MONITORING RESET FOR CONTAINERS

by **LAWRENCE HECHT**

Monitoring is not a new concept, but a lot has changed about the systems that need monitoring and which teams are responsible for it. In the past, monitoring used to be as simple as checking if a computer was still running. Cobe Chief Technical Officer (CTO) Dave Charles remembers monitoring as simple instrumentation that came alongside a product.

As James Turnbull explains in The Art of Monitoring, most small organizations didn't have automated monitoring — they instead focused on minimizing downtime and managing physical assets. At companies that actually had IT staff, operations teams used simple tools to check on disk, central processing unit (CPU) and memory usage, but focused mostly on dealing with emergencies related to availability. Larger organizations eventually replaced the manual approach with automated monitoring systems that utilized dashboards.

Even without the introduction of containers, recent thought leaders have advocated that monitoring should more proactively look at ways to improve performance. To get a better view of the monitoring environment,

we reviewed a survey James Turnbull conducted in 2015. Although it is a snapshot of people that are already inclined to care about monitoring, it provides many relevant insights.

Expectations of time and effort needed for monitoring is changing. With the current prevalence of automated systems, users often want to reduce the time needed to setup a monitoring tool and trying to find the problem in their stack. While monitoring may always been relatively time consuming, there are approaches that can improve the overall experience.

What's Different With Containers

To understand how to monitor containers and their related infrastructure, you need to understand what is different about containers. There are aspects of containerized environments that change previously established monitoring practices and the efficiency of traditional monitoring solutions. Understanding these changes will help explain how vendors are shifting to create new products to address changing metrics and a new, varied team of users involved in monitoring. The monitoring changes that come with containers can be explained in five points:

1. The ephemeral nature of containers.
2. The proliferation of objects, services and metrics to track.
3. Services are the new focal point of monitoring.
4. A more diverse group of monitoring end-users.
5. New mindsets are resulting in new methods.

Ephemerality and Scale of Containers

Cloud-native architectures have risen to present new challenges. The temporary nature of containers and virtual machine instances presents

tracking challenges. As containers operate together to provide microservices, they are in effect a distributed system. While distributed systems are not necessarily transitory at larger scales, they require targeting of many moving parts. This requires new methods of monitoring to make observations about their health. Due to their ephemeral nature and growing scale, it doesn't make sense to track the the health of individual containers; instead, you should track clusters of containers and services.

Traditional approaches to monitoring are based on introducing data collectors, agents or remote access hooks into the systems for monitoring. They do not scale out for containers due to the additional complexity they introduce to the thin, application-centric encapsulation of containers. Neither can they catch up to the provisioning and dynamic scaling speed of containers.

In the past, people would look at a server to make sure it was running. They would look at its CPU utilization and allocated memory, and track network bottlenecks with I/O operations. The IT operator would be able to know where the machine was, and easily be able to do one of two things. First, they could point an instrument to that specific location and collect data. In monitoring language, this is called polling a machine. Alternatively, an agent can be installed on the server, which then pushes data to a monitoring tool.

This push approach has achieved popularity because the ephemeral nature of containers and virtual instances makes it difficult to instrument tools to find and poll them. It also reduces the amount of intrusion or tainting of applications. This monitoring approach benefits from the key observability characteristics of containers, and enables solutions that operate efficiently, seamlessly and without intrusion to container execution.

Proliferation of Objects, Services and Metrics

The explosion of data being generated is a well known phenomenon. Ten years ago, people cared about how to store all that data. More recently, the focus has been on how to best utilize that data without storing it all. With the rise of Internet of Things (IoT) sensors and container adoption, there are now more and more objects than ever to monitor. While there is an instinct to try to corral all these objects into a monitoring system, others are attempting to identify new units of measurement that can be more actionable and easily tracked.

The abundance of data points, metrics and objects that need to be tracked is a serious problem. Streaming data presents many opportunities for real-time analytics, but it still has to be processed and stored. There are technical solutions that can handle the scale, but at significant cost to both finance and performance. While NoSQL and other next-generation databases have established their place in the IT ecosystem, they are not optimized for this use case; time series databases is a potential solution for storage. However, companies can't just store their log data indefinitely; much of the data is never used. Some older log files are never looked at, motivating users to focus less on log management tools and more on metrics, which is data collected in aggregate or at regular intervals.

Per Host Metrics Explosion			
Component	**# of Metrics for a Traditional Stack**	**for 10 Container Cluster with 1 Underlying Host**	**for 100 Container Cluster with 2 Underlying Hosts**
Operating System	100	100	200
Orchestrator	n/a	50	50
Container	n/a	500 (50 per container)	5,000 (50 per container)
Application	50	500 (50 per container)	5,000 (50 per container)
Total # of Metrics	**150**	**1,150**	**10,250**

TABLE 2: *Containers means more metrics than traditional stacks.*

Containers present two problems in terms of data proliferation. Compared to traditional stacks, there are more containers per host to monitor and the number of metrics per host has increased. As CoScale CEO Stijn Polfliet describes it, there would traditionally be 150 metrics to track per host: 100 about the operating system and 50 about an application. With containers, you're adding an additional 50 metrics per container and 50 metrics per orchestrator on the host. Considering a scenario where there a cluster is running 100 containers on top of two underlying hosts, there would be over 10,000 metrics to track (*Table 2*).

With so much potential data to collect, users focus on metrics. As Honeycomb co-founder and engineer Charity Majors wrote, "Metrics are usually bucketed by rollups over intervals, which sacrifices precious detail about individual events in exchange for cheap storage. Most companies are drowning in metrics, most of which never get looked at again. You cannot track down complex intersectional root causes without context, and metrics lack context." Even though metrics solve many operations problems, there's still too many of them, and they're only useful if they're actually utilized.

Services Are the New Focal Point

With a renewed focus on what actually needs to be monitored, there are three areas of focus: the health of container clusters; microservices; and applications.

Assessing clusters of containers — rather than single containers — is a better way for infrastructure managers to understand the impact services will have. While it's true that application managers can kill and restart individual containers, they are more interested in understanding which clusters are healthy. Having this information means they can deploy the cluster to a different infrastructure or add additional resources to support its optimal operation. Container orchestration solutions help by allowing

for efficient scheduling of containers on clusters of hosts.

Many microservices are composed of multiple containers. A common example is a microservice composed of five different containers, each running a different process. If one goes down, another can pop up in its place. However, if this failure is a consistent pattern in the long-term, there will be a degradation of the service. Looking at the microservice as a unit can provide insight into how an entire application is running.

According to CA Technologies SVP Product Management Sushil Kumar, in an interview with The New Stack on modern application monitoring considerations, "cross-functional DevOps teams and Site Reliability Engineers need insight into the services running within and across containers. An aggregated view of performance across microservices, apps and containers is key to ensuring a flawless customer experience.

Critical to this is massively scalable metric capture, including API communication and latency, traffic, errors and utilization as they relate to specific applications; analytics to remove noise and correlate information across the highly dynamic container and application fabric; and visually tracking services and the dependencies between containerized microservices."

More Diverse Group of Monitoring End-Users

The focus on monitoring applications instead of just infrastructure is happening for two reasons. First, a new group of people is involved in the monitoring. Second, applications are more relevant to overall business performance.

Monitoring is still generally reactive, despite progress in recent years. It's focused on the objectives of the IT team managing the actual infrastructure. This mindset does a disservice to developers because they generally receive data secondhand. Developers are increasingly being held

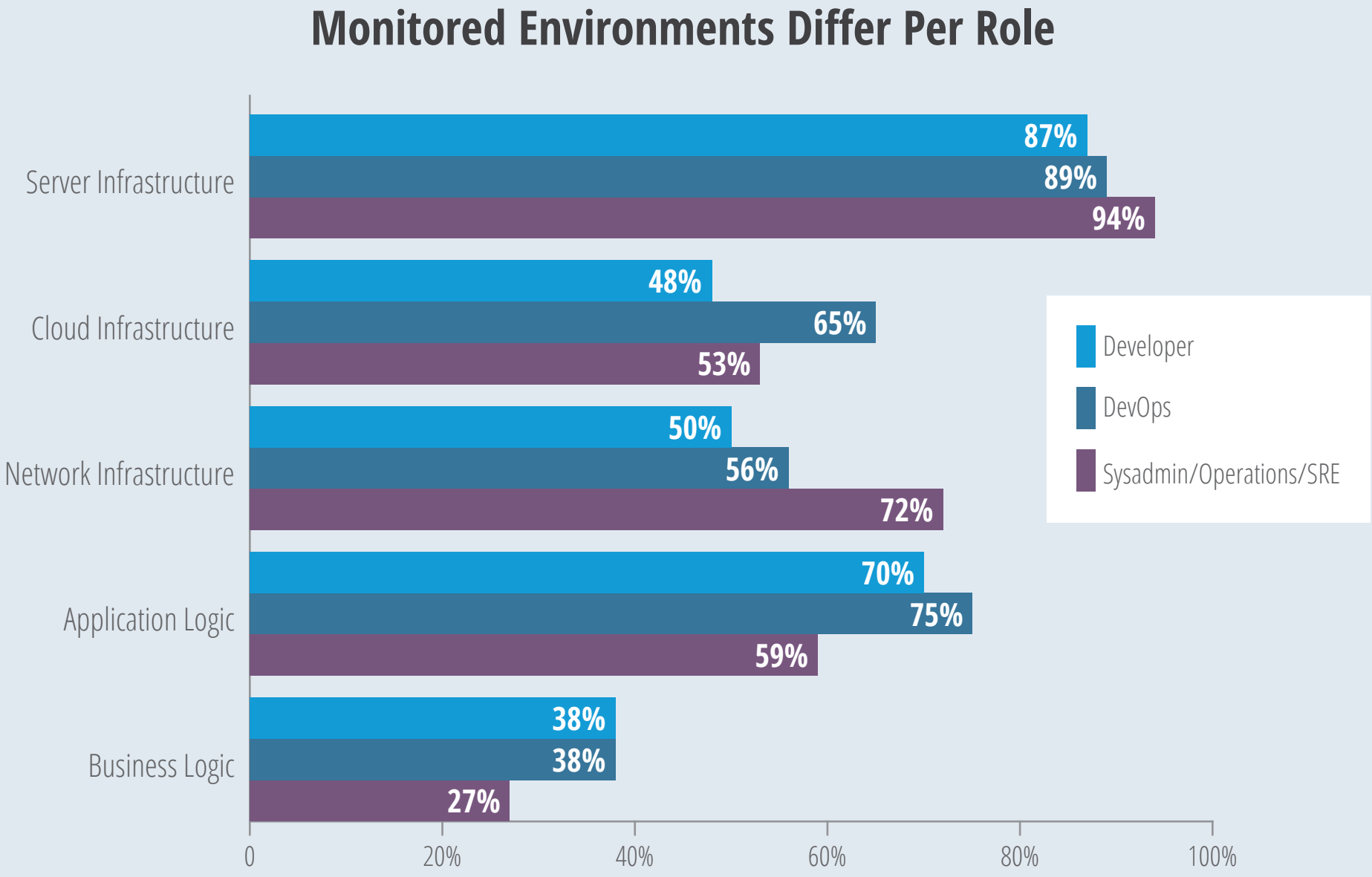
accountable for applications once they have been put into production. As Todd DeCapua and Shane Evans's Effective Performance Engineering notes, developers are being asked to "deliver the highest quality and performing product, and provide continuous feedback and optimization recommendations, so other teams can deliver quickly and in fully automated ways."

The DevOps movement has risen, at least in part, as a response to developers' desire for increased visibility throughout the full application life cycle. Now, DevOps roles are often the full stack managers and operators of applications.

Different roles care about different parts of the monitoring process. Our analysis of the aforementioned Turnbull survey of IT professionals that care about monitoring shows that beyond servers, their areas of interest vary significantly. The data shows a break between the developer and DevOps roles. Based on the survey, 48 percent of developers monitor cloud infrastructure, which is significantly below the 65 percent reported by DevOps roles.

The biggest differences are between DevOps and other IT staff. The data showed that 72 percent of system admins and IT Ops roles monitor networking infrastructure, which is about 20 percentage points higher than the developers and DevOps groups. On the reverse side, 70 percent of developers and 75 percent of DevOps roles monitor application logic, compared to only 59 percent of the IT operations-oriented respondents.

DevOps roles care as much about applications as they do infrastructure, but they care more about performance than availability. As James Turnbull writes in The Art of Monitoring:



Source: The New Stack Analysis of a 2015 James Turnbull survey. Which of the following best describes your IT job role? What parts of your environment do you monitor? Please select all the apply. Developers, n=94; DevOps, n=278; Sysadmin/Operations/SRE, n=419.

THENEWSTACK

FIG 1: *DevOps care more about monitoring cloud infrastructure (65 percent) and application logic (75 percent) as compared to their IT operations-focused peers.*

> *"Orienting your focus toward availability, rather than quality and service, treats IT assets as pure capital and operational expenditure. They aren't assets that deliver value, they are just assets that need to be managed. Organizations that view IT as a cost center tend to be happy to limit or cut budgets, outsource services, and not invest in new programs because they only see cost and not value."*

Luckily, we've seen a trend over the last few years where IT is less of a cost center and more of a revenue center. Increased focus on performance pertains to both IT and the business itself. Regarding IT, utilization of storage or CPU resources is relevant because of their associated costs. From the perspective of the business itself, IT used to only care about availability and mean time to resolve (MTTR). While

availability and resolvability are still critical, new customer-facing metrics are also important.

Along with DevOps, the practice of site reliability engineering (SRE) will affect how monitoring tools are used. From this perspective, monitoring will still largely be managed by an operations team, but responsibility for ensuring new applications and services are monitored may be delegated to application developers. Shariq Rizvi, co-founder of Netsil, said in an interview with The New Stack that SREs and DevOps engineers are different from software engineers. He believes SRE teams should split up the management of services, thus creating more specialization. Dan Turchin, co-founder and chief product officer of Neva, said in an interview with The New Stack that he believes DevOps positions are replacing network operations center (NOC) engineers, who were traditionally looking at things from a data center perspective. If the old-school networking stats are being displaced by cloud infrastructure metrics, then this may be true.

The market is responding to this changing landscape. CA Technologies added the "perspectives" functionality to their solution, which allows teams to view and administer using any combination of grouping attributes. A major monitoring benefit of this approach is that it significantly simplifies and distills the data into views that are customizable by roles, tasks or services — essentially allowing data to be presented in context. Another example of role-based monitoring is playing out in the Kubernetes world, where the project has been redesigning its dashboard based on the differing needs of application developers, application operators and cluster operators.

New Mindset, New Methods

Although monitoring is changing to meet the needs of different job roles, it is also moving to a more holistic approach. As Majors wrote on her blog,

instead of relying on a fixed set of questions and checks, people should move towards the "observability" of systems. This has to happen because those fixed data points will not provide the needed insights alone. New tools are needed to keep pace and provide the ability to predict what's going to break. Many of these tools use machine learning and analytics.

Observability recognizes that testing won't always identify the problem. Thus, Majors believes that "instrumentation is just as important as unit tests. Running complex systems means you can't model the whole thing in your head." Besides changes in instrumentation, she suggests focusing on making monitoring systems consistently understandable. This means actually defining what the data represents and using the same definitions as your peers do both within and outside the organization. Furthermore, there is a frustration with the need to scroll through multiple, static dashboards. In response, vendors like CA Technologies are making more intuitive, interactive dashboards. Companies are even using artificial intelligence to determine what information displays when for each service.

Approaches to Address the New Reality

Increasing automation and predictive capabilities are common approaches to address new monitoring challenges.

Increasing automation centers around reducing the amount of time it takes to deploy and operate a monitoring solution. According to Steven Acreman, founder and Chief Technical Officer of Dataloop.IO, in an interview with The New Stack, the larger the organization, the more likely it will require customized solutions that can collect and integrate data from all their inputs and applications. Vendors are trying to reduce the number of steps required in the setup process. This might mean that once a monitoring agent is installed on a host, you don't have to think about it. More likely, it means that the tools have the ability to auto-discover new

applications or containers.

You also want to automate how you respond to problems. For now, there is a difference between automating certain tasks and automation that takes humans entirely out of the equation. Monitoring systems continue to create automated alerts, but now the alerts are more sophisticated. As James Turnbull notes, alerting will be annotated with context and recommendations for escalations. Systems can reduce the amount of unimportant alerts, which mitigates alert fatigue and increases the likelihood that the important alerts will be addressed. For now, the focus is getting the alerts to become even more intelligent. Thus, when someone gets an alert, systems display actionable information and workflows needed to quickly pinpoint problems and assist triage efforts.

Automating the container deployment process is also related to how you monitor it. It is important to be able to track the setting generated by your configuration management. This is where container orchestrators can help. Kubernetes, Mesos and Cloud Foundry all enable auto-scaling.

Just as auto-scaling is supposed to save time, so is automating the recognition of patterns. Big Panda, CA Technologies, CoScale, Dynatrace, Elastic Prelert, IBM Bluemix, Netsil and SignalFx are just a few of the companies that use artificial intelligence to identify patterns and detect anomalies. A common result is that much of the noise created by older monitoring approaches gets suppressed. In an interview with The New Stack, Peter Arjis of CoScale says anomaly detection means you don't have to watch the dashboards as much. The system is supposed to provide early warnings by identifying patterns of behavior among how different services, applications and infrastructure behave.

For example, CA Technologies APM solution uses analytics and machine learning to detect anomalies, identify the root cause of problems and

build automated triage workflows. By employing proven statistical techniques, CA APM dynamically builds performance baselines across groups of microservices. If a threshold is surpassed or anomaly detected, then engineers get both alerted and presented with an assisted triage dashboard. This approach eliminates the traditional practice of manually predicting acceptable performance baselines — which are unsustainable in microservice environments and often result in "event storms."

Finding the Most Relevant Metrics

The number of container-related metrics that can be tracked has increased dramatically. Since the systems are more complex and decoupled, there is more to track in order to understand the entire system. This dramatically changes the approach in monitoring and troubleshooting systems. Traditionally, availability and utilization of hosts is measured for CPUs, memory, I/O and network traffic. Although these are still important for managing IT infrastructure, they do not provide the best frame of reference for evaluating what metrics to collect.

Although there are many different layers in this IT environment, services are a key unit of observation. Service health and performance is directly related to application performance. Services can be defined with common names, with their health and performance benchmarked over time. Services, including microservices running in containers, can be tracked across clusters. Observing clusters of services is similar to looking at the components of an application.

Google's book on Site Reliability Engineering claims there are four key signals to look at when measuring the health and performance of services: latency, traffic, errors and saturation. Latency describes the time it takes to service requests. Within a container, it can be helpful to look at how slowly API calls are handled. Traffic and errors are both commonly

tracked, and refer to the communicating and networking of services and the frequency of errors. Saturation describes how "full" the service is and emphasizes the most constrained resources. It is becoming a more popular way to measure system utilization because service performance degrades as they approach high saturation.

Using this viewpoint, we can see what types of metrics are most important throughout the IT environment. Information about containers is not an end unto itself. Instead, container activity is relevant to tracking infrastructure utilization as well as the performance of applications and infrastructure. Metrics about the saturation and latency of requests within a container are most relevant. Metrics about the health of individual containers will continue to be relevant. However, in terms of managing containers, measuring the health of clusters of containers will become more important.

It's important to remember that you're not just monitoring containers, but also the hosts they run on. Utilization levels for the host CPU and memory can help optimize resources.

As Sematext DevOps Evangelist Stephan Thies wrote, "when the resource usage is optimized, a high CPU utilization might actually be expected and even desired, and alerts might make sense only for when CPU utilization drops (service outages) or increases for a longer period over some max limit (e.g., 85%)."

In the past, it was possible to benchmark host performance based on the number of applications running on it. If environments weren't dynamic, with virtual instances being spun up and down, then it would be possible to count the number of containers running and compare it to historical performance. Alas, in dynamic environments, cluster managers are automatically scheduling workloads, so this approach is not possible.

Questions to Ask When Deciding What Metrics to Monitor

	Questions	Sample Metrics
Microservice In general, there is one process to track per container.	Where are new services deployed? What percentage of time is the service reachable? How many requests are enqueued?	Average percentage of time a request-servicing thread is busy. Number of enqueued requests. Percentage of time a service is reachable.
Application Multiple microservices running simultaneously constitute an application.	Do the databases respond quickly? Are the message queues fast enough? How does heap memory usage change over time? Are my application services responsive?	Query execution frequency, response time and failure rate. Response time, failure rate.
Container Separate from the underlying process being run within it, containers are also monitored.	How responsive are the processes within the container? Which images have been deployed? Are specific containers associated with over-utilization of hosts?	CPU throttle time. Container disk I/O. Memory usage. Network (volume, dropped packets).
Container Cluster Multiple containers deployed to run as a group. Many of the metrics for individual containers can also be summarized.	Are your clusters healthy and properly sized? Can applications be effectively run using fewer nodes?	Percentage of clusters remaining operational compared to those originally deployed.
Host Also called a node, multiple hosts can support a cluster of containers.	Do changes in utilization indicate a problem with a process or application?	Percentage of total memory capacity in use. Percentage of time CPUs are utilized.
Infrastructure Broadly speaking, this is the cloud in which the hosts are running.	How much does it cost to run each service or deployment? What is the ratio of microservices and/or containers per instance?	Network traffic. Utilization of databases, storage, and other shared services.
End User The end goal of the entire system is to serve this group.	What is the average web response time experienced by users per region?	Response time. Number and percentage of user actions that failed.

TABLE 3: *Saturation and latency related metrics are the most relevant when monitoring microservices-based applications. Instead of looking at individual services and containers, dashboards and alerts should focus on their operation in aggregate.*

Instead, observing the larger IT environment for anomalies is becoming a way to detect problems.

The Next Steps

The biggest changes in IT monitoring are the new groups involved and the new metrics they are using. IT operations still care about availability and cost optimization. DevOps and application developers focus on the performance of services. Everyone, especially the chief information officer, cares about the impact on business operations and customer interactions.

Of course, there are new metrics that have to be monitored. The Identifying and Collecting Container Data chapter provides an overview of how to collect this data. All of these metrics can be collected in different ways. Classes of Container Monitoring details the different components of an effective monitoring stack. From collection to logging to visualization, there are unique technical challenges to monitoring containers and microservices. Looking at next steps, The Right Tool for the Job: Picking a Monitoring Solution provides important criteria to think about.