexible and Powerful Alerting

One of the most important aspects of a monitoring system to evaluate is its alerting capabilities. Aside from very strict reliability requirements, the alerting system need to be flexible enough to notify operators through multiple mediums and powerful enough to be able to compose thoughtful, actionable notification triggers. Many systems defer the responsibility of actually delivering notifications to other parties by offering integrations with existing paging services or messenger applications. This minimizes the responsibility of the alerting functionality and usually provides more flexible options since the plugin just needs to consume an external API.

The part that the monitoring system cannot defer, however, is defining the alerting parameters. Alerts are defined based on values falling outside of acceptable ranges, but the definitions can require some nuance in order to avoid over alerting. For instance, momentary spikes are often not a concern, but sustained elevated load may require operator attention. Being able to clearly define the parameters for an alert is a requirement for composing a robust, trustworthy set of alert conditions.

Additional Terminology

As you explore the monitoring ecosystem, you'll start to encounter a set of shared terminology that is frequently used to discuss characteristics of monitoring systems, the data being handled, and different trade offs that require consideration. While in no way exhaustive, the list below can help introduce you to some of the terms you're most likely to come across.

- **Observability**: Although not strictly defined, observability is a general term used to describe processes and techniques related to increasing awareness and visibility into systems. This can include monitoring, metrics, visualization, tracing, and log analysis.
- **Resource**: In the context of monitoring and software systems, a resource is any exhaustible or limited dependency. What is considered a resource can vary greatly based on part of the system being discussed.
- **Latency**: Latency is a measure of the time it takes to complete an action. Depending on the component, this can be a measure of processing, response, or travel time.
- **Throughput**: Throughput represents the maximum rate of processing or traversal that a system can handle. This can be dependent on software or hardware design. Often there is an important distinction between theoretical throughput and practical observed throughput.
- **Performance**: Performance is a general measure of how efficiently a system is completing work. Performance is an umbrella term that often encompasses work factors like throughput, latency, or resource consumption.
- **Saturation**: Saturation is a measure of the amount of capacity being used. Full saturation indicates that 100% of the capacity is currently in use.
- **Visualization**: Visualization is the process of presenting metrics data in a format that allows for quick, intuitive interpretation through graphs or charts.

- **Log aggregation**: Log aggregation is the act of compiling, organizing, and indexing log files to allow for easier management, searching, and analysis. While separate from monitoring, aggregated logs can be used in conjunction with the monitoring system to identify causes and investigate failures.
- **Data point**: A data point is a single measurement of a single metric.
- **Data set**: A data set is a collection of data points for a metric.
- **Units**: Units are the context for a measured value. A unit defines the magnitude, scope, or quantity of a measurement to understand extent and allow comparison.
- **Percentage Units**: Percentage units are measurements that are taken as a part of a finite whole. A percentage unit indicates how much a value is out of the total possible amount.
- **Rate Units**: Rate units indicate the magnitude of a metric over a constant period of time.
- **Time series**: Time series data is a series of data points that represent changes over time. Most metrics are best represented by a time series because single data points often represent a value at a specific time and the resulting series of points is used to show changes over time.
- **Sampling rate**: Sample rate is a measurement of how often a representative data point is collected in lieu of continuous collection. A higher sampling rate more accurately represents the measured behavior, but requires more resources to handle the extra data points.
- **Resolution**: Resolution refers to the density of data points that make up a data set. Collections with higher resolutions over the same time frame indicate a higher sample rate and a more granular view of the same behavior.
- **Instrumentation**: Instrumentation is the ability to track the behavior and performance of software. This is accomplished by adding code and configuration to software to output data that can then be consumed by a monitoring system.
- **The observer effect**: The observer effect is the impact of the monitoring system itself on the phenomena being observed. Since monitoring takes up resources, the act of measuring behavior and performance will alter the values produced. Monitoring systems seek to avoid adding unnecessary overhead to minimize this impact.
- **Over-monitoring**: Over-monitoring occurs when the quantity of metrics and alerts configured is inversely related to their usefulness. Over-monitoring can cause stress on the infrastructure, make it difficult to find relevant data, and cause teams to lose trust in their monitoring and alerting systems.
- **Alert fatigue**: Alert fatigue is the human response of desensitivity that results from frequent, unreliable, or improperly prioritized alerts. Alert fatigue can cause operators to ignore severe problems and is usually an indication that alert conditions need to be reevaluated.
- **Threshold**: When alerting, a threshold is the boundary between acceptable and unacceptable values which triggers an alert if exceeded. Often alerts are configured to trigger when a value exceeds the threshold for a certain period of time, in order to avoid sending an alert for temporary spikes.
- **Quantile**: A quantile is a dividing point used to separate a dataset into distinct groups based on their values. Quantiles are used to put values into "buckets" that represent segments of a population of data. Often, this is used to separate common values from outliers to better understand what constitutes representative and extreme cases.

- **Trend**: A trend is the general direction that a set of values is indicating. Trends are more reliable than single values in determining the general state of the component being tracked.
- **White-box monitoring**: White-box monitoring is a term used to describe monitoring that relies on access to internal state of the components being measured. White-box monitoring can provide a detailed understanding of system state and is helpful for identifying causes of problems.
- **Black-box monitoring**: Black-box monitoring is monitoring that observes the external state of a system or component by looking only at its inputs, outputs, and behavior. This type of monitoring can closely align with a user's experience of a system, but is less useful for finding the cause of problems.

Conclusion

Gathering metrics, monitoring components, and configuring alerts is an essential part of setting up and managing production infrastructure. Being able to tell what is happening within your systems, what resources need attention, and what is causing a slowdown or outage is invaluable. While designing and implementing your monitoring setup can be a challenge, it is an investment that can help your team to prioritize their work, delegate the responsibility of oversight to an automated system, and understand the impact of your infrastructure and software on your stability and performance.

By: Justin Ellingwood

Tutorial Series

An Introduction to Infrastructure and Application Monitoring

In this series, we explore what metrics and monitoring are and how to best use them to gain visibility into your systems and the responsiveness of your team. Collecting metrics from your infrastructure gives you insight into the health and performance of your systems. Metrics can be used to create dashboards to troubleshoot issues or give a summary of the state of your applications and resources. Alerts can be defined to notify you as soon as situations require your attention.

Show Tutorials

Related Tutorials

Putting Monitoring and Alerting into Practice

Gathering Metrics from Your Infrastructure and Applications

An Introduction to Tracking Statistics with Graphite, StatsD, and CollectD

How To Use Traefik as a Reverse Proxy for Docker Containers on Debian 9

How To Install Elasticsearch, Logstash, and Kibana (Elastic Stack) on CentOS 7

0 Comments

Leave a comment...

Log In to Comment