Fairwinds

BENCHMARK REPORT **2021**

Kubernetes Configuration

See how your container and Kubernetes configurations compare to other organizations.

INTRODUCTION

Correct Kubernetes configuration is vital to the success of your cloud native adoption. Without it, improving application reliability, security and efficiency remains elusive. While configuration validation might be easier in a small team with one or two clusters, DevOps teams, along with platform and security leaders, lack visibility into what's actually happening in each cluster as the organization scales. The beauty of Kubernetes is its customization—but that customization can cause risk, downtime or wasted resources.

The Kubernetes Configuration Benchmark Report has been created based on results from over 100,000 workloads and hundreds of organizations using the Fairwinds Insights platform. It serves as a tool for Kubernetes users to benchmark against. Divided into three sections, the report focuses on:

Fairwinds is the trusted partner for Kubernetes governance and security. With Fairwinds, customers ship cloud native applications faster, more cost effectively and with less risk. We provide a unified view between Dev, Sec and Ops, removing friction between those teams with software that simplifies complexity.

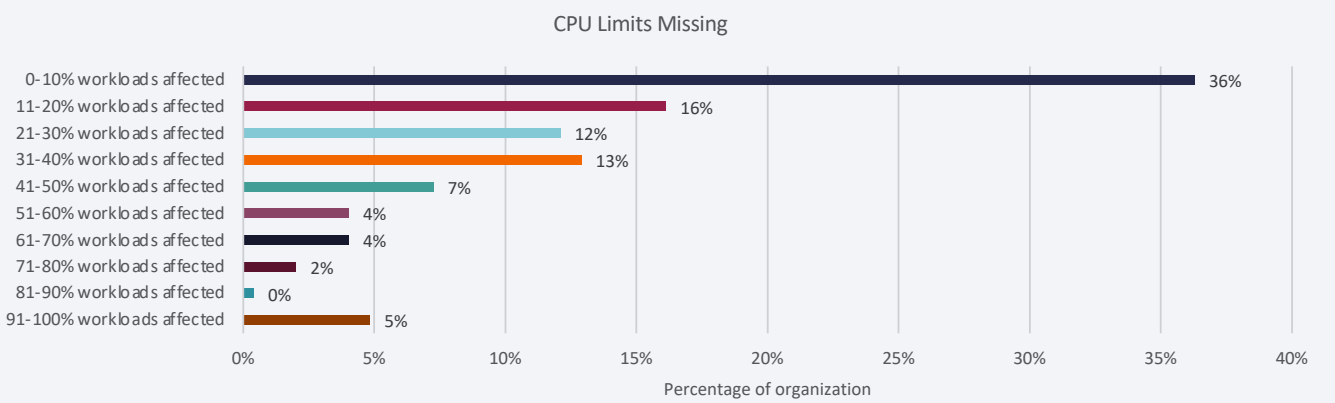
RELIABILITY

CPU Limits Missing

Polaris

Sixty-four percent of organizations are missing CPU limits on more than 10% of their workloads impacting application performance, stability and cost. In fact, 5% of organizations are missing CPU limits on the vast majority of workloads (91-100%). A large minority of organizations (36%) have successfully set CPU limits on at least 90% of their workloads, demonstrating that this is a desirable and achievable goal.

If you do not specify CPU limit then the container will not have any upper bound. This can impact reliability as the CPU intensive container slows down and could exhaust all CPU available on the node.

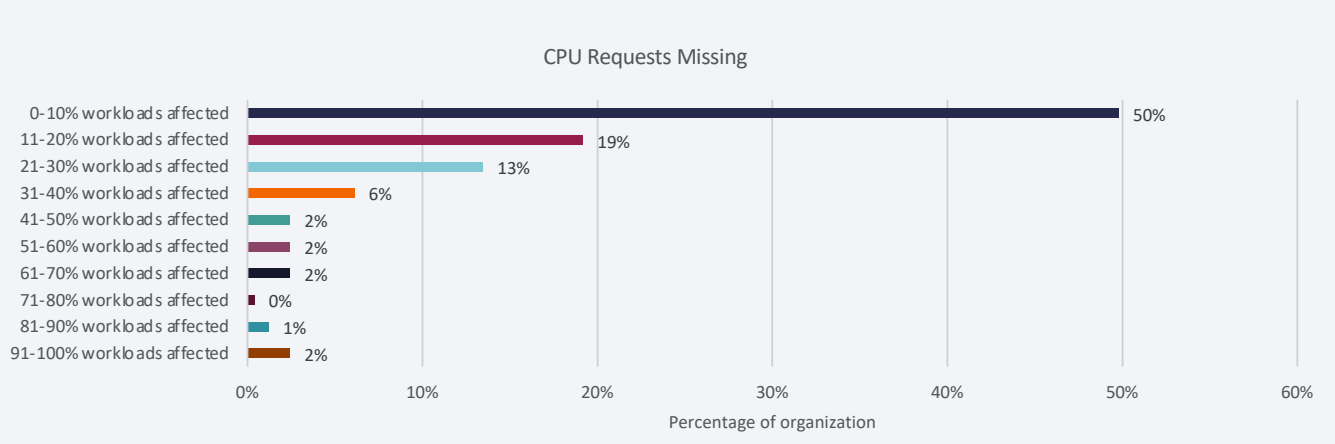


CPU Requests Missing

Polaris

Fifty percent of organizations are missing CPU requests on at least 10% of their workloads. But the other half have clearly prioritized setting CPU requests on the vast majority of their workloads.

If a single pod is allowed to consume all of the node CPU and memory, then other pods will be starved for resources. Setting resource requests increases reliability by guaranteeing the pod will have access to those resources—and preventing other pods from consuming all of the available resources on a node (this is referred to as the “noisy neighbor problem.”)

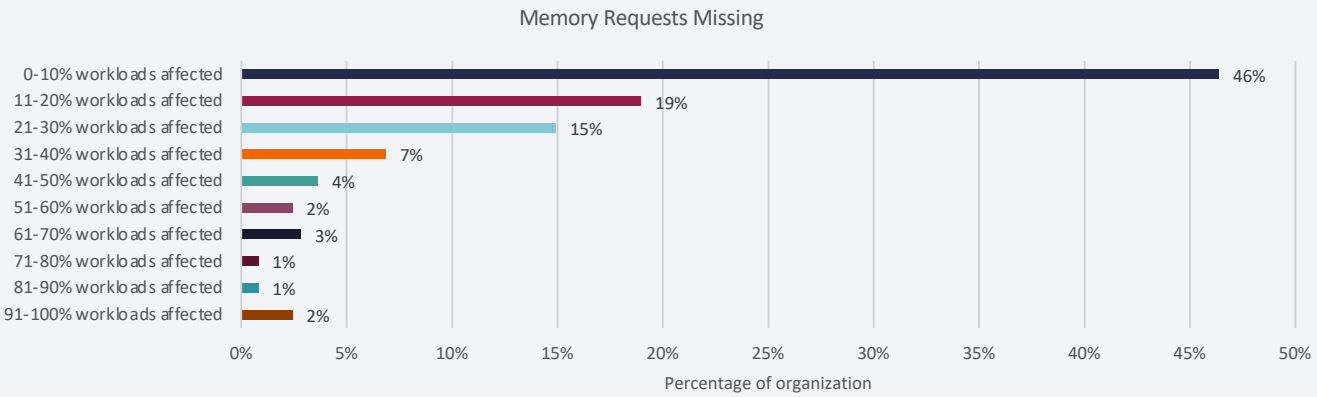


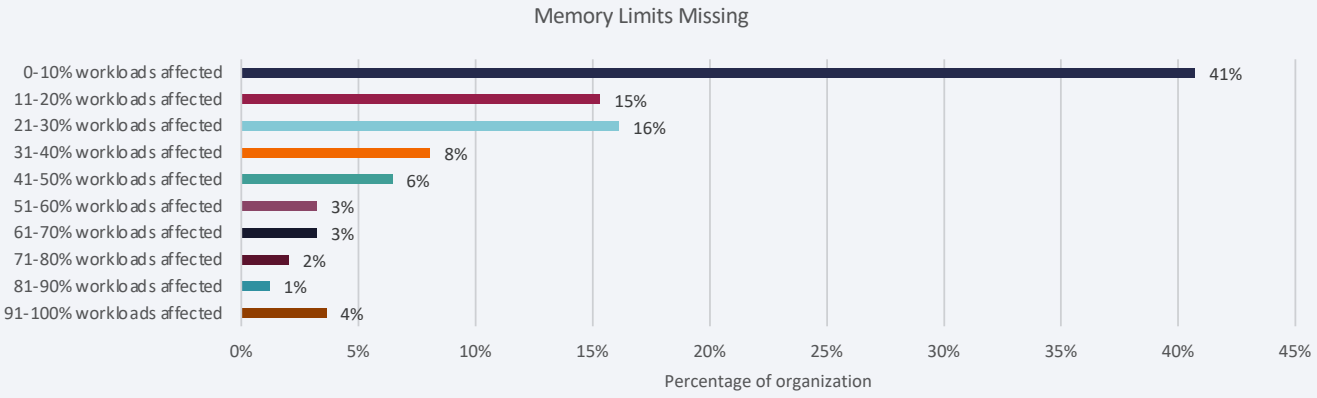
Missing Memory Limits / Missing Memory Requests

Polaris

While Kubernetes best practices dictate that you should always set resource limits and requests on your workloads, it is not always easy to know what values to use for each application. As a result, some teams never set requests or limits at all, while others set them too high during initial testing and then never course correct. Our findings show that almost 60% of organizations are not setting memory limits or memory requests for their workloads.

The key to ensuring scaling actions work properly is dialing in your resource limits and requests on each pod so workloads run efficiently. Setting resource limits and requests is essential to operating applications on Kubernetes clusters as efficiently and reliably as possible.





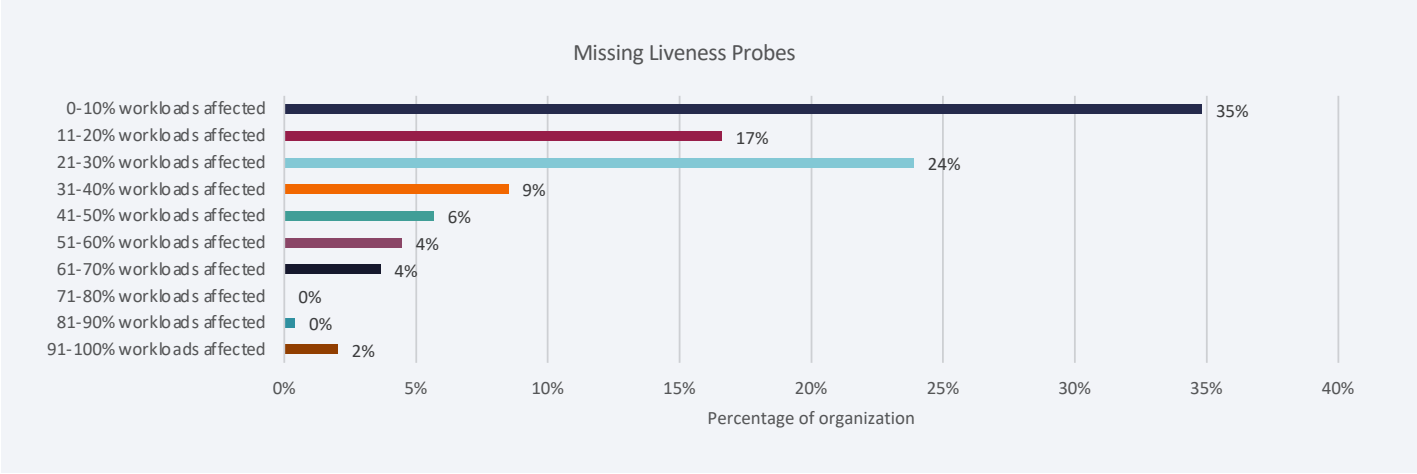
60% of organizations are not setting memory limits or memory requests for their workloads.

Missing Liveness and Readiness Probes

Polaris

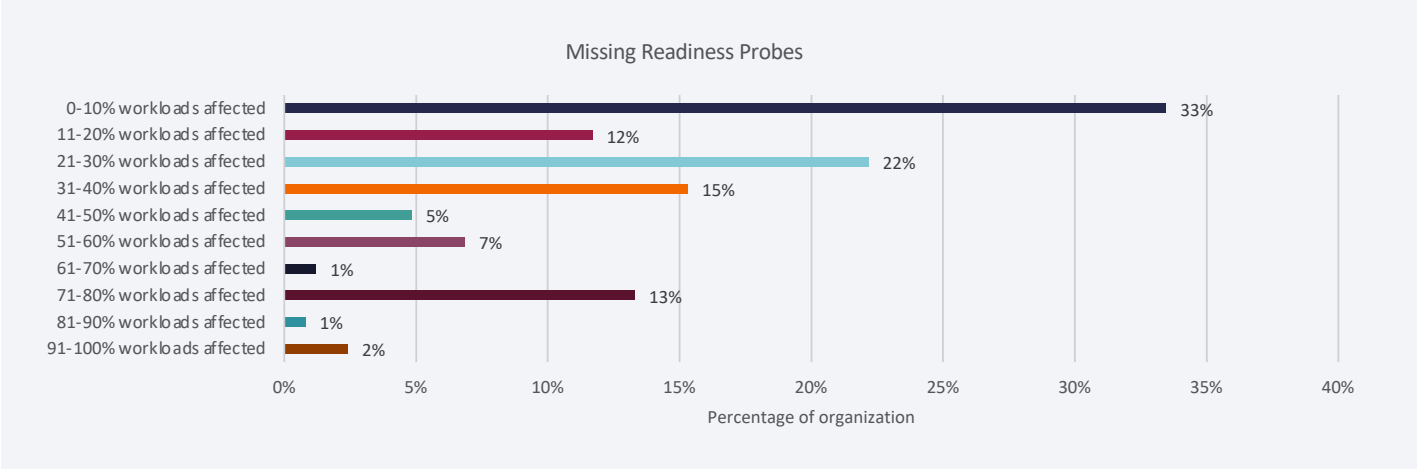
A liveness probe indicates whether or not the container is running or alive, a fundamental indicator of how a Kubernetes cluster should function. If this probe is moved into a failing state, then Kubernetes will automatically send a signal to kill the pod to which the container belongs. In addition, if each container in the pod does not have a liveness probe, then a faulty or non-functioning pod will continue to run indefinitely, using up valuable resources and causing possible application errors.

Sixty-five percent of organizations are not setting liveness probes for some workloads. In fact, 50% of organizations are missing liveness probes on at least 20% of their workloads. Not having liveness probes set, limits Kubernetes ability to self-heal.



A readiness probe, on the other hand, is used to indicate when a container is ready to serve traffic. If the pod is behind a Kubernetes service, it will not be added to the list of available endpoints in that service until all of the containers in that pod are marked as ready. This procedure allows you to keep unhealthy pods from serving any traffic or accepting any requests, thus preventing your application from exposing errors.

Almost a quarter of organizations (24%) are missing readiness probes in more than half of their workloads. Without these probes set properly, reliability issues often surface.

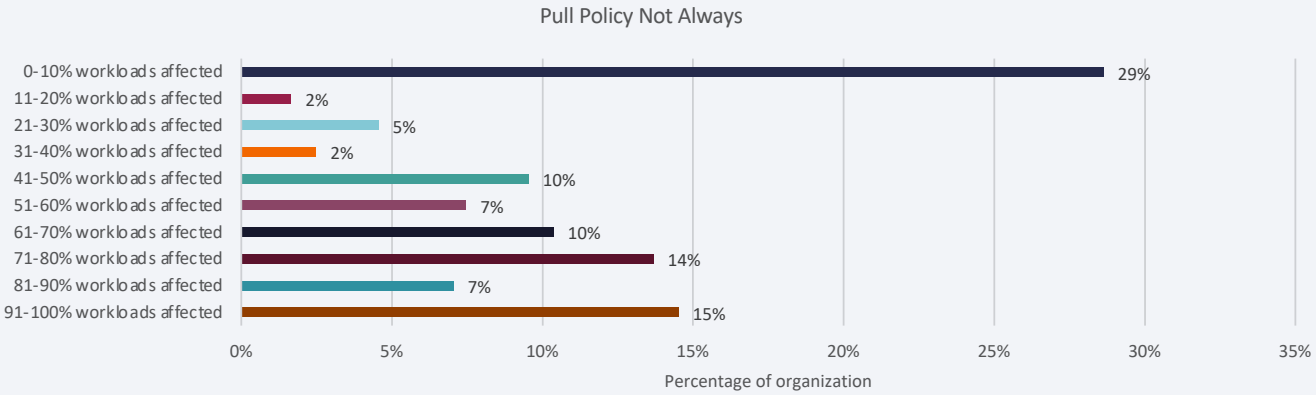


Pull Policy Not Always

Polaris

Relying on cached versions of a Docker image can become a reliability issue. By default, an image will be pulled if it isn't already cached on the node attempting to run it. This issue can cause variations in images that are running per node, or potentially provide a way to gain access to an image without having direct access to the ImagePullSecret.

More than 50% of organizations have at least half of their workloads affected by a suboptimal pull policy, but 29% have made a point of enforcing this policy for the majority of their workloads.



50%

of organizations have at least half of their workloads affected by a suboptimal pull policy.

29%

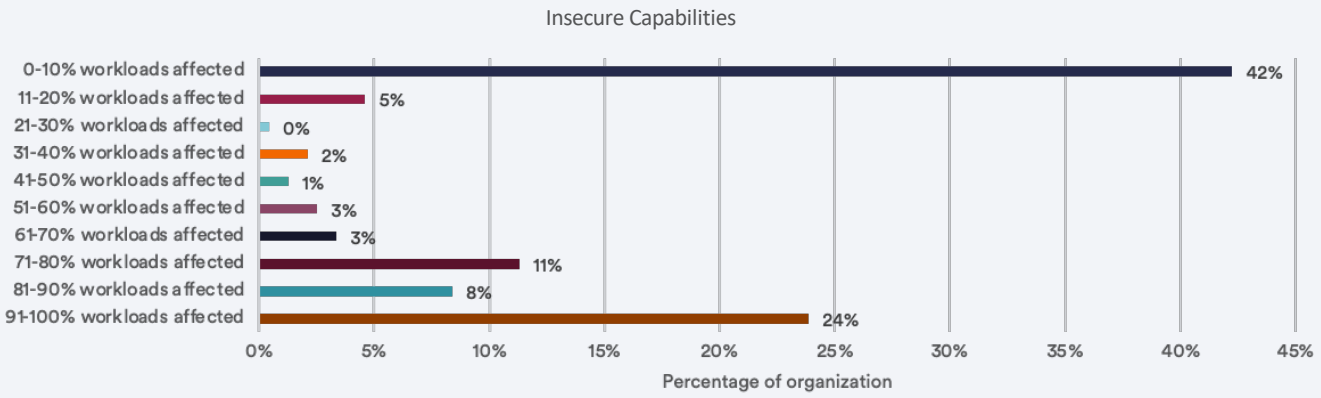
have made a point of enforcing this policy for the majority of their workloads.

SECURITY

Insecure Capabilities

Polaris

Certain Linux capabilities are enabled by default for Kubernetes workloads, though most workloads don't really need these capabilities. A large minority (42%) of organizations have made a concerted effort to pare back these capabilities on the majority of their workloads, but most organizations have not. Almost a quarter (24%) of organizations have 91-100% of their workloads running with insecure capabilities.

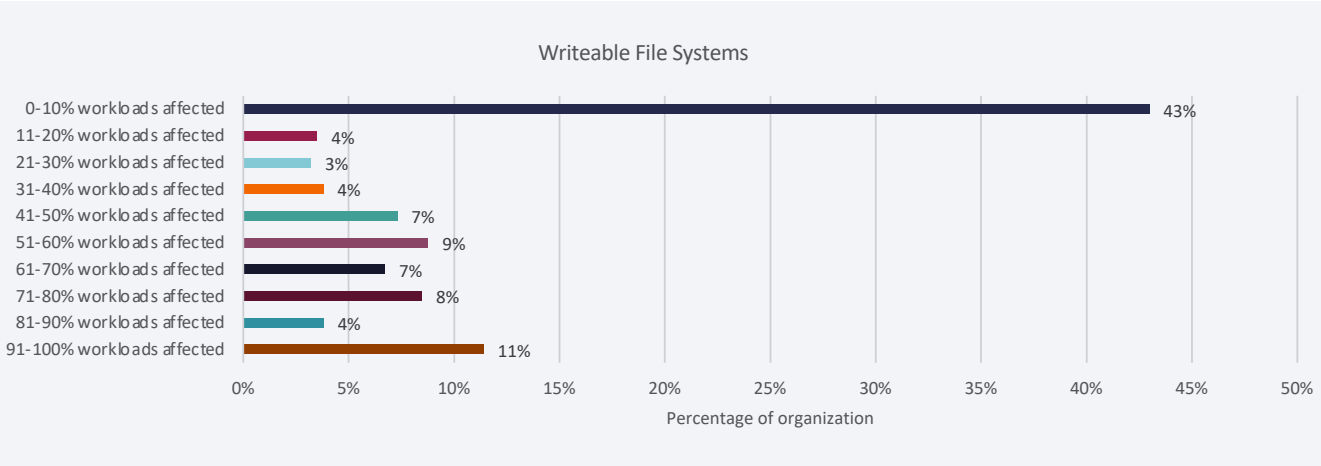


Writeable File Systems

Polaris

readOnlyRootFilesystem is a security setting that controls whether a container is able to write into its filesystem. It is a feature most organizations want enabled in the event of a hack. If an attacker gets in, they will not be able to tamper with the application or write foreign executables to disk. Unfortunately, Kubernetes workloads do not set this to true by default, which means teams need to explicitly ensure it happens to get the most secure configuration possible.

Interestingly, we see a fairly binary distribution here—some security-conscious organizations (43%) have clearly prioritized locking down the filesystems inside their containers. But many (39%) have not bothered to override the insecure defaults for the majority of their workloads.

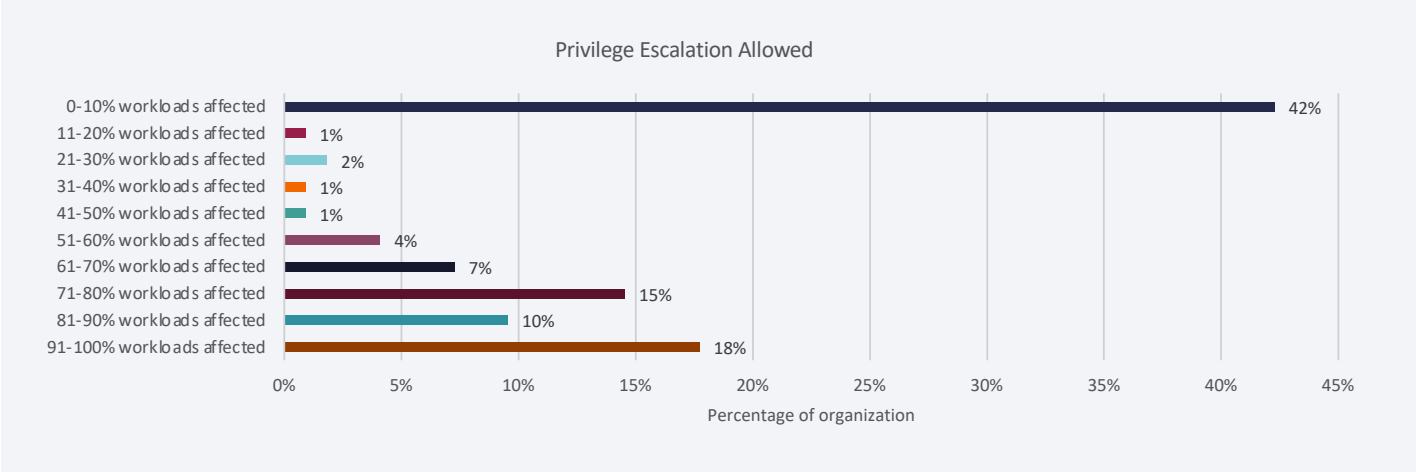


Privilege Escalation Allowed

Polaris

Under particular configurations, a container may be able to escalate its privileges. Setting allowPrivilegeEscalation to false will set the no_new_privs flag on the container process, preventing setuid binaries from changing the effective user ID. Setting this flag is particularly important when using runAsNonRoot, which can otherwise be circumvented. Because this, too, is not set by default, security-conscious teams need to explicitly set it.

Again, we see a fairly binary distribution here, with 42% of organizations managing to lock down the vast majority of their workloads. However, most organizations (54%) have left over half their workloads open to privilege escalation.

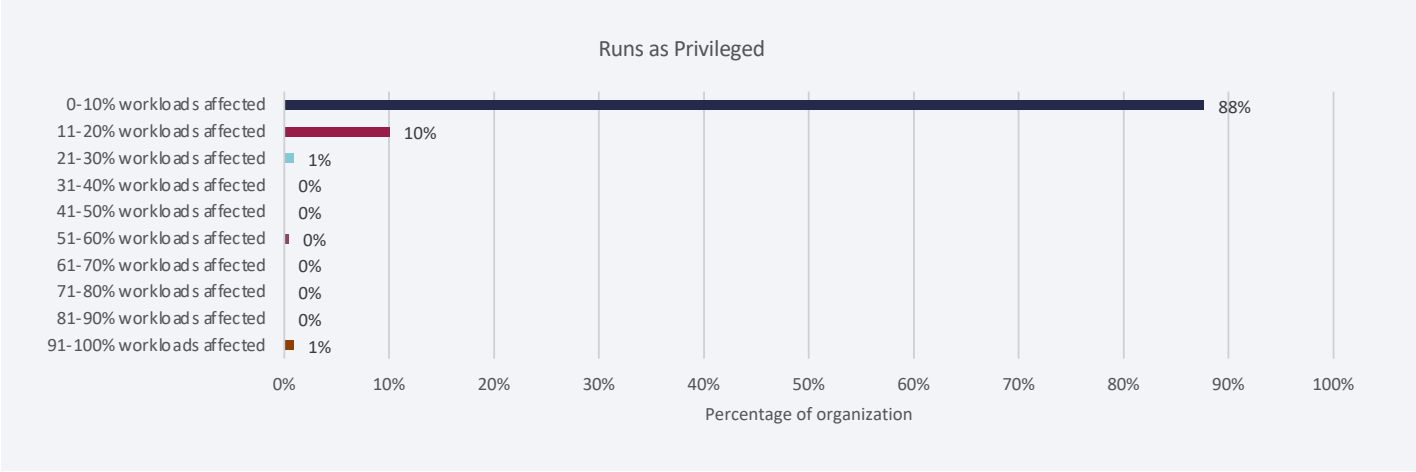


Runs as Privileged

Polaris

The command, privileged, determines if any container in a pod can enable privileged mode. By default, a container is not allowed to access any devices on the host, but a privileged container is given access to all devices on the host. This feature allows the container nearly all the same access as processes running on the host, which is useful for containers looking to use Linux capabilities, like manipulating the network stack and accessing devices.

Fortunately, the privileged flag is off by default, and 88% of organizations have stuck with that default, helping to ensure the security of their workloads.



Run as Root Allowed

Polaris

Recently announced CVEs target containers running as root when trying to enter an organization's environment. Unfortunately, many workloads are allowed to run as root. While this may be appropriate in some workloads, 70% of organizations are running 11% or more of their workloads as allowing root access.

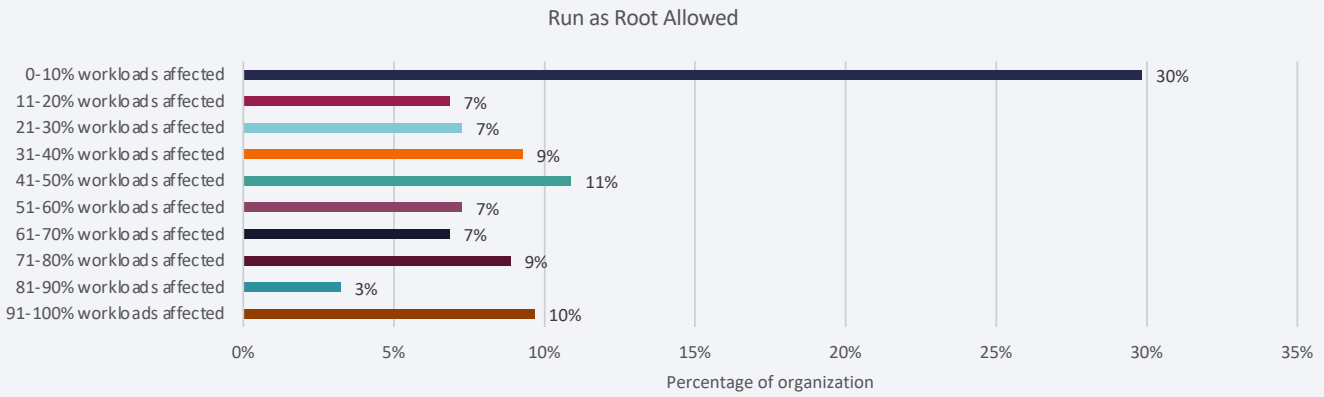
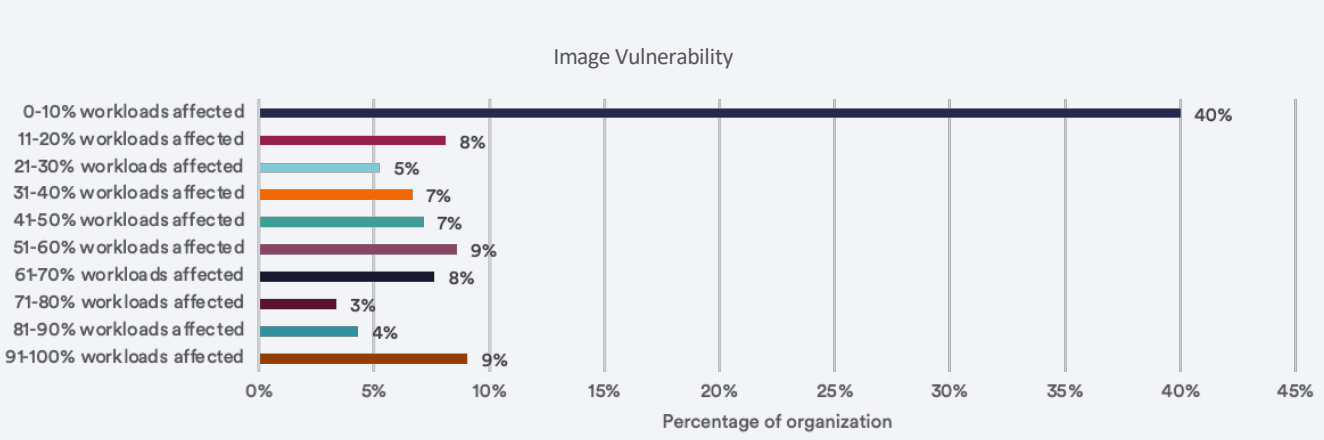


Image Vulnerability

Trivy

Nine percent of organizations have workloads with image vulnerabilities in 91-100% of their workloads. In fact, more than 60% of organizations are running some images with vulnerabilities in production. Known vulnerabilities can be exploited by malicious actors and need to be patched/remediated.



70%

of organizations are running 11% or more of their workloads as allowing root access.

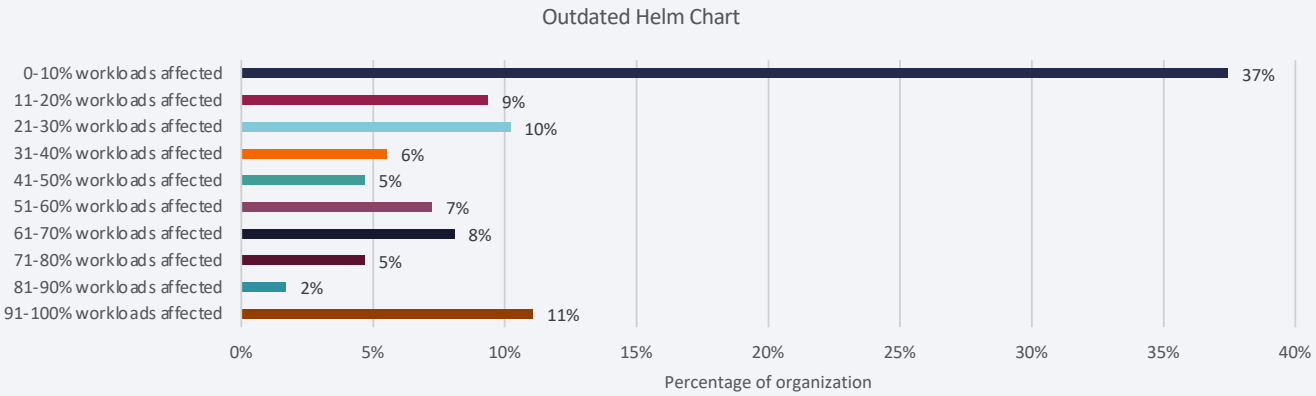
60%

of organizations are running some images with vulnerabilities in production.

Outdated Helm Chart

Nova

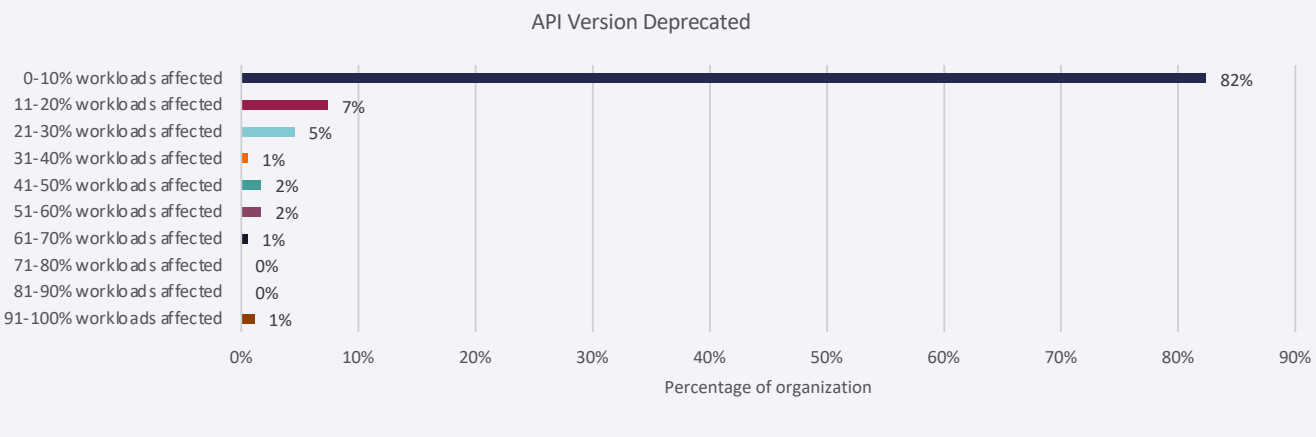
Outdated Helm charts are a pervasive issue across most organizations. For example, 33% of organizations have at least 50% of their workloads running with outdated Helm charts. Maintaining current Helm charts, for both public and private charts, is now considered best practices for keeping vulnerability exposure windows short.



API version deprecated

Pluto

Most organizations appear to have only a few workloads with deprecated API versions. However, monitoring for deprecated APIs remains an important step in de-risking Kubernetes upgrades.



33% of organizations have **at least 50%** of their workloads running with outdated Helm charts.

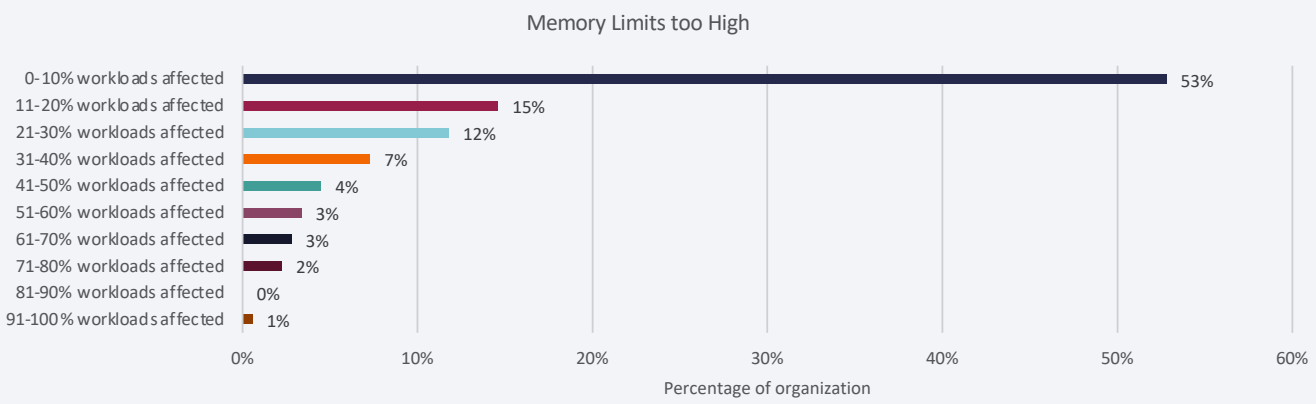
EFFICIENCY

To maximize the efficient utilization of your Kubernetes cluster, it is critical to set resource limits and requests correctly. Setting your limits too low on an application will cause problems. For example, if your memory limits and requests are too low, Kubernetes is bound to kill your application for violating its limits. Meanwhile, if you set your limits and requests too high, you're inherently wasting resources by overallocating, which means you will end up with a higher bill.

Memory Limits too High

Goldilocks

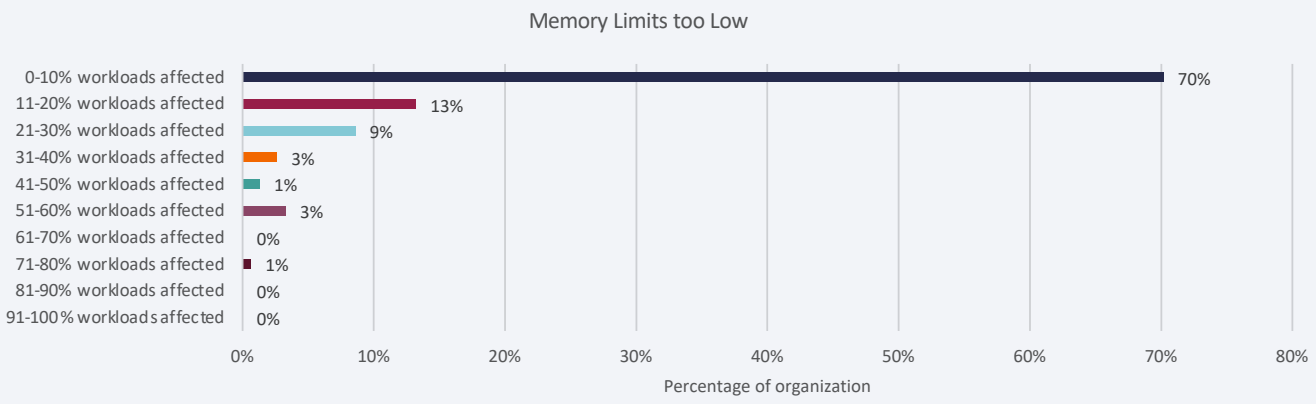
Almost half of workloads have memory limits set too high, which often results in wasted/unnecessary resource.



Memory Limits too Low

Goldilocks

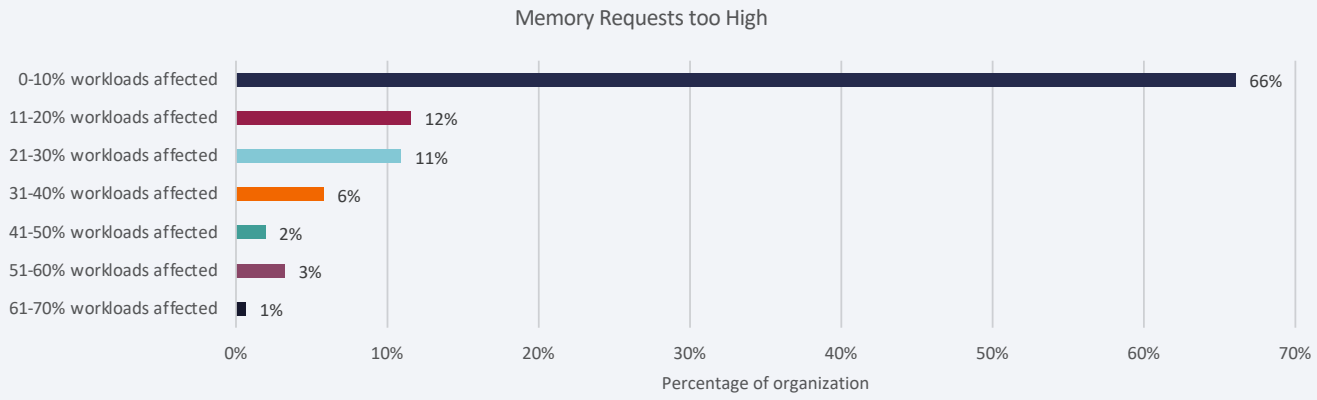
Organizations appear to set memory limits. Most workloads have limits set. Seventy percent of organizations have memory requests set too low on at least 10% of their workloads.



Memory Requests too High

Goldilocks

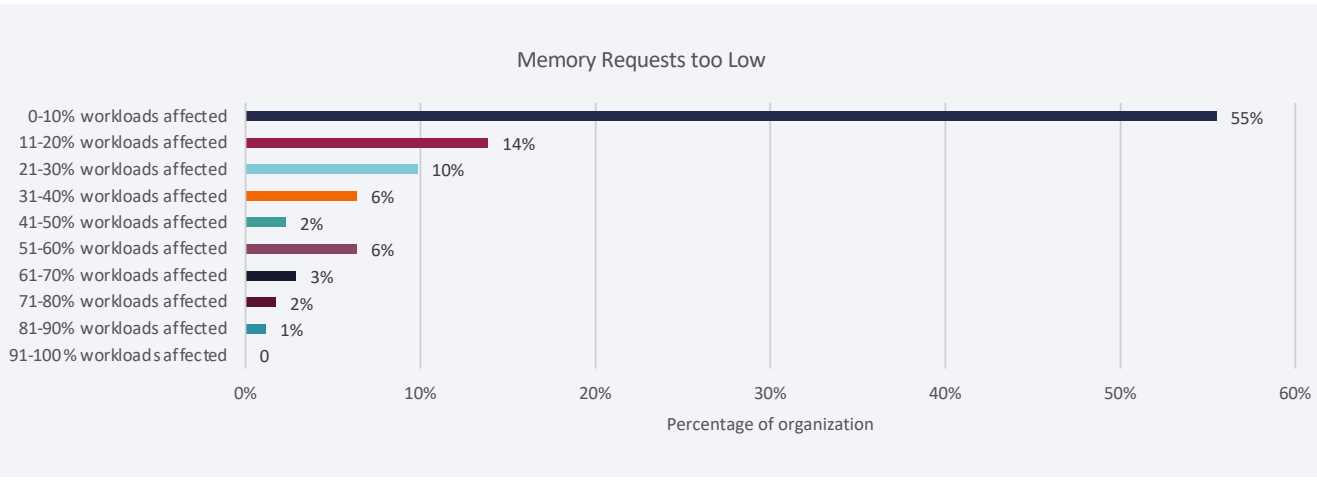
Thirty-four percent of organizations have memory requests set too high on at least 10% of their workloads.



Memory Requests too Low

Goldilocks

Forty-five percent of organizations have memory requests set too low on more than 10% of their workloads. Unfortunately, when set too low, ensuring application reliability becomes a challenge.



CONCLUSION

Fairwinds will continue to update the results of this benchmark data to help the cloud native community understand how they stack up. The important takeaway is this:Kubernetes configuration is essential to cloud native success. Misconfiguration is too risky.

Empower your team to ensure Kubernetes configurations are done right. Solutions like Fairwinds Insights can provide Dev, Sec and Ops to align on security risk, reliability and app-right sizing.

Fairwinds Insights

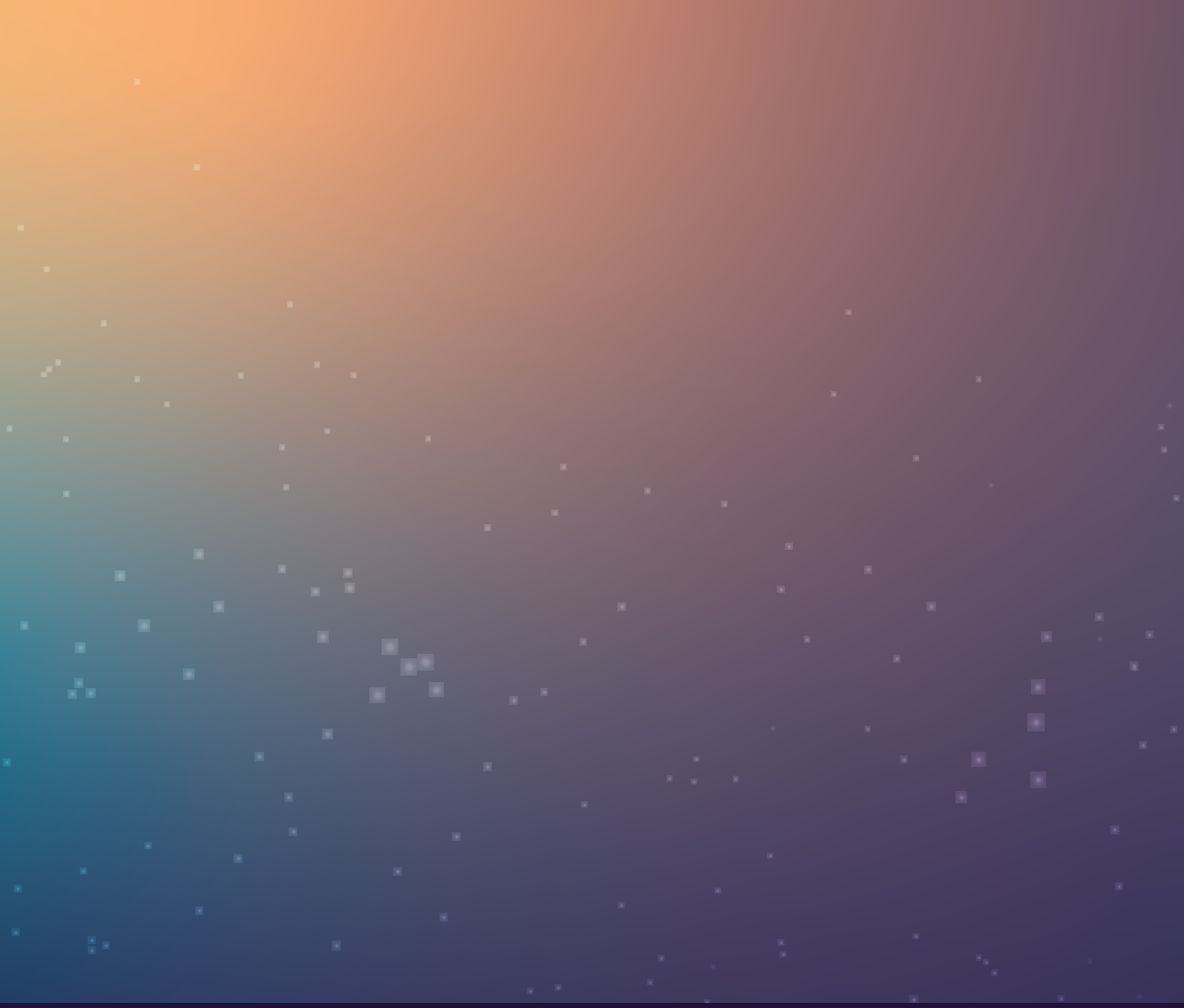
Fairwinds Insights is software that simplifies Kubernetes chaos to help you achieve your business goals. It provides DevOps teams a safety net for scalability, reliability, resource efficiency and security while empowering developers to ship applications faster.

Insights is a complete platform that offers cost optimization advice, security alerting, guardrails and compliance findings. The platform doesn't get in the way of your development life cycle, instead it speeds it up by offering a path for turning chaosinto success.

DevOps teams can prevent misconfigurations throughout the CI/CD pipeline and provide remediation advice to developers, free from manual intervention. Managing multiple clusters and teams becomes easy with Fairwinds Insights. Apply the guardrails and custom OPA policies your business requires. Developers are free to develop with safety nets in place.

Fairwinds Insights was developed based on years' of experience managing Kubernetes clusters for hundreds of clients. We created a suite of open source tooling to offer Kubernetes best practices, recommend resource requests and limits, and identify deprecated tooling. Insights operationalizes these tools into a single platform for managing multiple clusters across multiple teams.

Fairwinds Insights makes findings actionable thru tight integrations withGitHub Actions, CI/CD tooling including, CircleCI, Jenkins, GitLab, Travis, Azure DevOps, ticketing systems including GitHub, Jira, Slack and DataDog Custom Metrics. Empower your developers to work the way they want within the tooling they already know and love.



Fairwinds

WHY FAIRWINDS

Fairwinds is your trusted partner for Kubernetes security, policy and governance. With Fairwinds, customers ship cloud-native applications faster, more cost effectively, and with less risk. We provide a unified view between dev, sec, and ops, removing friction between those teams with software that simplifies complexity. Fairwinds Insights is built on Kubernetes expertise and integrates our leading open source tools to help you save time, reduce risk, and deploy with confidence.

WWW.FAIRWINDS.COM