Making Everything Easier!™

SolarWinds Special Edition

Systems Monitoring

FOR DUMMIES®

A Wiley Brand

Learn:

- Why you need server and application monitoring
- Monitoring frameworks and technologies
- Best practices
- Tips for cloud and hybrid IT

Brought to you by

solarwinds

Leon Adato

About SolarWinds

SolarWinds provides powerful and affordable IT operations management software to more than 150,000 customers worldwide — from Fortune 500 enterprises to small businesses. SolarWinds' products are downloadable, easy to use and maintain, and provide the power, scale, and flexibility needed to manage today's IT environments. SolarWinds' growing online community, THWACK®, offers users problem-solving and technology-sharing for all of SolarWinds' products. This active user-community input is combined with decades of IT management experience to deliver a wide range of solutions and tools to address the real-world needs of IT professionals.

Systems Monitoring

FOR DUMMIES®

A Wiley Brand

SolarWinds Special Edition

by Leon Adato

FOR DUMMIES®

A Wiley Brand

Systems Monitoring For Dummies, SolarWinds Special Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN: 978-1-119-46049-7 (pbk); ISBN: 978-1-119-46059-6 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

Project Editor: Carrie A. Burchfield
Acquisitions Editor: Amy Fandrei
Editorial Manager: Rev Mengle
Business Development Representative: Kimberley Schumacker
Production Editor: G. Vasanth Koilraj

Introduction

Systems crash unexpectedly, users make bizarre claims about how the Internet is slow, and managers request statistics that leave you scratching your head wondering how to collect them in a way that's meaningful and doesn't consign you to the headache of hitting Refresh and spending half the day writing down numbers on a piece of scratch paper just to get a baseline for a report.

The answer to all these challenges (and many, many more) lies in systems monitoring — effectively monitoring the servers and applications in your environment by collecting statistics and/or checking for error conditions so you can act or report effectively when needed.

About This Book

To build an effective systems monitoring solution, the true starting point is understanding the fundamental concepts. You have to know what monitoring *is* before you can set up what monitoring *does*. For that reason, this book introduces you to the underpinnings of monitoring techniques, theory, and philosophy, as well as the ways in which systems monitoring is accomplished.

Oh, and please note that at no point do I discuss specific software. At the end of the day, ping is still just ping, no matter how pretty a wrapper you put around it.

Icons Used in This Book

This book uses the following icons to call your attention to information you may find helpful in particular ways.

The information marked by this icon gives you certain details that are important to remember. This way, you can easily spot noteworthy information when you refer to the book later.

This icon points out extra-helpful information, including ways to save time, money, and headaches.

Paragraphs marked with the Warning icon call attention to common pitfalls that you may encounter.

This icon gives you more information that you don't necessarily need to know to grasp systems monitoring, but which you may find interesting.

Beyond the Book

This book can help set your feet on the path to discover the glory that is systems monitoring. But 48 thin pages is nowhere near the whole story. If you want to continue your journey, check out the following books, webinars, and videos:

- ***4 Essential Skills to Master Your Virtual Universe:*** This e-book is a deep dive into the DART/SOAR framework and how it can help make your monitoring an asset to the business, not just the data center. Head to `www.solarwinds.com/assets/resources/ebook/4-essential-skills-to-master-your-virtual-universe.aspx` for more information.
- **SolarWinds Lab: Monitoring 101:** SolarWinds Lab is a monthly video series that teaches, entertains, and informs you about all things monitoring. In this episode, you do some hands-on investigation of the Monitoring 101 e-book's content, complete with closed-captioning and transcripts. Visit `thwack.solarwinds.com/community/labs_tht/solarwinds-lab/lab-37`.
- **"Don't Hate Your Monitoring":** This resource is a video from one of the sessions from THWACKcamp 2016, SolarWinds' yearly free online conference. Find it at `thwack.solarwinds.com/community/thwack-event-session.jspa?sessionId=1002`.
- **Network Performance Monitor:** Reduce network outages and improve performance with advanced network monitoring software. Download a free, 30-day fully functional trial at `solarwinds.com/network-performance-monitor`.

Chapter 1

Monitoring As a Discipline

In This Chapter

- Exploring monitoring as a job description rather than a task
- Understanding that creating good server, application, and systems monitoring is simple (but not always easy)
- Identifying what systems monitoring is (and isn't)
- Listing the steps to roll out a new monitor or alert
- Discovering sources of inspiration for new monitors and alerts

I want you to imagine for a moment that systems monitoring is your actual job. Imagine that monitoring is its own discipline within IT, rather than it being one more item on someone's to-do list. Imagine that your role is to be the systems monitoring expert. The most important benefit of having a dedicated role is that someone focuses on taking data points from various monitoring utilities and translates them into actionable insights by looking at the data from a holistic vantage point instead of from within narrowly-siloed areas.

Now, I'm not naïve. I realize that having a dedicated individual (let alone a team) of monitoring engineers is currently a reality mostly in larger and more forward-thinking organizations. Nevertheless, small- and medium-sized businesses may want to take note because their environment of servers, applications, and services will only get larger and more complex in the coming years. This complexity presents the need to create such a dedicated role.

Don't believe me? Think about how common hiring a dedicated information security (InfoSec) professional was ten years ago — it was nearly unheard of. Back then, security was simply a checkbox on the to-do list of a network engineer who had a knack for writing Access Control Lists (ACLs), or

a server administrator with a love of logfiles. But today, this position is considered a necessity given the constant threat of security breaches.

Now reflect on how the scale of server and application deployments within the corporate IT landscape has grown, both in size and complexity. In turn, monitoring those systems has equally grown in complexity. In fact, due to hybrid IT (discussed more in Chapter 3), it has become increasingly more challenging to pinpoint the root causes of issues — whether they lie with a cloud-based service, with customer-owned applications running on a cloud provider's systems, or on-premises in the organization's data center.

The "old way" of monitoring systems, where the administrators of the network, server, storage, and so on each monitor in silos, is no longer a viable option. An expert who monitors as a cross-functional discipline can provide a cohesive view across an organization, making root cause analysis more efficient and accurate, and reducing costs in the process.

In this chapter, you discover what monitoring is (and what it isn't), how organizations get distracted from building good (which is to say robust, effective, and efficient) monitoring, how to go about building and testing monitors so they won't wreak havoc in your environment, and where to go for inspiration on which systems monitors and alerts you should build next.

Dance of the Special Snowflakes

In the world of IT, it seems to be a commonly held belief that your environment is somehow special — measurably different from all those other companies. "You don't understand," you tell vendors and colleagues. "We're different." And then you list attributes of your company that apply to 99 percent of the businesses out there. You justify how best practices, common techniques, and standard solutions don't apply to the beautiful snowflake that is your IT architecture.

I see this perception most emphatically when it comes to systems monitoring. I've lost count of the number of organizations that insist they require an in-house, custom-crafted, artisanal solution that more resembles an interpretive dance than a technology.

Many vendors don't help matters. Each spins a story about how it leverages special Application Programming Interfaces (APIs) and context-sensitive commands that somehow only it knows, due undoubtedly to the combination of its great skill, long beards, and certifications from Hogwarts-like institutions.

With 30 years in IT and almost 20 of those focused on the monitoring space — not to mention having used just about every solution on the market since 1998 — I'm here to tell you a little secret I've learned: Systems monitoring is simple.

Creating *good* monitoring, which is robust enough to collect the statistics you need without injecting observer bias, is hard work. But it is simple in its fundamental essence.

What is Systems Monitoring?

I'd like to get something out of the way: Systems monitoring isn't a ticket, or a page, or a screen. Systems monitoring is nothing more than the ongoing, consistent collection of metrics from and about a set of devices. Everything else — reports, alerts, tickets, and automation — is a happy by-product of the first part.

Job One in systems monitoring is to make sure you're collecting all metrics, such as the following:

- Interrupt-based messages, such as Simple Network Monitoring Protocol (SNMP)-trap and syslog
- Ad hoc polling, such as Windows Management Interface (WMI) and SNMP-get
- Protocol-centric options, such as IP Service Level Agreement (IPSLA) and NetFlow
- Simple techniques, such as Internet Control Message Protocol (ICMP, also known as good old "ping")
- Complex solutions, such as Deep Packet Inspection (DPI)
- Vendor-specific techniques that take advantage of a system's API

Job Two in systems monitoring is setting alerts for when things go off the rails. Doing so lets you know when a system is down, gives you a record of when a router configuration was changed, and much more.

The Siren Song of the Black Swan

In business, when an unforeseen and unexpected failure occurs, management often acquires a dark obsession to understand the event and make sure it never happens again. Postmortems are called, task forces are assembled, money is spent.

But if the event was a so-called black swan, a term coined by economist Nassim Nicholas Taleb when he wrote *The Black Swan: The Impact of the Highly Improbable* in 2007 that describes things that can't, by their very nature, be predicted, all that effort is ultimately wasted.

I'm not saying that businesses shouldn't learn lessons from their failures. But it's important to ask whether the failure was predictable in the first case. If it wasn't, then hunting down that black swan is a waste of time and money.

A single big and unpredictable event can distract everyone from the common, every day, and ultimately more expensive and solvable opportunities around you. The upshot is that hunting down and building a system to avoid a black swan event could cost more than the event itself and isn't guaranteed to fix anything because the event is, by definition, unlikely to reoccur.

The Four Phases of Development: Develop, Test, Test, and Test

Hopefully, there are a few examples in this book that get you excited, spark your creativity, or maybe just make you say, "I need to see that for myself." Whatever your reason, diving right in and putting one of them into action in your production environment is never a good idea.

Not ever. I'm serious.

Before you create new monitors and/or alerts, take a moment and hear me out about proper testing.

It's amazing to me that people with long careers in IT still aren't clear on how to create good (which is to say safe, repeatable, common-sense) tests. Therefore, I'm going to share with you a few of the lessons I've learned:

- **Understand the difference between the scope of an alert and the trigger conditions.** "Operating System = Windows" isn't an alert trigger; it's a scoping statement. When testing, ratchet the scope down as small as you can and expand it slowly after each successful test.
- **Identify your test machines first.** Whether that is lab gear set aside for the purpose or a set of production systems that are less critical or a specific "canary in the coal mine" system identified by the team that requested the monitor in the first place, set up your alert so it only triggers for those machines.
- **Learn to use reverse thresholds.** While your ultimate alert will check for "CPU>90%," you probably want to avoid spiking the systems repeatedly. But "CPU<90%" is going to trigger a whole lot more often and reliably (at least I hope so!).
- **Verbose is your friend.** Not at cocktail parties or a movie theater, but in this case, you want to have every possible means of understanding what's happening and when. Insert, "I'm starting the XYZ step now," and "I just completed the XYZ step" liberally. You'll be glad you did.
- **Eat your own dog food.** If you thought you'd test by sending those alerts to the production team, think again. You're going to be getting those alerts yourself until they're just right.
- **Serve the dog food in a very simple bowl.** When you *are* ready to share the results, don't fire them through email. Send them to a local logfile or the display. Worry about formatting after the data has been accepted.
- **Set PHASE-ers to full.** After the monitor or alert is working on your test systems, plan on a phased approach as you roll into production. Widen the net a bit — maybe 10 to 20 systems. Make sure that no unintended side effects exist in production. Then expand out to 50 or so. And so on.

If you follow these guidelines, any new monitor or alert should have a high degree of success, and more importantly, you'll catch bad monitoring and alerting before it does too much damage.

Decisions, Decisions!

So, how do you find that first volunteer — the situation that will provide you with the chance to show off your monitoring chops? I suggest looking for issues that have the biggest bang for the least effort.

A great place to start is to look at your current help desk tickets. Look for monitoring-based alerts that have a high incidence of "mass close" operations (if your ticket system supports that feature) or where large numbers of the same type of ticket are closed at about the same time (within three minutes of each other). These are likely the monitors and alerts that happen too often with no actionable result.

Also, don't plan too far ahead. As apprehensive as you may feel right now, after one or two solid, if small, successes, you will find that teams are seeking you out with suggestions about ways you can help.

Chapter 2

Monitoring 101

In This Chapter

- Putting a frame around monitoring theory with the FCAPS model
- Looking at the various monitoring techniques

In this chapter, you explore the foundational concepts of systems monitoring. There's a difference between commands, scripts, and data used in the course of a typical day of IT work and those performed for monitoring. Sometimes those differences are cosmetic; in other cases, the difference is great — where entirely different commands, protocols, and techniques are at play.

So it's important to define which of those commands, protocols, and techniques fall into the realm of systems monitoring, along with some of the theory and philosophy.

Monitoring Conceptualized: The FCAPS Model

Sometimes it helps to structure a concept into a model or framework, so you can fit all the new concepts into an overarching structure. Luckily, systems monitoring already has one of these, called FCAPS (short for Fault, Capacity, Administration, Performance, and Security). Using the image of an airplane flying from one city to another, the FCAPS model would look something like Table 2-1.

Table 2-1 Flying with FCAPS, an Illustration

FCAPS	*Tells You*	*On the Airplane*
Fault	What's up?	Is the plane in the air, at the gate, or has it crashed?
Capacity	How much?	How many people are on the plane? How many seats are unoccupied? How much fuel is in the tank?
Administration	Who could?	Who purchased tickets for this flight, and, therefore, who's allowed on the plane (regardless of whether they're on right now)?
Performance	How fast?	How fast is the plane going? Is it using the optimal fuel-oxygen mix? How many miles per gallon is the plane getting?
Security	Who did?	Who got on the plane?

While this is certainly a simplification of what can be a very broad subject, it holds true most of the time.

Monitoring is largely concerned with the F, C, and P of FCAPS. Administration (who has access to a system) and Security (who actually accessed the system at any particular time) are usually the purview of the security team and/or your RADIUS/TACACS-type tools.

Systems Monitoring Technologies

This section focuses on the techniques that are suited to systems (meaning server, application, and the support subsystems like storage) monitoring.

If you want to know more about the commands and protocols that are specific to network devices, check out the e-book *Network Monitoring For Dummies,* SolarWinds Special Edition, at `go.solarwinds.com/network-monitoring-for-dummies`.

Windows Management Instrumentation

Windows Management Instrumentation (WMI) is, at its heart, a scripting language built into the Windows operating system. A wide variety of tasks can be performed via WMI, but for the purpose of systems monitoring, the focus is on collecting and reporting information about the target system.

The benefit of WMI is that a remote system with the right privileges can run WMI scripts without having to be "on the box" using some sort of local agent.

Performance monitor (perfmon) counters

Perfmon counters are another Windows-specific monitoring option and can reveal a great deal of information, both about errors on a system and ongoing performance statistics. Perfmon counters can be collected remotely by a machine that has privileges on the server to be monitored. In addition to the hardware and operating system information that's present by default, many applications come with their own perfmon counters as well.

Service and process monitoring

Monitoring services and processes is both as simple as it sounds and more involved. On the one hand, the information can be collected fairly simply, using WMI (on Windows systems) or SNMP (on just about anything). The tricky part is being thorough about what's collected. In its most simplistic form, you find out if a service is running (or not). But for the same price, you can also collect information on the following:

- CPU percentage used by the service or process
- RAM percentage (physical or virtual) used
- I/O operations per second

SQL query monitors

Just as the name suggests, SQL query monitoring involves running a query against a database. The point of this query is to find out information about the database server itself. In this process, you typically query the system tables to find out about the number of client connections, transactions per second, and (most importantly) the impact of locking, blocking, and waits on the system.

Monitoring with scripts

Monitoring with scripts is one of the easiest systems monitoring methods to understand, but it's also possibly the most challenging to execute well. Running a script to collect information can be as simple or complicated as the author chooses to make it. It can also be as efficient or disruptive as the author's skills permit. Therein lies the glory or the downfall.

Synthetic transactions

The core idea behind a synthetic transaction is that it will imitate the actions of a legitimate user, but it does so regularly (even when users wouldn't normally be on the system) and therefore provides consistent insight into the performance of the monitored element.

Synthetic transactions won't always tell you the actual experience of a user because variables such as their specific location can't always be taken into account. Nevertheless, they're a great way to monitor complex, multi-element systems from the perspective of the user rather than the status of individual components.

Examples of synthetic transaction monitors include (but are certainly not limited to) http/https, DNS, DHCP, and FTP.

Logfile monitoring

Logfile monitoring is usually applied to four different and mutually exclusive areas:

- Windows event log
- Syslog
- Logfile aggregation
- Monitoring individual text files on specific servers

Windows Event Log

Event log monitoring is specific to Windows. By default, most messages about system, security, and (standard Windows) applications events are written here. When an event is written, the standard data elements are

- **Message:** Detailed information about the event generated by the application, service, or subsystem
- **Category:** A numeric value that identifies which broad area of computing this message relates to: a disk, printer, service, network element, shell, and so on
- **Keywords:** A series of words used to filter events
- **Event ID:** Classifies the event type
- **Level:** A classification of the event severity (information, warning, error, critical, and so on)
- **Source:** The software that logged the event —can be the name of an application ("SharePoint") or a subcomponent, such as a driver or service, of a larger application

Syslog

Syslog is a standard that describes how to send a message from one machine to another on User Datagram Protocol (UDP) port 514. The messages must fit a predefined structure. This protocol is most often found when monitoring network and *nix (Unix, Linux) devices — although network and security devices send out more than their fair share as well.

Logfile aggregation

Logfile aggregation involves sending (or pulling) log files from multiple machines and collecting them on a central server. This collection is done at regular intervals. The data from each log is "normalized" so it has a consistent structure even if it's coming from various sources.

A second process then searches across all the collected logs, looking for trends or patterns across the enterprise.

Monitoring individual text files on specific servers

This activity focuses on watching a specific (usually plain text) file in a specific directory on a specific machine, looking for a string or pattern to appear. When that pattern is found, an alert is triggered. Now it certainly can get more involved than that, but the goal is the same — to find some kind of text within a file somewhere on the system.

Chapter 3

Monitoring Systems in the Cloud

In This Chapter

- Digging into what hybrid IT means
- Discovering what's almost as good as authority (hint: it's not chocolate)
- Looking at monitoring techniques like tracing and APM, which are well-suited to the cloud

According to the Cisco Global Cloud Index, by the year 2020, about 98 percent of all compute workloads will be processed by a cloud-based architecture. Out of that jaw-dropping percentage, 66.5 percent (or 68 percent of all cloud-based compute workloads) will be in public-cloud spaces, such as Amazon Web Services and Microsoft Azure, leaving 31.5 percent in private-cloud environments. Just these few numbers should explain the current gold-rush atmosphere surrounding investment in and development of cloud-centric tools and solutions.

This has created a techno-economic feedback loop where companies move their data and workloads to the cloud, creating interest and investment, which creates new and better solutions, which generates excitement that causes even more companies to move their data and workloads to the cloud.

This growth has continued for several years now, but one area has lagged conspicuously behind: systems monitoring. Part of the reason for this is the rate of change itself. Every time you think you've nailed down what "cloud-centric systems monitoring" is, the technology shifts, and you have to reevaluate.

But that's no longer the case, and any IT pro with an eye to systems monitoring must consider the cloud — both in terms of systems monitoring that runs *from* the cloud as well as solutions that can monitor systems *in* the cloud.

Hybrid IT: The Technology Nobody Ever Asked For

Despite its value in describing the state of corporate cloud usage, the Cisco Global Cloud Index report doesn't capture the reality for today's boots-on-the-ground IT professionals. For that perspective, you must look at a different set of numbers.

Each year, SolarWinds publishes an IT Trends Report (`it-trends.solarwinds.com`). The 2017 study tracked cloud adoption. It revealed that although 95 percent of those surveyed have moved something to the cloud in the past 12 months, just 1 percent of the respondents are 100-percent cloud-based. The majority have 1 to 25 percent of its architecture in the cloud; the rest remains on-premises. This has come to be known as *hybrid IT*.

You know what nobody ever said? Nobody has ever walked into her boss's office and said anything like this:

"Hey, you know what would be awesome? If we split all of our critical infrastructure so that some of it was here in our office, and some was operated by complete strangers. And I don't just mean remote data centers — I mean those nameless, faceless people would write the software, build the servers, do the updates, *everything*. But our business would totally rely on that software. And not only that, but also all of the connectivity between our office and that business-critical application would be owned and operated by any number of other organizations, and we'd have no idea who they were. But wait a minute boss, you haven't heard the best part! The icing on this techno-cake is that we — you and me — we'd still be responsible for all of it. If something happened on interface 3 of router 17 of our ISP's ISP's ISP, it would still be our responsibility to figure it out and get it fixed. Doesn't that sound wonderful?"

And that, my friends, is hybrid IT.

Hybrid IT is the condition in which part of your infrastructure is in the cloud and part of it remains on-premises, with your team responsible for all of it — including the Internet, which connects the two. It's not only a reality, but also it's the norm among companies, according to the SolarWinds IT Trends Index. Although very few companies have resisted the urge to move something to the cloud, very few companies have moved everything "up there."

The reason I bring this up isn't to depress you but to point out that a "pure cloud" systems monitoring solution isn't going to cut it any more than a purely on-premises systems monitoring solution will. You need tools and techniques that do both and *also* bridge the gap between the two.

The Secret To Success

To be successful as an IT professional, you need three things:

- Responsibility
- Accountability
- Authority

Responsibility? Most of you probably get that before you even figure out where the bathroom is at your first IT job. Heck, just passing within five feet of a server can make you responsible for it in some organizations.

The same goes for *accountability,* which is why you've probably had phone calls like this one: "Hey, Leon, the database on Server 19 is acting funny. Yeah, I know you're not the DBA. Yes, I know you're not the server admin either. But I clearly heard you from two cubes away mention Server 19 last month, so I figured you were doing something on it, and now the database is acting all weird, and . . . look, I know it's 2:00 a.m., and I'm not happy about having to call you either, but could you just take a look at it?"

Authority, on the other hand, is what you usually have to fight tooth and nail to get, and, of course, it's the one thing that's crucial to be successful. In order to be effective at your job

and able to fix anything, you need to have the authority to make decisions, approve purchases, and issue guidelines that are respected and followed.

While all this has been true in IT since the time when Moses had to reboot the two tablet-based systems at Mount Sinai, it's even more tenuous in the modern hybrid IT world. In fact, the SolarWinds IT Trends Index showed that over half of IT professionals consider a lack of control over the performance of cloud-based workloads a top challenge and still a considerable barrier to migration.

But this is IT, so of course there's a hack — a work-around that gives you something almost as good as authority, and that's *visibility*. If you can see the environment and its current state, you have a better chance of getting ahead of any issues that crop up. And if you can see the specifics of those issues — the more details the better — and communicate those details to the true owner, you have a much faster route to resolution, which in turn relieves some of the stress associated with moving a workload to the cloud. This sounds a lot like good, plain old "monitoring". And the fact is that a lot of it *is* good, plain old monitoring. But hybrid IT and the cloud have forced IT pros to reimagine some old techniques with a new spin and invent other techniques out of whole cloth.

Unraveling The Internet

In a hybrid IT environment, the biggest challenge by far, in terms of the authority/visibility conundrum, is the Internet. With every other piece of the puzzle, you have some fundamental insight. On-premises systems are onsite, so these should be a piece of cake. And with the cloud-based systems, you still can lean on the vendor-customer relationship (at least a bit).

With the Internet itself, you need to overcome at least two hurdles:

- **It's massively multipath in nature.** Any given packet can take any one of a number of different routes to the destination, and unlike your network, the routes (or paths) through the Internet can (and do) change at any moment.

- **Your connection to your cloud-based environment may start with your ISP, but it doesn't end there.** Your ISP has an ISP that may have its own ISP. And working from the other end, your cloud provider has an ISP that likely has an ISP as well.

Like many IT challenges, after you can identify it, you're halfway to solving it. The answer lies in finding a system that maps out all those nodes across the Internet. You need something that does it in a way that gives you information about each of the devices (hint: not ping) and isn't blocked by intermediate firewalls (spoiler: not traceroute).

It's Not Really Just Someone Else's Computer

Despite jokes to the contrary, cloud-based infrastructure isn't just a computer running deep within the bowels of Amazon, Microsoft, Google, or some other Johnny-come-lately cloud vendor. Without digging into the complexity of containers, instances, and virtualized workloads, it's easier to simply accept that, while a server running in the cloud looks like a duck and even alerts like a duck, it's *not* actually a duck. It's not even one of those duck decoys you can buy out of a catalog.

A cloud-based "server" is more of a computing cooperative. You're asking for a four-CPU server with 64Gb of RAM, but one CPU comes from a machine over here, two from a machine over there, and the last CPU from this other machine. And the RAM is actually carved out of a set of resources that spans three other machines. Making matters more complicated, at any moment, the physical source of your computing resources can (and does) change. Even if you were to get the key into the cloud provider's data center, you still wouldn't be able to put your hands on "your" machine.

The challenge of is that you, the consumer of cloud computing, need to know if the resource/service you paid for is underutilized, holding steady, or maxed out. You need to know if it's stable, healthy, and that there are no issues. You need to monitor it. If you monitor your cloud servers the same way you monitor your on-premises servers, you'll still receive metrics, but those metrics often won't match reality.

Instead, follow these tips:

- **Use the monitoring and status capabilities that are provided by the cloud vendor.** These will (usually) more closely match the reality of what the hardware is really doing. At the same time, if your in-house monitoring solution can integrate with those cloud metrics and correlate them to traditional monitoring information, all the better.
- **Don't concentrate on the server at all. Monitor the application.** If that's operating well, whatever is happening in the abstracted hardware layer below is (often) unimportant.

Application Performance Monitoring (APM)

As an IT professional, I'm used to words that mean different things to different people. This is why I'm not surprised that application performance monitoring (APM) can mean so many different things depending on the context. But what is most confounding is that these usages aren't mutually exclusive. There is overlap. There are different kinds of APM:

- **Code-centric or "traditional" APM (cAPM):** The focus is on code execution, transactions moving through the message queue, transforms, and so on. This type of APM is often applied to custom-developed code, or applications that are highly transactional in nature.
- **Operations-centric APM (oAPM):** This type of APM is more concerned with what's often called "shrink-wrapped" software, which can be everything from single-purpose business utilities to enterprise-class tools, such as Microsoft Exchange and even foundational things like the operating system itself. The point isn't that they're any less sophisticated than the programs that use code-centric APM, but the needs are different.
- **Web-centric performance management or digital experience monitoring (WPM/DEM):** As the name implies, this APM is focused on monitoring web applications. So it's less about the code execution or the stability of the underlying server application and more about how the user of the web application is experiencing the service.

- **Database-centric APM (dbAPM):** In this iteration, it's all about the things that make your database go bump in the night: long-running queries, locking, blocking, and wait states.
- **Network performance management (NPM):** This area is just as much a member of the xPM family as any of the other APM variations because without solid network performance, everything else falls apart fairly quickly.

With all these types of APM, it's easy to see the overlap. cAPM still cares that the application itself is healthy, and it can provide insight into things such as services and processes, performance counters, and log messages. But that's not the primary focus. Similarly, oAPM has the ability to expose issues with transactions but not to the level that cAPM does. Where it shines, however, is in operational metrics. And the same is true for WPM and dbAPM.

This has all always been true, but it wasn't as clear until recently. The emergence (and convergence) of cloud, DevOps, hybrid IT, and everything-as-a-service (EaaS) has highlighted both the overlap and the differences.

All of this is important to your understanding of how cloud and hybrid IT resources are monitored. Despite claims to the contrary, APM (regardless of what that means in any particular context) wasn't specifically created to monitor cloud applications, nor is it the best monitoring for any and all cloud-based applications. Like all other monitoring techniques, APM has situations where it works well and others where some other tool or protocol is a better fit.

Tracing Your Steps

In a cloud and hybrid IT context, APM often refers to program trace analytics. A tracker is added to every piece of data sent through an application. That tracker is then checked at various steps in the process. From this, it's possible to see how quickly applications process data, where the holdups are, where packets are dropped, and more.

Combine this information with data on hardware resources, application performance, and more and you have a powerful

solution that can provide insight into actual user experience in near real time.

Application tracing has been around for a long time. The wrinkle that cloud and hybrid IT throw into the mix is how spread out the component parts have become. According to SolarWinds IT Trends Index, 69 percent of the people surveyed were at organizations that used up to three different cloud providers simultaneously.

This presents a unique challenge because many tools will perform tracing within one cloud provider — some of those tools are even provided by the cloud provider itself. What you need is a tool that can trace an application's activity across multiple cloud instances and on-premises.

Chapter 4

Understanding How Automation Fits into Systems Monitoring

In This Chapter

- Discussing why automation is part of a solid monitoring strategy
- Looking at the prominent areas of automation

SolarWinds recently reached out to IT professionals to find out what they thought about monitoring and managing their environments. What stood out from this industry-wide conversation was that automation was at the top of everyone's wish list.

Good automation is enabled by, and is a result of, good monitoring. In the end, monitoring and alert automation are limited only by your imagination and ability, assuming you've got a good monitoring tool in place.

Why Automation?

With a perfectly good monitoring system, you may wonder why automation is so important. The answer is that spending your day responding to tickets, alerts, and emails is both tedious and boring, and your time is valuable. Too valuable for this.

When I help implement new automation, there are two questions I make sure to ask: The first is "How do you know when

something has gone wrong?" The second (and the more important of the two when it comes to automation) is "Okay, then what?"

The discussion that comes after "Okay, then what?" is where the magic happens. Talking through a process helps people realize just how repeatable it all is and how it doesn't take decades of programming skill to turn into an automated process.

What's more, after you get into that "then what" mindset, you start to see automation opportunities all around you. "Okay, then what" doesn't just come into play when there's a problem. It's also relevant to receiving a request or running a report. "Okay, then what" even applies to "I arrive in the morning and sit down at my desk."

Sometimes, in the course of this discussion, I encounter some healthy skepticism. I'm asked, "If I automate too well, won't I be out of a job?" It's a fair question. My answer is "Maybe." But if you have no job left after automation, you'll be out of a really tedious, boring job. You're welcome! And in the course of creating all this amazing automation, you'll have proven that you have solid skills that are far more valuable than your ability to repeat the same boring process over and over.

What to Automate

Automation brings solid, measurable value to the monitoring discipline within IT. Automatically responding to issues (a topic I cover in Chapter 5) is a big win, but there are less obvious, yet equally important, aspects of automation. In this section, you look at three of the most prominent areas of automation that aid in systems monitoring: hardware discovery, software discovery, and application monitoring assignments.

Hardware discovery: A very good place to start

One of the first things you do when you fire up a new monitoring system is load devices. While this can be done manually, in any environment larger than about 20 systems, this

becomes an exercise in tedium and frustration. Scanning the environment is far more convenient.

For about a week.

IT environments of all sizes are remarkably fluid, with devices being added, changed, or removed all the time. Manually kicking off a scan of the environment, reviewing the results, weeding out the redundant results, selecting interfaces, and so on get old after the third week. This is often where the itch for automation starts.

Device discovery has three aspects:

- The ways new devices can be discovered
- The way a discovery can be executed automatically
- What to do with devices when they're found

Every systems monitoring solution has its own spin on executing a device discovery. But no matter which solution you have, one set of techniques you should avoid is when the only way to discover devices is by using ping (ICMP) to sweep an entire network subnet (192.168.1.0/24, for example). If no option exists for partial scans or fixed lists of IPs, that should raise some red flags. More red flags should be waving if there's no way to include IP addresses to avoid. And red flags should continue to wave if each device that's found is absolutely attacked by SNMP by using the most aggressive method possible.

Why is all of that bad? First of all, the first time you run discovery on your network you want to be thoughtful about it. If you have no intention of monitoring PCs or IP phones, why would you waste time scanning them? Secondly, that kind of aggressive behavior can often send the target system into a tailspin (or more accurately, a CPU-spin) and cause it to lock up.

There are also some positives. You should look for solutions (or options within your current solution) that let you scan your network in a variety of ways, combining the ability to

- Enter a subnet
- Enter a list of IP addresses

- Enter a seed device where the network is discovered by finding connected devices from that first one
- Specify an Active Directory OU, and scanning computers in that OU

The systems monitoring solution should also provide the ability to specify a set of devices that should *not* be scanned, and that ability should take the exact same input as the discovery itself, either a subnet, list of IPs, or so on. The result is a robust discovery profile that only looks at the parts of the network you and your team want scanned.

After a set of devices is found, the discovery should use the gentlest possible approach, interrogating a few basic pieces of information, such as the device name, vendor, and model. That information should then kick off a discovery of specific elements unique to that vendor/model, rather than a walk of every known numeric combination. In addition, device scans should be able to utilize protocols other than SNMP, including Windows Management Interface (WMI) and vendor Application Programming Interfaces (APIs). Finally, part of the scan should determine connectivity to other devices. This should reveal everything from which switch a server is connected to, to VM-host-cluster-data-center hierarchy, and beyond.

But what about subsequent scans? A good monitoring system should also allow you to take the entire profile and schedule it to run

- Every xx hours/days/weeks
- Every x day of the week (every Monday, every Sunday, and so on)
- Every x day of the month
- At specified times of the day

In addition, there should be controls to end a scan if it runs past a particular time of the day or longer than a set duration. In this way, you can break down large environments into manageable slices, set up robust, sophisticated discoveries, and not overwork both the monitoring solution itself, or the environment being scanned.

In addition to running as a scheduled job, the scan should be able to be triggered by an event. For example, if an interface on a router has been down for more than 30 minutes, start a scan on the subnet the interface was part of, to see if a new interface has been brought up and if any new far-end devices have come online. But regardless of the triggering event, the functionality you want is to set off a controlled discovery based on real-time events on your network.

Automatically discovering a bunch of devices is good. Knowing how they all connect is better because of dependencies. *Dependency* describes a hierarchy where, when one piece of the hierarchy has a problem, everything below it becomes unreachable. Let me offer an example. Imagine a simple scenario where you have one router connected to two switches, which are, in turn, connected to ten servers. (You can see this example in Figure 4-1). If the router goes down, you *should* get only one device down alert for the router, but it's likely that you would get a bunch more alerts.

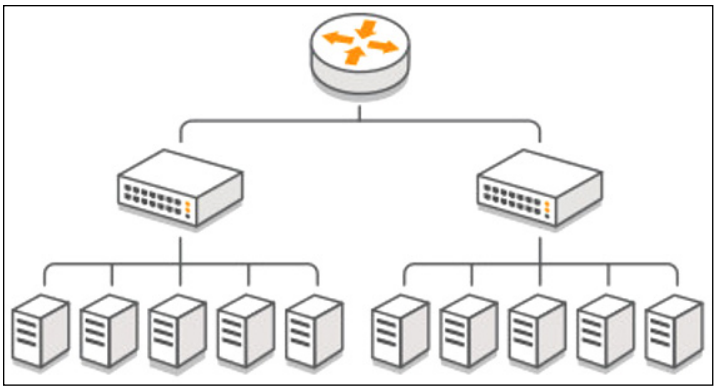

Figure 4-1: A simple example of dependency hierarchy.

The solution to avoid multiple alerts is to set up dependencies within your systems monitoring solution, so when the router goes down, the rest of the downstream devices are put in an "unreachable" state rather than a "down" state (which is what many systems monitoring solutions do if you don't create these dependencies). While the process of setting up dependencies can be done manually, it's much better to have a monitoring tool that goes out and identifies those dependencies for you.

The importance of application discovery

So, you've found every device that was hiding in the nooks and crannies of your network. You've scanned them to find out how the router is connected to the switch. You've also established your virtualization hierarchy, from VM to host to cluster to data center. You've even determined what kind of hardware each device is, how many interfaces and drives they have, and the status of their physical hardware, such as fans, temperature sensors, RAID controllers, and more. You've got it all figured out.

Now if you could only remember which of those little stinkers is running your critical production Exchange email service.

While device discovery is important, it tends to be the simplest aspect of the total systems monitoring story. Applications are their own ball of wax. Figuring out what's installed on a server, what's running, and what's actually being used continues to be a challenge for even the most sophisticated tools.

Despite that, it's critical to the business to have robust and accurate application monitoring in place, an area that demands significant attention. Your business doesn't succeed or fail based on the hardware. It's successful because of the software and the things your employees are able to achieve through that software.

That's why it's so important to monitor the applications that are critical to your business.

If you've read this chapter up to this point, hopefully I've established the fact that application discovery is at least as important as hardware discovery (see the preceding section). How this is done is less of a point of focus in this book than how often you do it and how much manual effort it requires. Because, sure, you can scan a server when it's connected to the network and determine that it's running Exchange or SharePoint, but you'll never know (until it's too late) that three weeks later, the developers enabled Internet Information Services (IIS). Or worse, turned it into a DHCP server. The problem here is that then the server is running

an application that monitoring (and, likely, everyone except the developers in question) knows nothing about. They won't know when it conflicts with something else in the environment (like the official DHCP server) or that it's an element of the business service that needs to be checked in the event of a failure (like IIS).

If anything, you need automation to regularly scan for new, changed, or deleted applications on existing systems even more than you need it to scan for new, changed, or deleted devices on your network. But unfortunately, automatically discovering and monitoring applications (whether on new or existing equipment) is actually more complicated than just performing a scan and then saying, "Oh, it's running IIS now. Good to know." You also need a way to automatically assign the correct application monitor. You get more details on that in the next section.

Monitoring assignments

It's one thing to know that a server is running IIS (or DHCP, SharePoint, or so on). It's another thing to understand what your company thinks about that server. Is it a critical SharePoint box, or just a SharePoint server running in the QA network? Is the application a new build that's being tested, or does it have full production status and is therefore considered business critical?

The same application can be running on the same hardware in two very different situations, which causes the perceived criticality of that application — and therefore which application components need to be monitored — to be very different. This might be due to load balancing, location within the network, business service, whether the server is part of the production or QA environment, and more. These differentiations are essential to knowing whether to apply the "critical order system MS-SQL" monitoring package, or the "low-importance departmental MS-SQL" package. (By the way, I just made up those two monitoring packages. It's all about how you designate the monitors based on your — and your company's — needs.)

Believe it or not, this is the key to understanding and designing good application monitoring assignments. The same

application in two different contexts may need different levels of systems monitoring, so you shouldn't use a one-size-fits-all approach. Instead, allow yourself the flexibility to have "SQL Server Production Core" and also "SQL Server Order Entry Testing." However, knowing that the same application in two different contexts may need different levels of systems monitoring means being vigilant to ensure that you create *only* as many different monitoring profiles as you need and no more. Otherwise you'll be buried under 300 different "SQL Server" monitoring profiles.

Leverage your systems monitoring solution to clearly categorize your systems (Production Windows SQL server). Because one system may occupy multiple roles, that categorization needs to be equally flexible. After the system categorization is in place, the monitoring assignments follow suit. Getting to this level is no small feat, but it's worth the effort when application monitors are automatically added and removed from systems as their roles change throughout the life cycle.

Chapter 5

All About Alerts

In This Chapter

- Exploring the philosophy of alerting
- Effectively triggering alerts
- Informing the alert recipient with meaningful alerts
- Considering automated actions as part of the alert response
- Understanding the DPAR cycle

For many companies, teams, and IT professionals, alerting is seen as the reason for monitoring. If you can't get an alert when something is going wrong (so the thinking goes), why bother monitoring at all? At the same time, alerting is also seen as the curse monitoring brings because it's a source of constant interruptions, false alarms, and "noise." Or at least, that's how it is sometimes perceived.

The reality is that monitoring alerts as a blessing or a curse depend largely on the design and implementation of those alerts, more so than any specific monitoring tool or technique.

In this chapter, I lay out alerting concepts and techniques so you can avoid the noise and create alerts that are efficient, effective, and actionable.

The Philosophy of Alerting

Before digging into the specifics of how to set up alerts, it's important to understand a bit more of the why and what. *Why* are alerts viewed with everything from annoyance to downright loathing in many organizations? *What* are some things IT

professionals can do to create effective systems monitoring alerts? This section helps you answer those questions.

Alerts should not be noise

Oftentimes you hear the accusation that systems monitoring alerts are just noise. One reason for this is the common mistake of turning on all alerts that come out-of-the-box in the monitoring solution. The logic goes that if the vendor thought they were good enough to include by default, they must be good for most (or all) companies. Nothing could be further from the truth.

The alerts you find built in to monitoring software are often included as examples of specific techniques or types of alerts. For example, if the systems monitoring solution can send a message when Linux CPU is over a certain threshold and include a list of the top ten processes running at that time, you'll likely find a sample alert that does just that — not because CPU over 90 percent is a particular best practice, but because it's easily understandable for someone who wanted to see that technique in action.

Treat the out-of-the-box alerts as polite suggestions because often they're included by the vendor simply to show you what could be done with alerting, and how to set it up.

Alerts are considered noise in organizations for many reasons:

- **Too many:** Alerts trigger too often, so feel like you've barely responded to one when another shows up.
- **Not enough detail:** The alert message says something like "system down" without any other details — not helpful.
- **False alarms:** Of course, when the alert turns out to be just plain wrong, nobody is happy.
- **Too sensitive:** The alert indicated something happened, but wasn't bad enough to require someone to respond.

The good news is that these issues are all extremely solvable with a bit of planning and forethought.

Alerts should be actionable

In a relay race, only one person carries the baton at a time. But that doesn't mean that the only person who cares about the baton is the one carrying it. In fact, many people care about both the baton and the race itself — everyone from the coach to the other teammates to the spectators in the stands. But even if thousands of people care about the baton's progress around the track, the fact is that there is still only one person carrying it at any given moment.

In the race to mean time to repair (MTTR) that IT professionals run every time there's an issue in their environment, systems monitoring alerts are the baton. Only one recipient gets the alert.

You can take this "relay race" metaphor too far. When I say that the alert goes to just one recipient, I don't mean that it has to go to a single individual. The recipient can be a team. My point is that the alert goes to the person or people who have responsibility for fixing the issue, and not to people who are interested or supportive but not involved in fixing the situation. That means there is no such thing as an "FYI" alert. If you get the alert, you're responsible for fixing the issue. If you aren't responsible for fixing the alert, then you shouldn't be getting the alert.

Alerts are not just emails

Many IT professionals and monitoring teams are under the mistaken impression that the only output of an alert is an email. There are many actions that an alert can perform once triggered — most of which I talk about in the later section "Automated Alert Actions."

For the moment, I want to focus on the messaging that comes from alerts. Of course it can be an email. But it's just as valid for an alert notification to take the form of a ticket within the corporate help desk system. It's also possible for an alert notification to go to a messaging system such as Slack, Skype for business, or a self-hosted messaging platform. It's also possible that the alert goes to a logfile, which can be sent in real time to a variety of displays.

Think about the output of your alerts in ways that go beyond the simplistic or out-of-the-box behavior. Match your messaging to the need you have of how that information will be consumed.

Alerts should not be noise. Alerts should be actionable. Alerts are not just email. Those ideas are important, but they certainly don't tell you *how* to create good monitoring. Hop to the next section to dig into the details.

Getting Trigger Happy

The first stop in your quest to create alerts that are meaningful, effective, and actionable is the alert trigger itself. If many alerts are guilty of triggering too often or too soon (and they are), then you should look at why that is and what you can do about it.

Setting a trigger delay

Let's take one of the simplest alerts — the old "Tell me when the system is down" (when it stops responding to ping). Should you send an alert when a single ping is missed? When five are missed? How about 20?

The reality is that in any environment pings momentarily fail all the time. So triggering after a single failed test would create an uncomfortable amount of alerts that ultimately are proven to be false alarms. Inserting a delay allows the systems monitoring solution to determine whether the device is truly down or just momentarily busy and unable to respond to that one ping.

This concept of a delay becomes even more important when you expand beyond simplistic "is it down" alert types and into examples where you want to know when an application has high CPU at the same time there's a high-user connection count and long-running queries.

But there's a gotcha that you need to keep in mind. All monitoring systems have three aspects that you must fully understand before you start injecting alert trigger delays into the mix. They are

- **The polling cycle:** The polling cycle is the frequency with which the systems monitoring solution goes out and gathers data from the target devices. It could be a blanket 5 minutes; or it could be 2 minutes for up-down data, 5 minutes for hardware metrics, 7 minutes for network statistics, and 15 minutes for disk information.
- **The alert trigger query cycle:** The alert trigger query cycle refers to the concept that an alert trigger is simply a query that is run against the systems monitoring database, checking (querying) for various conditions. When the trigger returns data, that means there's an alert condition. This alert trigger query is run at some interval, whether that's every 60 seconds or every 5 minutes.
- **The alert trigger delay you're introducing:** The alert trigger delay is the delay this section is asking you to consider including.

So when you introduce an alert trigger delay, make sure you understand the polling cycle and alert trigger delay and account for it so that your delay is really a measurement of how many polling cycles you want to delay, not just how many minutes you want to wait.

Single element triggers

IT environments are a complex web of interdependent connections between networks, servers, services, and data. Systems monitoring alert triggers are, of necessity, going to reflect that complexity. Take a "simple" alert as an example: disk full. At first blush, you'd think that it's just a matter of setting a threshold to trigger when "percent disk utilization" is over some number (for the sake of argument, let's say 90 percent). But what about those spiffy 3 terabyte (TB) drives you can pick up on Amazon for under $100? Your simple 90 percent threshold will trigger when the drive is down to a measly 300 gigabytes (GB). Three. Hundred. Giga. Bytes. I don't know about you, but if someone woke me up at 3:00 a.m. because there was "only" enough space to store 7,500 CD-quality song files, I'd find a way to tie that person down and make them listen to all that music continuously.

Instead, it's perhaps better to add a second element that looks at the actual space. So the trigger becomes this:

WHEN "percent disk space utilization" > 90%

AND "Disk space remaining" < 20 GB

Another situation I encounter is when I ask, "When you get that alert, what do you do?" The answer is "Well, nothing. At least not after the first one. But if I get three of them in 15 minutes, I know it's a problem, and I jump on it." I don't get paid by the bushel for sending out alerts.

Trigger delays are all about letting you know when a condition has persisted for a certain amount of time. But there's also the ability to alert when a problem has occurred a certain number of distinct times. This is most evident when you're monitoring logfiles. For example, if you want to alert when the message "Danger, Will Robinson!" appears three times within ten minutes. But it's just as likely in other situations.

In addition to "when it happens X times in a row," some systems monitoring solutions trigger when a condition occurs X times out of Y polling cycles. So maybe you want to alert when the number of customer connections spikes over 100, but only if that happens three out of five polling cycles. So of course you'd get an alert if customer connections were 100+ on polling cycle 1, 2, and 3. But you would also get an alert if the value was high on polling cycles 1, 4, and 5.

Learn to look past the simple and simplistic alert triggers (often these are the ones that come out of the box from the vendor) and think about the conditions that truly indicate an actionable problem.

Hitting reset

Another aspect of alert triggers that frequently gets overlooked is actually the anti-trigger: the reset threshold. The idea of a reset is that it's a test, which, when met, tells the systems monitoring solution to consider the problem to have reversed.

In most cases, IT professionals set this to "when the trigger is no longer true" and call it a day. Trigger when CPU is over 90 percent? Well, if it's 89 percent we're obviously good. Close that ticket and grab a beer! While this logic is true in some cases, I have found that more often it's a wasted opportunity to help make alerts less noisy.

To use the example of a disk alert from the previous section, "Single element triggers," I show you how this might work. An alert would trigger when "percent disk utilization" was over 90 percent *and* "disk space remaining" was under 20GB.

Using a simple reset, if the disk utilization goes to 89 percent *or* the disk space jumps to 21GB, the alert will clear. That's not particularly reassuring.

Instead, you can set the logic to reset when "percent disk utilization" is under 80 percent *or* "disk space remaining" is over 50GB. On top of that, you should also add a time delay. That way, you don't end up resetting the alert until it is truly all clear.

Making Meaningful Messages

Even if you have insightful trigger logic that only creates alert notifications when there's a verifiable and actionable issue, all your hard work can be foiled by a confusing or terse message that causes the recipient to waste valuable time trying to figure out what actually happened, or worse, simply ignores the alert all together.

The good news is that solving this is very simple: Add more information to your messages.

Where many IT pros get stuck is understanding exactly what information is useful and what creates confusion or obscures the issue. And that's where I can help. Here is your official checklist of the data elements that I believe must be included in every message:

- **The "identification" of the device:** Includes the local name, the DNS name, the IP address, and more
- **Information about the operating system:** OS name, version, and so on

- **Other information to identify the device:** Includes location, group, owner, and so on
- **The time the alert occurred:** Which may be different than the time the notification is sent out
- **The current value of the problem elements:** Including the values at the time of the problem *and* at the time the alert is sent out
- **The threshold values which were breached:** So that the responder understands how far over or under the current state is
- **The duration of the problem:** The time of first detection versus the time the alert notification went out
- **A live link:** Allows the recipient to click to see the current state of the devices, elements, and/or items that have a problem
- **Information about the alert itself:** The name of the alert that was triggered, the polling engine or machine that collected the data, and so on

While it means more work during alert setup, having an alert with this kind of messaging means that the recipients have several answers to "Why did I get this alert?" at their fingertips.

Automated Alert Actions

Back in the Chapter 4, "Why Automation", I said that I had two questions I loved asking when it came to helping someone set up a new systems monitor or alert. The first was: "How do you know when something has gone wrong?" The second (and the more important of the two when it comes to automation) is "Okay, then what?"

Maybe after the IT person gets an alert, she clears a queue, or restarts a service, or deletes all the files in a temp directory. Whatever the action is, it's very likely going to be something that could've been performed without human intervention.

That's a quick win for automation — as long as it solves the problem. Not just once. Not just some of the time. Always. Automated alert responses must always solve the problem, or else you have another problem to solve.

As anyone who has been working in IT for more than 15 minutes should know, while many problems are tediously repetitive, sometimes you get the one weird case that resists all your usual tricks. This is why sophisticated monitoring solutions allow you to build an alert that triggers an initial action, then waits a specified amount of time. If the condition persists, a second (or third, or fourth, or whatever) action will be triggered.

Therefore, a disk-full alert could first clear the temp directory, wait ten minutes, then remove specific application log files more than one month old, wait another ten minutes, create a ticket for the server team, wait one hour, and, if the problem continues to persist, page the team lead as an escalation point.

But let's say that there isn't a definitive action that can be taken. Maybe the answer is to check the last 15 lines of this log file, look at this other counter, and run a test query from the application server to the database. Based on the results of that information, the technician will know what to do next.

In that case, your automated action is to do all those steps and then insert those results into the alert message.

Then, instead of a message that says "Service XYZ is down," the technician receives a ticket that already contains greater insight into the conditions at the time of the failure, as opposed to 15 minutes later, when she's dragged herself out of bed, fired up the laptop, and started to dig into the situation. By doing so, the monitoring system has effectively given staff 20 minutes of its life back. It has simplified the troubleshooting process.

And the best part of all this is that even if the information you pull isn't 100 percent necessary, the reality is that the information is useful *most* of the time. Gathering that information proves that the monitoring system can be leveraged as an always-on, Level One diagnostician.

Doing this kind of thing will make heroes out of you and the monitoring system.

The Elephant in the Room

As exciting as they are, I would suggest that, before you dig automated alert responses, you take a moment to understand a critical concept. In some circles, it's called the detection-prevention-analysis-response (or DPAR) cycle. While any conversation about monitoring will focus on detection, (or monitoring), and response, (automated alert actions), it would be negligent for you to ignore prevention and analysis.

Alerts are way of catching errors when they occur (or at least, before they do too much damage). But then it's up to us as IT professionals to determine why they occurred and find a way to keep them from occurring again. When we make the effort to create an alert, we should also commit to doing the hard work of analyzing the situation going forward, looking for patterns and root causes, to help ensure the issue is prevented in the future.

Automatic responses to alerts keep your business running and help ensure you get the beauty sleep you need, but in the morning, you and your team need to be able to see that something happened and do the hard work of figuring out why it happened, so you can prevent it from happening in the future.

Chapter 6

Ten Tasks to Consider in Systems Monitoring

In This Chapter

- Preparing for the five questions of monitoring
- Setting logical trigger and reset conditions
- Creating meaningful messages
- Adding actionable automation
- Ensuring the new monitor works in the real world

I'm often asked (with varying degrees of exasperation, urgency, or enthusiasm), "What is it going to take to get systems monitoring up and running?" This might refer to a single monitor or to an entire solution. While everyone's environment has its own peculiarities, I can describe a set of tasks that you'll more likely than not have to consider.

There are, in fact, ten of them — hence their appropriateness to the traditional Part of Tens, which appears in all *For Dummies* books.

Be Ready for the Five Questions of Monitoring

Before you accept your first monitoring request, you need to understand a universal truth about being "the monitoring expert." Inevitably, you'll find yourself answering the five questions of monitoring. These questions are ones that people never really had to ask when they were monitoring

in silos, often under the radar and on the side. But now that you're in charge of all monitoring, people are going to ask them constantly. They are

- **Why did I get an alert?** The person isn't asking, "Why did this alert trigger at this time?" He's asking why he got the alert at all. This tells you your alert messages don't have enough information. Go back and add more.
- **Why didn't I get an alert?** Something happened that the owner of the system felt should have triggered an alert, but he didn't receive one. This is your clue that you didn't ask enough questions when you set up the monitor and alert in the first place, and now it's triggering at an unexpected time.
- **What's being monitored on my system?** This person wants reports and data she can pull for her system so she can look at trending, performance, and forensic information after a failure. This tells you that you need to beef up your reporting to show this information.
- **What will alert on my system?** The person asking this question wants to be able to predict the conditions when he'll get an alert for this system. How you address this depends on the capabilities of the systems monitoring solution you have. Some can output monitors, alerts, thresholds, and assignments with ease; others not so much. If you aren't lucky enough to have a tool that does, you'll need to spend a little extra time documenting as you put monitors and alerts in place so you have this information at your fingertips when needed.
- **What do you monitor "standard?"** Asking what metrics and data are typically collected for systems is the inevitable (and logical) response when you say, "We put standard monitoring in place." You can't make a statement like that without being able to back it up with a handy list.

Fully Describe the Trigger Conditions

When someone requests a monitor, make sure she's able to give you all the details. She should not only be able to identify the system(s) to be monitored, the sub-elements, and so

on but also what "normal" operation for the monitored item should be, what the threshold for a warning and critical state might be, and so on.

However, the truth is that many folks don't have a clear sense of what the operating boundaries are, which is why I also recommend setting up a monitor in a test environment for a period of time to collect that data. Then you and the requestor review that data as a baseline and fill in the blanks.

The same level of granularity is needed for alerts. Make sure that you discuss exactly which factors contribute to a problem condition — which elements, the duration, the count of events, and so on.

Fully Describe the Reset Conditions

As with the trigger conditions (see the preceding section), take the time to understand what "all better" looks like. Often, it's more than simply when the problem goes away. Reducing monitoring and alerting noise means resetting an alert only when the problem has truly cleared up. Otherwise, you'll end up with *sawtoothing* — alerts that repeatedly trigger and reset.

To that end, your reset logic may have a longer delay than the trigger as well as a different threshold.

Alert Messages Should Be a Tell-All Story

After your systems monitoring solution has gathered the data and determined there's a problem, it's no time to play coy. Giving all the information available could save valuable time for the person responding to the alert. That also means thinking creatively about what information you may regularly collect but what the systems monitoring tools could obtain at the time the problem is detected and include in the alert message. An example would be pulling a list of the top ten processes (sorted by percent CPU utilization). You don't need

that information all the time; you just need it when the CPU alert triggers.

Add Automation

If the response to an issue is repeatable (and most are), it can be automated. The value of automated responses can't be understated. It allows for immediate reaction to a problem condition, often clearing up an issue before a human has time to react, and at every moment of the day.

The best automation is where the problem is completely resolved. But don't discount having an automation script that gets the technician halfway "home" — it's still time back in your pocket and builds faith and reliance in the monitoring tool.

Have a Knowledge Base

Whether you're setting up a new systems monitor or an alert (or both), one of the things I insist on is a knowledge article of some kind to go with it. Why, you ask? Because while the monitoring solution is turned on 24/7/365, the humans need a break. Just because Claude on first shift knows what to do when this alert comes in doesn't mean Gertrude on third shift will. Asking (nay, demanding!) that Claude document his process ensures that whoever is on call that day will know what to do.

Make It Happen On Purpose

Imagine you do everything you're supposed to, and the alert never triggers. Then, a few weeks later, you get an angry call from the manager of the team, asking why your crappy monitoring isn't doing its job. You investigate, only to find that the error message in the original request has never been seen because the error message you were given doesn't exist.

IT professionals with the best of intentions still can get it wrong. Maybe the message in the documentation has changed after a recent patch. Maybe the request contained a spelling

mistake or the words were in a different order. Whatever the cause, this is the reason I also ask the requestor to make the problem happen on purpose. If he can't, then the monitoring is highly suspect in the first place.

Some alerts can't be replicated. If you're checking to see if the data center is on fire, you certainly don't want to test your new monitor with a can of gasoline and a match. However, you *can* test the alert by temporarily changing the parameters to "*not* on fire."

Find One in the Wild

Even after testing the new monitor or alert by causing it to trigger on purpose, I still consider that new addition to be in the "testing" phase until I've seen an occurrence "in the wild." Very often, IT professionals develop a case of techno-paranoia and ask to monitor errors that simply never come to pass. Marking a monitor as "testing" until seeing one actually occur in production allows you to evaluate monitors regularly and de-commission those that have never happened. This process frees resources for the events that are provably occurring (and therefore have a higher return on investment).

Describe the Cost of (not) Monitoring

Knowing (and being able to articulate) the value that systems monitoring brings to the table can make the difference in whether you have the respect of the business or not. If you can't explain, with data, the business benefit that systems monitoring provides, then the business doesn't have a reason to keep putting time and resources into systems monitoring.

Start with a baseline of the environment pre-monitoring. How often does a particular problem occur? How long does it take staff to discover the problem? How long does it take to resolve? What is the business impact to this problem? This information can often be provided by the person/people requesting the new monitor in the first place.

What you ultimately want is a dollar value for the event: something that says "Every time X happened, we spent $123 in staff time to fix it and lost $456 in sales." Then observe the differences after the new systems monitor or alert (or both) is rolled out. Find the differential, expressed as dollars, saved.

Now watch the number of times the alert triggers. You can now confidently report that systems monitoring represents a value to the business of $XX (the savings for that monitoring or alert times the number of occurrences) for that one event alone. Do that for all (or even most) of your monitors and alerts, and you'll soon have the executive team begging you to find new monitors to roll into production

GOTO 1

At this point in the process, you may think the final task is to crack open a frosty beverage and bask in the fame and glory that systems monitoring is raining down on you. While that is true, there's one more thing you should do: Set a calendar reminder to check back with the requestor in three to six months. The happy and sad truth is that a monitoring engineer's work is never done. Not only will you get more requests based on early successes, but also you need to go back and touch base with the folks you've created a systems monitor for in the past to make sure it's still performing as expected and desired.

Applications change; patches introduce new features, fix bugs, and even introduce new peculiarities. The person who requested the alert may not realize, until you reminder her about it, that the monitor you created for her at the beginning of the year has been strangely silent since the last patch and should be revisited. Whatever the reason, make it part of your process to go back, check in, and offer to make improvements if possible.

Continue your systems monitoring journey and check out these additional resources:

Network Monitoring For Dummies

This book covers the foundational concepts of network monitoring, from why you need it to how it's actually done (not to mention how to do it RIGHT!).

Monitoring 101

If you're familiar with computers and IT but not with monitoring, this guide is for you. *Monitoring 101: A primer to the philosophy, theory, and fundamental concepts involved in systems monitoring* is your 100-percent, vendor-agnostic, quick-start guide to understanding basic monitoring terms, techniques, and technologies.

Monitoring 201

If you're comfortable with the basics of monitoring and are ready to up your game, *Monitoring 201: Moving beyond simplistic monitors and alerts to #MonitoringGlory* is a great place to start. This guide focuses on making monitors and alerts efficient, effective, meaningful, and actionable.

Server & Application Monitor

Server & Application offers server performance, capacity, and hardware health monitoring software that allows you to monitor your applications from a single dashboard, with more than 200 pre-made templates available out of the box. It features automatic application discovery and dependency mapping. Download a free, 30-day, fully functional trial now.

Monitoring is a discipline

See how you can become a disciple. Systems monitoring has been around for decades, but some still treat it as an arcane art. But the truth is that monitoring is simple (even if it's not always easy). This book introduces the fundamental concepts, tools, and techniques so you can steer clear of common mistakes. Along the way, you also discover how to take the noise out of alerts, shave hours off incidents, and create monitoring that gets you noticed (in a good way) by the leaders in the data center and the business.

- ***Monitor as a discipline* — see the benefits of monitoring and the difference between monitoring and managing**
- ***Start with Monitoring 101* — learn the foundational frameworks, techniques, terminology, and tools**
- ***Get tips for cloud and hybrid IT* — discover ways to monitor your servers, applications, and infrastructure as they migrate to the cloud**
- ***Discover best practices* — for monitoring servers, applications, and infrastructure as well as tips on creating meaningful alerts and automation**

Open the book and find:

- **Why you should monitor**
- **Monitoring frameworks and technologies**
- **How to automate and amaze your coworkers**
- **How to take the noise out of alerts**
- **Best practices of systems monitoring**
- **Ways to continue your growth as a monitoring specialist**

Go to Dummies.com®
for videos, step-by-step examples, how-to articles, or to shop!

FOR DUMMIES®
A Wiley Brand

ISBN: 978-1-119-46049-7
Not for resale

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.