



Sysdig 2021 Container Security and Usage Report

Contents



Executive Summary	3
What Container Platforms are Being Deployed?	5
Security and Compliance	7
What Services are Customers Running?	15
Containers	18
Alerts	22
Kubernetes Usage Patterns	26
Demographics and Data Sources	29
Conclusion	31



You can find
all our past
reports [here](#).

Executive Summary

For the past four years, we've provided insights into container usage through real-time, real-world customer data. As our security and monitoring capabilities grow, our unique vantage point lets us discover details about how companies are dealing with security and compliance, in addition to how the usage of infrastructure, applications, and containers is evolving over time. Armed with these insights, we bring you the Sysdig 2021 Container Security and Usage Report.

Our customers tell us that security and compliance concerns surrounding their container environments are top of mind due to their ephemeral nature. Consistent with last year's report, about half of containers live for less than five minutes. This highlights the need for a detailed record that can be used for incident response, forensics, and troubleshooting. With that in mind, we have added a deeper look at the state of security to shed light on the challenges that face our customers. One highlight in our analysis revealed that for many companies, the trend of shifting left is extending to Kubernetes security with three-fourths of organizations scanning their container images in the CI/CD build phase prior to

deployment. While many teams have a high awareness around identifying vulnerabilities, their misconfigurations are leaving the door open to attackers. In fact, our analysis showed that the majority of container images are still configured to be overly permissive with 58% of them running as root, which has serious security implications. As container environments mature, organizations realize that scanning is not enough. They also need runtime security to deal with ongoing threats. As a way to deal with these concerns, we have seen tremendous growth in the adoption of the Cloud Native Computing Foundation (CNCF) Falco project, which helps organizations detect runtime threats inside containers, hosts, and Kubernetes environments.

While the use of Kubernetes for container orchestration didn't change in 2020, there is a clear shift in the choice of container runtimes as organizations move away from Docker and toward containerd and CRI-O. In fact, the Kubernetes project announced it will be officially deprecating the use of Docker later in 2021. With container density growing again this year, organizations are shifting toward Prometheus as the standard

way to monitor these environments. The use of Prometheus metrics among our customers grew 35% year-over-year and the top three exporters are node-exporter, blackbox-exporter, and jmx-exporter, based on GitHub statistics. The Quay registry saw increased adoption this year while Golang, a popular programming language choice for cloud-native developers, made a big jump in usage among organizations.

The data in this report is derived from an analysis of millions of containers that a subset of our customers are running every day and the nearly one billion unique containers that our customers have been running over the past year. In this report, you will find further detail about security, compliance, services, alerting, and Kubernetes usage patterns. This information can be useful for determining the real-world state of security and usage for container environments at companies around the world, from a broad range of industries.

Key 2021 Trends

Security


74% of customers are scanning images during the CI/CD build stage


58% of containers run as root

49% of containers live less than 5 minutes

Open Source

3x increase in Falco adoption  **Falco**

35% growth of Prometheus metric use 

4.7x increase in usage of Go 

Container Usage

4x increase in containerd and CRI-O  **cri-o**

33% growth in container density

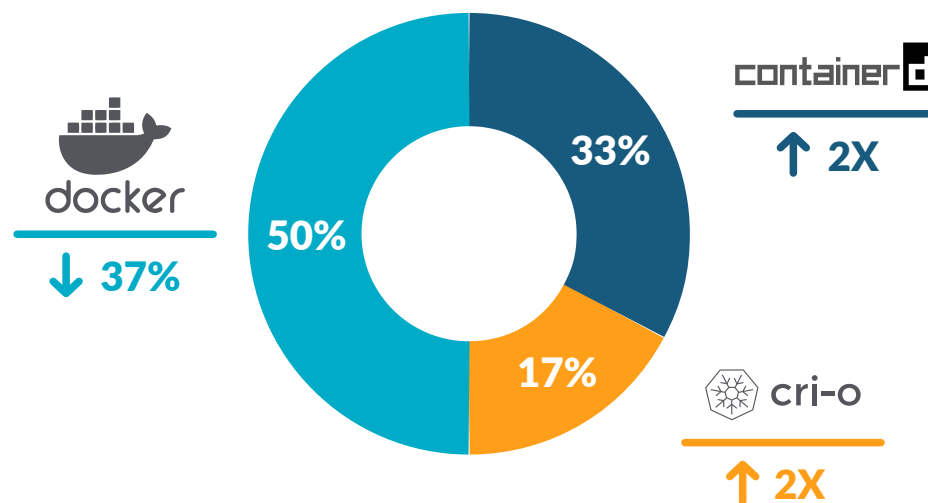
60% increase in Quay.io usage 

What Container Platforms are Being Deployed?

Container runtimes

Over the past year we have seen significant growth for both containerd and CRI-O (up from 18% and 4% last year respectively) over Docker which came in at 79% last year but is down to 50% this year. It is also notable that the Kubernetes project announced it will be officially deprecating the use of Docker in late 2021. To be fair, it's important to note that containerd is used by Docker. The Docker engine previously implemented both high-level and low-level runtime features. These are now broken out into separate containerd and runc projects. Choosing which container runtime to use may seem a little unclear given the

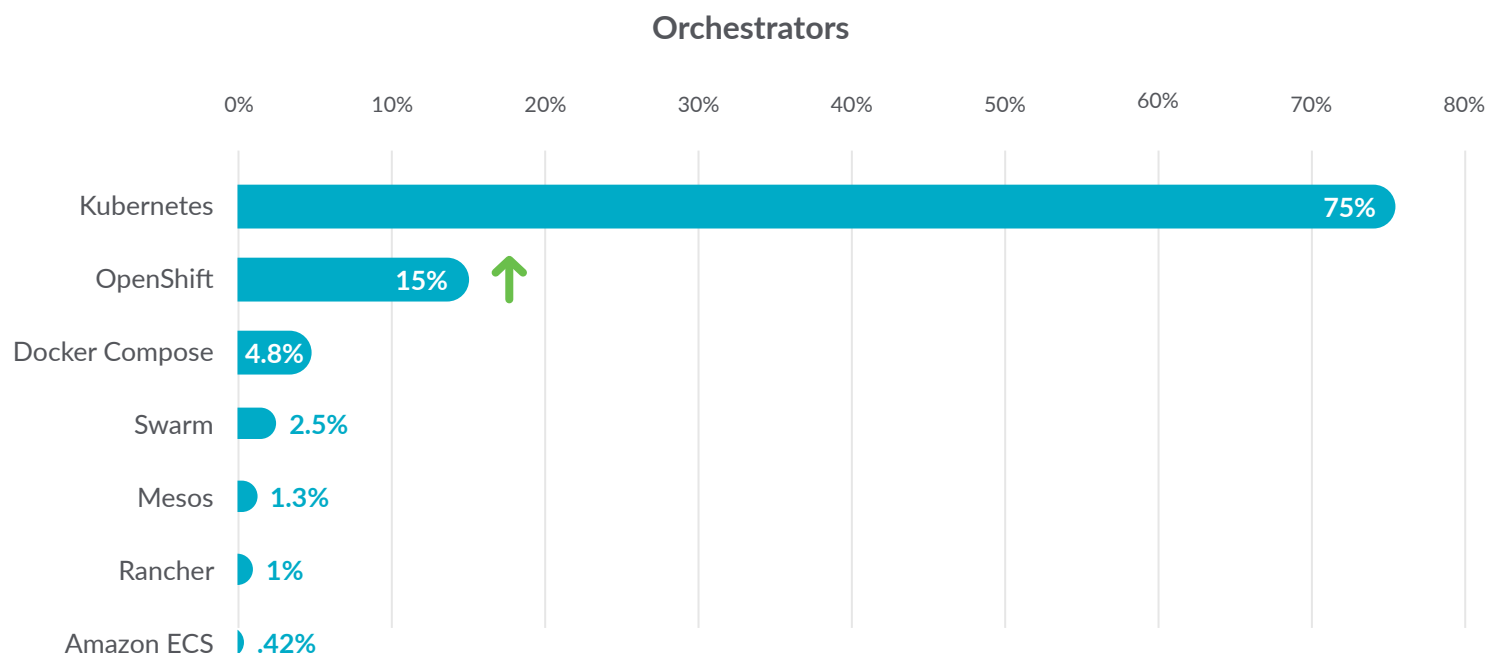
emergence of several options. Different solutions cite aspects like reduced overhead, stability, extensibility, and container registry compatibility as advantages. Now, however, because of the open standards, concerns about making the wrong choice and lock-in have evaporated. To make it even easier, popular platforms like OpenShift, GKE, and IKS support using multiple container runtimes in parallel and have typically designed in a runtime of choice, removing the need to spend any cycles on deciding which one to use.



Container orchestration platforms

Kubernetes holds a steady lead at this point over the other orchestrators, shifting only slightly from last year's report. The chart at the bottom of the page shows the current breakdown. Surprisingly, Swarm and Mesos stayed at about the same usage levels, 2.5% and 1.3% respectively, compared to last year. OpenShift takes the biggest jump from 9% to

15% as more of our users seem to be relying on OpenShift due to its ability to run in multiple cloud environments. Docker Compose, which is used to manage multiple containers only on a single host, has been added this year even though it may not be considered a direct corollary to multi-host orchestrators like Kubernetes.



Security and Compliance

As organizations move container workloads to production, they are recognizing the need to integrate security and compliance into the DevOps workflow. “Shift security left” has become a buzz phrase that often refers to scanning containers for vulnerabilities. Scanning is clearly critical given the high percentage of container images pulled from public registries and the high failure rate of scanned images. But the data also highlights the need for compliance checks and stringent runtime policies to reduce risk. To provide insights into the state of security and compliance in Kubernetes and cloud-native environments, we’ve analyzed data points that include vulnerability scanning, runtime security, and compliance.

Image scanning

Regardless of the source of the container images, it is critical to perform image scanning and identify known vulnerabilities prior to deploying into production. To quantify the scope of the risk of vulnerabilities, we sampled pass and fail rates for images scanned over a seven-day period. Over half of the images failed, meaning they were found to have known vulnerabilities with a severity of high or greater.

Scanning Results

Median of Containers Scanned



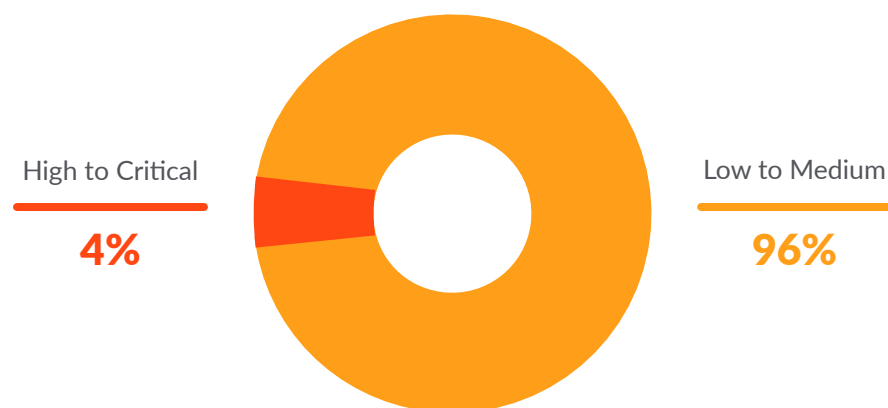
"The more you scan, the faster flaws get identified and fixed. We have found that scanning in the CI/CD pipeline allows us to reduce risk and ensure security is not a blocker for fast app delivery."

- Natnael Teferi
Lead DevSecOps Cloud Security Architect at FIS

OS vulnerability snapshot

We noticed that 4% of OS vulnerabilities are high or critical. Although this may seem low, if an OS vulnerability is exploited, it can compromise your entire image and bring down your applications. This is also why there is a heavy focus on scanning for OS vulnerabilities, especially by cloud providers that provide this capability as part of registry scanning (i.e., ECR, GCR, etc.).

OS Vulnerabilities by Severity



Non-OS vulnerability snapshot

What many teams don't check for are vulnerabilities in third-party libraries. We found that 53% of non-OS packages have high or critical level severity vulnerabilities. Developers might be unknowingly pulling in vulnerabilities from these non-OS open source packages, like Python PIP, Ruby Gem, etc., and introducing security risk.

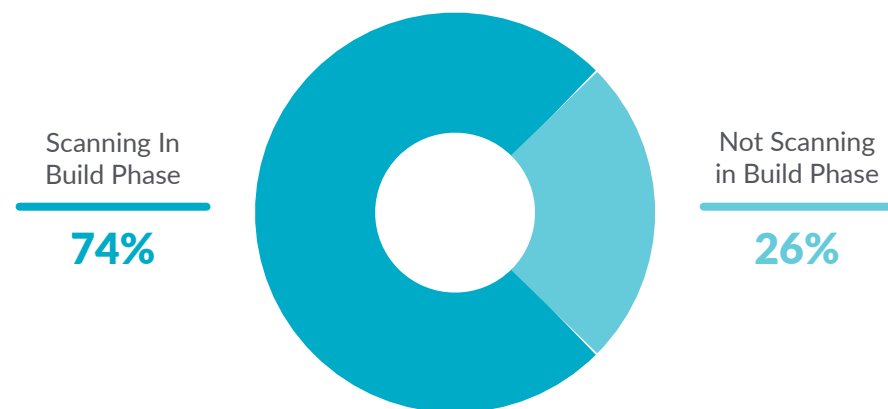
Non-OS Vulnerabilities by Severity



Scanning in build phase

DevOps teams are “shifting left” with the goal of starting to consider important implications earlier in the development lifecycle. Security is key among these concerns. As part of our analysis, we looked at the number of organizations that are scanning their images as part of the build phase in their CI/CD pipeline and those that are not. 74% of our customers are in fact scanning pre-deployment. This is a good sign because scanning in the build phase helps teams address potential security risks with images before they make it into production.

Images Scanned In Build Phase



Where does scanning occur: inline vs. backend scanning

There are two fundamental approaches customers can take to scan images:

Backend Scanning — When using backend scanning (i.e., directly in the UI or via an integration that uses the `sdc-cli`), the Sysdig backend will pull the entire image from the registry and execute both the image analysis (extraction of the image metadata such as installed packages, versions, file attributes, Dockerfile instructions, etc.) and the evaluation (detection of OS/non-OS vulnerabilities, misconfigurations, and bad security practices). Many teams leverage backend scanning. While inline scanning might be the goal since it provides better security, it is the more advanced step.

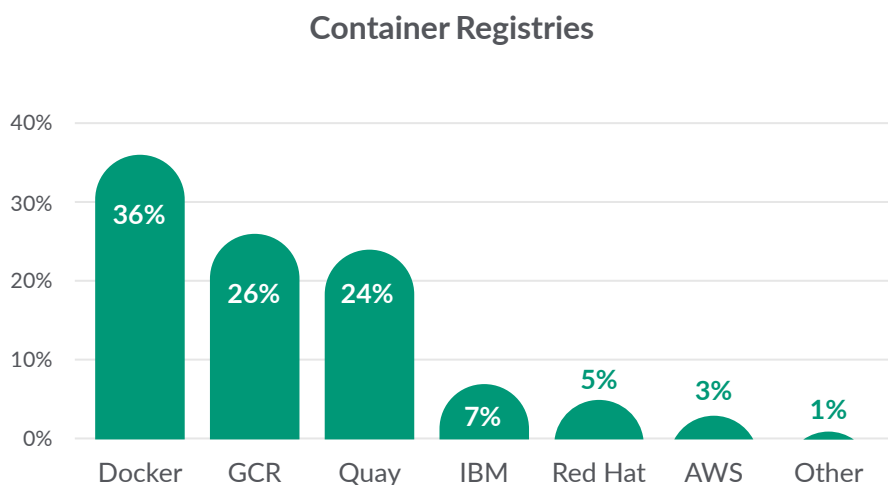
Inline Scanning — When using inline scanning, the image analysis phase takes place directly within your CI/CD pipeline, registry, or at runtime. The resulting metadata is sent to the Sysdig backend for evaluation and the policy evaluation is sent back to the worker (i.e., as a PDF or JSON artifact). This allows you to have full control of your image data, without sharing image contents or exposing registry credentials outside. The scan results are directly seen in Sysdig.

Inline vs. Backend Scanning



Public and hosted container registries

Container registries provide repositories for hosting and managing container images. Docker registries are most frequently used — common among 34% of our customers. This measure includes both private hosted and public repositories. Registry solutions hosted by cloud providers are increasingly popular. Similar to the past few years, the Google Cloud Registry is once again the top public cloud repository, used by 26% of our Sysdig users. However, Quay has picked up some growth from last year, increasing from 14% to 24%.



Within these various offerings, we looked at the percentage of containers pulled from public vs. private repositories. We found that public sources are being trusted more and more with an increase from 40% last year to 47% this year. The risk of using container images from public repositories is that few are validated or checked for security vulnerabilities. However, with more companies improving their security procedures and processes in Kubernetes environments, the convenience of using public repositories may outweigh the risk. Our customers are creating policies to define which container registries are approved for use in their organizations.

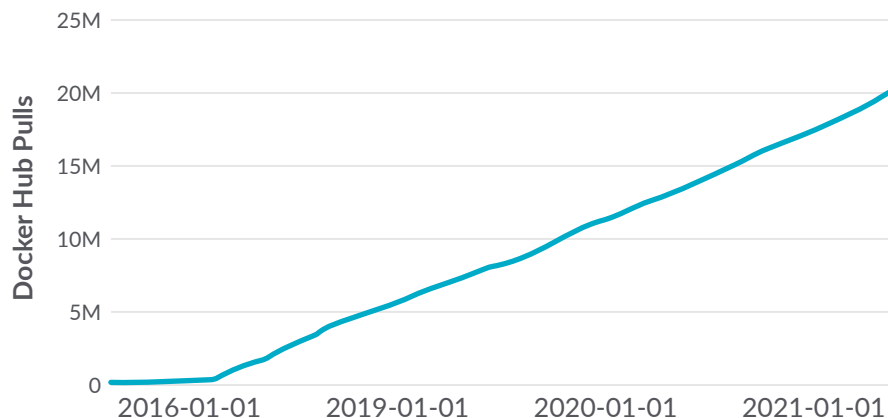
Images Pulled from Public vs. Private Registries



Runtime security threats

Once known vulnerabilities have been addressed in the build phase of the container lifecycle, teams need to set policies that will detect anomalous behavior and trigger security alerts at run time. Runtime security for Kubernetes is something organizations are just starting to address. Falco, the CNCF open-source project contributed by Sysdig, is quickly gaining momentum and interest, as seen in the project stats below. The project now has over 20 million Docker Hub pulls, which represents a 300% growth compared to last year's 252% increase. Falco enables the definition of runtime policies that detect security violations and generate alerts.

Growth of Falco



616 contributors

9,560 code commits

1,635 pull requests

25K contributions

3,168 stars

"With increasing concerns about security in container environments, the continuing growth of Falco means more users are taking advantage of community-based rules. As the Falco project grows, Kubernetes security is strengthened by the collective group working together against bad actors."

- Chris Aniszczyk, CTO, CNCF

Containers running as root

While teams understand the need to scan for vulnerabilities, they may not be scanning for common configuration mistakes. What we see is that 58% of images are running as root, allowing for privileged containers that can be compromised. There are some containers that should run as root (for example, security and system daemons), but this is a small portion of total containers. From talking to our customers, in practice, even if risky configurations are detected at runtime, teams don't stop containers in order to continue deploying quickly. Instead, they run within a grace period and then decide on the remediation step.

58%

run as root

Top runtime policy violations

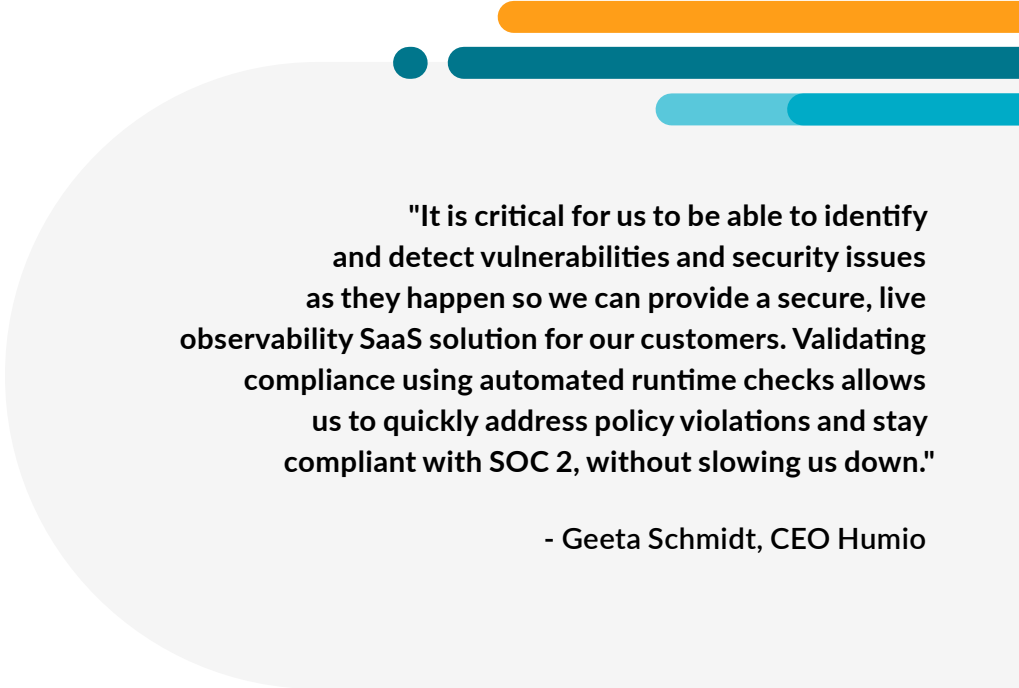
We looked at policy violations as measured by the volume of alerts customers are receiving. This indicates the types of runtime security risks that container users are uncovering most frequently. This year we saw a rise in suspicious filesystem and suspicious container violations.

Each of the following violations are detected by Falco security policies that are enabled by default in Sysdig Secure. Below, we provide the top seven violations in order of frequency, along with a description of each to explain the possible threat.

Violation	What is it	Why it's a security threat
Write below etc	Attempt to write to any file below the /etc directory.	Adding or altering files in the /etc, could be an attempt to change the application behavior.
Launch Privileged Container	Starting a privileged container.	Privileged containers can interact with host system devices, cause harm to the host OS, and gain access to other containers.
Write below root	Attempt to write to any file directly below /or / root.	Modifying data in these directories could be an unauthorized attempt to install software on the container.
Suspicious Filesystem Changes	Newly identified suspicious filesystem activity that might change sensitive/important files.	Attacker might be trying to get access to sensitive data.
Launch Sensitive Mount Container	Starting a container that has a file system mount from a sensitive host directory.	Indicates the container has access to data volumes that may contain sensitive files.
Suspicious Container Activity	Identified suspicious container-related activity (execs into containers, etc.).	Could be an indicator of compromise within the container system.
Terminal shell in container	A shell was used as the entrypoint/exec point into a container with an attached terminal.	Enables an attacker to manipulate the system, download malware, or initiate other malicious activity.

Compliance

Since today's enterprises face a number of governance and regulatory compliance requirements including PCI-DSS, HIPAA, and GDPR, taking steps to follow best practices in order to comply with regulations is imperative. The Sysdig platform runs compliance checks against monitored clusters to check hosts, containers, and other aspects of the environment against a defined set of best practices. This includes the Center for Internet Security (CIS) benchmark tests, CIS benchmark for Kubernetes, and CIS benchmark for Docker. We chose a sample from over 80 benchmark rules from the CIS benchmark for Docker to highlight the state of compliance against these best practices with Sysdig users. The eight benchmarks evaluate container images residing on each host for configuration issues related to permissions, security tooling, and configuration that have the potential to expose an organization to risk. We took the median score for each of these eight container checks. The score, in this case, is the measure of containers per host that fail the test and do not adhere to the recommended best practice for reducing risk.



"It is critical for us to be able to identify and detect vulnerabilities and security issues as they happen so we can provide a secure, live observability SaaS solution for our customers. Validating compliance using automated runtime checks allows us to quickly address policy violations and stay compliant with SOC 2, without slowing us down."

- Geeta Schmidt, CEO Humio

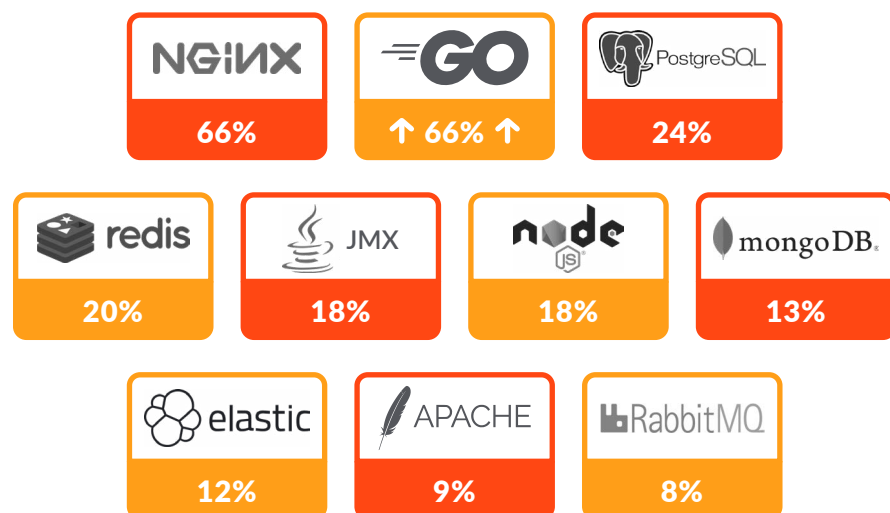
Benchmark	Median number of vulnerable containers per host	Why it's a threat
Containers with no <i>healthcheck</i> instruction configured	49	An important security control is that of availability. Adding the <i>HEALTHCHECK</i> instruction to your container image ensures that the Docker engine periodically checks the running container instances against that instruction to ensure that containers are still operational. Based on the results of the health check, the Docker engine could terminate containers which are not responding correctly, and instantiate new ones.
Container without a dedicated <i>cgroup</i>	34	At run time, it is possible to attach a container to a different <i>cgroup</i> other than the one originally defined. This usage should be monitored and confirmed, as by attaching to a different <i>cgroup</i> , excess permissions and resources might be granted to the container and this can therefore prove to be a security risk.
Containers with disabled default <i>seccomp</i> profile	34	A large number of system calls are exposed to every user and process with many of them going unused for the entire lifetime of the process. Most applications do not need all these system calls and would therefore benefit from having a reduced set of available system calls. Having a reduced set of system calls reduces the total kernel surface exposed to the application and thus improves application security.
Containers with unlimited restart ability	29	If you indefinitely keep trying to start the container, it could possibly lead to a denial of service on the host. It could be an easy way to do a distributed denial of service attack, especially if you have many containers on the same host. Additionally, ignoring the exit status of the container and always attempting to restart the container likely means the root cause of the container getting terminated will not be investigated.
Containers with default <i>ulimit</i>	29	<i>ulimit</i> provides control over the resources available to the shell and to processes started by it. Setting system resource limits in a prudent fashion to protect against denial of service conditions. On occasion, legitimate users and processes can accidentally overuse system resources and cause systems to be degraded or even unresponsive.
Containers with no AppArmor profile	28	An alternative to SELinux, AppArmor is available by default on most Linux distributions. AppArmor enables the association of a security profile to each application and restricts access to the underlying system.
Container with / root mounted in read-write mode	28	The container's root filesystem should be treated as a 'golden image' by using Docker run's—read-only option. This prevents any writes to the container's root filesystem at container runtime and enforces the principle of immutable infrastructure.

What Services are Customers Running?

The top 10 open-source solutions running in containers

Open source has changed the face of enterprise computing. It powers innovation across not just infrastructure, but especially application development. Sysdig's ability to auto-discover the processes inside containers gives us instant insight into the solutions that make up the cloud-native services that our customers run in production. Below are the top 10 open source technologies deployed by Sysdig customers:

Go is going places!



The 2021 list includes a wide range of services — each critical to the function of modern applications, including:

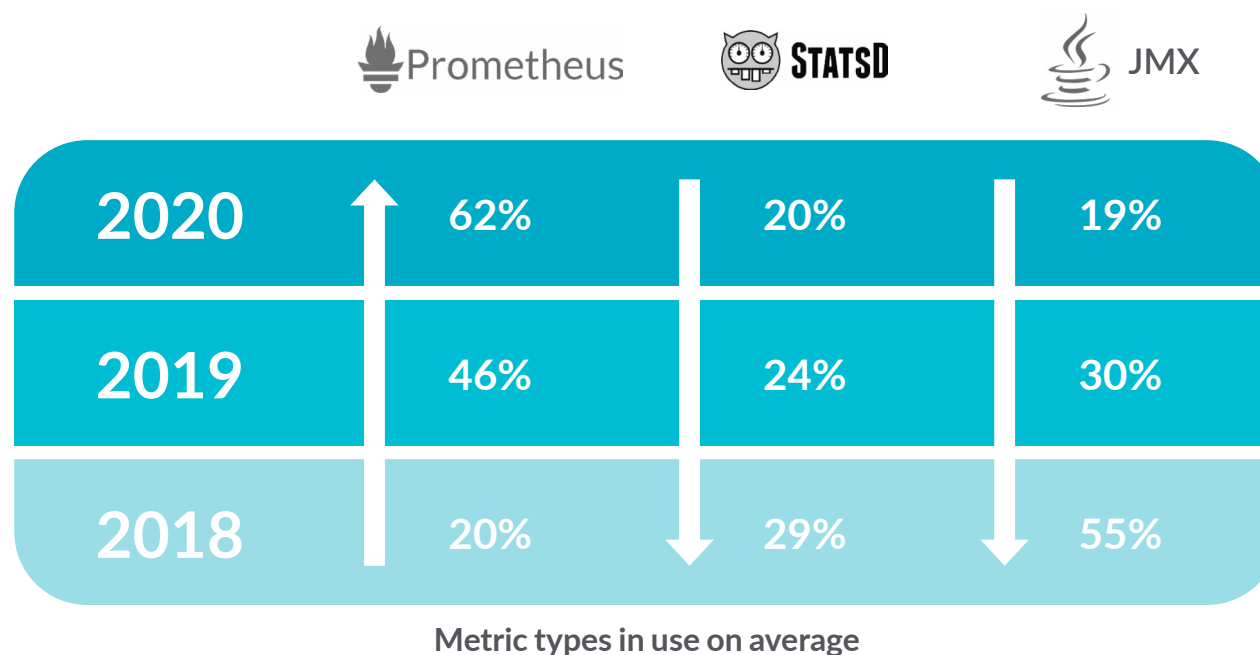
- HTTP server and reverse proxy solutions — NGINX
- NoSQL, relational, and in-memory database solutions — MongoDB, Postgres, and Redis
- Logging and data analytics — Elasticsearch
- Programming languages and frameworks — node.js, Go, and Java/JVMs
- Message broker software — RabbitMQ

Given the wide range of options available in the open source community, it's surprising that the services on our list have remained fairly consistent over the past three years. We purposely omitted Kubernetes components like etcd and fluentd as well as Falco. Since these are deployed by default, they end up at the top of the list for every Kubernetes user. Last year, both Node.js and Go (aka golang) overtook the use of Java. This year, Go has shot up in usage from 14% to 66%, an increase of 470%. Go, created by Google engineers, is quickly becoming the language of choice for developing cloud-native applications. The top 10 solutions above are widely deployed and trusted services. If you're in the market for similar services, you can't go wrong with taking advantage of what these open source solutions offer. There is, however, a long tail of software solutions available.

Custom metrics

Custom metric solutions give developers and DevOps teams a way to instrument code to collect unique metrics. This approach has become a popular way to monitor applications in production cloud environments. Of the three mainstay solutions, JMX, StatsD, and Prometheus, it was Prometheus that gained for the second year in a row. Year-over-year,

Prometheus metric use increased to 62% across our customers — compared to 46% last year. As the use of new programming frameworks expands, alternatives like JMX metrics (for Java apps) and StatsD continue to decline, down 35% and 15% year-over-year respectively.



Top Prometheus exporters

One of the most successful open-source projects to emerge from the CNCF, Prometheus has become synonymous with cloud-native monitoring. It is now widely adopted as a metric standard in projects like Kubernetes, OpenShift, and Istio. In addition, an increasing number of “exporters” are available to provide metric output for a wide range of third-party solutions. We expect the popularity of Prometheus to continue its growth within our customer base, particularly as Sysdig now offers full Prometheus compatibility for large-scale environments.

For this ranking, we looked at each github project listed on prometheus.io and measured the number of issues, stars, and forks for each project, and correlated the results against the number of Dockerhub or other repository pulls.

"Prometheus, in combination with Sysdig, gives us the ability to adapt scraping to our needs while filtering what's important to us. On top of that, we can provide our users with strong visualizations and alerting thanks to PromQL, which notably improved our monitoring system."

- Mario Simko, Observability Team
Leader SAP Concur

Top 10 Prometheus Exporters

Name	Maintainer
node_exporter	prometheus
blackbox_exporter	prometheus
jmx_exporter	prometheus
redis_exporter	oliver006
windows_exporter	prometheus
postgres_exporter	wrouesne
elasticsearch_exporter	justwatchcom
mysqld_exporter	prometheus
snmp_exporter	prometheus
kafka_exporter	danielqsi

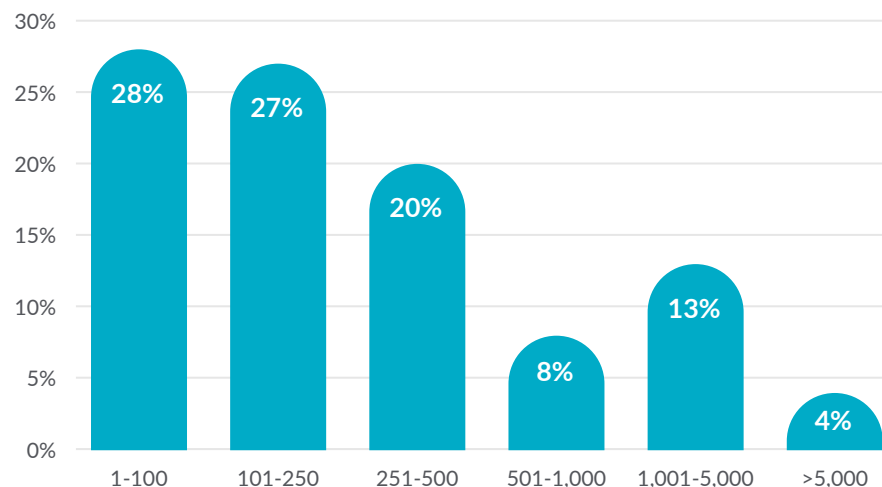
Containers

Each year, we take a look at details specific to the count and activity around containers, including density and lifespans. This provides insight into the rate of adoption, but also illustrates the scale and efficiencies being achieved.

Containers-per-organization

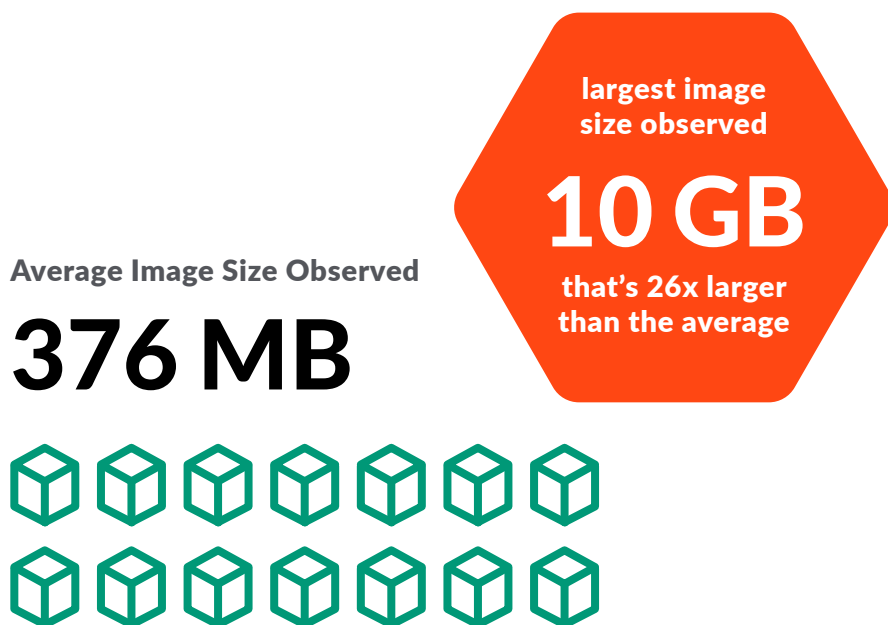
To get a sense of the scale at which enterprises are currently operating, we looked at the number of containers each customer runs across their infrastructure. Over half of customers run 250 or fewer containers. At the high end, only 4% of customers are managing more than 5,000 containers. It is common for adoption to begin at a small scale, sometimes born from developers who push for containerization as a means to accelerate software delivery. DevOps and cloud teams report that once the benefits are proven, adoption accelerates as more business units look to onboard to the new platform. However, the raw number of containers running should be taken into account, along with the size of those containers (see right).

Number of Running Containers



How big are images?

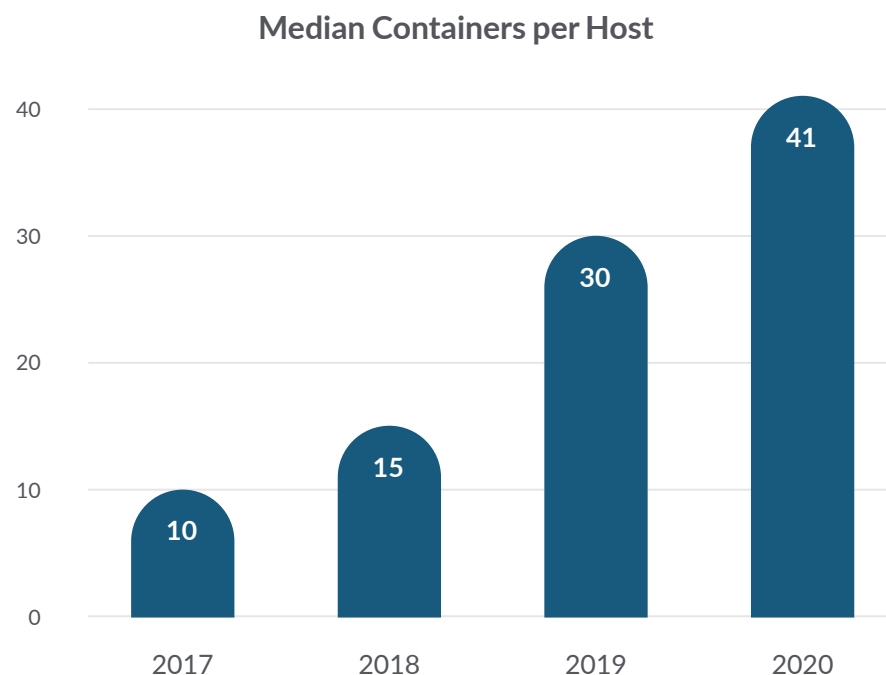
Although image size depends on the application, based on our data, the average image size observed is 376 MB. The large 10GB Alpine image seems to be an outlier, as it is not a good practice to have a large image unless absolutely necessary. Large images not only take longer to deploy, slowing down release velocity, but they also expose more opportunities for attack.



Container density

Containers-per-host density increases 33%!

Over the past four years, the median number of containers per host increased in every report. However, this year showed only a 33% increase year-over-year compared with the 100% increase of last year. It is possible that the number will continue to increase slightly in the future, but that density will probably come at the cost of overall image size. While the primary goal of containers is to speed development and deployment, many organizations are benefiting from increased utilization of hardware resources thanks to container efficiencies.



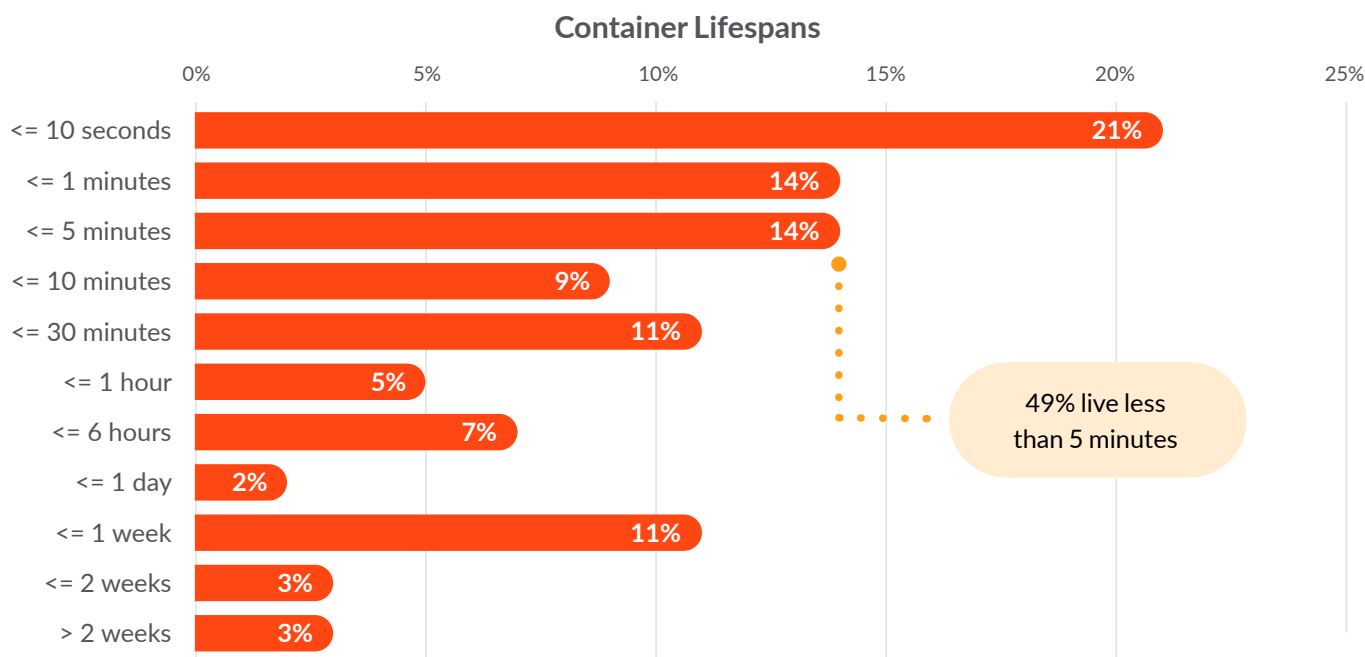
Container, image, and service lifespans

The measure of how long (or how short) containers, container images, and services live was one of the most popular data points from our 2019 report. It reflects just how dynamic modern applications are from both a development and a runtime perspective.

The short life of containers

Comparing container lifespans year-over-year, we see a similar pattern where a majority of containers are alive for less than a week. In fact, our newest data sample shows that the number of containers that are alive for 10 seconds stayed relatively unchanged at 21% compared to 22% for last year.

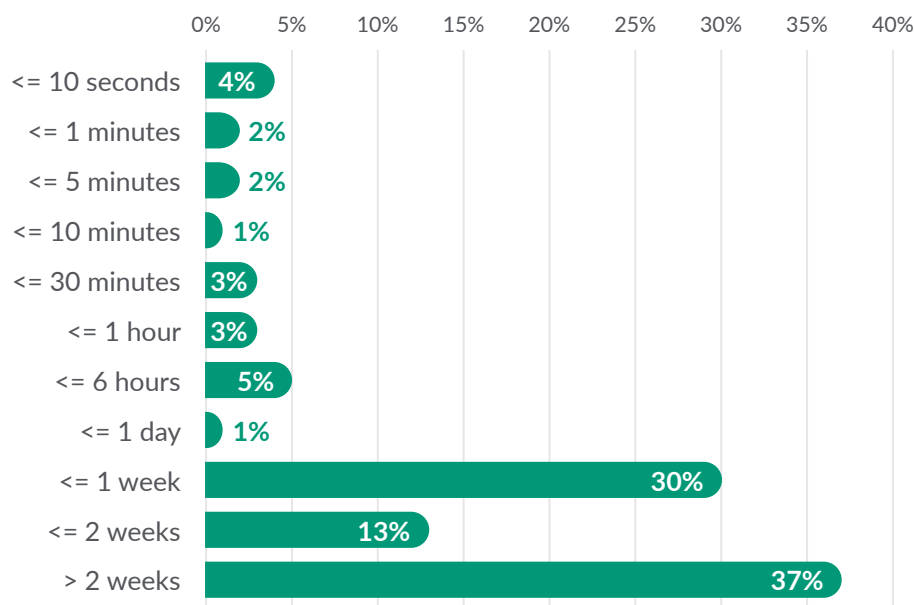
Many containers need to only live long enough to execute a function and then terminate when it's complete. Seconds may seem short, but for some processes, it's all that is required. The ephemeral nature of containers remains one of the technology's unique advantages, but presents new issues to consider for monitoring, security, and compliance. As adoption of cloud serverless technologies grows, it is possible that the pendulum will swing back the other way as short lived processes and services are moved away from containers and toward hosted functions. However, this would not represent a change in the overall makeup of the workloads running in an environment. Rather it could represent a shift of those short-lived workloads from one technology to another.



Continuous development and image lifespans

Containers are a perfect companion to the agile movement, accelerating the development and release of code, often as containerized microservices. Our image lifespan data reflects the shift in the time between code releases and the reality that CI/CD pipelines are helping developer teams deliver software updates at a faster cadence than ever before. The data shows that over half of container images get replaced – also known as churn – in a week or less. For most, if not all, of today’s businesses, speed to market matters and makes all the difference in maintaining competitiveness. Code deployment is being deployed more frequently, which means new container images. Containers support what businesses need to turn great ideas into reality, fast.

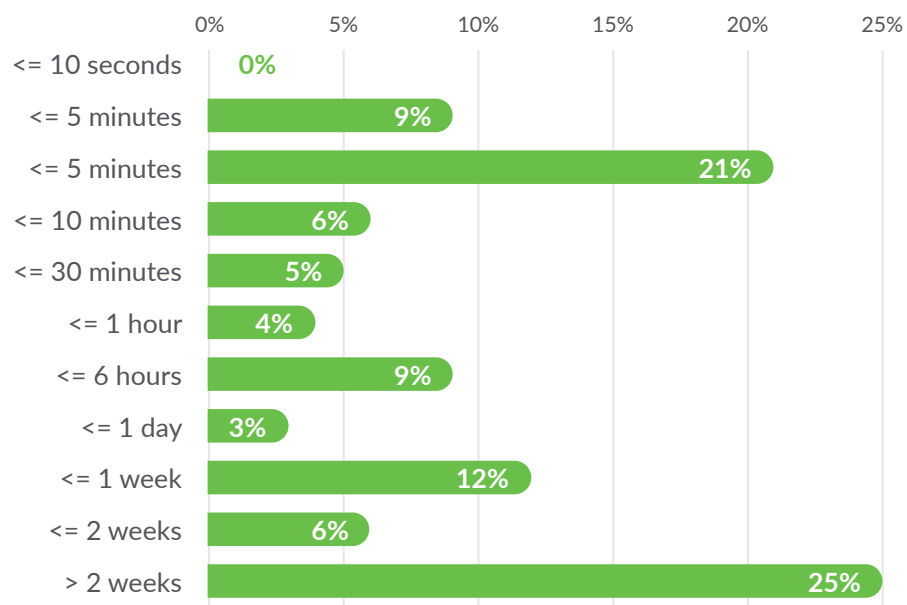
Container Image Lifespans



Service lifespan

For our last view into lifespans, we examined the data around services and uptime. Services – the functional software components of our applications like database software, load balancers, and custom code – might be continuously improved. However, at the same time, it’s important (at least for most 24/7 businesses) to keep services up and running around the clock. Unlike the past few years where over half of our customer services were up for two weeks or more, this year we saw more variance in uptime, especially in the less than five-minute range.

Service Lifespans



Alerts

Analysis of trends with the types of alerts set by our customers helps us understand the kind of conditions that our users identify as having the most potential for disruption to their container operations.

The top 10 alert conditions

There are more than 800 unique alert conditions being used across our customers today. The graphic below represents the most commonly used alert conditions, along with the percentage of customers using each. The makeup of these alerts has changed since our last report, shifting in favor of Kubernetes node availability while focusing slightly less on resource utilization and uptime.



Alert scopes

Sysdig alerting supports customization by “scoping” to a specific tag or Kubernetes / cloud label. For instance, using an example from the above alerts, you can specify memory.used.percent alert for an individual namespace like “istio-system,” or for a specific Pod name like “envoy” inside that namespace. Tagging and labeling play a critical role in cloud-native environments, providing unique identifiers that help organize

and isolate items. In this case, the tagging specifies a group of “things to watch.” Specifying alerts by Kubernetes labels is now one of the most common practices, including namespace, cluster, deployment, and host in the top five. Agent tags — the metadata attached to the Sysdig agent when deployed — rise to the most popular alert scoping across Sysdig users.

2019

Scope label type	% of users
Kubernetes namespace	94%
Agent tags	78%
Kubernetes cluster	76%
Kubernetes Deployment	71%
Kubernetes Pod	38%

2020

Scope label type	% of users
Agent tags	88%
Kubernetes namespace	75%
Kubernetes cluster	52%
Kubernetes Deployment	37%
Host	31%

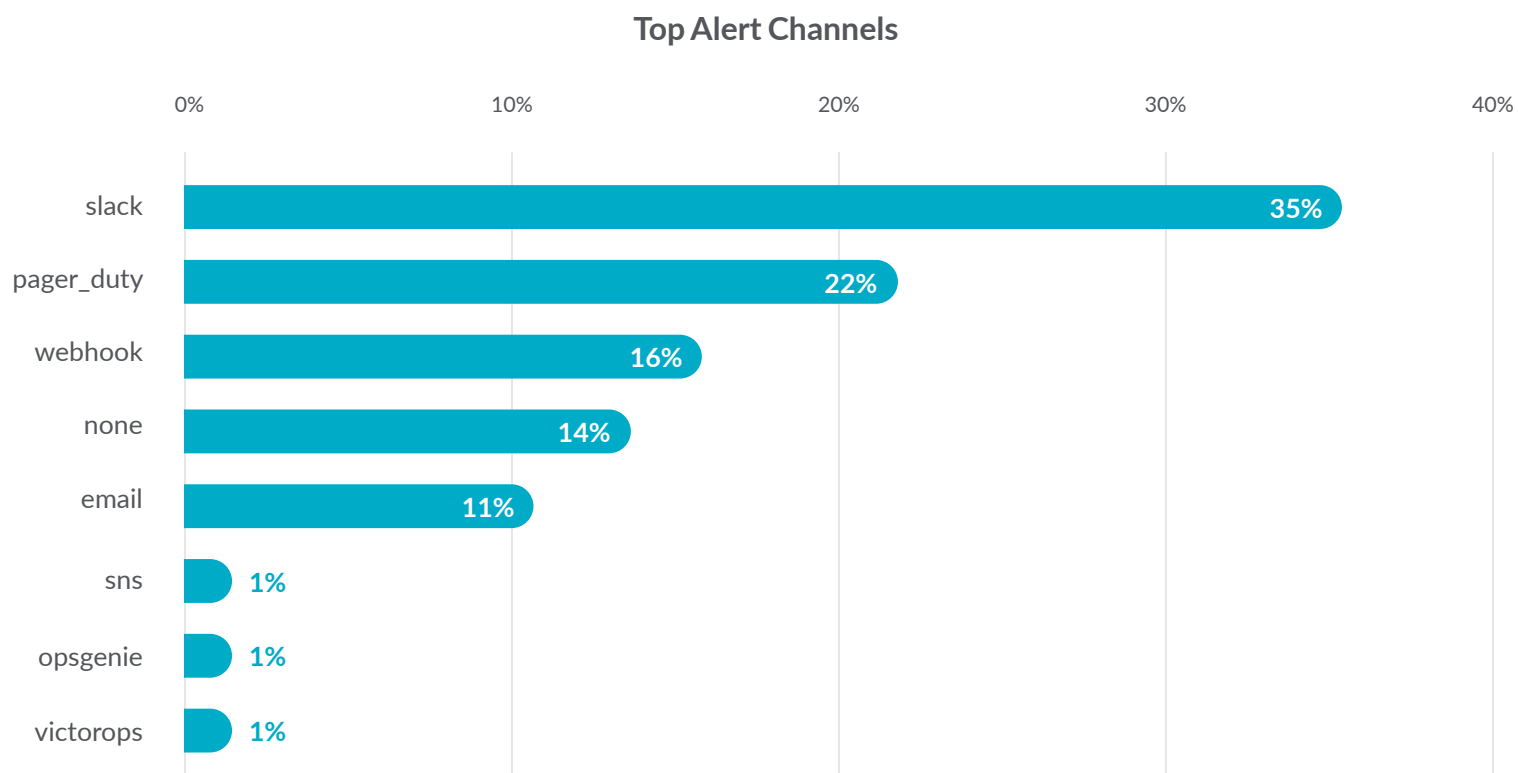
The word cloud below shows a breakdown of some of the other tags and labels our customers were using to scope their alerts.



Alert channels

We looked at the communication channels users have configured to receive alerts. Slack took the top position, greater than purpose-built incident response platforms and even email. We find the results interesting because unlike PagerDuty and Opsgenie, for instance, Slack is not considered an incident response platform. It's likely that Slack is being used for non-critical alerts handled during working hours, while solutions like PagerDuty are being used for "waking people from bed."

This year, we decided to include a category for alerts that didn't have a notification channel configured. This could be because the alert was for informational purposes only or because the Sysdig platform itself provided enough information to satisfy the demands of the alert in question.



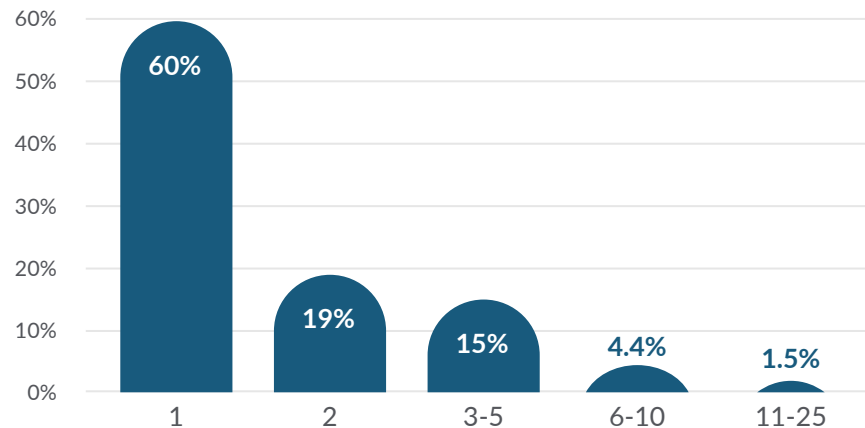
Kubernetes Usage Patterns

How many clusters are customers operating? How many pods run per node? In this section, we answer these questions and more. We look at a range of details about what customers are doing with Kubernetes, from clusters to ReplicaSets. Because Sysdig automatically collects Kubernetes labels and metadata, we're able to provide cloud-native context for all of the data insights we discover, from performance metrics and alerts to security events. This same capability enables us to capture each of the following usage metrics from the cluster all the way to pods and containers, all with a simple query.

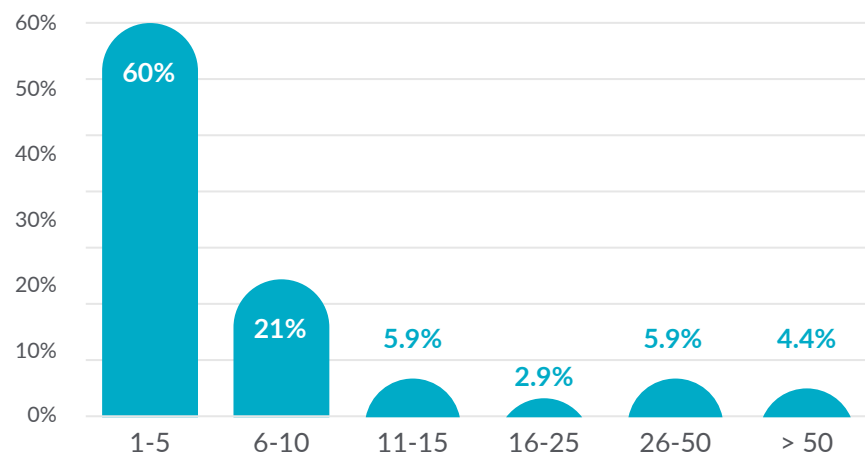
Kubernetes clusters and nodes

Some customers maintain a few clusters — some small, some large — while others have a sizeable estate of many clusters of varying sizes. The charts to the right provide a distribution of cluster count and nodes per cluster for users of the Sysdig platform. The large number of single clusters per customer, and relatively small number of nodes, is an indication that many enterprises are still early in their use of Kubernetes. We've also recognized that the use of managed Kubernetes services in public clouds is another factor that impacts these data points. With services like Amazon Elastic Kubernetes Service (EKS), Google Kubernetes Engine (GKE), Azure Kubernetes Service (AKS), and IBM Cloud Kubernetes Service (IKS), users can spin up and tear down clusters quickly as needed.

Number of Clusters



Nodes per Cluster

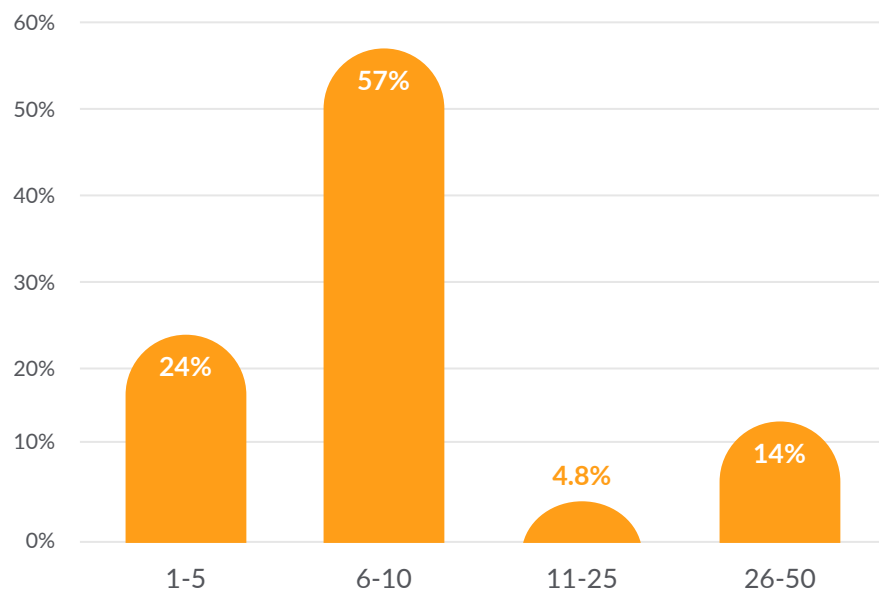


Kubernetes namespaces, deployments, and pods

Namespaces per cluster

Kubernetes namespaces provide logical isolation to help organize cluster resources between multiple users, teams, or applications. Kubernetes starts with three initial namespaces: default, kube-system, and kubepublic. How namespaces are used varies across organizations, but it is common for cloud teams to use a unique namespace per application.

Kubernetes Namespaces per Cluster

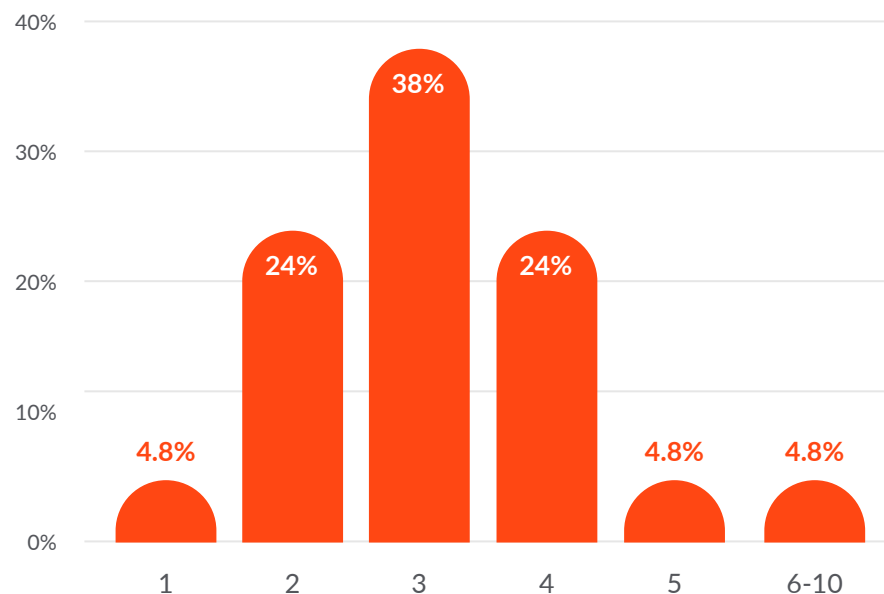


Deployments per namespace

Deployments describe the desired state for pods and ReplicaSets and help ensure that one or more instances of your application are available to serve user requests. Deployments represent a set of multiple, identical pods with no unique identities such as deployments of NGINX, Redis, or Tomcat. The number of deployments per namespace provides an idea of how many services compose our users' microservices applications.

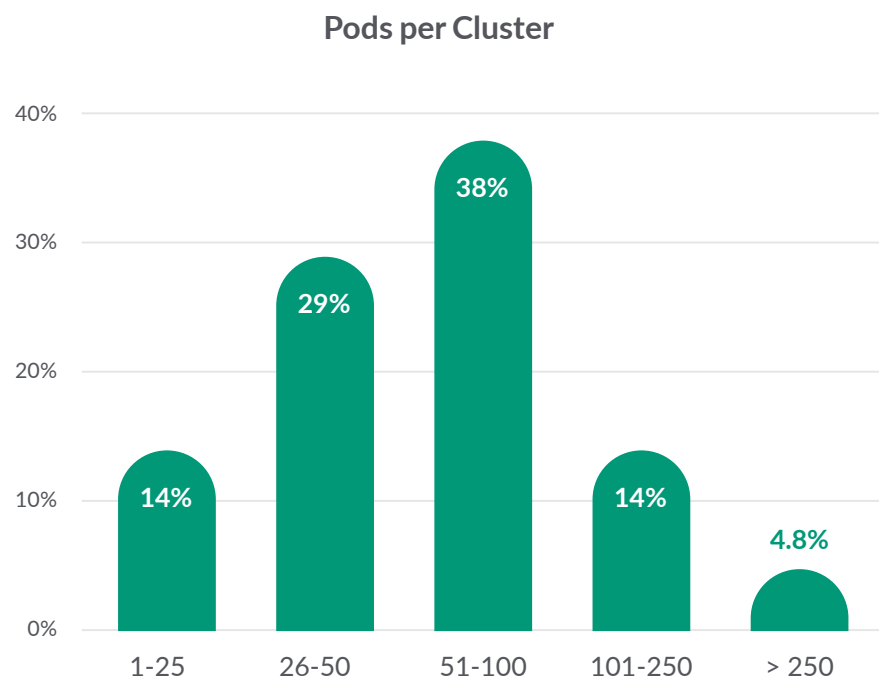
We saw a slight shift this year toward fewer deployments per namespace. Segmenting access to environments is easiest by namespace, so by having fewer deployments in each namespace could enable teams to do more segmenting to limit access to the applications they are responsible for.

Deployments per Namespace



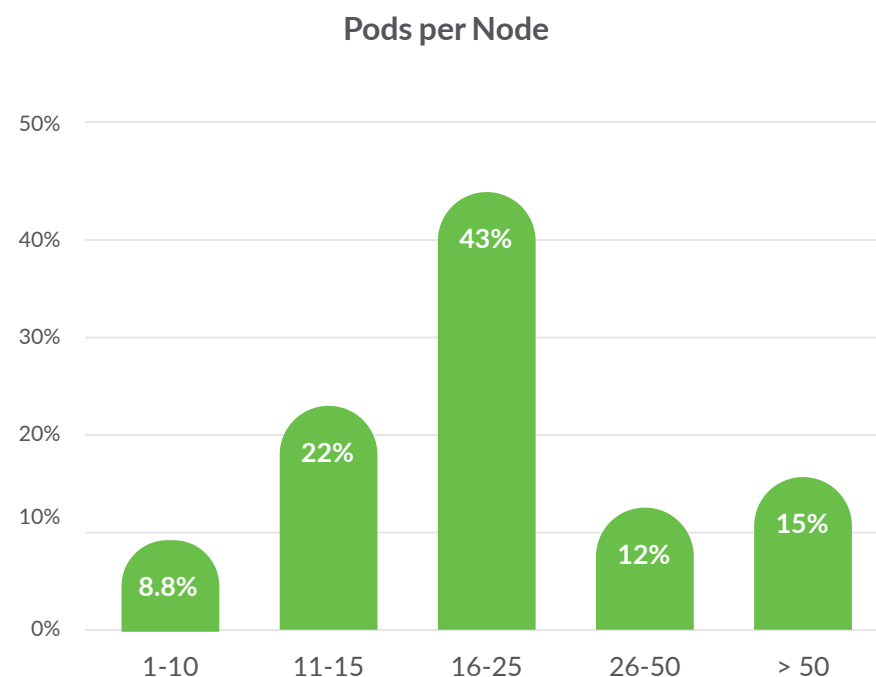
Pods per cluster

Pods are the smallest deployable object in Kubernetes. They contain one or more containers with shared storage and network, as well as a specification for how to run the containers.



Pods per node

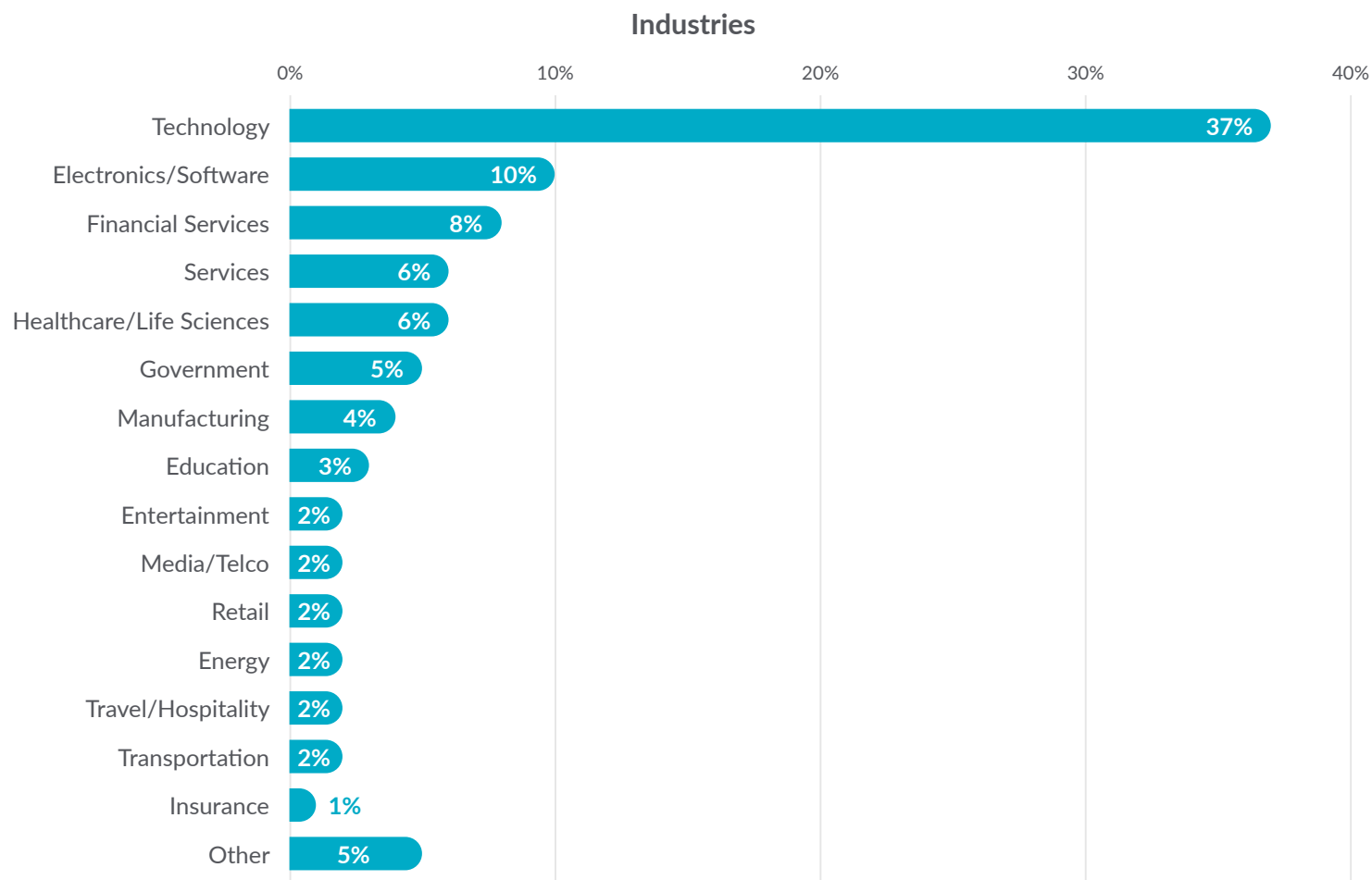
A pod remains on a node until its process is complete, the pod is deleted, the pod is evicted from the node due to lack of resources, or the node fails.



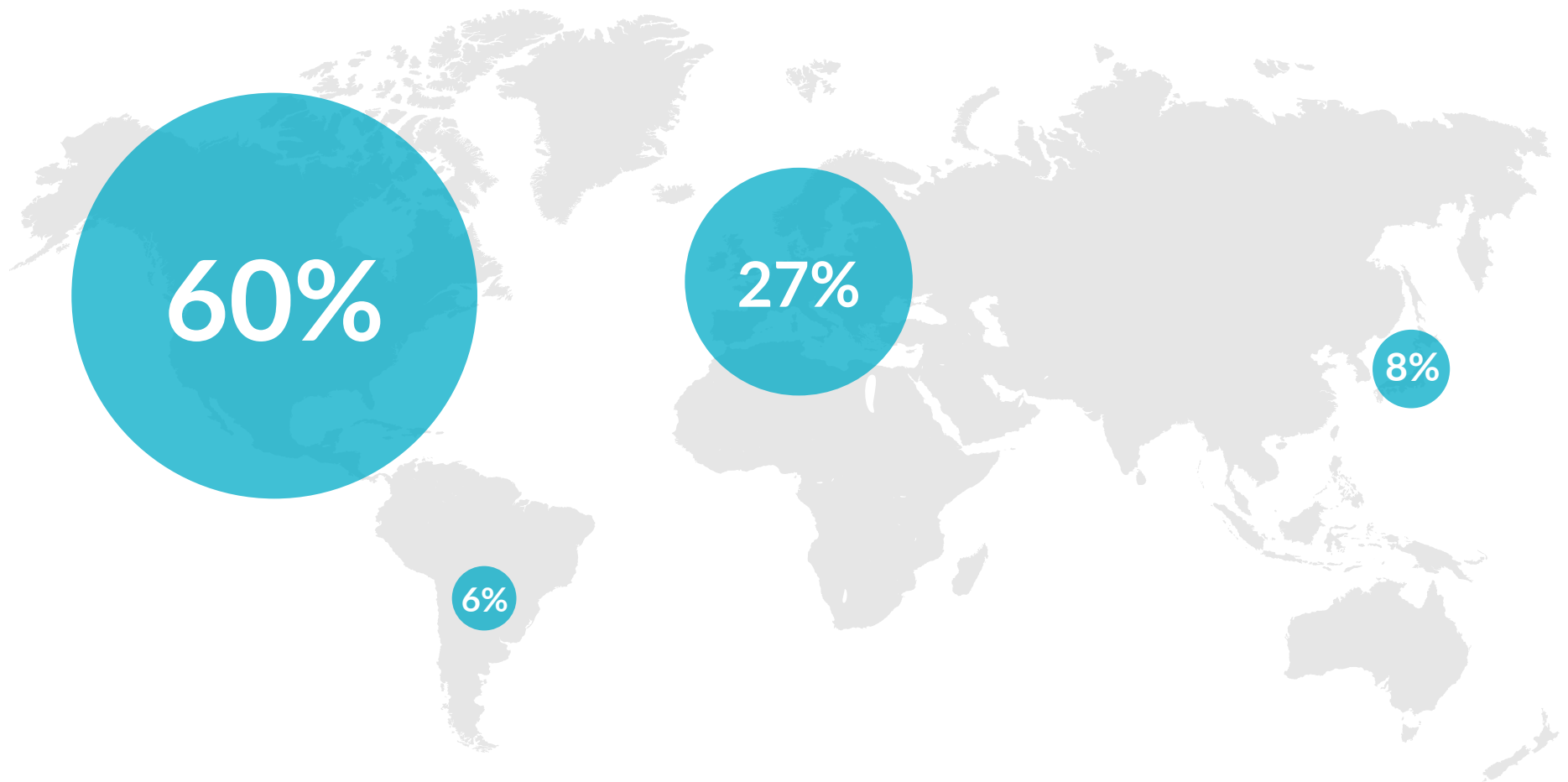
Demographics and Data Sources

The data in this report is derived from an analysis of nearly 2 millions containers, a subset of the containers our customers are running on a daily basis. We also pulled from public data sources like GitHub,

Docker Hub, and the CNCF. The data originates from container deployments across a wide range of industries and regions, with organizations ranging in size from mid-market to large enterprise.



Regions



Conclusion

Container technologies continue to expand their role in transforming how organizations deliver applications. With security in Kubernetes becoming a growing concern among DevOps teams, it is good to see that teams are implementing security during the build process. However, more work is needed to secure the containers themselves to prevent possible vulnerabilities from entering production. The key trends from our fourth annual report highlight the continued growth in container environments, and the growing dependency on open source-based solutions to run them:

- Organizations are shifting left in their Kubernetes environments by scanning images in the build phase and taking advantage of inline scanning. Vulnerabilities continue to flourish and robust security tools are needed to identify, audit, and provide validation for compliance.
- While Kubernetes remains the clear orchestrator of choice, the selection of container runtimes is shifting with containerd and CRI-O growing fast. With container density increasing, organizations should invest in Kubernetes-native tools to simplify operating at scale.
- Open source is growing as a core component in Kubernetes environments. The growth of Falco, Prometheus, and Go illustrate the desire for high-quality open source solutions to solve the problems of securing and monitoring containers that are short lived.

Thank you for reading the Sysdig 2021 Container Security and Usage Report. We look forward to following and documenting the evolution of the container market in the coming year. See you then!



Sign up for a free 30-day trial and get visibility for security, compliance, and monitoring in minutes.

LEARN MORE

www.sysdig.com