Sysdig Platform Architecture Guide

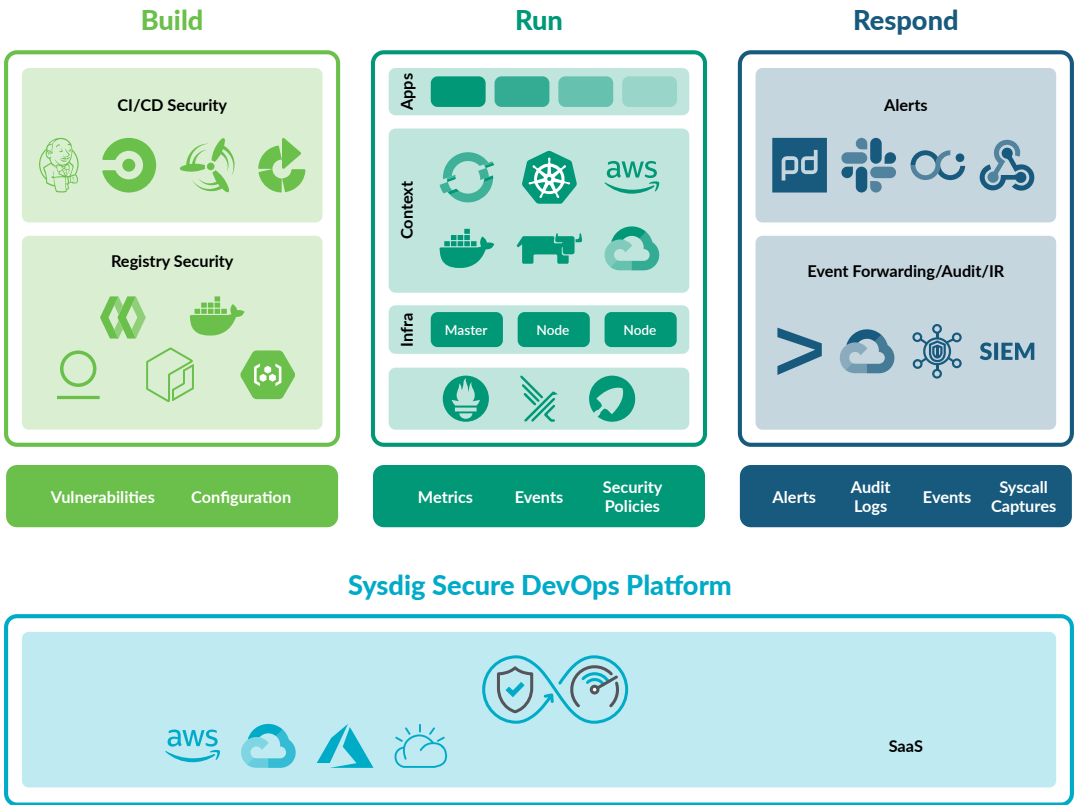
sysdig

Contents

The Sysdig Secure DevOps Platform

As cloud and containers become the standard for application deployment, DevOps practices become vital. Your cloud teams are accountable for application security, compliance, performance, and availability. Your teams need tools that support a secure DevOps workflow to accelerate deployment and confidently run cloud applications in production.

Integrated into Your Cloud-Native Ecosystem

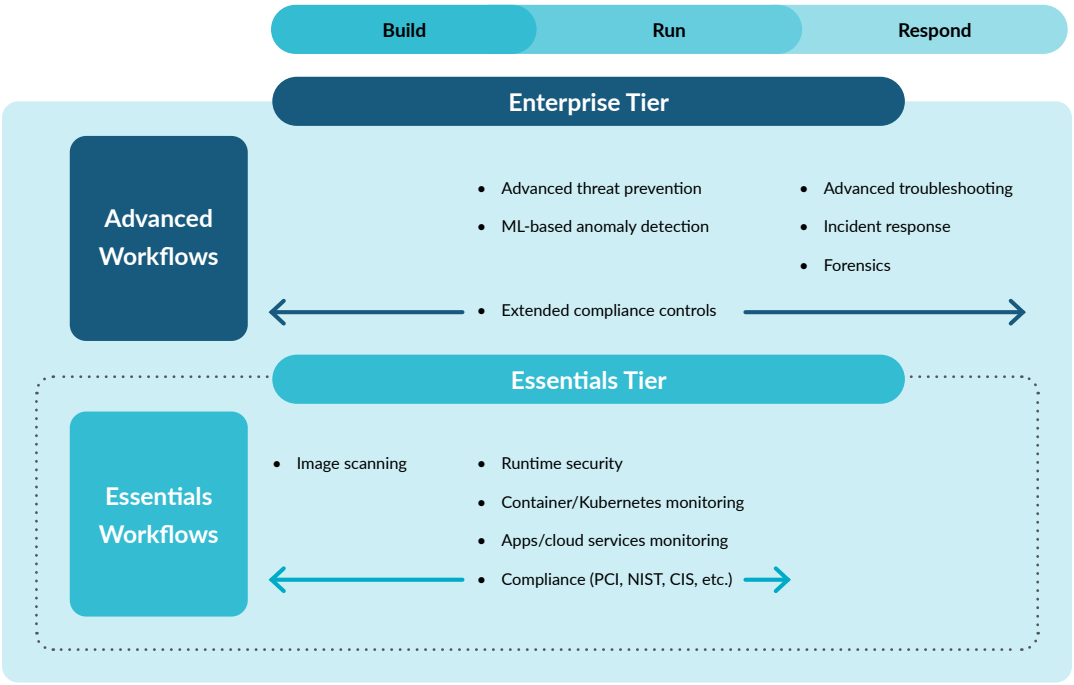


The Sysdig Secure DevOps platform embeds security, compliance, and monitoring into your DevOps workflow. It is the only unified container security and monitoring platform. With a single source of truth, Sysdig eliminates silos of information between development, DevOps, and security teams. This unified data platform enables DevOps teams to accurately triage an incident, quickly determine if the incident is caused by misconfiguration or malicious attempt, and perform forensics even after the container is gone. Your DevOps teams can report on vulnerabilities affecting running images in specific namespaces and clusters. For example, if a new CVE comes up, Sysdig Secure DevOps platform can help you quickly identify the affected images in a particular public cloud region, namespace, cluster, etc., as well as the team that owns the fix. With this approach, organizations can resolve issues

quickly by analyzing granular system data automatically correlated to cloud and Kubernetes context.

Sysdig helps you deliver reliable and secure cloud applications and addresses the essential use cases of running Kubernetes and containers in multi-cloud production environments.

Integrated into Your Cloud-Native Ecosystem



Unique ContainerVision with ServiceVision

- **ContainerVision** - Sysdig's unique instrumentation that allows you to get deep visibility into containers, network, application, and system activity without any invasive instrumentation. Access to this deep system call data accelerates both incident response and troubleshooting, helping your DevOps teams determine root cause and resolve problems faster.
- **ImageVision** - Identifies vulnerabilities and misconfigurations by automating scanning within CI/CD pipelines and registries, as well as implementing registry scanning inline. Blocks vulnerabilities pre-production and monitors for new CVEs at runtime. Helps you map a critical vulnerability back to an application and dev team.

- **CloudVision -** Enables a consolidated view of cloud activity using cloud logs. Allows you to use cloud activity logs, like AWS Cloudtrail, for run-time threat detection. Alert on changes to your AWS user permissions, S3 buckets, access keys etc. by analyzing CloudTrail logs with Falco. Enables event audits, detects events that are a threat, and raises a notification so you can quickly address those security events.
- **ServiceVision -** Provides rich context using the metadata provided from Kubernetes and your Cloud services to all of the data collected with ContainerVision. Drill-down views for dashboards, metrics, and events gives your team the flexibility to view security status across Kubernetes logical objects, such as namespaces, deployments, and pods. This allows you to quickly identify which team is responsible for resolving a vulnerability or an alert and then focus on what data is relevant to them.

Platform agnostic security and monitoring capability that easily integrates into your DevOps pipeline

- **Platform agnostic -** Consolidated monitoring, security, troubleshooting, and forensics for multiple clusters across any cloud or infrastructure ensures a consistent view.
- **DevOps pipeline integration -** Out-of-the-box integrations with the tools you use today save time across the lifecycle.
- **Common data platform -** The same data provides the foundation for both security and monitoring workflows, collected by a common agent and managed in a common back-end.

Easy to adopt and scale

- **SaaS-first delivery model -** Get up and running in minutes and avoid the hassle of managing on-premise software.
- **Guided adoption and usage -** Curated workflows and customizable dashboards deliver immediate value and accelerate adoption.
- **Scale -** Sysdig scales to support the largest cloud deployments in the world without compromising performance and stability.
- **Isolation with Teams -** The ability to define specific user groups by service, application, or infrastructure helps DevOps teams give exclusive permissions to support different user roles and isolate and secure data access.

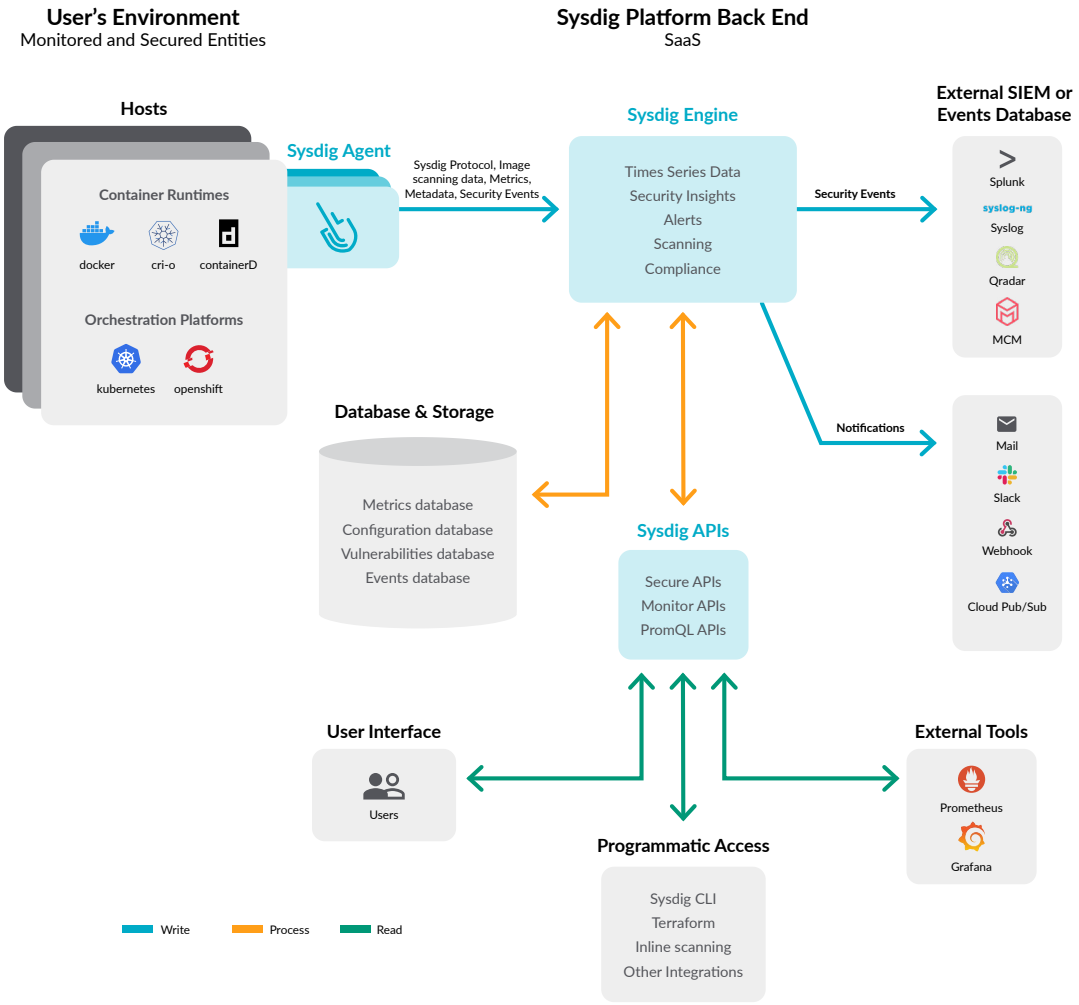
Built on open source standards for investment protection

- **Full Prometheus compatibility -** Sysdig delivers scale and long-term metric retention for Prometheus metrics. By maintaining full compatibility, including support for Prometheus Query Language (PromQL), developers can build on their investment in Prometheus and continue to use existing dashboards and scripts.
- **Falco runtime threat detection -** The Falco open source project provides the rules engine for runtime threat detection and compliance validation.
- **Sysdig open source project -** The Sysdig open source project is widely known, with over 10 million downloads, and is used by the community for troubleshooting low level issues.
- **Anchore engine -** The Anchore engine open source container image scanning tool analyzes packages and third-party libraries present in container images to find known software vulnerabilities and report on content and licenses.

This paper provides a framework for describing how, where, and why Sysdig can accelerate and ease your move to cloud-native applications and the ongoing benefits it brings to your entire software development lifecycle.

This paper will also take you deep into the architecture of our Kubernetes visibility and security platform. Not only will it help you understand how it works, but it will help you understand how Sysdig's unique approach can effectively support your cloud-native journey.

Key elements of the Sysdig Platform



The Sysdig agent

Sysdig will collect monitoring and security information from all of the target entities. To achieve this, one Sysdig agent should be deployed in each host. These hosts can be:

- The nodes that make up a Kubernetes or OpenShift cluster
- Virtual machines or bare metal
- Residing on-premises in a customer data center

The Sysdig agent can be installed as a container itself using a Helm chart, Kubernetes operator, etc. The process of installing the agent is completely automated following the steps detailed on the platform onboarding instructions.

Once the agent is installed on the host it will automatically start collecting information from

- the running containers
- container runtime
- the orchestration API (Kubernetes, OpenShift, etc.)
- metrics from defined Prometheus endpoints
- auto-detected JMX sources
- StatsD
- integrations via app checks
- the host itself using kernel headers or modules and extended Berkeley Packet Filter (eBPF)

The Sysdig agent will maintain a communication channel with the Sysdig backend that will be used to encapsulate messages containing the monitoring metrics, infrastructure metadata, and security events. The channel is protected using standard TLS encryption and transports the data using binary messages. Using this channel, the Agent can transmit data but also receive additional configuration from the backend, such as Security runtime policies or benchmarks.

Sysdig Agent Resource consumption is subjective to the size of and load on your host. At a minimum, the agent requires 2% of total CPU and 512 MiB of memory.

Sysdig backend

The Sysdig backend can be directly used in its SaaS version, thus being managed transparently by Sysdig, or it can also be installed on the customer premises. This distinction does not affect the actual operation of the platform described below.

Once the agent messages are received in the backend, they are processed and extracted into data available to the platform - time series, infrastructure and security events, and infrastructure metadata.

Sysdig backend can not only scale vertically to handle the load of individual data needs, such as capacity for metrics, but can also scale horizontally for processing and scalability, such as the needs of thousands of agents or ingestion of millions of metrics.

Main data flow for the Sysdig platform:

- Extraction and post-processing of the metric data from the agent so that full time-series, with all of the necessary infrastructure metadata, is available to the user.
- Maintenance of the infrastructure metadata (most notably Kubernetes state) so that all events and time series can be enriched and correctly grouped.
- Storage of time series and event data.
- Processing of time series data to calculate alert triggers.
- Queue security events triggered by the agents to be shown on the event feed, pushed to notification channels and forwarded via event forwarder to platforms like Splunk, Syslog or IBM MCM / Qradar.
- Aggregate and post-process other security data like container fingerprints used to generate container profiles or security benchmark results.
- Store post-processed data in Sysdig platform internal databases where it can be combined by the API service to create dashboards, event feeds, vulnerability reports, or security benchmarks.

Sysdig APIs

The Sysdig platform provides several ways to consume and present its internal data. All APIs are RESTful, HTTP JSON-based, and secured using TLS. The same APIs are used to power the Sysdig front-end, as well as any API clients such as **sdc-cli**.

These API's enable use cases like:

- User access to the platform via the Sysdig user interface.
- Programmatic input and extraction of data, for example:
  - Automatic user creation.
  - Terraform scripts to save or recover configuration state.
  - Inline scanning to push scanning results from the CI/CD pipeline.
  - Instrumentation using the sdc-cli.
  - PromQL API interface that can be used to connect any PromQL compatible solutions, such as Grafana.

The Sysdig architecture: Built on open source foundation

The Sysdig Secure DevOps Platform starts with the idea that *effectively operating containers in production is fundamentally a data problem.* Our belief from the beginning has been that if we can take a radically different approach to create visibility in complex, distributed, ephemeral container environments, we could solve a number of operational challenges over time, including monitoring, run-time security, vulnerability management, incident response, and more.

At the same time, driven first by Docker and now by Kubernetes, the cloud-native movement is heavily rooted in open source building blocks. This community effort allows enterprises to get started with little to no cost and provides a massive community that's constantly improving key projects. Sysdig is no exception.

Sysdig believes that open source tools are an essential part of the cloud-native ecosystem. Developers shouldn't be required to pay up-front to experiment or use tooling as they figure out their cloud-native strategy. Instead, they should have access to key building blocks through open source.

Sysdig is based on the open-source Linux troubleshooting and forensics project by the same name *(sysdig)*. The open-source project allows you to see every single system call down to process, arguments, payload, and connection on a single host. As we'll describe, this data is dynamically mapped to containers, microservices, clouds, and orchestrators in a way that is at once powerful and simple to use in the commercially supported Sysdig platform.

Instrumentation needs to be transparent

In static or virtual environments, it was simple to run an agent on a host and to configure the agent based on the relevant applications. In container environments, however, this approach doesn't work:

- You can't place an agent within each container without destroying a key value of containers : simplicity.
- With applications and containers coming and going, you can't manually configure agent plug-ins to collect relevant app-level metrics. And you certainly cannot depend on manual configuration for per-service security policies.

With these new demands, the act of instrumenting must be as transparent as possible, requiring as little human intervention as possible. Events or actions on the target system, infrastructure metrics, application metrics, service response times, custom metrics, and resource/network utilization should be ingested without any effort from within the container. It certainly shouldn't require efforts with the spin-up of each additional container.

There are two possible approaches to achieving this. First are pods, a concept created by Kubernetes. A pod is a group of containers that share a common

namespace. As a consequence, a container inside a pod can see what the other containers in the same pod are doing. For instrumentation agents, this is often referred to as a "sidecar" container.

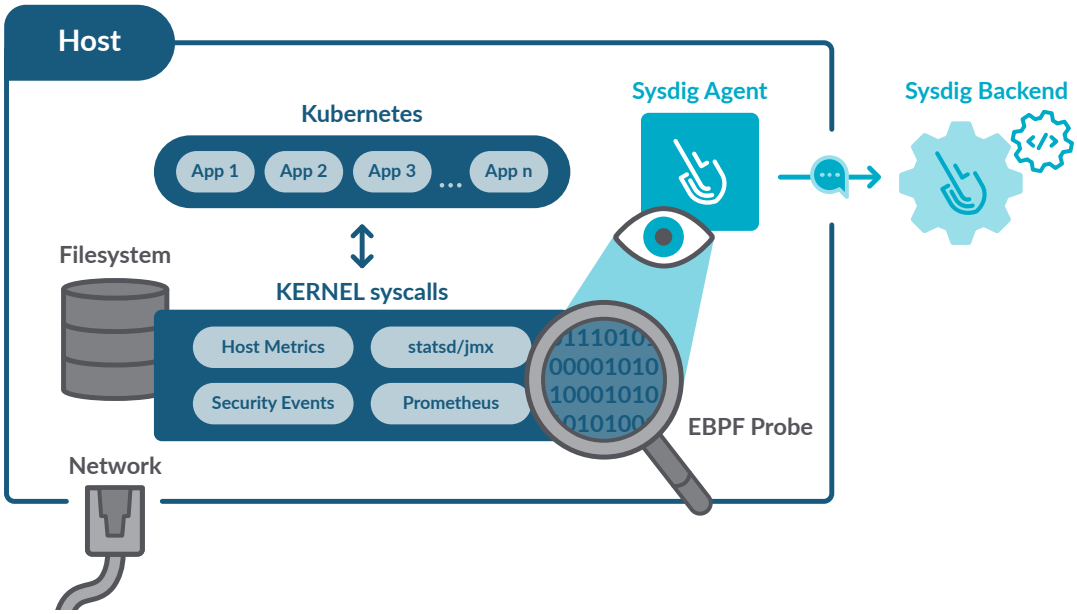
The positive here is that this is something relatively easy to do in Kubernetes. The downsides, however, are worrisome:

- Resource consumption can be high if you have many pods on a machine - it's a bit like having a monitoring agent per process(!).
- You create dependencies as well as additional attack surfaces within that pod. That means that if your monitoring sidecar has performance, stability, or security issues, it can wreak havoc on your applications.
- Another variant of the side-car method the ld_preload method where binaries are injected into a container, much like auto installing an agent into every container. This has the downsides of both agent based and sidecar based instrumentation, as well as often causing stability issues due to uncontrolled / untested changes of a container.

The second model is per-host, transparent instrumentation. This transparent instrumentation captures all applications, containers, custom metrics (Prometheus, StatsD, and JMX) and host metrics with a single instrumentation point and sends it to a container per host for processing and transfer. This eliminates the need to turn everything into a custom metric, something we've seen many people resort to.

Unlike sidecar models, per-host agents drastically reduce resource consumption of monitoring agents and require no modification to application code. It does, however, require a privileged container and a kernel module.

Sysdig chose to do the latter. We call this "ContainerVision," our shorthand for this technical approach. You'll see here that choosing this model was one of the most important design decisions we made.



Despite the greater complexity this represented in building the Sysdig agent, we believed this approach would provide us with several advantages:

- To unify many of the common operational activities necessary for production systems like security, monitoring, troubleshooting, forensics.
- To collect more data with lower resource overhead.
- To reduce operational overhead for small, agile teams.

We were also confident that our instrumentation approach represents a *reduced threat to environments versus complex networking and a high density of agents in your environment. To reduce concerns, we open-sourced our kernel module as part of the Sysdig Linux and container visibility command-line tool.*

ContainerVision in-depth

ContainerVision is a core element of what makes Sysdig's approach different. Our architecture is very similar to that of tcpdump and Wireshark. This is not by chance - Sysdig was created by one of the co-creators of Wireshark. First, events are captured in the kernel by a small driver, called sysdig-probe, which leverages a kernel facility called tracepoints.

Tracepoints make it possible to install a "handler" that is called from specific functions in the kernel.

Currently, Sysdig registers tracepoints for system calls on enter and exit, as well as for process scheduling events. Sysdig-probe's handler is limited to copying the event details into a shared read-only ring buffer, encoded for later consumption. The reason to keep the handler simple, as you can imagine, is performance, since the original kernel execution is "frozen" until the handler returns. The freeze is on the order of nanoseconds, that's all the driver does. The rest of the magic happens at the user level.

The event buffer is memory-mapped into user space so that it can be accessed without any copy, minimizing CPU usage and cache misses.

Two libraries, libscap and libsinsp, then offer support for reading, decoding, and parsing events. Specifically, libscap offers trace file management functionality, while libsinsp includes sophisticated state tracking functionality (e.g., you can use a file name instead of an FD number) and also filtering, event decoding, a Lua JIT compiler to run plug-ins, and much more.

Finally, open-source Sysdig tops it off as a simple wrapper around these libraries, while the agent for the commercial platform uses the same core but simultaneously (a) forwards data to the backend and (b) enforces security policies.

Sysdig also now supports eBPF as an alternative to our kernel module-based architecture described above. eBPF – extended Berkeley Packet Filter – is a Linux-native in-kernel virtual machine that enables secure, low-overhead tracing for application performance and event observability and analysis.

There are several motivations for tying ContainerVision into eBPF. One is simply to take advantage of maturing technology that is already a part of the base operating system. This makes management of observability that much easier and frictionless. Another reason why eBPF makes sense is the advent of container-optimized operating systems. These solutions, like Container-Optimized OS (COS) from Google Cloud Platform or AWS Bottlerocket, feature an immutable infrastructure approach that disallows kernel modules altogether. By tapping into eBPF, users of these newer OS approaches achieve the same level of container observability Sysdig has delivered for some time with our kernel module.

ContainerVision is a key advantage of Sysdig's architecture. The tracing overhead of Sysdig's instrumentation is very predictable and makes it ideal for running in production environments. To go a little deeper on this, read **DTrace vs strace vs sysdig: A technical discussion.**

The Sysdig open source troubleshooting tool gives you both a command-line interface and a curses-based interface to all of the rich data collected with ContainerVision for a single host.

For our monitoring application, we use these same system calls to extract the relevant metrics up and down your stack, and create an htop-like interface across your distributed environment. And our security application uses the same agent, same data, and same backend, but focuses on events that represent security violations.

Our belief, however, is that performance metrics or security events alone are not enough once you're dealing with containers. Next, let's see if we can make sense of it all.

ServiceVision adds rich context to ContainerVision

In a container and microservices environment, Linux system calls are a rich source of data about containers, Kubernetes, hosts, and application health. By tapping into system kernel traces, Sysdig agent has the ability to see every process, every container, every service, every action, and every metric in a uniquely multi-dimensional way without compromising cardinality or accuracy. The Sysdig agent can auto-detect and auto-ingest additional data sources such as Prometheus, statsd, container, or orchestrator events. This rich data source provides unique insights into health and security of containerized applications through their full lifecycle from development to production. Thus, it protects your containerized applications against malicious players as well as vulnerabilities created by common misconfigurations.

Granular syscall data is dynamically mapped to containers, microservices, clouds, and orchestrators in a way that is immediately powerful and simple to use in the commercially supported Sysdig platform. A rich set of great metrics are useless if they aren't relevant. With **ServiceVision**, your granular data is mapped to your container orchestrator to understand how you've built your systems. The core advantage here is that you don't waste time designing groupings or laying-out your

services, you've already done this within Kubernetes. You'll have familiar keys / names to reference when looking for data or views. This unique ability to quickly zero in to the few relevant metrics from the millions is one of the key reasons customers choose Sysdig to help mange the complexity of Kubernetes at scale.

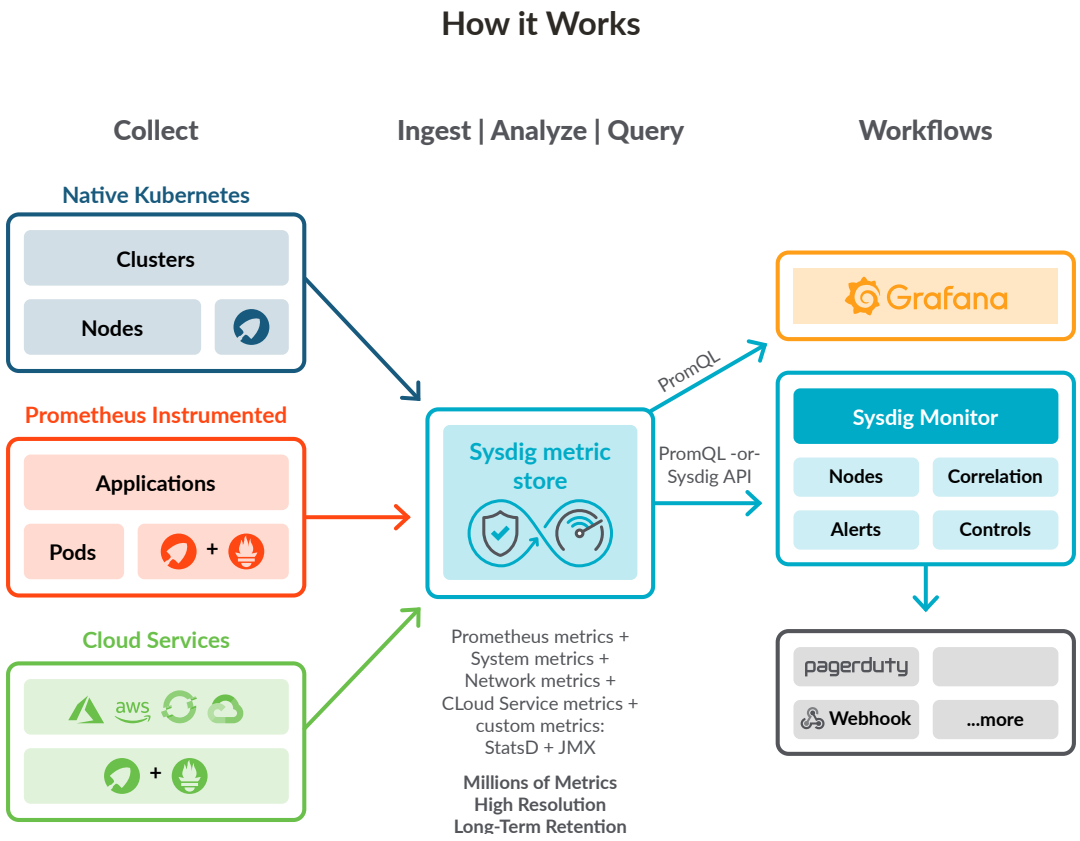
The ability to pull this metadata and apply it to your view can be a really simple way to make your data relevant to different users. Instead of forcing users to wade through thousands, possibly even millions of different objects (security events, containers, metrics, etc.), we allow you to focus and present only what is relevant to them. This saves time, provides easy snapshots of what is healthy, and accelerates root cause analysis (and as such, time-to-fix!). We call this ability to collect and apply metadata, ServiceVision.

Kubernetes Native Monitoring & Security

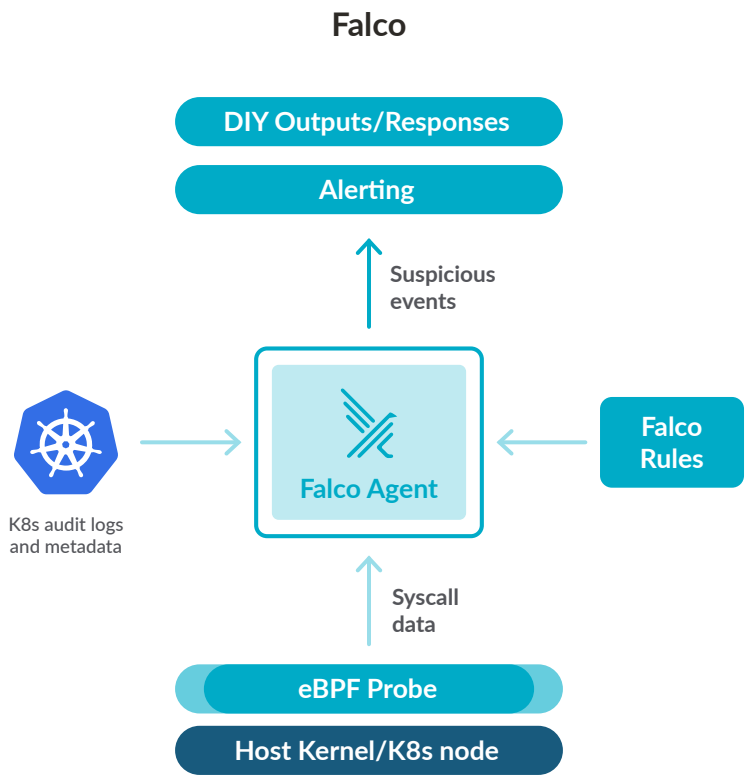
ServiceVision highlighted earlier is core to a lot of the Kubernetes integration. You don't want to have to manually create groupings and individually apply policies to each one. With Sysdig, you can use what you've already built and apply policies and rules in a flexible way that adapts to both what you've deployed today, as well as what you will deploy tomorrow. Tag a policy as 'production' and any new workload using this key will automatically inherit it. Protect a production cluster, and everything that happens in that cluster in the future will also be protected.

Application and Cloud Services Monitoring

Prometheus monitoring is another key building block in the open-source community, providing a metrics format and a method to scrape metrics from applications. While Sysdig didn't create Prometheus, we actively contribute to the project and have incorporated its capabilities into our platform. For example, you can use standard PromQL queries to view and analyze your metrics stored in Sysdig. Sysdig has created promcat.io, a website for the Prometheus community where users can find application monitoring resources for Prometheus, ready to use in Sysdig platform, or with other open source tools. These resources have been collected, actively tested, and maintained by a dedicated team of engineers.



To further leverage the unique visibility created by our original project, we built an open-source security tool called Falco. Falco combines the visibility of open source *sysdig* with a rules engine that constantly monitors system events for violations of policies at run-time. Our enterprise offering then allows for the enforcement of these policies, compliance, and auditing on top of this rich data.



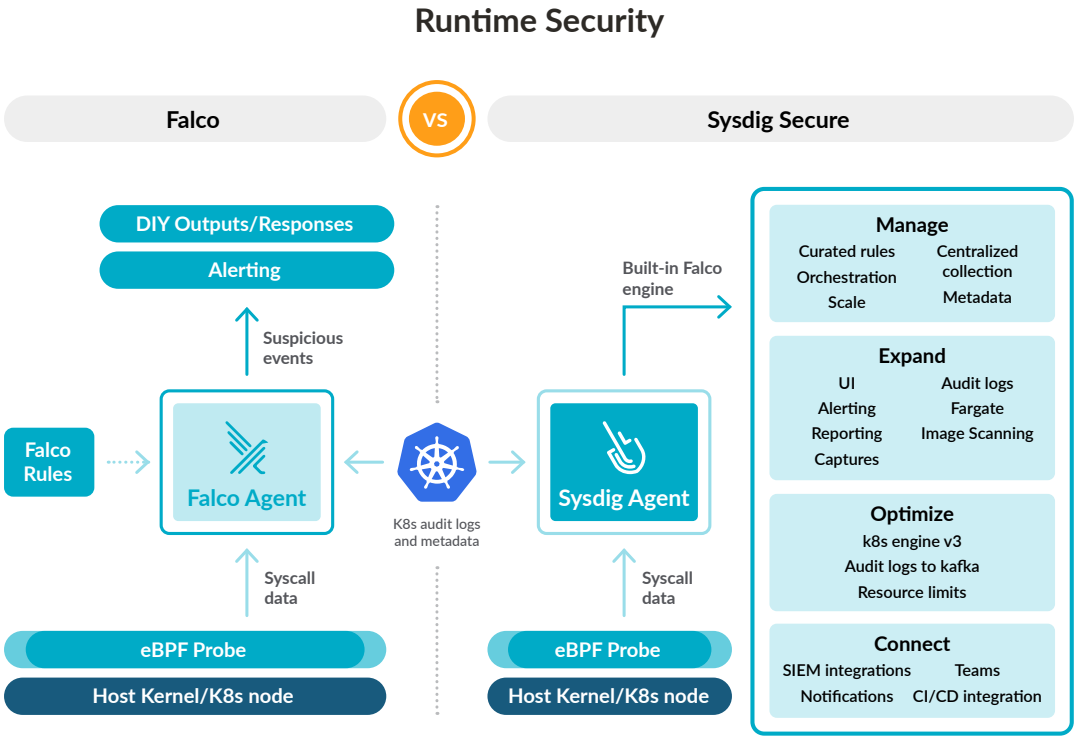
Falco for Runtime Threat Detection

Falco is becoming the standard CNCF project for runtime security detection and is already widely adopted in thousands of clusters. It has an extensive community that not only helps develop and improve the product, but also helps crowdsource new rules and continually validate existing ones. The Falco community is encouraged by the Sysdig open source team (who initially developed Falco), and continues to work with the community and CNCF to drive the future of the tool.

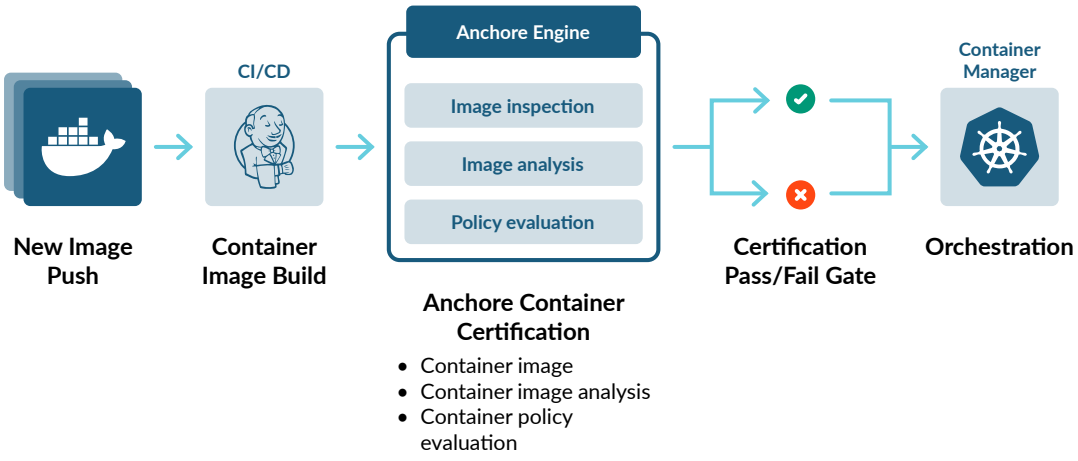
Falco's data sources are both low level system calls (i.e., being able to detect SSH connections at the host or container level, detecting a database server making outbound network connections, a message-bus forking processes and running command line scripts, etc.) as well as Kubernetes audit API (i.e., a user launches a privileged container, a configmap containing clear text credentials, sensitive host filesystem areas are mapped to a container, etc.).

Falco is embedded into the Sysdig Secure DevOps platform, providing runtime security visibility. Within the platform, the open source functionality is expanded with a simplified UI, easy rules editor, and rules tagging to quickly filter relevant

rules. It is also extended with the Sysdig native Kubernetes integrations, allowing quick and flexible application of rules to different Kubernetes objects. Falco rules can detect anomalous activities and flag compliance issues. Sysdig provides full commercial support for Falco rules used within the Secure DevOps platform.

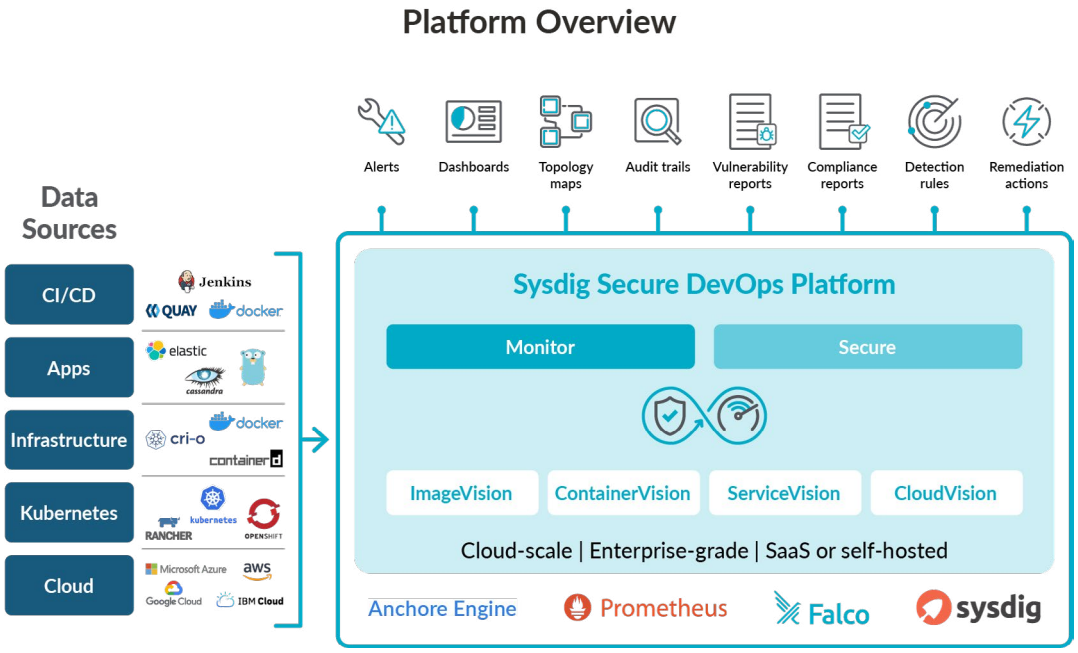


To further enrich the data used to secure your environment, Sysdig has also integrated Anchore into the platform. What Falco does for run-time, Anchore does for build-time; it allows you to implement and enforce vulnerability management policies and scan your container images before they ever go into production.



The Sysdig Secure DevOps Platform also integrates with VulnDB, a comprehensive and actionable source of vulnerability intelligence, to provide richer findings around vulnerabilities in third-party libraries and dependencies. Combined with the wide range of vulnerability databases the Sysdig platform checks against, the comprehensive data from VulnDB enables you to more effectively identify, track, and reduce security risk.

Sysdig's commercial offering unifies all of your operational data and turns it into insight. Starting with thousands of metrics and events for every application, container, and host, the Sysdig platform then enriches the data to give you precise, in-context, views of your applications and microservices. Sysdig then provides you with apps that deliver key visualizations to help you achieve your specific workflows.



The **Sysdig Secure DevOps platform** provides a unified view of the risk, health, and performance of your entire environment. It is designed to intelligently surface the services and components in your environment with the most issues and highest severity.

- **Sysdig Monitor** delivers performance and health monitoring with deep telemetry data collected from your containers, applications, orchestration, and clouds.
- **Sysdig Secure** enables vulnerability management, compliance and security policy enforcement, auditing, and run-time protection.

All of this capability is provided with unified host instrumentation, a unified data backend, and powerfully simple user interfaces that allow your DevOps, DevSecOps, service owner, and developer teams to take advantage of all of the available rich data. With this as a baseline, let's dig in further so you can understand how Sysdig impacts and supports customer use cases across each facet of the container lifecycle.

Development - Build:

- Developers must ensure they push vulnerability-free applications and are responsible for security scanning early in the CI/CD pipeline.
- Sysdig helps detect vulnerabilities and misconfigurations with a single workflow.

Operations - Run:

- Sysdig also provides the ability to report on vulnerabilities affecting running images in specific namespaces, clusters, and more. For example, if a new CVE comes up, Sysdig can help you quickly identify the affected images in a particular AWS region, namespace, cluster, etc., as well as the team that owns the fix.
- Sysdig helps you prevent and detect threats at runtime without impacting performance.
- Sysdig also allows you to monitor cloud applications, infrastructure to maximize availability, and performance.

Incident Response and Troubleshooting - Respond:

- Your DevOps team needs to accurately triage an incident and quickly determine if it is a misconfiguration or a malicious attempt.
- Sysdig allows you to do incident response and troubleshooting, and conduct forensics even after the container is gone.

Validate compliance end to end:

- Verify configuration meets CIS best practices such as containers running as root, and ensure ssh is not run within containers.
- Secure application compliance with NIST, PCI, SOC2, etc.
- Use falco rules to detect anomalous activity and flag compliance issues.

Along with these areas, we'll next dig further into the architecture so you can see each piece in more detail.

Build: Accelerating secure development

Businesses need to embed security earlier in the development lifecycle. Given how containers will change your development process, proactive security measures within the development of containerized and microservice-based applications move from a desire to a necessity. And given the dramatic acceleration of code delivery, it's no surprise that developers expect these additional requirements to be met with enhanced resources and tools to give them visibility into the performance and security posture of their code before it goes into production. Sysdig is a critical tool for allowing development teams to deliver more reliable, secure code more efficiently.

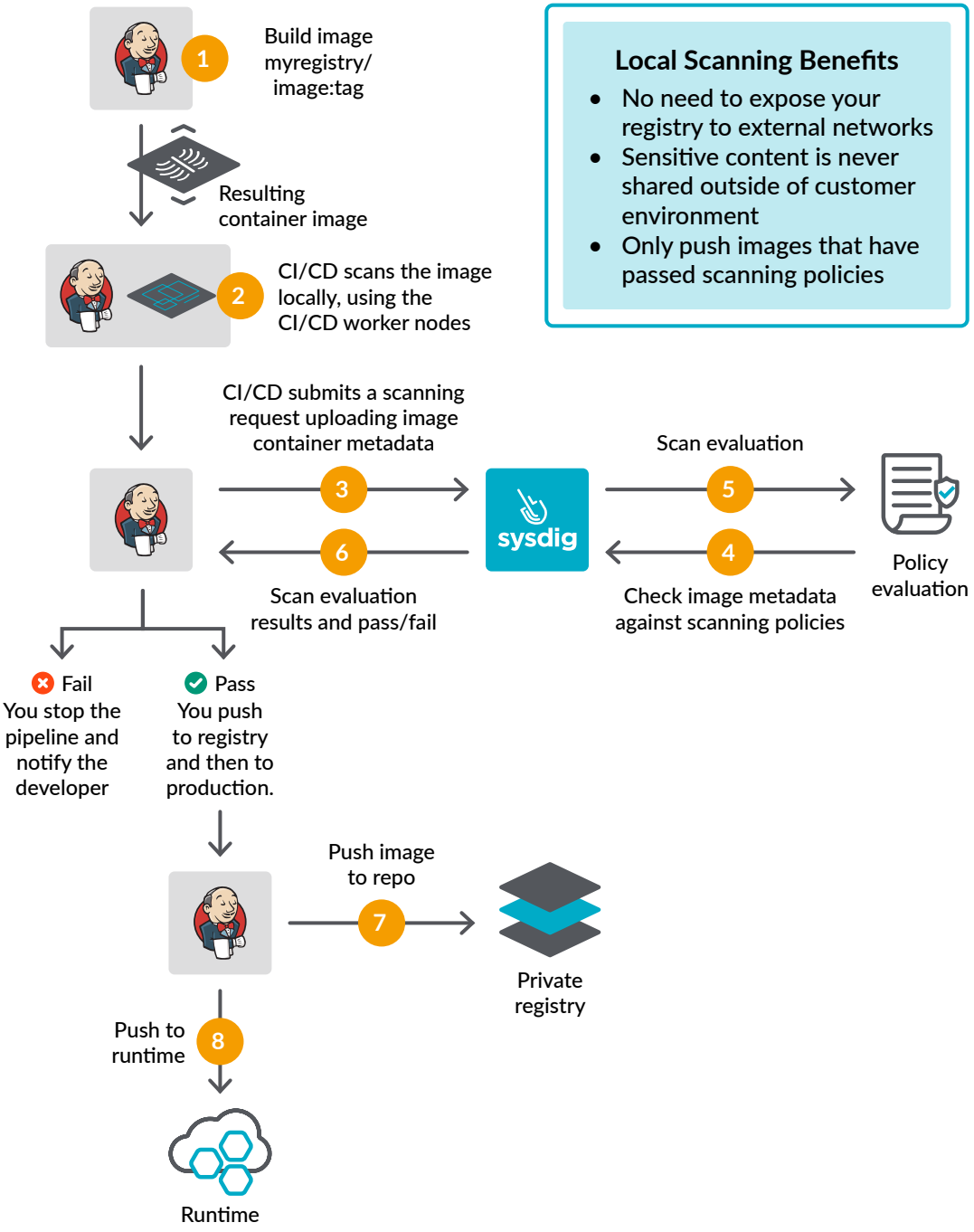
Shifting Security Left: Introducing image scanning and vulnerability management into the development process.

Sysdig Secure's vulnerability management capabilities help organizations bring application security, compliance, and quality closer to the developer. Through native integrations with common tooling in the software delivery chain, Sysdig Secure enables teams to scan for, block, and remediate security issues before a build is completed or a container is ever deployed. Sysdig Secure starts with image scanning to perform an inspection of an image and generate a detailed analysis of the contents of the image, including:

- Official OS packages
- Unofficial OS packages
- Configuration files
- Language Modules – NPM, PiP, GEM, and Java Archives
- Image metadata and more

Sysdig Secure then automatically correlates the contents of the image with vulnerability feeds to give insight into known vulnerable packages, and files. Our vulnerability feeds are continuously updated vulnerability and package data from OS vendors, package repositories, VulnDB, and the National Vulnerability Database (NVD).

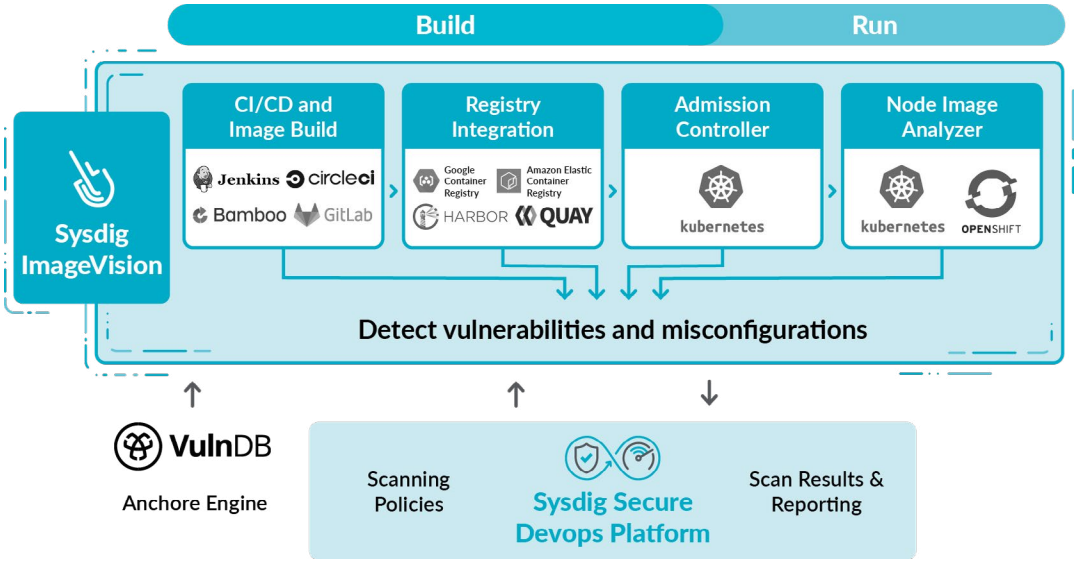
Inline Scanning



Local Scanning Benefits

- No need to expose your registry to external networks
- Sensitive content is never shared outside of customer environment
- Only push images that have passed scanning policies

Integrate Image Scanning and Governance Throughout the Entire Image Lifecycle



You can easily configure Sysdig Secure to automatically scan images as part of your build process through either a native Jenkins plugin or APIs. You can fail builds, trigger warnings, and enforce compliance easily by including image scanning every time a container goes through your build process.

Sysdig Secure scans images stored in any Docker V2 compatible registry such as CoreOS Quay, Amazon ECR, Docker Private Registries, Google Container Registry, JFrog Artifactory, Microsoft ACR, SuSE Portus, and VMware Harbor.

The solution makes it easy for you to configure policies for your build pipeline or your registries to evaluate images against user-defined policies for vulnerabilities, operating system packages, third-party packages, software libraries, Dockerfile checks, file contents, configuration files, and image attributes.

All of these procedures are tied to alerting. If unscanned images are deployed into production environments, if a new vulnerability is discovered in a package of an image that's running in production, or if the scan status of one of your running images changes, you will be proactively notified.

For all run-time vulnerability management, Sysdig ties back information about unscanned images or scan results to Kubernetes clusters, namespaces, and deployments to categorize risk and prioritize image patching and upgrades. This unique ability to isolate issues by Kubernetes metadata allows you to put your efforts in the right place at the right time.

Software performance during development and test

Even if your latest software is vulnerability free, do you know how well it performs? After all, a development team that only delivers secure software isn't doing its job - they need to deliver high performing, reliable, secure software that makes your company more competitive and your customers happier.

This process of performance testing (or the reverse view of regression testing) is well understood by developers: build software, run it through a battery of tests, and compare its performance to alternative or previous versions of the code. Typical questions a developer asks include:

- What's the response time of my service?
- Do common activities consistently cause any errors?
- What is the underlying resource utilization (CPU, Memory, Disk) of the code compared to previous versions?
- What are the slowest API endpoints of my service? Not surprisingly, these are the same kinds of questions operators might monitor for in production, and the process to capture this information looks very similar. Instead of repeating it all here, let's talk about monitoring software in production, and you will quickly see how the same techniques can be used in dev/test.

Run: Running containers in production
Monitor, detect, enforce, and comply

Monitoring an enterprise production environment at scale reveals the complexity of the data challenge with operating containers in production.

- Mapping your data to your applications, hosts, and containers.
- Leveraging orchestrators.
- Deciding what data to store.

How to relate monitoring and security data to your applications, hosts, containers, and orchestrators with Sysdig ServiceVision

As your environment increases in complexity, the ability to filter, segment, and group metrics and policy violations based on metadata are essential. Tags allow you to represent the logical blueprint of your services and application architecture in addition to the physical reality of where containers are running. For example, you may want to look at your data by dev/prod, by service, by pod, by container, by cluster, by data center, or by host.

As you explore your data, you should be able to dynamically select labels and tags on metrics to produce just the right view you need for the concern or issue at hand. By scoping and grouping these tags, you can view the performance of a service at large, or drill down into a deployment or even a container. The dynamic selection gives different users and roles within your organization the ability to quickly visualize data and answer questions about their part of the application stack.

There are two ways to think about tagging metrics:

- Explicit – attributes you'd like to annotate and store.
- Implicit – orchestrator tags like Kubernetes descriptors – namespace, pod, etc.

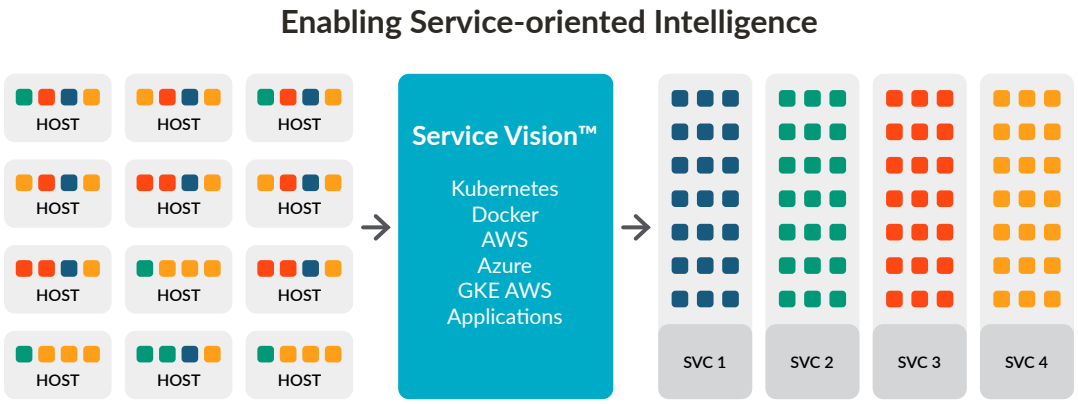
You should have a mechanism and a best practice for the use of explicit tags so that anyone on your team can add them as needed – and – implicit tags should be captured by default.

Orchestrators radically change the scheduling management approach for containers, and impact users' monitoring strategy along the way. Whether it's Kubernetes, DC/OS, Mesos, or Nomad, you'll see a similar change to the required monitoring approach. Individual containers become less important, *while the performance of service becomes more important*. The service is made up of potentially many containers, and more importantly, the orchestrator can move those containers as needed to meet performance and health requirements.

Being able to dynamically and automatically understand what an orchestrator is doing to your application is so critical, we built a capability called ServiceVision. It

allows Sysdig to manage all of this tagging without input from the user. This, in turn, allows you to understand the performance of your system regardless of how distributed or dynamic it is.

Service context make metrics relevant and actionable



As our platform evolved, we found that we typically discover and add 12 to 25 tags to any given metric or event by *default*. Power users of our product may have significantly more. Think of each unique combination of tags as a separate line-item that you need to store, process, and then recall on-demand to extract a precise view.

- Your security, monitoring, and forensics systems must implicitly tag all metrics and events according to the metadata of your orchestrator.
- This tagging applies to events, user commands, and inputs, as well as system metrics, container metrics, application component metrics, and even custom metrics.

Custom metrics bear repeating: Whether these are Prometheus, StatsD, JMX, or Golang expvar, your developers should be able to simply output the custom metric and the monitoring system should keep state regarding where the metric is from.

Here's what we recommend steering clear of challenges with custom metrics and tagging:

- Your agents should be able to discover the application components running on a host, in a container, without any manual input.
- Given that an orchestrator can move containers at any point in time, the burden falls to your monitoring system to auto-discover what's running and then collect the correct metrics.
- Auto-discovery might require writing a custom "check" for your custom code. You should be able to do this once, but it should never have to be repeated or manually told when to run.

For more on monitoring Kubernetes and orchestrators, check out our Kubernetes Monitoring Guide

Alerting and Security Policy Enforcement. Operational systems should be designed to simplify operators' lives. One of the most direct ways of doing that is to make sure the system can:

- Find issues.
- Alert when rules are broken or performance exceeds thresholds.
- Take automated actions to stop or mitigate issues where appropriate.

Sysdig has built these capabilities into the Secure DevOps Platform. All of these have been built with ContainerVision and ServiceVision in mind. That means Sysdig sees your hosts, containers, and applications, but can also automatically manage actions across services and not just individual containers. This in itself provides more value to the operator; you can manage fewer actions and be confident that they will adapt across your dynamic, ephemeral container environment.

- **Metric Alerts** can be triggered off of any metric in the system. Metric thresholds can be manually set across sums, averages, and rates. You can set multiple, required conditions using boolean logic to drive alerts. Alerts can then be sent to a range of downstream tools such as Slack, PagerDuty, Email, and more.
- **Event Alerts** use discrete system events as opposed to calculations based on metrics. Examples include a Kubernetes CrashLoopBackOff, Docker Kill, or a user spawning a shell inside a container. You trigger an alert after a specific count of events.
- **Anomaly detection** is another form of alerting which, instead of using manual thresholds, relies on Sysdig algorithms to determine normal behavior and when these bounds have been exceeded. Sysdig can both detect anomalies against historical metric patterns and an outlier within a group (e.g., a group of hosts).
- **Action triggers** can be tied in to any alert. Typically using a webhook, the operator can trigger a scheduler to modify a deployment, run a script, or take almost any other action.
- **Policy enforcement** takes a slightly different approach to actions. In the event that any sensitive security policy is violated during run-time, the system can pause or kill the container in question, so as to prevent further malicious behavior or intrusion attempts. This is in addition to the build-time policy enforcement we discussed earlier.
- **Captures** allow deep forensics and troubleshooting in the event of an incident. The value of captures is discussed in detail in the next section. Captures can be triggered by alerts.

Being Kubernetes native goes beyond just understanding the topology of a workload, it also means leveraging the tools and controls that are already provided. You don't need yet another firewall, or another method for preventing unauthorized container usage. Sysdig believes that using the right tools for the right job is very important, so we embrace the tools already provided within Kubernetes. This

allows for very efficient roll-out of security controls, simplified troubleshooting, and minimal performance overhead. Sysdig helps you to leverage the tools already available within Kubernetes, such as:

- **Admission Controllers -** Prevent pods from even starting if they use container images that breach specific policies.
- **Pod Security Policy -** Create native security controls around your workloads, either by analyzing them or by modelling the impact against them.
- **Kubernetes Audit API -** Understand and analyze what your users are doing, what workloads they're deploying, and what actions they're taking.
- **Kube-state-metrics -** Kubernetes already collects hundreds of data points about its own health.

We decided that it wasn't reasonable to deploy smaller, isolated backends per-service or per-application. We instead made a design decision to build a horizontally scalable approach to metric and data storage.

To us, prescribing multiple backends didn't seem like a manageable approach, neither for us to run in a cloud environment or for a customer to manage in an on-premise deployment of our software. If you're wondering why this would be a consideration at all, you'll see some open source monitoring projects default to the isolated backend model, and thereby push concerns about scalability to the user resulting in significantly more management work. Instead, we wanted to build a horizontally scalable backend, with the ability for our application to then isolate data, dashboards, alerts, and more based on a user or service.

To offer retention that wasn't time-bound, we decided to roll up data over time. We store full-resolution data for six hours, and then begin aggregating data after that.

Enterprise Grade Scale

While our backend continues to evolve, today it consists of horizontally scalable clusters of Cassandra (metrics), ElasticSearch (events), MySQL (configuration data), Apache Kafka (audit logs), and Redis (intra-service brokering). Building on these components gives high reliability and scale to store years of data for long-term trending and analysis.

Enterprise scale is one of the key factors driving Sysdig adoption. Sysdig scales to support the largest cloud deployments in the world without compromising performance and stability. Sysdig platform uses an orchestrator mechanism to scale individual components and has been load tested on thousands of production clusters. It can not only scale deep to handle the load of individual data needs, such as capacity for metrics; but can also scale wide for processing and scalability, such as the needs of thousands of agents or ingestion of millions of metrics.

All of the data that we collect and store is accessible by a REST API. This scalable backend is used for Sysdig's cloud service, but can also be deployed by any enterprise as software that they operate in a private cloud for greater security and isolation. Because of our design choices in addition to avoiding the need to have to run multiple systems for monitoring and others for security, long term analysis, data retention, or compliance, if you're part of a fast-growing enterprise, you can scale and grow the system along with your business.

In addition, we want to make it simple; Sysdig comes as standard with many out-of-the-box health dashboards, alerts, and security policies. Typically, our customers start seeing benefit from Sysdig visibility within minutes of deploying, and spend minimal time making tweaks to get their views and policies personalized.

Respond: Reducing Mean Time to Respond (MTTR)

Troubleshooting and forensics in containerized environments

Containers are designed to be small, lightweight, and distributed. This is all fantastic for deployability and repeatability, but impacts your ability to gain visibility in the event of a performance issue or security event.

Remember our old friends ssh, top, ps, ifconfig, and the like? You will likely not have them in your containers. If you're operating in a controlled PaaS environment, you may not have access to those tools even if they are available. And... did we mention the container may not even exist anymore? If the orchestrator is doing its job, an affected container is probably long gone before you get to troubleshooting it. There's no going back to that developer to ask for another tcpdump from the host.

In short - it's going to be complicated to get the information you need. And, on top of that, having the appropriate context from the orchestrator for troubleshooting will be essential. Thus, it's essential to enable your developers to be able to get this in-depth information, ideally without polluting your production environment. We had to address this issue because we decided that simplifying troubleshooting was just as important as enabling monitoring for container workloads.

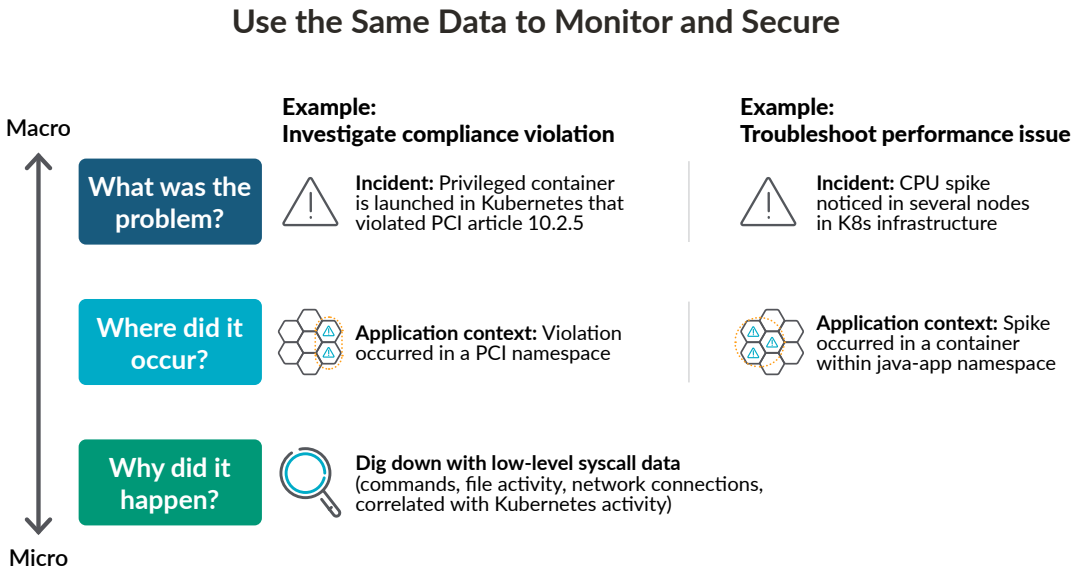
This is where Sysdig's container troubleshooting capabilities come into play. The ability to capture every single system call on a host gives you deep visibility into how an application, container, host, and the network were performing at the time of an issue or event. Sysdig capture provides the mechanism for in-depth incident response and forensics. By capturing system call information from the host during and even before an event, the capture provides a persistent copy of activity for the ephemeral workloads of containers. A container may have long been destroyed, but a copy of what it was doing, what it interacted with, what errors it generated, and much more is stored safely within the Sysdig Secure DevOps platform for later retrieval and analysis.

Sysdig's ability to record all system call activity into a standalone file means that you can capture data from production but troubleshoot on your laptop. And you can do this long after the containers are gone, letting you perform a proper post-mortem when your hair is no longer on fire.

Our automatic integration and communication with your orchestration master means that we collect relevant metadata in real-time to capture the context and the state of your distributed system – not just the state of an individual machine.

Sysdig captures can be manually created for ad-hoc troubleshooting, but more importantly can be fully automated based on custom events triggers, such as performance thresholds, network activity, security events and more. The creation of these can also be integrated with SIEM platforms to allow more complex incident

response workflows which may traverse several systems and applications. This helps accelerate root cause analysis of application troubleshooting, as well as forensic analysis of security incidents



For example, imagine you get an alert because you are seeing that databases in a particular service are spawning outbound connections. That alert within Sysdig can trigger a capture, recording all system calls for a needed time on that host. These captures will contain data from before the violation in addition to after. Exploring that in Sysdig Inspect, you can get the correct container context and then drill down into its network connections.

To get a sense of what you can do with Sysdig Inspect, read our Introduction to Sysdig Inspect.

Ensuring Infrastructure compliance

Enterprises need to ensure configurations across their infrastructure are compliant with standards such as CIS benchmarks – from hosts and nodes to the service configuration files inside containers.

Another value-added feature of the Sysdig platform is the ability to run and report on container and Kubernetes compliance benchmarks, like docker-bench and kube-bench, to validate configuration at every logical layer of your infrastructure. In the event of a Kubernetes and Docker CIS benchmark configuration drift (meaning things are out of compliance with best practices), Sysdig provides guided remediation tips to recommend the changes needed to maintaining container compliance, saving security professionals, and DevSecOps time when issues arise.

Auditing activity

In addition to troubleshooting activity, having an audit trail of *who did what* in your container environment is essential to meet auditors' requirements in the event that you have a breach or other infraction. Again, containers' dynamic, ephemeral behavior makes this task much harder than previous generations of infrastructure.

Sysdig has the unique ability to see deeply into all the container level activity, combine syscall audit trail with the Kubernetes audit logs and provide a timeline view of all activity in your container environment.

This auditing data can be supplied to external parties, fed into another events storage system, notification tools (e.g., Slack, Webhook, Mail, etc.), or a Security information and event management SIEM (e.g., Splunk, Syslog, Qradar, MCM, etc.).

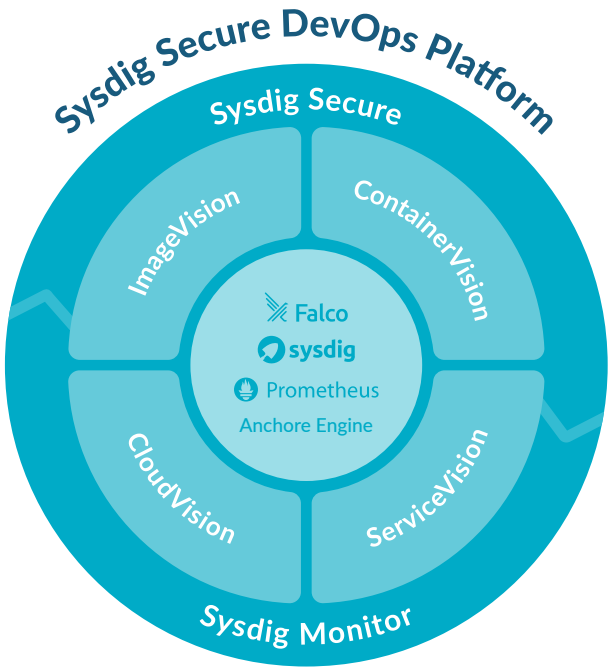
With just a few clicks, it's possible to isolate data for a bad actor. In case of a breach where the bad actor was isolated, Sysdig provides a powerful way to understand the scope of the actor's activity and instantly understand what microservices or data may have been affected.

Conclusion

Building a highly scalable, distributed data platform for visibility and security into your containers is not an easy task. A focus on getting the right data with the right context is essential to a robust system that can serve developers, operators, and security professionals across the entire container lifecycle.

Important questions to ask as you investigate tools to help you deliver secure DevOps for your business:

- How will you ensure that only the most reliable, secure code is going into production?
- How are you going to instrument containers in production for monitoring and security?
- How will you interface with orchestrators for the right context to your data?
- How will you implement security and compliance policies on your containers?
- How will you simplify the collection of application data and custom metrics?
- What data will you decide to retain?
- And will you enable troubleshooting and forensics in dynamic, ephemeral container environments?



Embed security and validate compliance

Maximize performance & availability

Get results quickly

Ship cloud apps faster by converging monitoring and security

Please check out the resources section of Sysdig.com to find more in-depth guides to assist you in your transition to containers. Your account team can support with proof of concept to meet your specific needs, or start running Kubernetes in production with confidence with your free trial of Sysdig Secure DevOps platform sysdig.com/company/free-trial

sysdig.com/company/free-trial