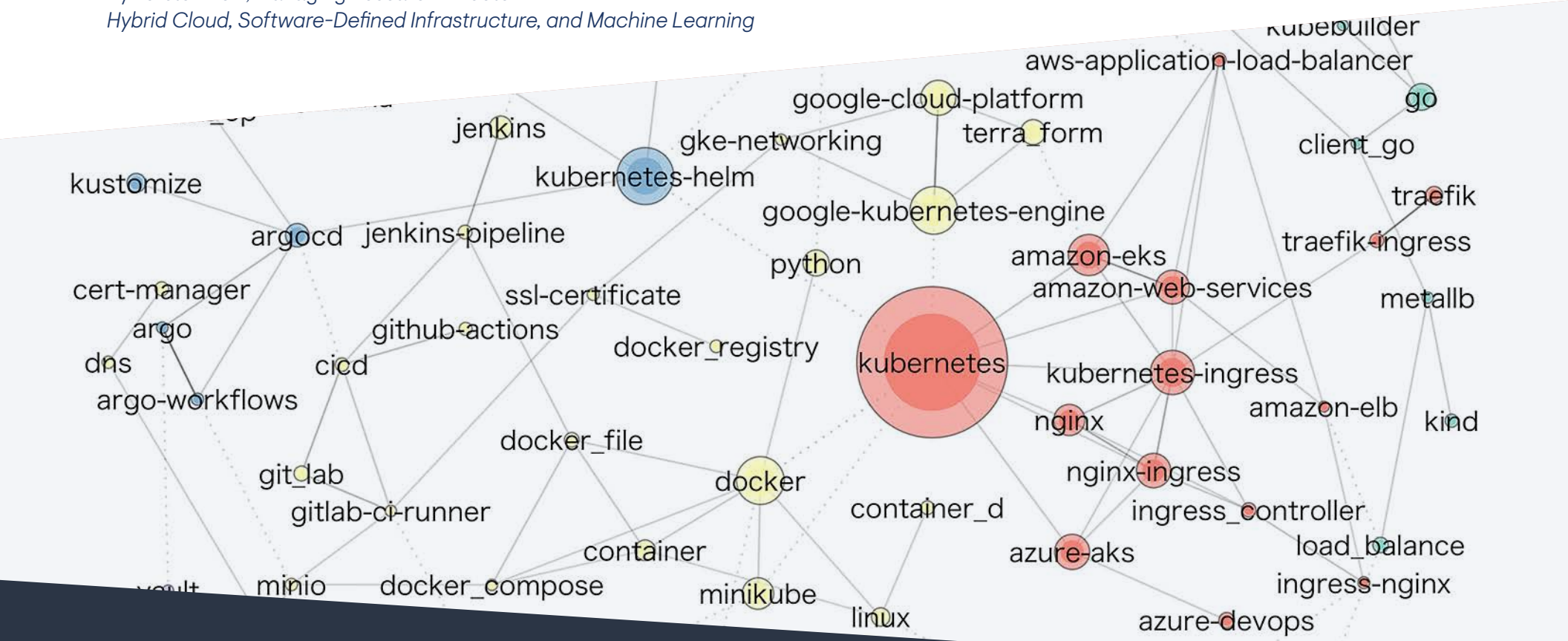
IT & DATA MANAGEMENT RESEARCH,
INDUSTRY ANALYSIS & CONSULTING

Kubernetes at Scale: Challenges, Priorities, Adoption Patterns, and Solutions

Q3 2023, EMA Research Report

By Torsten Volk, Managing Research Director
Hybrid Cloud, Software-Defined Infrastructure, and Machine Learning



Spotlight on Developer Productivity

The equation is simple: to maximize the productivity of application developers, we need to minimize the time they have to spend on overhead tasks. These are tasks that go beyond writing, maintaining, and testing new application code. They include infrastructure provisioning and configuration, deployment, management, upgrades of Kubernetes clusters, application deployment and management, ensuring security and compliance of the overall application stack, configuring backup and disaster recovery, configuring observability and visibility, and optimizing cost.

The Five Pillars of the Ideal Kubernetes Management Platform

The optimal Kubernetes management platform offers five fundamental capabilities that are key to relieving developers from their extensive list of overhead tasks.

1. **Declarative Management:** DevOps teams define the desired state of an application stack and the Kubernetes management platform continuously enforces this state (e.g., pod configuration, network policies, performance and reliability metrics, storage types and volume sizes, endpoints and DNS names of services, secrets, configuration maps, and service discovery).
2. **Policy-Driven Automation:** Automated application deployment, configuration, operations management, scaling, and upgrades based on a core set of policy rules are critical for minimizing overhead tasks for developers.
3. **Unified Operations Management:** A single, unified management plane to control declarative management and policy-driven automation across application stacks is essential for optimal operational efficiency and organization-wide consistency.
4. **Observability:** The ability to continuously collect and analyze metrics, logs, and traces within their application context is crucial for proactive and business-centric monitoring and remediation of issues that could lead to application downtime, poor performance, and decreased user experience.
5. **Continuous Optimization:** The ongoing fine-tuning of Kubernetes environments to ensure optimal performance, cost-efficiency, resiliency, compliance, and security is essential to continuously meet and exceed business goals.

The ideal Kubernetes platform provides enterprises with a comprehensive solution to alleviate the burden of time-consuming overhead tasks for developers. Declarative management, policy-driven automation, unified management, observability, and continuous optimization enable development teams to focus on delivering value to end users, thereby achieving faster time-to-market, better application performance, and improved overall efficiency for the organization.

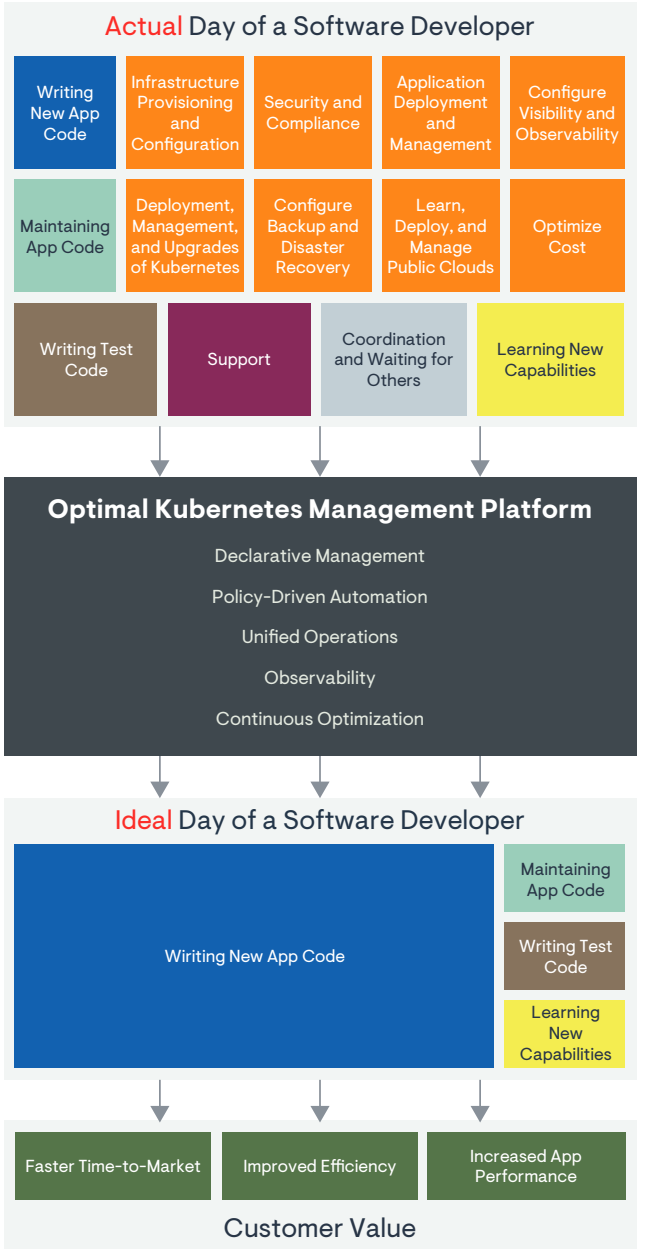


Table of Contents

Part 1: Kubernetes Growth, Adoption Patterns, and Choice

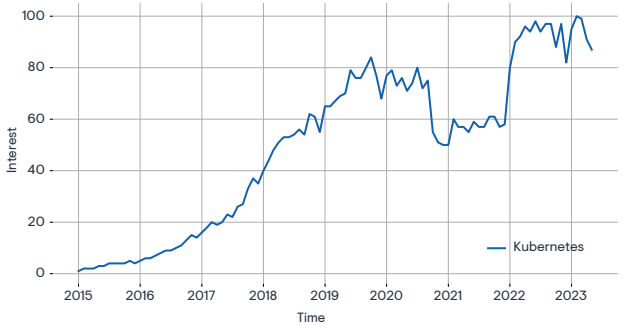
The Rapid Rise of Kubernetes Continues

There is no shortage of metrics that illustrate the rapid rise of Kubernetes as the application platform of choice for many enterprises. The number of monthly Google searches doubled between 2021 and 2023. Meanwhile, the number of downloads of the Kubernetes client library for Python increased by 1,714% between 2019 and 2023. The number of operator questions on the Stack Exchange Server Fault support forum increased by 40% year over year, and the number of publicly accessible Kubernetes clusters also increased by 40% over the same timeframe.

Today, there are 1,609 Python libraries related to different Kubernetes distributions and components.

GOOGLE TRENDS

Google Trends Interest Over Time for "Kubernetes"

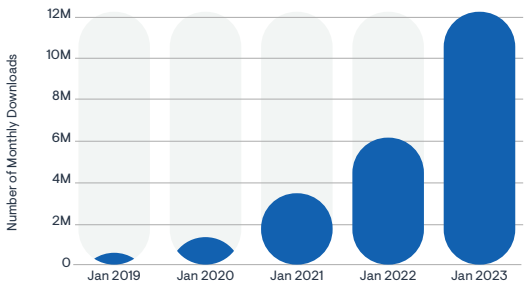


The total number of monthly Google Searches (worldwide) that include 'Kubernetes' as a search term increased by approximately 100% from 320k in 2021 to 639k in 2023

Source: Google Trends

APP DEVELOPERS

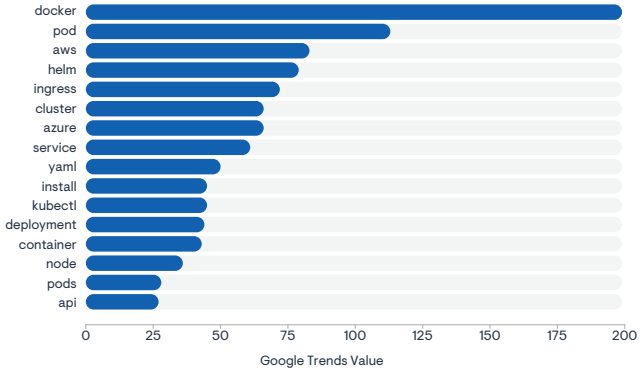
Monthly Kubernetes Python client library downloads between 2019 and 2023: from 700k to 12.4 Million (+1,714%)



The number of monthly downloads of the Kubernetes Python client library increased by 1,714% from 700,000 to 12.4 Million between 2019 and 2023. This illustrates the rapid progression of Kubernetes toward becoming the de-facto standard platform for cloud native applications. The Python library enables developers to programmatically create, read, update, query, scale, monitor and manage Kubernetes clusters.

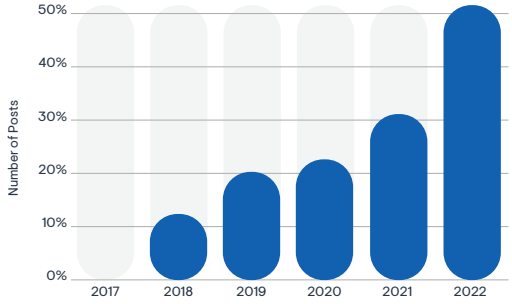
Source: PyPi, March 2023

TOP 20 KUBERNETES-RELATED GOOGLE SEARCH TERMS



IT OPERATIONS

40% YoY Increase of Operator Questions on StackExchange

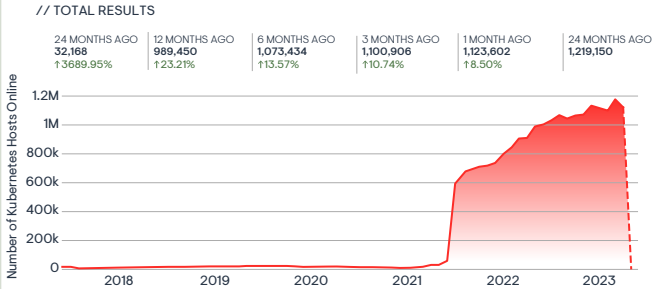


The 40% increase in IT operator questions that were directly or indirectly related to 'Kubernetes' on the StackExchange Serverfault forum points to the significant impact of Kubernetes on the daily lives of operators.

Source: StackExchange

ACCESSIBLE CLUSTERS

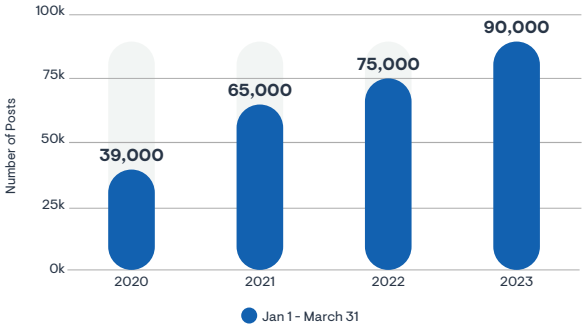
40% YoY Increase of Operator Questions on StackExchange



Based on the Shodan search engine for connected devices, the total number of publicly accessible Kubernetes clusters increased by 40% over the past 12 months.

Source: Shodan

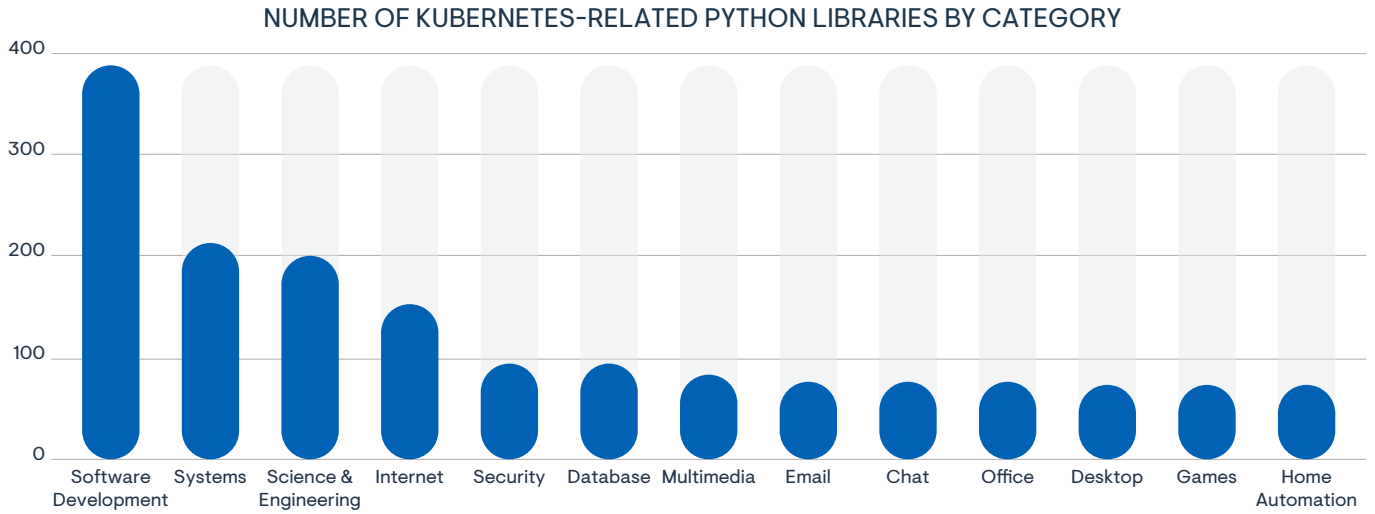
CNCF SLACK ACTIVITY



Activity in the official CNCF Slack channel increased by 20% YoY and by 130% between 2020 and 2023.

EMA Perspective

This data demonstrates the rapid and sustained growth in Kubernetes adoption. Initially conceived as a joint platform for developers and operators, developers set the adoption pace. However, in recent years, we observed IT operators catching up. It turned out that Kubernetes clusters, even the managed ones in the public cloud, do not operate themselves. When looking at the top 20 most popular Kubernetes-related Google search terms, we find the Helm package manager and "ingress" near the top. This points toward an increased interest in Kubernetes deployment tools and networking configuration. The trend suggests that users are actively seeking solutions for managing and deploying applications on Kubernetes, as well as managing external access to services in a cluster—both crucial aspects of Kubernetes administration and operations.



Each Python library that is part of these categories provides app developers with the code functions that are required to deploy, manage, or integrate Kubernetes clusters.

Source: PyPi

Kubernetes Community Metrics

Over the past 12 months, 47 open source Kubernetes distributions received updates in the form of new code commits. This resulted in a total of 31,329 code commits, with developers continuously working on improvements, bug fixes, and net new features and capabilities. The fact that there are currently 11,531 developers contributing to these 47 Kubernetes distributions demonstrates the high level of vendor commitment to the Kubernetes platform since it is the vendors that pay these 11,531 salaries.

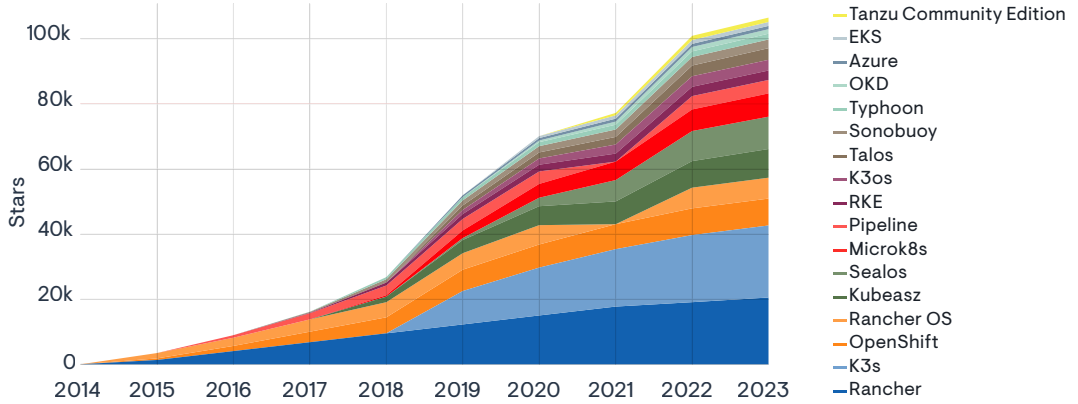
CNCF Metrics

From an end-user perspective, 95,000 individuals enrolled in one of CNCF's certification programs in 2022, with approximately one-third receiving the desired certificate. The most popular certifications were Certified Kubernetes Administrator (45%), Certified Kubernetes Application Developer (20%), Certified Kubernetes Security Specialist (12%), and Linux Foundation Certified System Administrator (5%).

EMA Perspective

The fact that over the previous 12 months there were 4,603 developers making 6,949 commits to the official Kubernetes repository on GitHub, in addition to the 11,531 contributors to the various Kubernetes distributions, shows the strong momentum of the overall Kubernetes platform. This momentum can benefit organizations adopting Kubernetes because they can tap into the collective knowledge and experience of this community for help, guidance, and best practices.

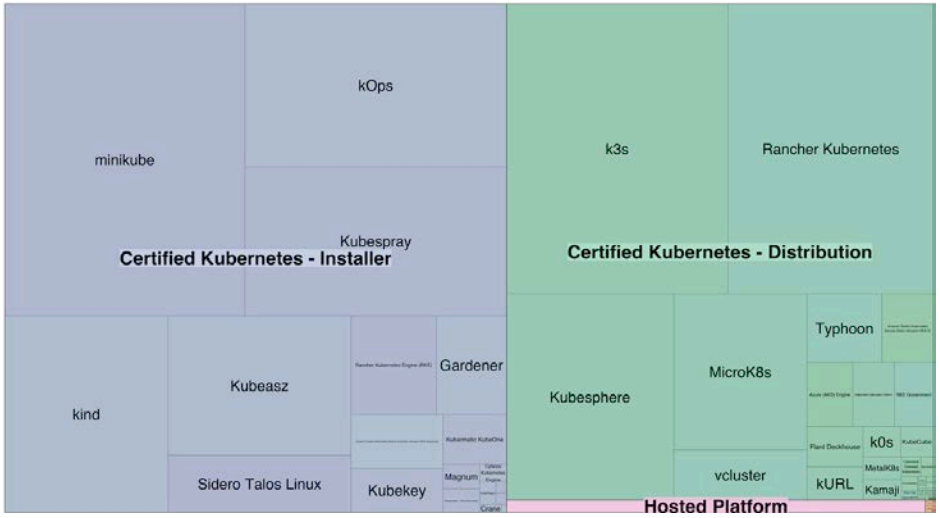
KUBERNETES DISTRIBUTIONS AND INSTALLERS BY GITHUB STARS OVER TIME



The chart shows the largest and fastest-growing Kubernetes distributions based on their number of GitHub stars. Combined, these distributions have collected over 120,000 stars.

Source: GitHub

31.329 ANNUAL CODE COMMITS



The chart shows Certified Kubernetes Distributions, Certified Kubernetes Installers, and Hosted Kubernetes Platforms sized by the number of code commits over the past 12 months.

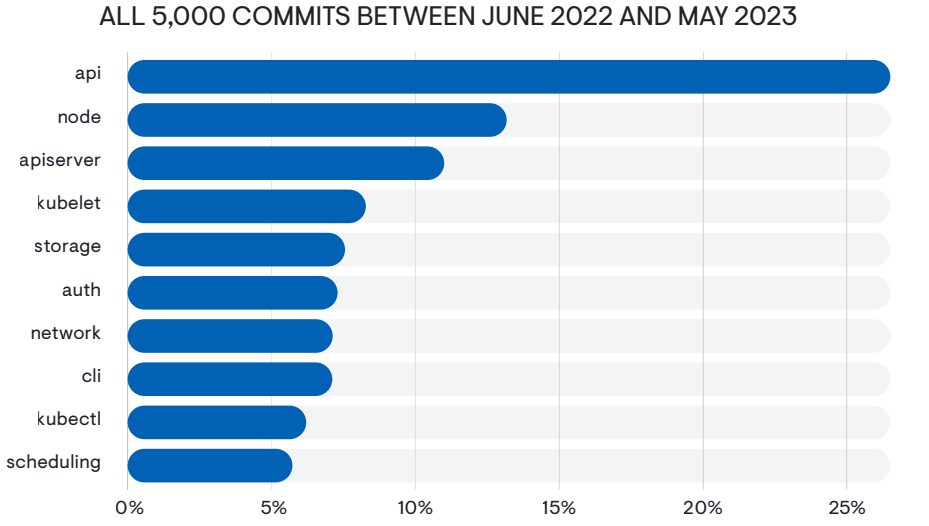
Source: GitHub

Key Topics From the Last 5,000 Kubernetes Code Commits

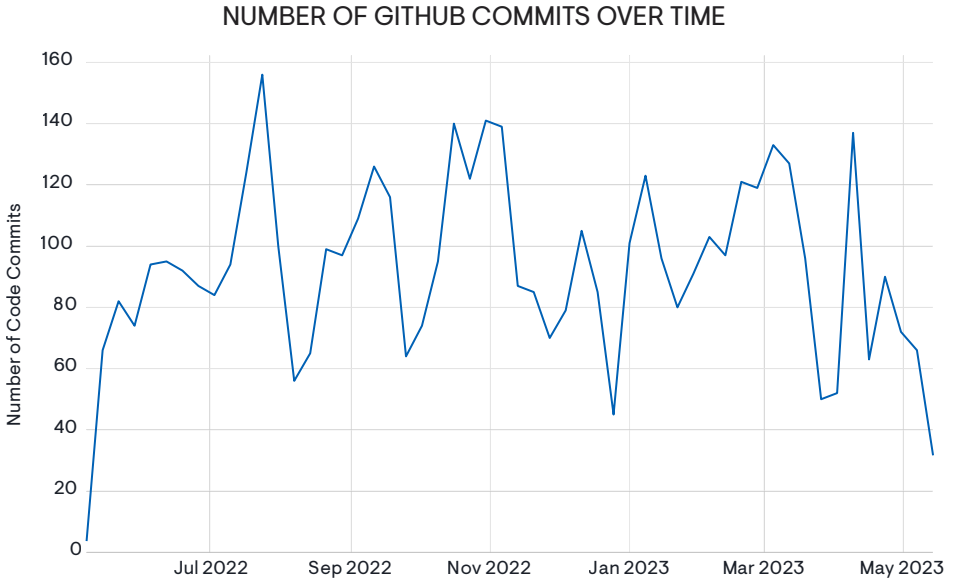
Based on a complete analysis of the titles and labels of the most recent 5,000 code commits to the Kubernetes/Kubernetes repository, we find that API fixes and enhancements are the predominant topics by a wide margin (27% + 12% for "API server"). The focus in terms of API capabilities includes enhancing the performance and efficiency of existing APIs, exploring the possibilities of new discovery APIs, and improving error handling and monitoring.

Improvements to Kubernetes nodes and their ability to run pods come in at second place (13% + 8% for "Kubelet"). The emphasis is on enhancing node capabilities by addressing bugs, improving performance and resilience, and refining the management of resources and tests.

Storage topics (including updated dependencies, optimization of existing features, and bug fixes), authentication (API updates and documentation), networking (cleanup, bug fixes, performance optimization, APIs, error handling), and CLI topics (usability, performance, interoperability, security, reliability, and compliance) all come in between 7% and 8%.



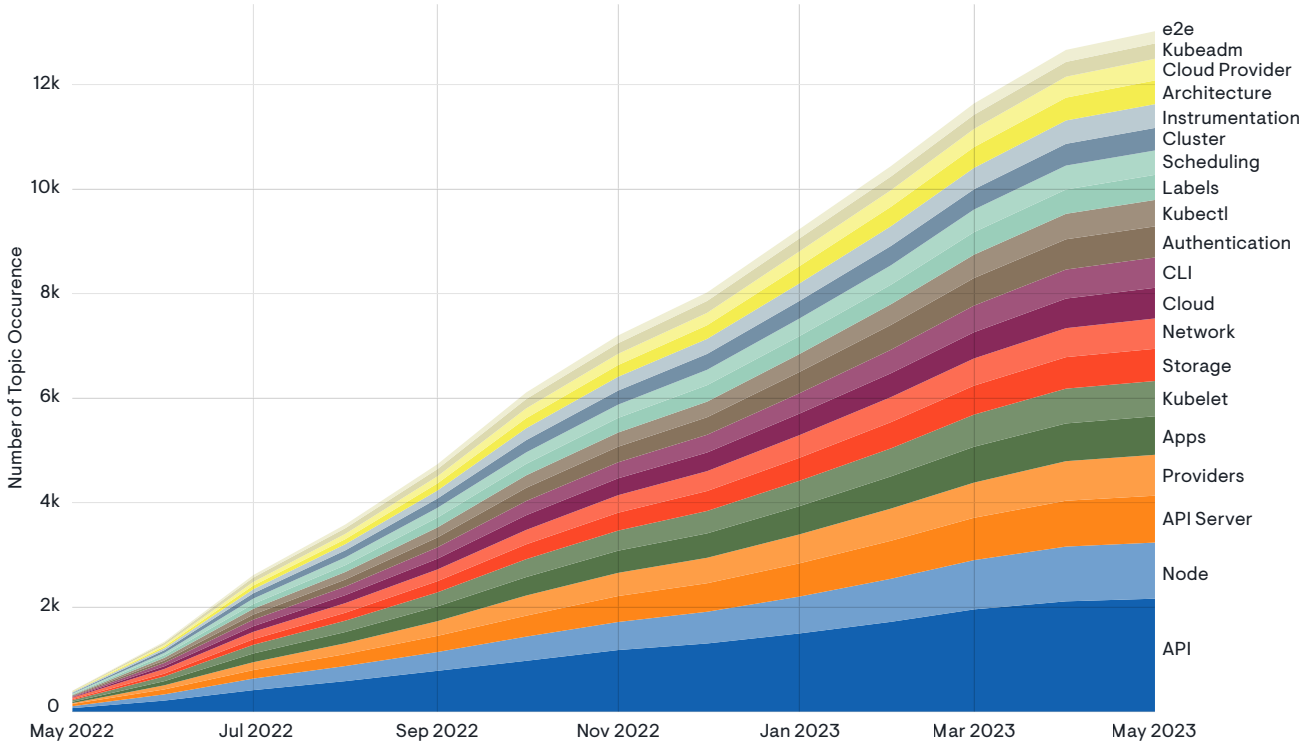
Source: GitHub



Source: GitHub

Research Spotlight: Code Commits Over Time

Code commits related to APIs, nodes, and the Kubernetes API server consistently emerge as the top three topics over time, as demonstrated in the chart (which is based on the same 5,000 code commits as the chart on the prior page).



Source: GitHub

Current Kubernetes Focus

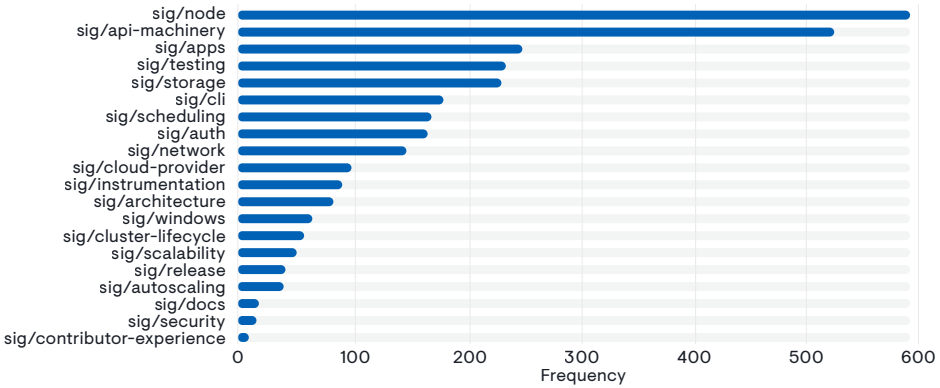
Special Interest Groups

Special interest groups (SIGs) within the Kubernetes community are focused teams working to improve various aspects of the Kubernetes project. Managing the interactions between Kubernetes pods and nodes is the number one topic when it comes to SIGs with the most issues assigned. This includes ensuring security between pods and the underlying host server, assigning specific hardware (e.g., GPUs) to pods, ensuring performance metrics for specific pods, supporting different container runtimes, discovering new hardware for the node to utilize, and many more node-related topics.

The Kubernetes Cluster API comes in a close second. It allows enterprises to manage application dependencies that are not part of Kubernetes through the Kubernetes control plane. Popular examples include managing a service mesh, ingress controller, CI/CD workflow tools, persistent storage, and even entire applications, such as WordPress, MongoDB, Nginx, Apache Web Server, Istio, Knative, etc.

Next come topics around deploying and managing application workloads, persistent storage volumes, the Kubernetes CLI, and workload scheduling. All these topics demonstrate that Kubernetes is going mainstream, and therefore the community is working to eliminate the key hurdles.

TOP 20 MOST USED LABELS IN OPEN ISSUES OF KUBERNETES REPOSITORY ON GITHUB WITH PREFIX SIG



Working Groups

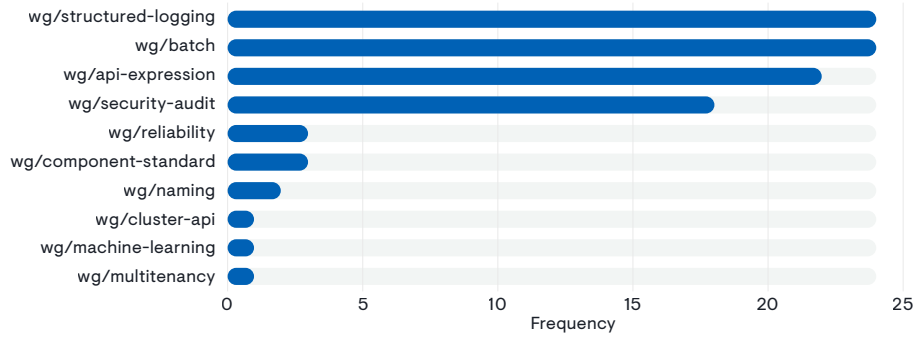
Working groups (WG) are formed on a temporary basis to coordinate between SIGs in order to implement specific features and capabilities.

The WG for Structured Logging aims to optimize the consistency of Kubernetes logs, simplifying parsing and maximizing the overall usefulness of these logs for both app developers and operators. This is part of a larger effort to implement observability for Kubernetes applications, clusters, and infrastructure.

Enterprises have recognized the benefits of running data processing and machine learning jobs on Kubernetes clusters, where they can take advantage of significant scalability and flexibility over traditional virtual machine infrastructure. Focusing on data processing and machine learning makes a lot of sense because these workloads temporarily require large amounts of high-performance CPUs, RAM, and storage. Different teams "taking turns" executing these workloads on the same infrastructure resources can lead to significant cost savings.

The API Expression WG focuses on making Kubernetes APIs consistent, customizable, secure, and easy to use for both developers and operators. These focus points are closely related to the tasks of the SIG API Machinery, which is responsible for implementing a consistent control plane for Kubernetes clusters.

TOP 20 MOST USED LABELS IN OPEN ISSUES OF KUBERNETES REPOSITORY ON GITHUB WITH PREFIX WG



Source: GitHub

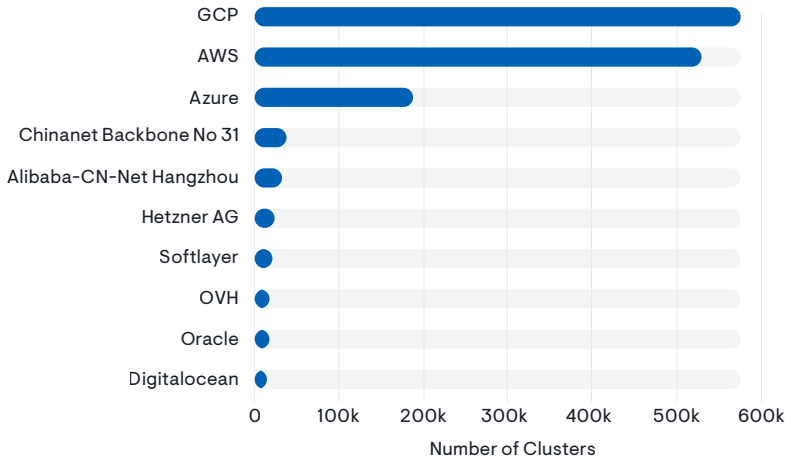
Adoption Patterns

We found that leveraging a managed Kubernetes offering on Azure, GCP, and AWS is the most popular method of Kubernetes adoption. Approximately 50% of companies use multiple Kubernetes distributions, including commercially supported ones, free open source distributions, and plain upstream Kubernetes. Most on-premises solutions run on Debian/Ubuntu and Fedora/RHEL, and 20% of organizations deploy Kubernetes clusters to bare metal.

A 50/50 Split Between Cloud and On-Premises Kubernetes

Through a simple internet scan, we can determine that there are currently 1.7 million Kubernetes clusters online, with 75% running in the public cloud. Assuming that Kubernetes clusters in the cloud are more detectable (less locked down) than on-premises ones, we can estimate parity between the number of Kubernetes clusters in the cloud and those on-premises. This would bring the total number of Kubernetes clusters worldwide to 3.5-4 million. Ninety-nine percent of these clusters run on Linux and on AMD CPUs.

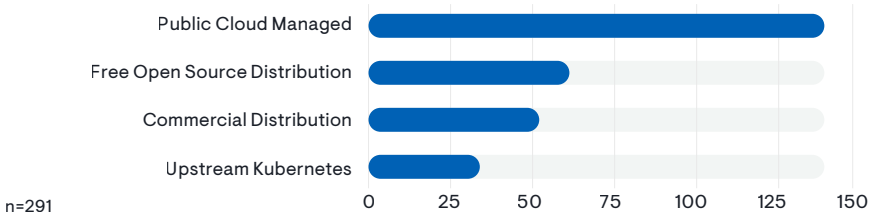
NUMBER OF KUBERNETES CLUSTERS ON PUBLIC CLOUD



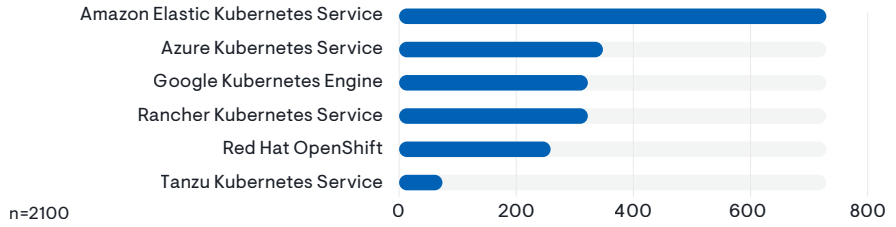
These numbers are based on a global scan of the internet and reflect all Kubernetes clusters that have their control plane exposed online. Since this is the default setting, we can assume that these are approximately 75% of overall Kubernetes clusters hosted online.

Source: Shodan

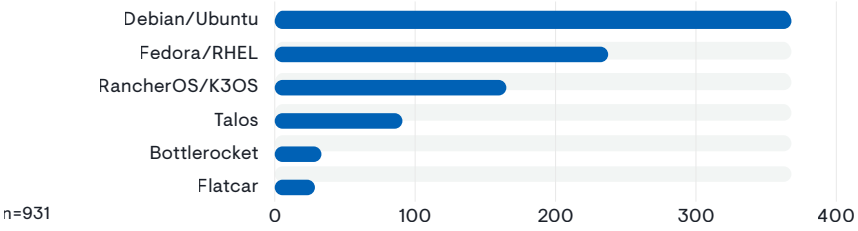
WHAT TYPE OF KUBERNETES DISTRIBUTION DOES YOUR ORGANIZATION USE?



WHICH KUBERNETES DISTRIBUTION ARE YOU CURRENTLY USING?



WHAT BASE OPERATING SYSTEM DO YOU RUN FOR YOUR ON-PREMISES KUBERNETES DEPLOYMENTS?



"99% of Kubernetes clusters run on AMD64 hardware."

"20% of organizations deploy Kubernetes to bare metal."

"99% of Kubernetes clusters run on Linex."

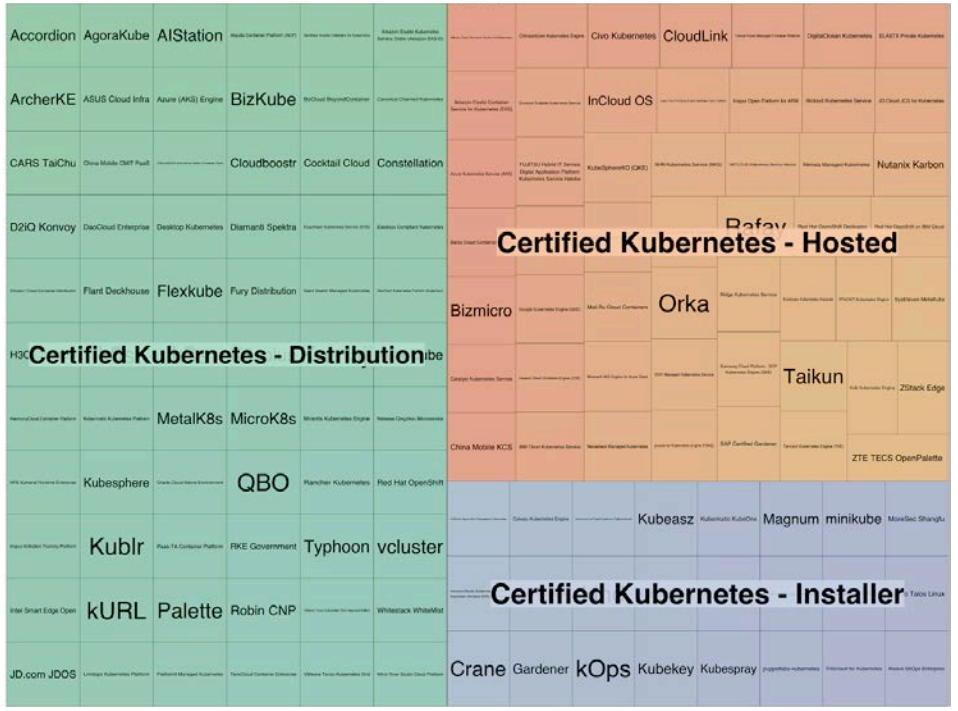
Source: EMA Quick Poll

10 Reasons to Use Multiple Distributions

An estimated 50% of organizations use multiple distributions, typically for the following reasons:

1. **Different Requirements.** Various teams within an organization might have different needs, skills, budgets, constraints, and preferences, leading them to choose a specific Kubernetes platform that best serves their specific use case.
2. **Multi-Cloud Strategy.** Some organizations might opt for a multi-cloud approach to avoid vendor lock-in or to ensure high availability, utilizing managed Kubernetes offerings from different cloud providers.
3. **Hybrid Environments.** Companies may have both on-premises and cloud infrastructure, requiring different Kubernetes platforms.
4. **Regulatory Compliance**. Some industries are subject to strict regulatory requirements that may necessitate the use of specific Kubernetes distributions or cloud providers. For example, organizations in the health care industry might need to ensure that their infrastructure is HIPAA compliant, leading them to choose different distributions depending on the specific requirements.
5. **Workload Optimization.** Different Kubernetes distributions may offer different performance characteristics and optimizations for specific workloads or applications. By using multiple distributions, organizations can ensure they're using the most suitable platform for each workload, maximizing efficiency and performance.
6. **Support for Legacy Systems.** Organizations with existing legacy systems might require specific Kubernetes distributions to maintain compatibility with older applications or infrastructure. Adopting multiple distributions can help ensure a smooth transition as the organization modernizes its infrastructure and applications over time.

141 CERTIFIED KUBERNETES DISTRIBUTIONS, INSTALLERS, AND MANAGED OFFERINGS



"Approximately 50% of organizations use multiple Kubernetes distributions."

Source: CNCF

7. **Ease of Management.** Some Kubernetes distributions come with specialized tools and features that make managing and deploying clusters easier, especially for organizations with limited in-house expertise. By adopting multiple distributions, organizations can leverage the best tools and features from each platform to simplify their Kubernetes management and operations.
8. **Cost Optimization.** Different Kubernetes distributions and managed services might have different pricing models and cost structures. By using multiple distributions, organizations can optimize their spending on infrastructure and resources, choosing the most cost-effective solution for each use case.
9. **Customization and Flexibility.** Some Kubernetes distributions offer more customization options and flexibility than others. By adopting multiple distributions, organizations can take advantage of the unique features and customizations available in each platform to tailor their Kubernetes environment to their specific needs.
10. **Testing and Comparison.** Organizations might test and compare different platforms to evaluate their capabilities before deciding on the best solution for their needs.

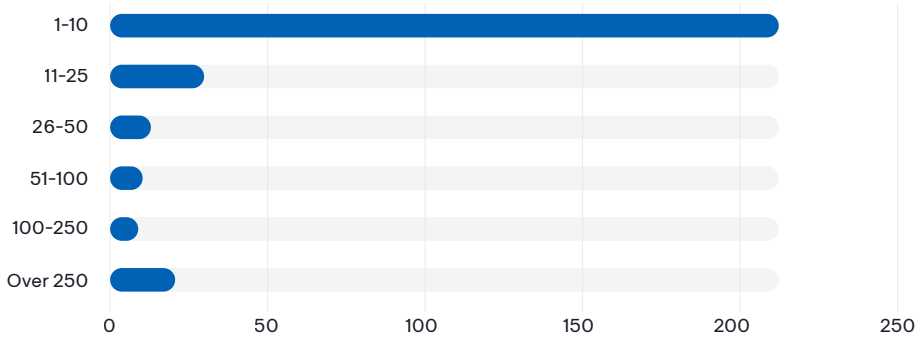
EMA Perspective

Ultimately, there isn't a single optimal Kubernetes distribution that fits all use cases. Typically, organizations grant their teams the autonomy to make choices based on specific project requirements and organizational context factors. For instance, while Google Kubernetes Engine (GKE) ranks among the most popular Kubernetes platforms, Google's Anthos hybrid cloud platform doesn't enjoy as much popularity compared with other hybrid cloud offerings, like Suse Rancher, VMware Tanzu, and Red Hat OpenShift.

DevOps teams appreciate GKE for its ease of use but often select a different Kubernetes platform for on-premises scenarios. In making this selection, they consider ease of integration with key infrastructure components, such as a specific Linux distribution, a certain hypervisor, or a particular storage platform. This same principle applies when leveraging different Kubernetes distributions in the public cloud.

A project team might choose a specific public cloud-managed Kubernetes platform to stay close to key data sources or to benefit from integration with preconfigured security services. Alternatively, a team might simply have more skills and experience with a certain cloud provider, enabling them to reuse deployment code, integrations, and automation workflows from previous projects. This can lead to decreased risk and cost for the current project. Certain enterprise applications and workloads may also come with specific requirements or recommendations regarding the cloud platform on which they should run.

HOW MANY KUBERNETES CLUSTERS ARE YOU CURRENTLY RUNNING IN PRODUCTION IN YOUR ORGANIZATION?



n=295

In our sample, 72% of organizations operate between 1 and 10 Kubernetes clusters while 7% are currently operating over 250 production clusters. Factors such as isolation for compliance reasons, support for specific hardware like GPUs, and reliability drive this distribution. Because the respondents to this survey were individuals from midsize to large organizations, they may not always have full knowledge of the total number of Kubernetes clusters within their organization. Therefore, the data in the chart might be skewed toward smaller numbers of clusters.

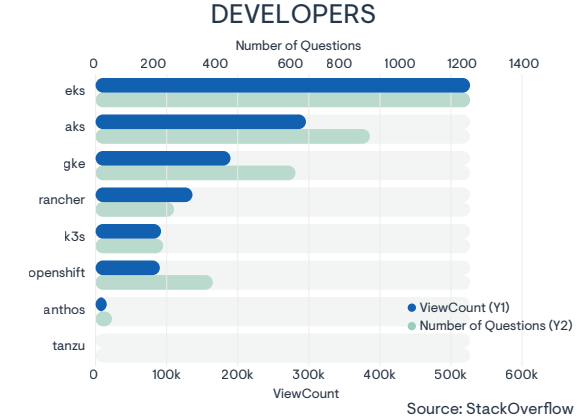
Source: EMA Poll

Market Topology

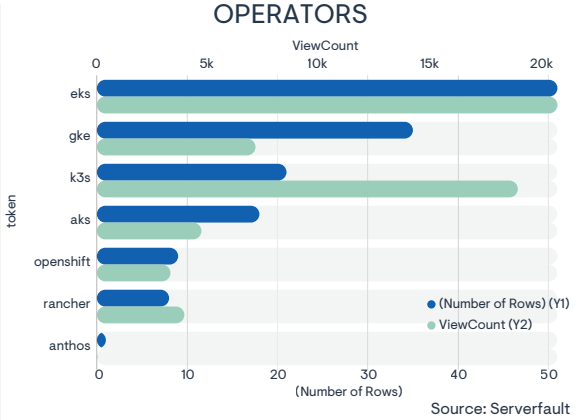
While EKS, GKE, and AKS currently represent a significant portion of the Kubernetes clusters in use, the remaining half of Kubernetes clusters is composed of cloud-independent Kubernetes platforms. Notably, Suse Rancher, VMware Tanzu, and Red Hat OpenShift emerged as popular choices for enterprises seeking to address the growing operational complexity associated with deploying Kubernetes clusters across numerous public cloud accounts. These platforms offer Kubernetes distributions and, more importantly, deliver unified governance, management, and developer services. Their ability to handle EKS, GKE, and AKS at an enterprise-grade level is particularly valuable. Enterprises are able to adopt Kubernetes rapidly without compromising on operational risk and cost thanks to policy-driven orchestration and automation for a large number of diverse Kubernetes clusters.

The strategic importance of these cloud-independent Kubernetes platforms is in their capability to streamline and simplify the management of Kubernetes clusters across different cloud providers, the corporate data center, and edge locations. By providing a unified approach to governance and management, enterprises can maintain control, enforce consistent policies, and ensure compliance across their diverse Kubernetes environments. This is crucial when dealing with a multitude of clusters deployed on various cloud platforms.

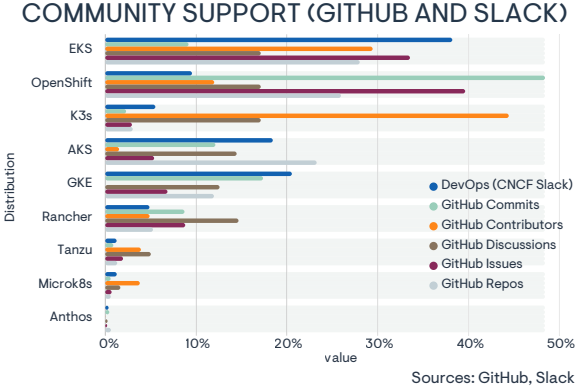
Ultimately, the strategic value of cloud-independent Kubernetes platforms like Suse Rancher, VMware Tanzu, and Red Hat OpenShift is in their ability to simplify and standardize Kubernetes management across hybrid and multi-cloud landscapes. They enable enterprises to navigate the operational complexities of managing diverse Kubernetes clusters while providing the necessary governance, automation, and scalability required for successful Kubernetes adoption at scale.



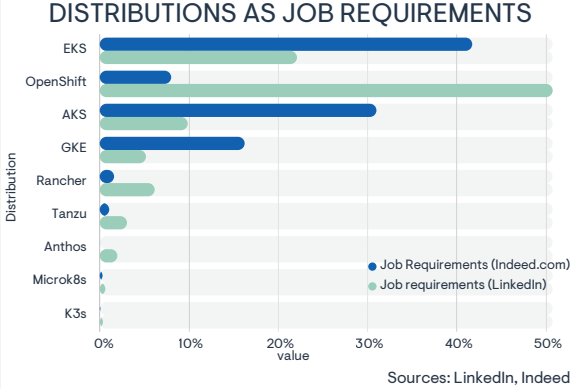
Source: StackOverflow



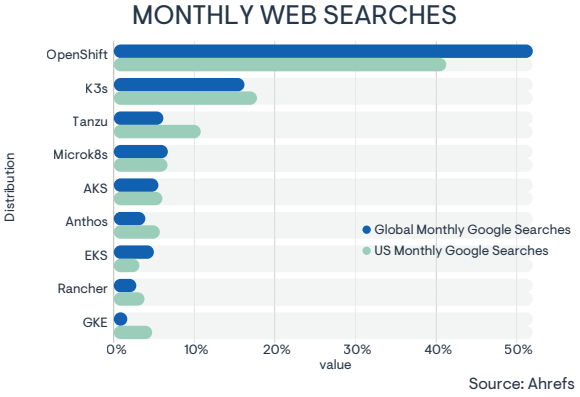
Source: Serverfault



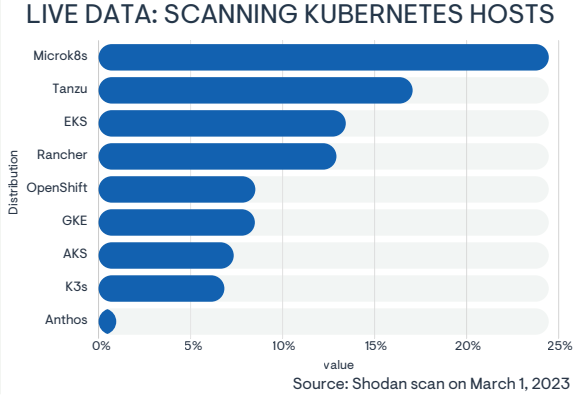
Sources: GitHub, Slack



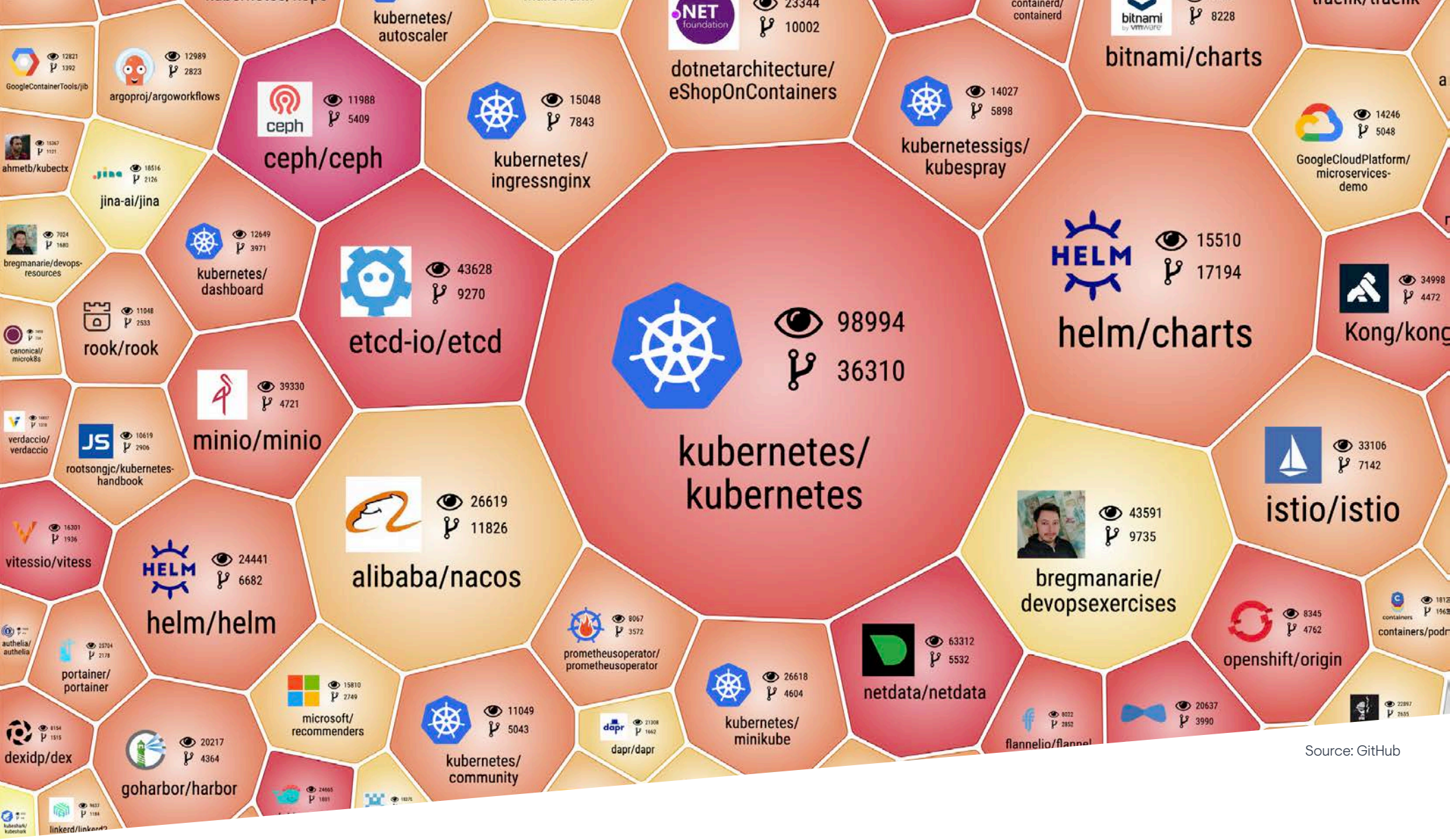
Sources: LinkedIn, Indeed



Source: Ahrefs



Source: Shodan scan on March 1, 2023



Source: GitHub

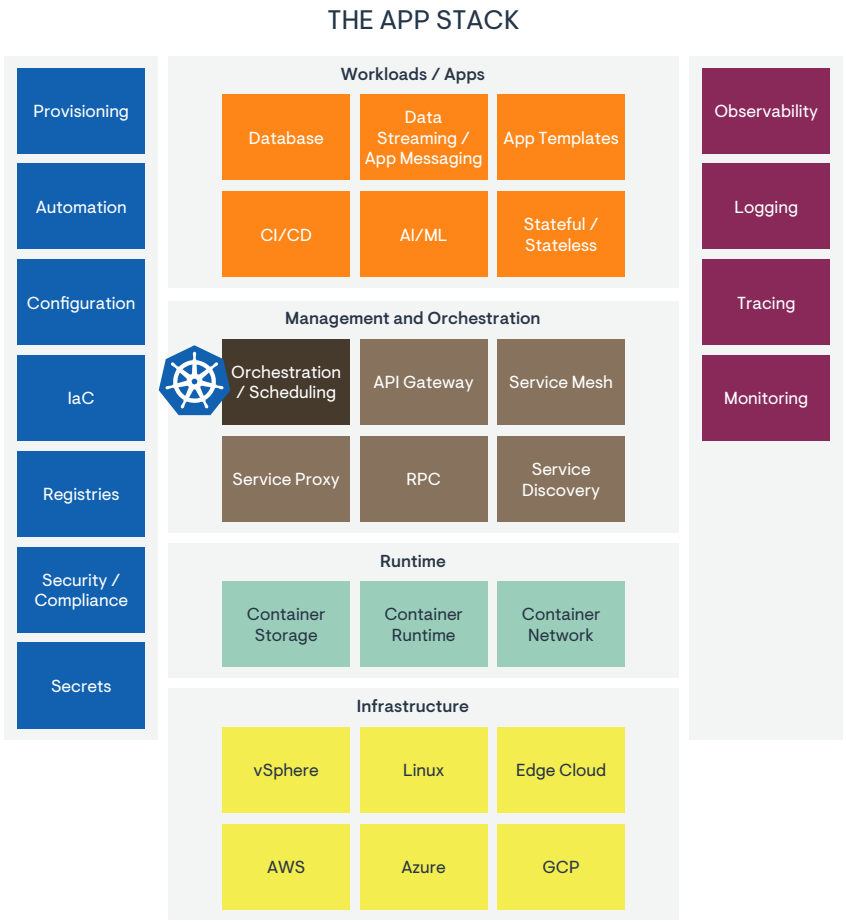
Part 2: Kubernetes Application Stack, Open Source Universe, Personas, and Technology Trends

Kubernetes is About the Stack

The true potential of Kubernetes as a container orchestration and scheduling platform can only be fully appreciated when it is examined within the context of the entire cloud-native application stack. Analyzing Kubernetes in isolation only provides a limited perspective because it is just one component of a larger, more complex ecosystem.

At its core, Kubernetes streamlines the deployment, scaling, and management of containerized applications. However, its effectiveness is significantly enhanced when it is integrated with other tools and services that form the cloud-native stack. This stack comprises various layers, including infrastructure, runtime, orchestration, observability, and application services, which collectively enable the efficient development and operation of modern applications.

The cloud-native stack is designed to be modular and extensible, allowing organizations to leverage the most suitable solutions for their specific needs. Kubernetes acts as a foundational layer in this stack, working in tandem with container runtimes, storage solutions, and networking plugins to provide the necessary orchestration capabilities. Furthermore, it seamlessly integrates with other tools and services to enhance observability, facilitate application deployment, and improve overall system resilience.

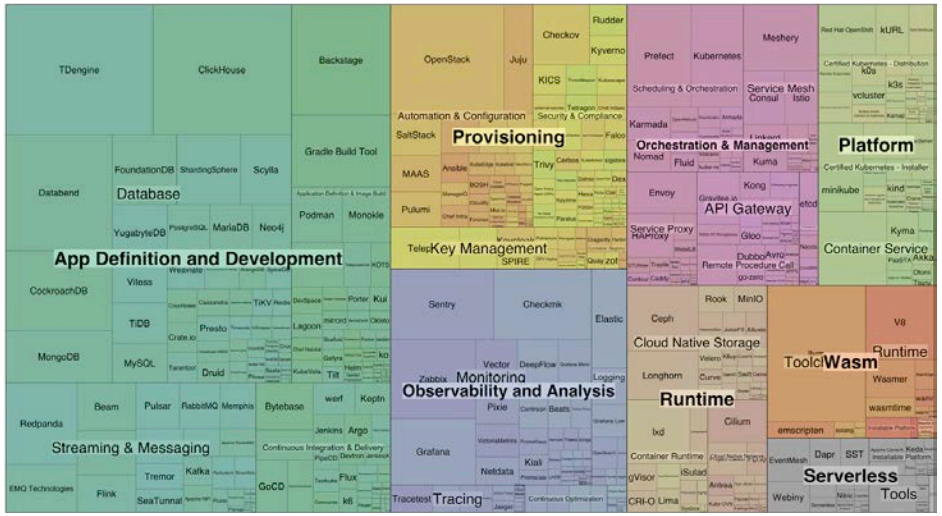


The Kubernetes Universe is Vast

There are 45 product categories with a total of 2,089 products available for organizations to choose from when assembling a Kubernetes application stack. Individual product teams often select different components for their application stack, leading to a very large number of permutations within the overall enterprise. This high level of complexity led to an explosion in the number of Kubernetes-related technologies that are part of daily challenges for developers, operators, and DevOps professionals.

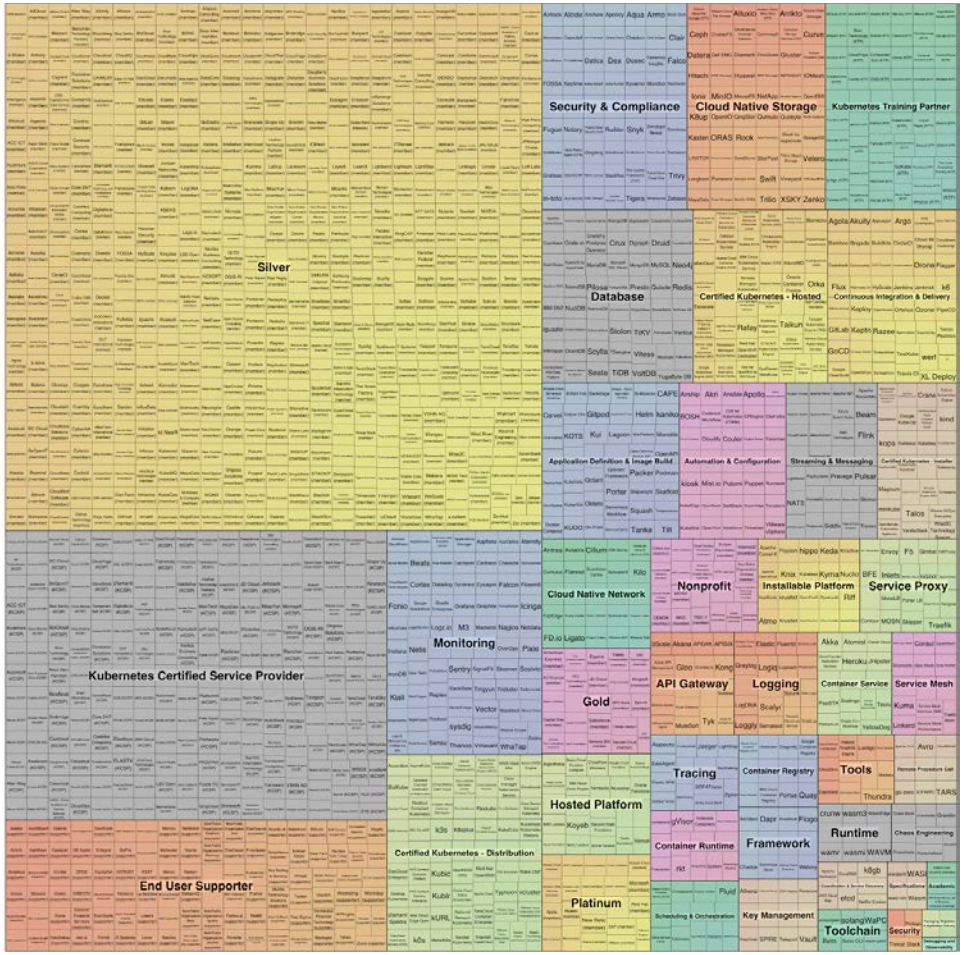
3,914,444 GITHUB STARS

614,431 CODE COMMITS

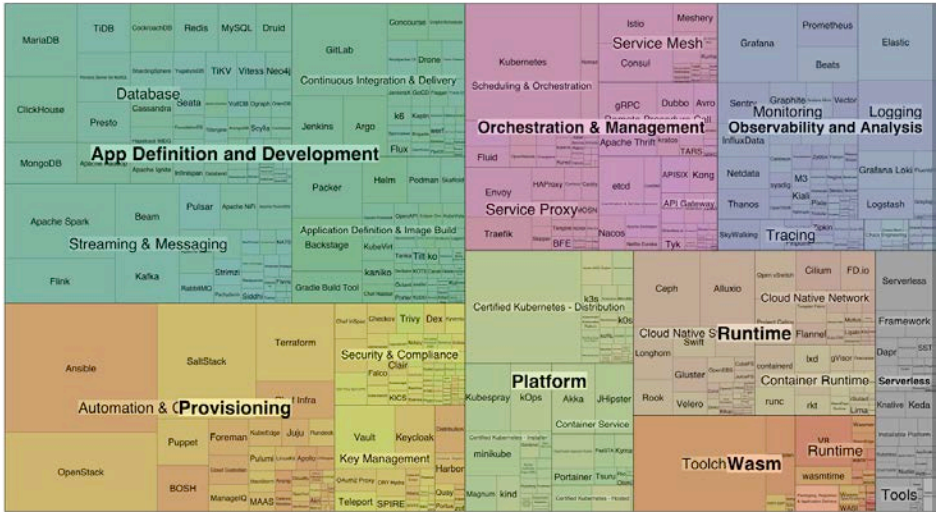


Source: GitHub

2089 CNCF PRODUCTS IN 45 CATEGORIES CAN BE PART OF THE KUBERNETES APPLICATION STACK



160,219 CONTRIBUTORS



Source: GitHub

Three Perspectives: Developers, Operators, and DevOps

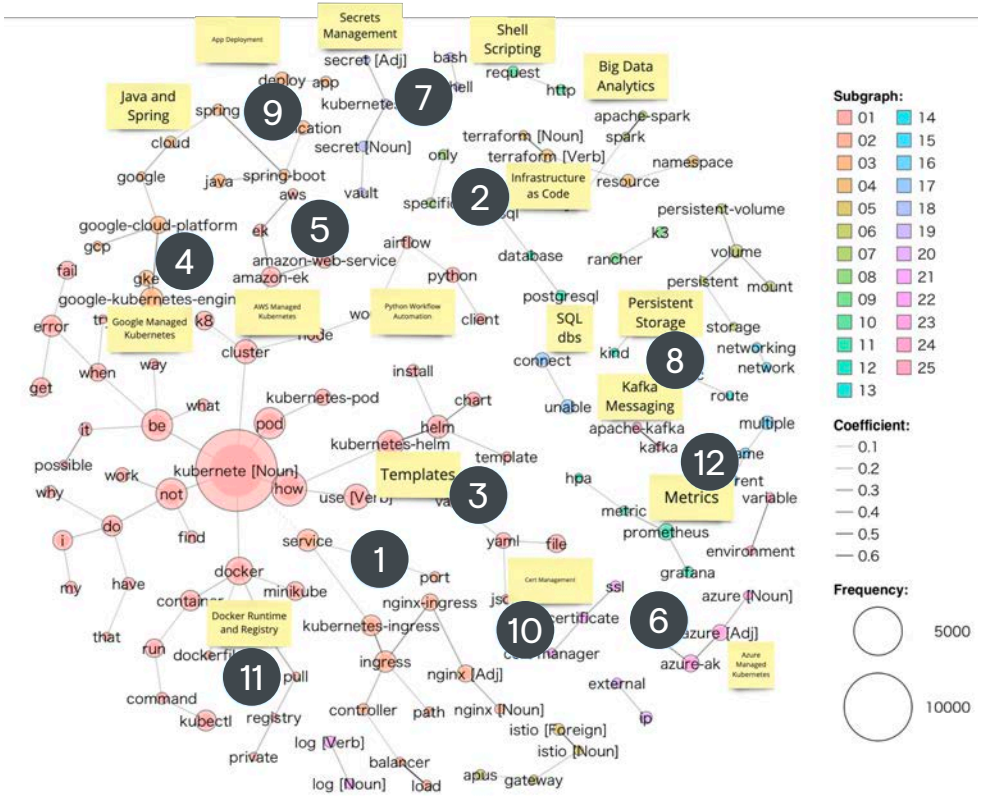
In a world of complex cloud systems, developers, operators, and DevOps professionals grapple with a variety of challenges while navigating Kubernetes, a potent container orchestration platform. This narrative captures the essence of three distinct perspectives and their experiences with Kubernetes issues.

Developers

From the perspective of developers, the journey commences with challenges such as accessing pods through ingress and service (1), defining application infrastructure as code (2), creating and configuring application templates (3), and coping with issues associated with public cloud-managed Kubernetes on AWS (4), Azure (5), and Google Cloud Platform (6). As they delve deeper into the realm of Kubernetes, developers encounter a broad spectrum of hurdles, including secrets management (7) and state (8), app deployment (9), certificate management (10), utilizing container runtimes (11), and ensuring observability (12).

From 2017 to 2022, the number of Kubernetes-related technologies for app developers surged by 166%.

DEVELOPER CHALLENGES BASED ON STACKOVERFLOW DATA 2022/2023: DEPENDENCY MODEL



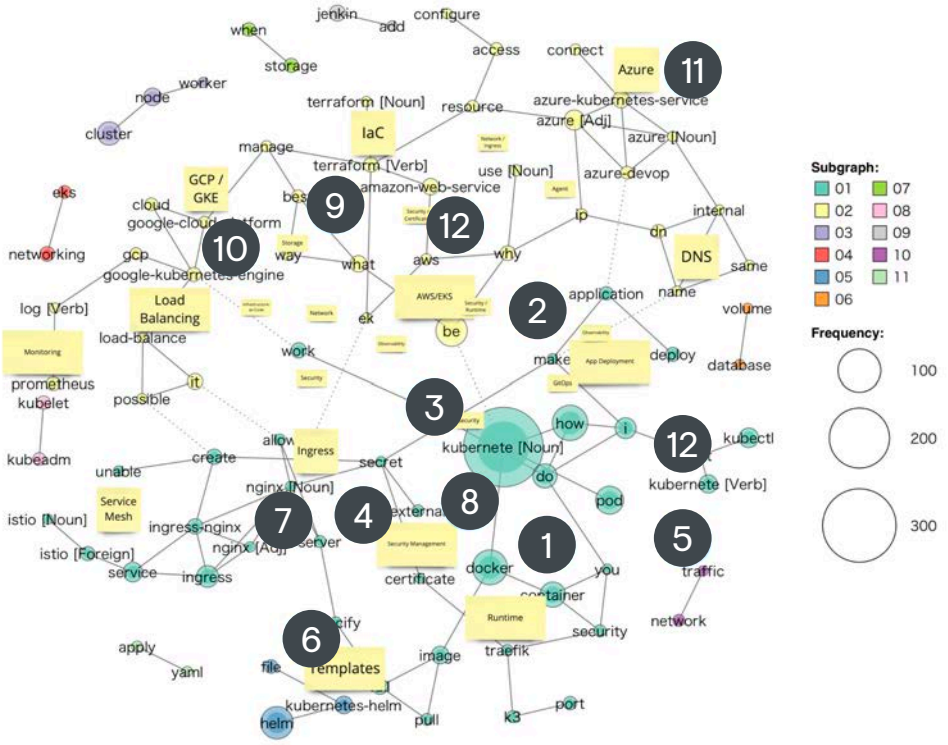
Source: StackOverflow

DevOps

DevOps professionals are tasked with bridging the gap between developers and operators. They confront Kubernetes issues that revolve around container management (1), app deployment (2), maintenance (3), security (4), and networking (5). Their expertise in related tools and technologies, such as Helm (6), Ingress (7), Docker (8), Terraform (9), GKE (10), AKS (11), and EKS (12), enables them to navigate the intricate landscape of Kubernetes successfully.

From 2017 to 2022, the number of Kubernetes-related technologies for DevOps engineers increased by 390%.

DEVELOPER CHALLENGES BASED ON SLACK DATA 2022/2023: DEPENDENCY MODEL



Source: StackOverflow

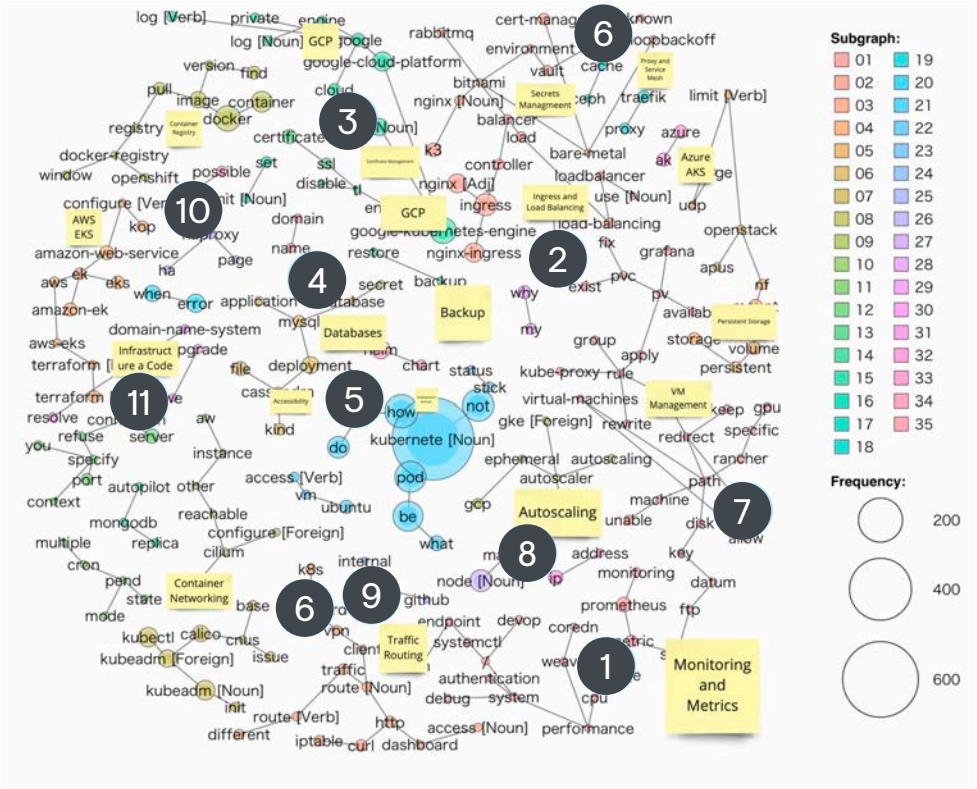
Operators

Meanwhile, operators face their own set of Kubernetes-related challenges, ranging from CoreDNS issues (1) to assigning external IPs (2) and managing certificates (3). Their daily responsibilities involve connecting to databases (4), optimizing resource usage (5), sharing disk caches (6), and deploying applications (7). They are troubleshooting issues with API servers (8), cert management (9), and external cluster connections (10) while also leveraging Terraform (11) to manage AWS EKS infrastructure.

From 2017 to 2022, the number of Kubernetes-related technologies for IT operations increased by 325%.

Together, these three perspectives weave a captivating narrative that highlights the diverse challenges faced when working with Kubernetes and the importance of collaboration in conquering these obstacles.

OPERATOR CHALLENGES BASED ON SLACKOVERFLOW DATA 2022/2023: DEPENDENCY MODEL



Source: StackOverflow

10 Critical Kubernetes-Related Technologies Over Time

Reflecting on the timeline of all 63,231 Kubernetes-related developer questions asked between 2014 and 2023, we discern a number of key trends:

NGINX ingress control tops the list of the most frequent developer issues (since 2020).

Helm, the standard Kubernetes package manager, stands as the second most important developer topic today.

Managed Kubernetes in the form of EKS, AKS, and GKE rose up the ranks swiftly, illustrating that leveraging cloud-native application principles entails more than just the management of Kubernetes itself.

Terraform quickly became the standard for consistent Kubernetes infrastructure management through GitOps in multi-cloud environments.

Python, the default language for data science projects, gained popularity as Kubernetes emerged as a platform for data science.

Jenkins lost in importance due to the rise of Kubernetes-specific DevOps tools, such as Argo and Flux.

Grafana established itself as the standard for health and performance dashboards.

MongoDB emerged as the most crucial database platform on Kubernetes.

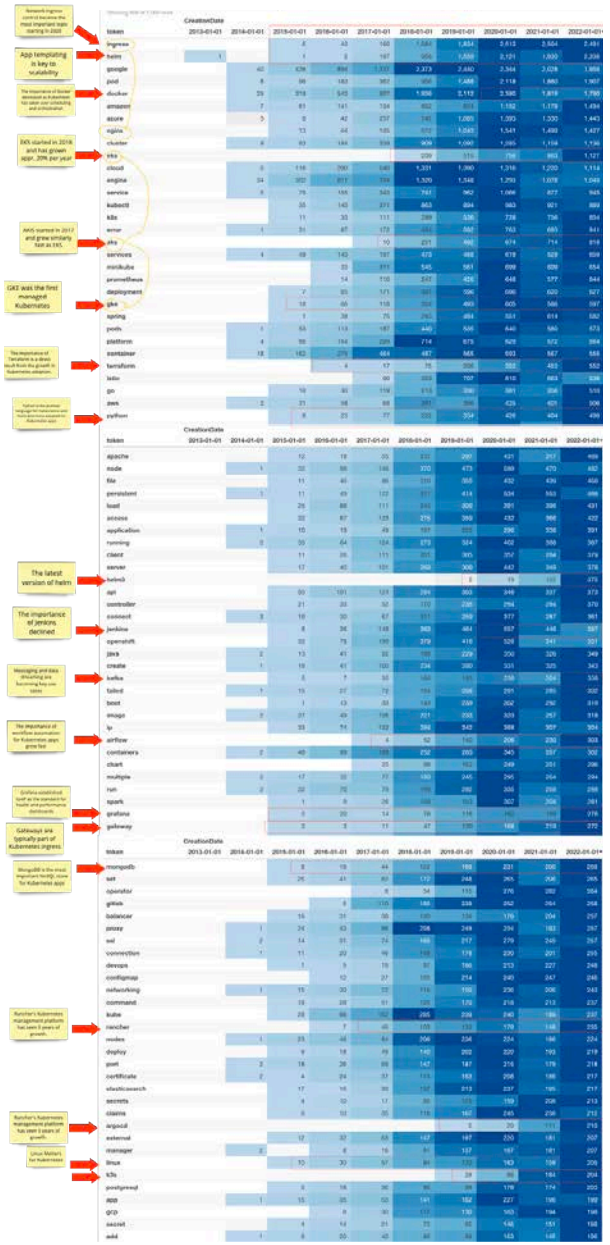
Rancher's rapid rise can be attributed to the platform's ability to provide consistent Kubernetes management across clouds.

K3s became the standard for lightweight Kubernetes deployments.

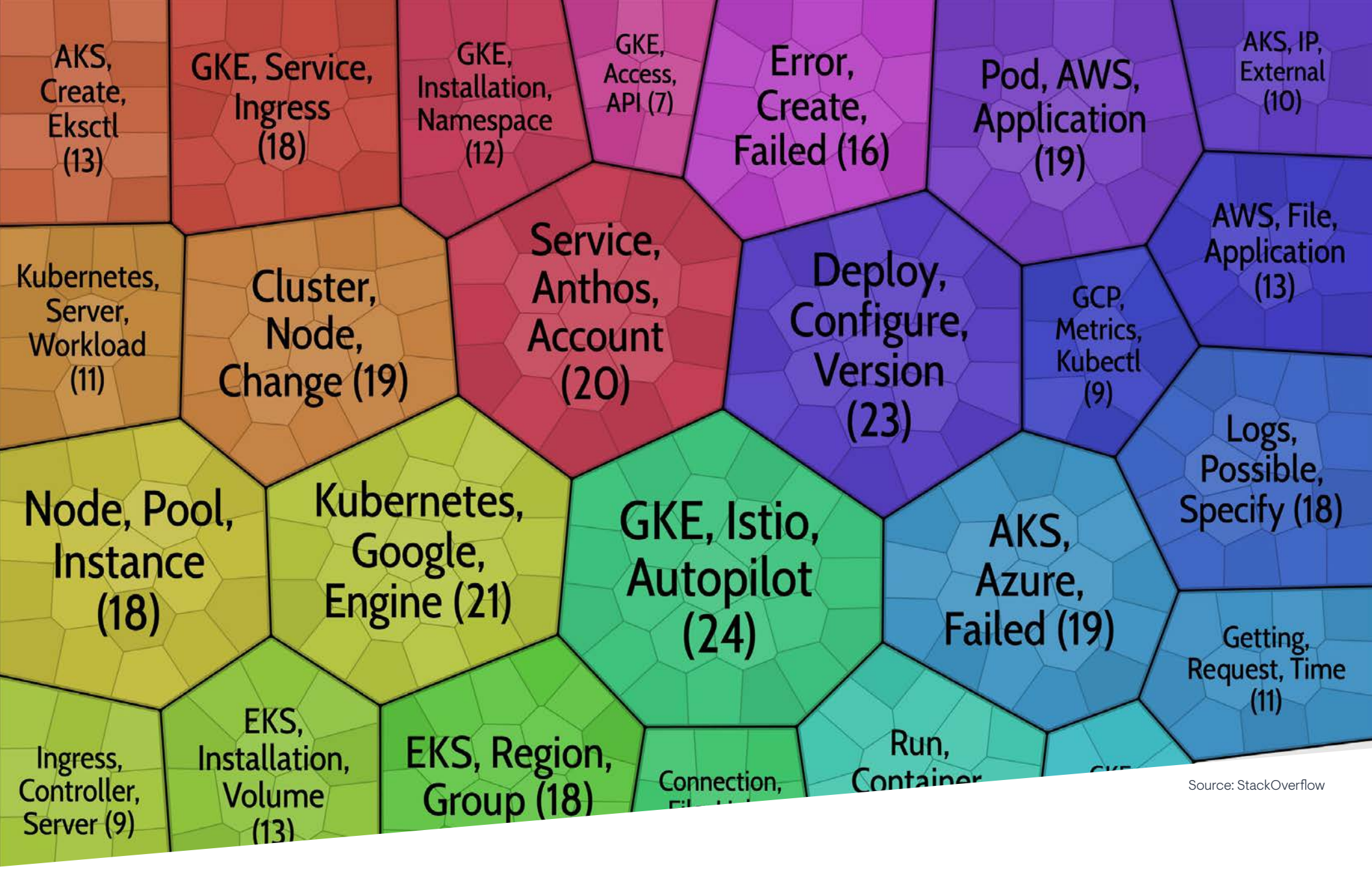
In summary, all 10 themes have one thing in common: they lay the foundation for Kubernetes adoption at scale by providing the integrations and management platforms that enable developers and operators to easily consume and manage Kubernetes clusters.

QUANTITY OF KUBERNETES-RELATED QUESTIONS ON THE STACKOVERFLOW DEVELOPER FORUM

The matrix shows a complete inventory of the quantity of Kubernetes-related questions on StackOverflow over time. The higher the number of these questions, the more prominent a specific topic appeared within the context of Kubernetes.



Source: StackOverflow



Source: StackOverflow

Part 3: Public Cloud-Managed Kubernetes

The Business Drivers and Perceived Advantages of Public Cloud-Managed Kubernetes

Product teams identify four key advantages of managed Kubernetes, such as Amazon Elastic Kubernetes Service (EKS), Google Kubernetes Engine (GKE), or Azure Kubernetes Service (AKS).

Easier Maintenance: Managed services handle the maintenance of the control plane and infrastructure, reducing the need for in-house expertise.

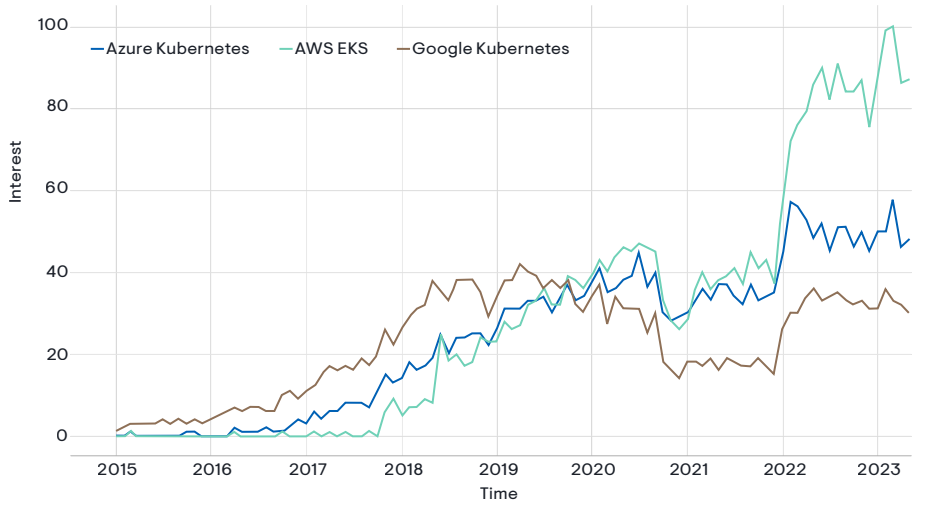
Cost-Effectiveness: Some users report that managed services prove cheaper in the long run because they eliminate the need for in-house maintenance, which can be expensive in terms of time and resources.

Stability: Managed services tend to be more stable and reliable than self-managed clusters.

Scalability: Managed services can handle scaling more effectively than self-managed clusters because cloud providers with extensive experience in managing largescale infrastructure maintain them.

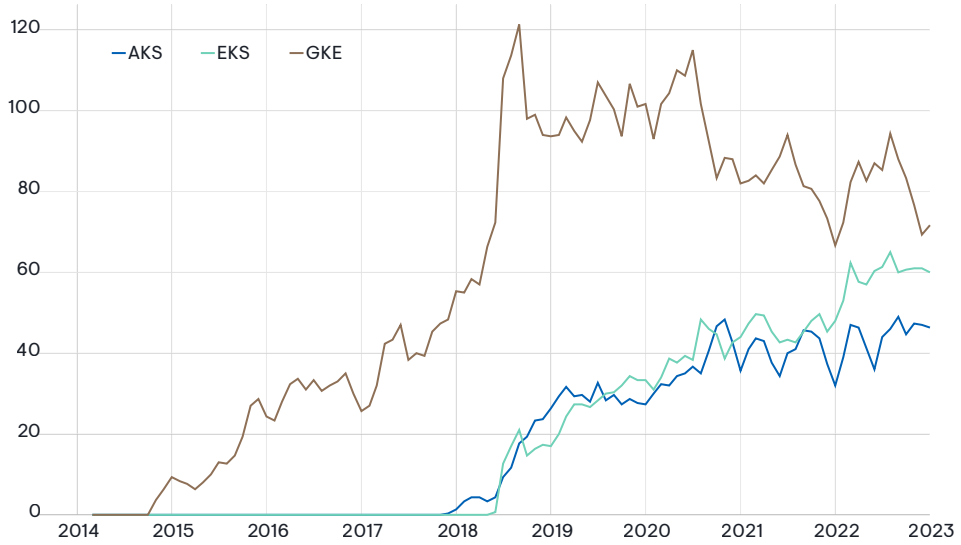
However, some users argue that self-managed clusters can be more cost-effective in certain scenarios and offer more control and flexibility. The choice between public cloud-managed and self-managed Kubernetes ultimately depends on an organization's needs, resources, and expertise.

GOOGLE TRENDS INTEREST OVER TIME



Source: Google

STACKOVERFLOW MONTHLY QUESTIONS OVER TIME



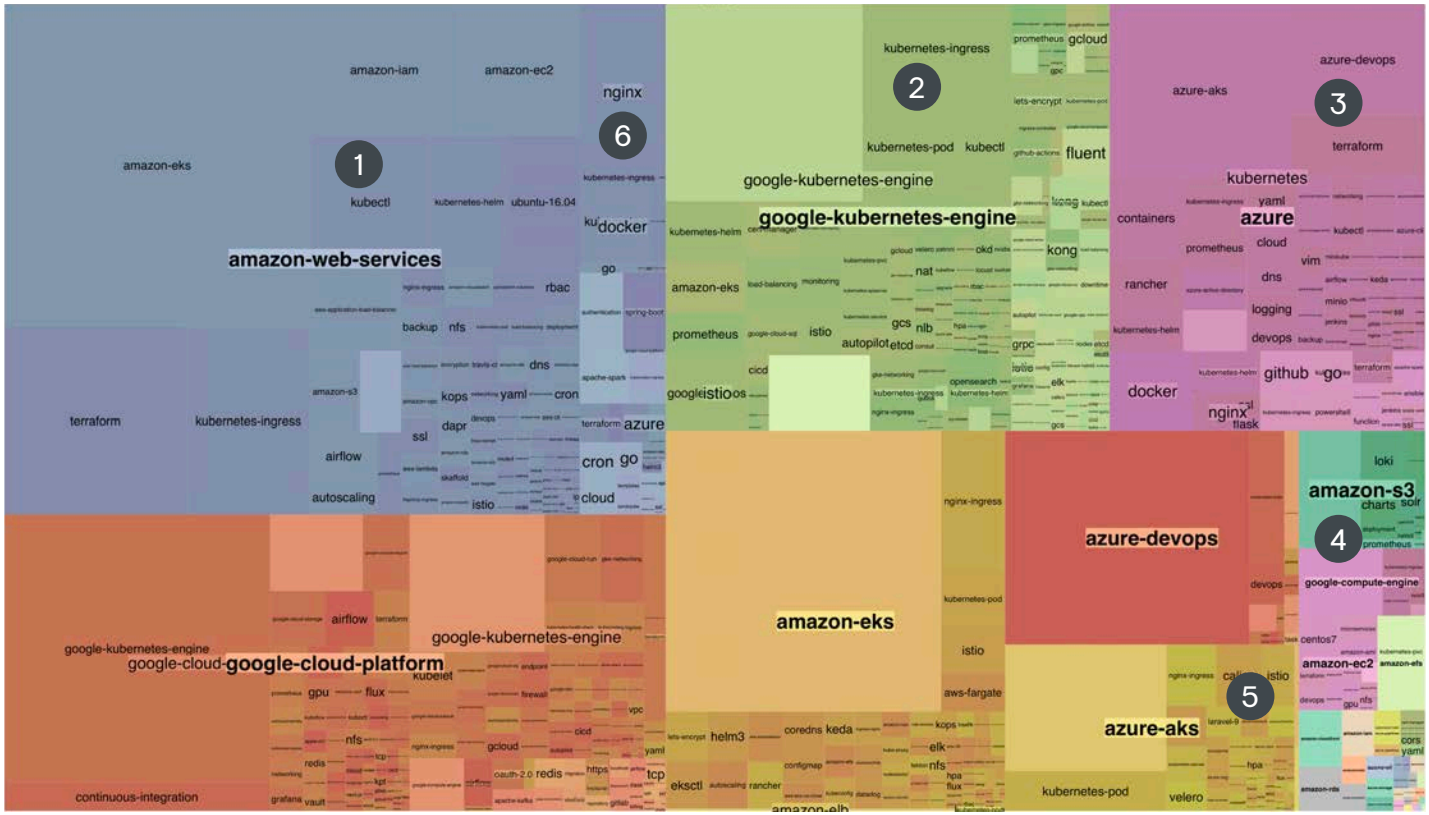
Source: StackOverflow

Top Pain Points of Public Cloud-Managed Kubernetes

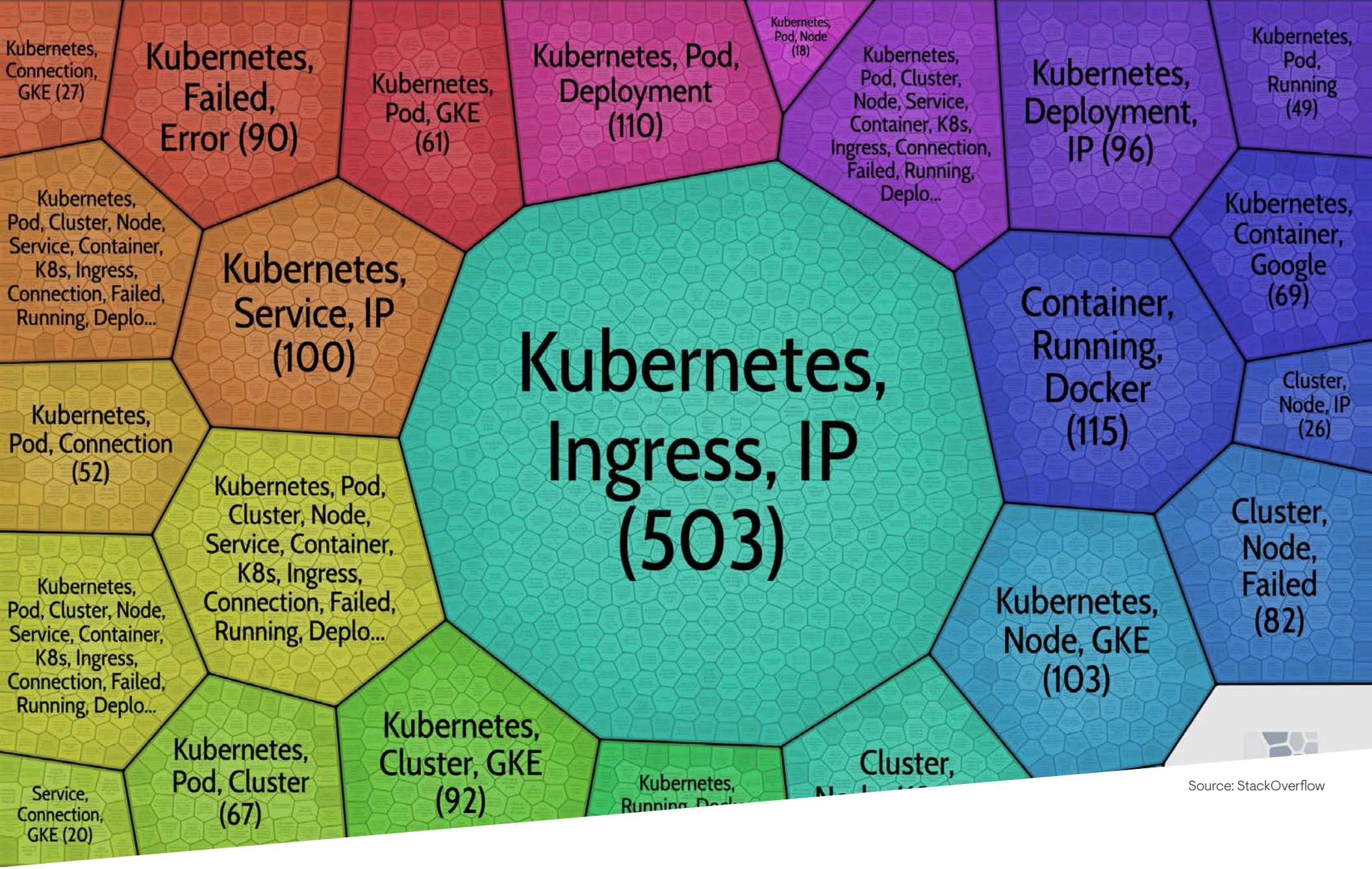
The three most popular managed Kubernetes offerings in the public cloud share eight critical clusters of developer challenges, as presented in this matrix. The diversity of these issues demonstrates that EKS, AKS, and GKE are not turnkey solutions. They still require developers to handle a variety of application-stack issues instead of allowing them to fully focus on writing business code.

EMA Perspective

Managing Kubernetes clusters in EKS, AKS, and GKE can be complex due to challenges in cluster management (1), security (2), application deployment (3), storage (4), networking (5), and integration with third-party tools (6). To succeed, organizations must invest in proper training, robust tooling, and well-defined processes for managing these environments. By effectively addressing these challenges, businesses can better leverage the benefits of container orchestration, improve scalability, enhance security, and streamline application deployment and management across various cloud platforms.



Source: StackOverflow



Source: StackOverflow

Part 4: Critical Kubernetes Challenges for Developers and Operators

Developers, IT operations, and DevOps engineers all place a similar level of importance on AWS, AKS, and GKE, with the highest priority given by DevOps engineers. This suggests that cloud platforms and services are a key focus across all three roles.

Network is a higher priority for IT operations and DevOps engineers compared to developers, indicating that network management and infrastructure are more critical aspects of their daily responsibilities. In particular, DevOps engineers place the greatest emphasis on this area.

Deployment is a standout area of focus for DevOps engineers, who prioritize it significantly more than both developers and IT operations. This highlights the unique role that DevOps engineers play in bridging the gap between development and operations teams by streamlining the deployment process.

Security is another important aspect for all three roles, with IT operations and DevOps engineers placing more emphasis on it than developers. This reflects the increased responsibility these roles have in ensuring the security of systems and applications.

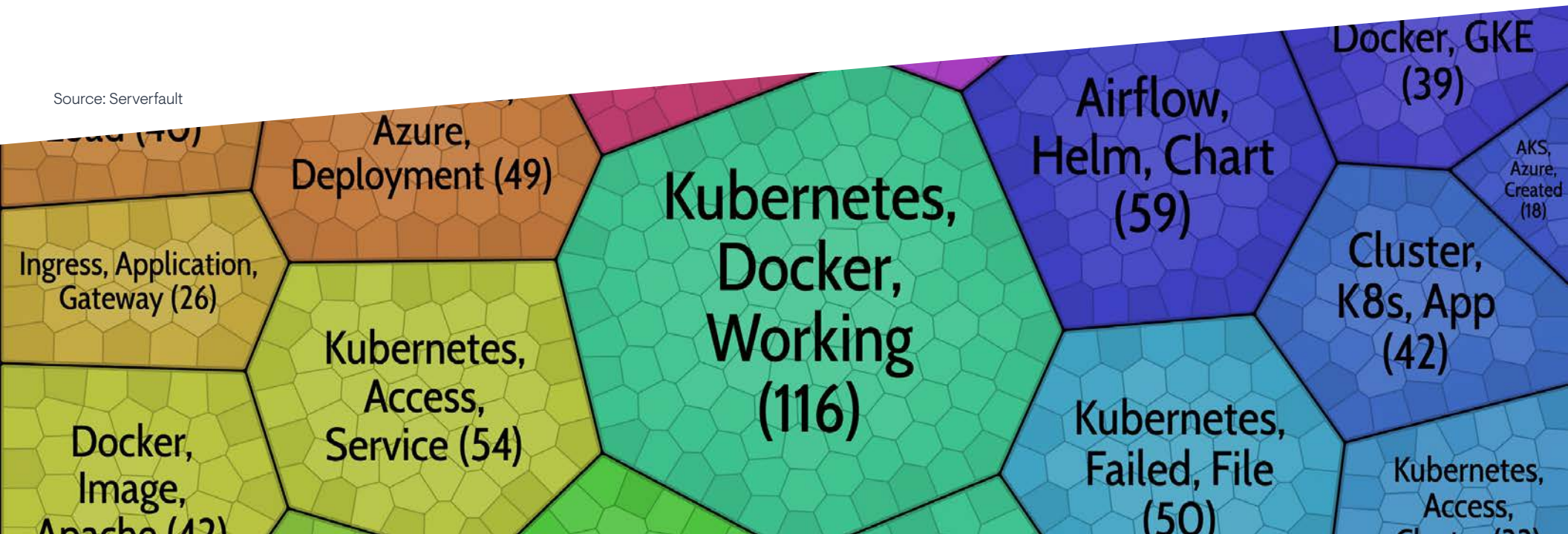
Storage and configuration are more important for DevOps engineers than developers and IT operations, demonstrating their role in managing and maintaining the systems that support application development and deployment.

Observability, API, server, automation, and data are also areas that DevOps engineers prioritize more than developers and IT operations. This showcases the diverse responsibilities of DevOps engineers who work to improve monitoring, integration, server management, process automation, and data handling across the development and operations spectrum.

Lastly, Linux is more important for IT operations compared with developers and DevOps engineers, suggesting that IT operations professionals may be more involved in managing Linux-based systems and infrastructure.

In conclusion, while all three roles share some common priorities, such as AWS, AKS, GKE, and security, each role also has distinct areas of focus. Developers prioritize aspects related to application development, IT operations emphasize network and Linux management, and DevOps engineers bridge the gap between the two with an emphasis on deployment, configuration, and system management.

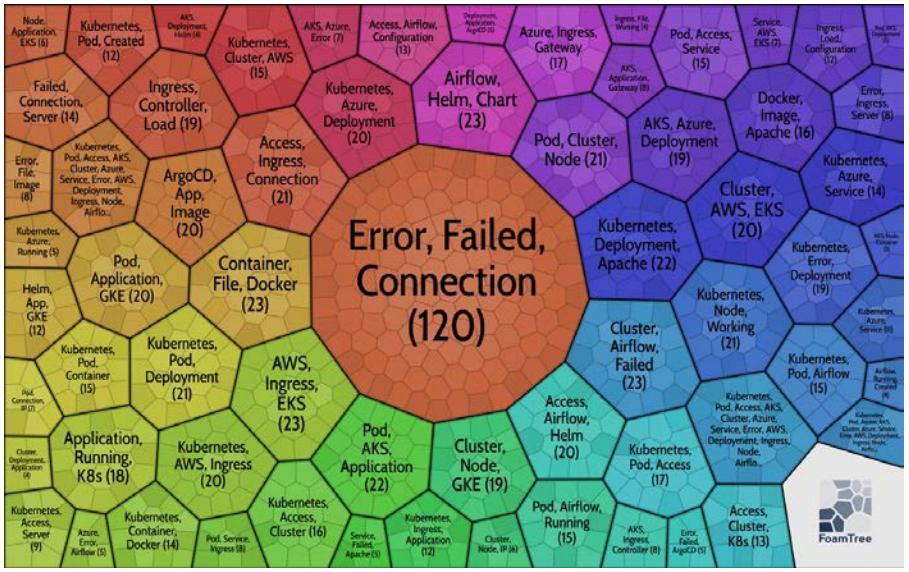
Source: Serverfault



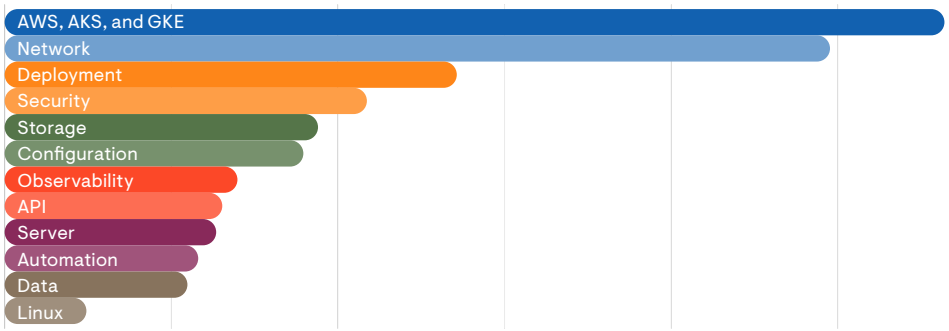
Top 10 Critical Kubernetes-Related Challenges for Application Developers

Learning from real-world developer challenges

1. **Public Cloud Managed Kubernetes:** EKS, AKS, and GKE are managed Kubernetes services offered by AWS, Azure, and Google Cloud, simplifying cluster management, scaling, and upgrades for developers.
2. **Networking:** Kubernetes networking deals with pod-to-pod communication, including ingress and egress traffic, network policies, and service discovery, which can become complex as applications scale.
3. **Deployment:** Deploying applications in Kubernetes involves creating and managing resource manifests, such as deployments and StatefulSets, ensuring smooth rollouts and updates.
4. **Security:** Kubernetes security encompasses role-based access control (RBAC), pod security policies, and network policies to protect cluster resources and maintain isolation for applications.
5. **Storage:** Kubernetes storage involves configuring persistent volumes (PVs) and persistent volume claims (PVCs) for stateful applications, which can be backed by various storage providers and options.
6. **Configuration:** Application configuration in Kubernetes requires the usage of ConfigMaps and secrets for managing environment variables and sensitive data, enabling secure and dynamic application configurations.
7. **Observability:** Monitoring, logging, and tracing are critical for maintaining and troubleshooting applications in Kubernetes, utilizing tools like Prometheus, Grafana, and Jaeger to understand and analyze system performance.
8. **API:** The Kubernetes API enables communication with the Kubernetes cluster, allowing developers to create, modify, and manage resources programmatically.
9. **Server:** The Kubernetes control plane is composed of multiple components, such as the API server, etcd datastore, and controllers, which work together to manage the cluster state.
10. **Automation:** Kubernetes automates scaling, rolling updates, self-healing, and load balancing, simplifying application deployment and management.



Source: StackOverflow



Based on the number of developer questions submitted on the StackOverflow developer forum

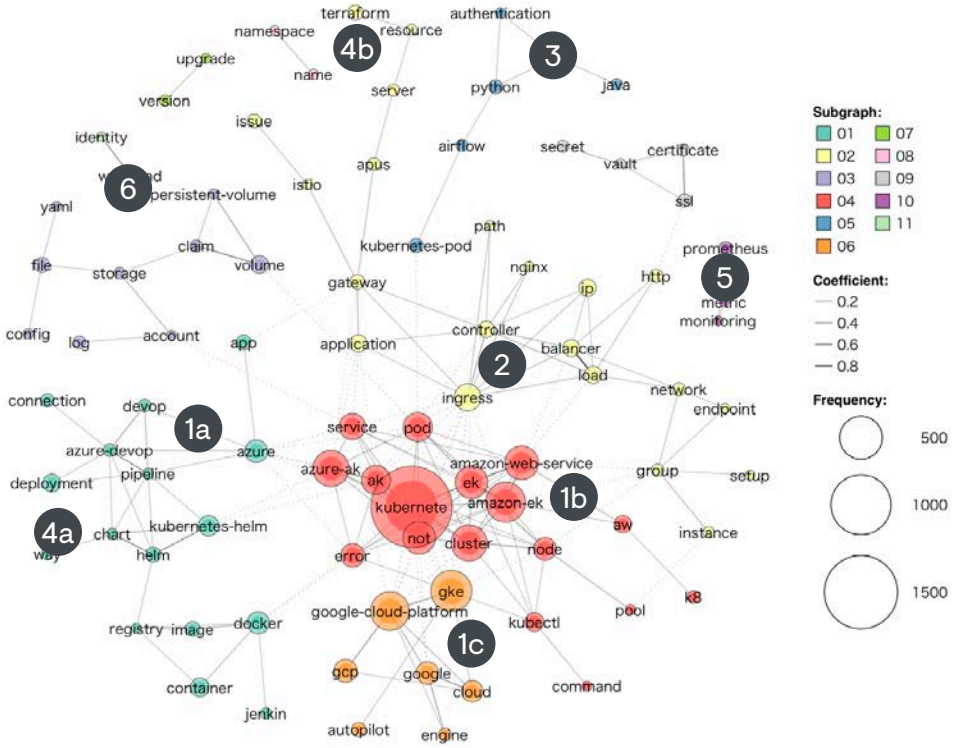
#1 Challenge: Public Cloud-Managed Kubernetes

5 Key Topics

1. **Cloud-Specific Topics:** AWS, AKS, and GKE all offer different deployment, runtime, configuration, and integration options that customers have to manage and control.
2. **Ingress, Load Balancing, and Networking:** Ingress controllers, load balancing, and networking in Kubernetes clusters, including issues with NGINX, ALB, and GKE ingress, as well as load balancing configurations for various cloud providers, are key challenges.
3. **Security, Authentication, and Authorization:** Role-based access control (RBAC), IAM integration, and handling secrets in Kubernetes clusters are critical customer challenges.
4. **Application Deployment, Helm, and Terraform:** Deploying applications, managing Kubernetes resources using Helm charts, and managing infrastructure using Terraform are important challenges. Managing resources and configurations for Kubernetes clusters and applications is often time-consuming.
5. **Observability:** Setting up monitoring with Prometheus, configuring log collection with Fluent Bit, and managing logs and metrics are essential challenges that come with the adoption of AWS, AKS, and GKE.

EMA Perspective

While EKS, AKS, and GKE are popular Kubernetes offerings, it is important to remember that they are not fully managed or turnkey products. Platform engineering teams still have to work on numerous critical topics in the areas of networking, storage, security, deployment, and observability.



Source: StackOverflow

Problem Area: Cost Control

There are several cost management challenges related to Amazon EKS, Google GKE, and Azure AKS. A few major challenges include:

1. **Cluster Sizing and Node Provisioning.** Choosing the right size and type of nodes can be a challenge because it requires accurate estimation of the resources required for your workloads. Overprovisioning can lead to higher costs, while under-provisioning can result in performance issues.
2. **Scaling and Auto-Scaling.** All three services offer auto-scaling features that adjust cluster resources based on demand. However, fine-tuning these settings can be difficult and improper configuration can lead to increased costs.
3. **Resource Wastage.** Wasted or unused resources are a common issue that can drive up costs. This includes unused compute capacity, idle load balancers, and unnecessary storage volumes.
4. **Multi-Cloud Deployment.** If you're using more than one cloud provider (such as EKS, GKE, and AKS), managing costs across different services and providers can be complex.
5. **Monitoring and Alerting.** Maintaining visibility of resource utilization and setting up appropriate cost and performance alerts is crucial for cost management. All three cloud providers offer various monitoring tools, but choosing the right combination for your needs can be challenging.
6. **Data Transfer Costs.** Data transfer between clusters or different cloud providers can be costly. You should be aware of the data transfer costs associated with using EKS, GKE, and AKS.

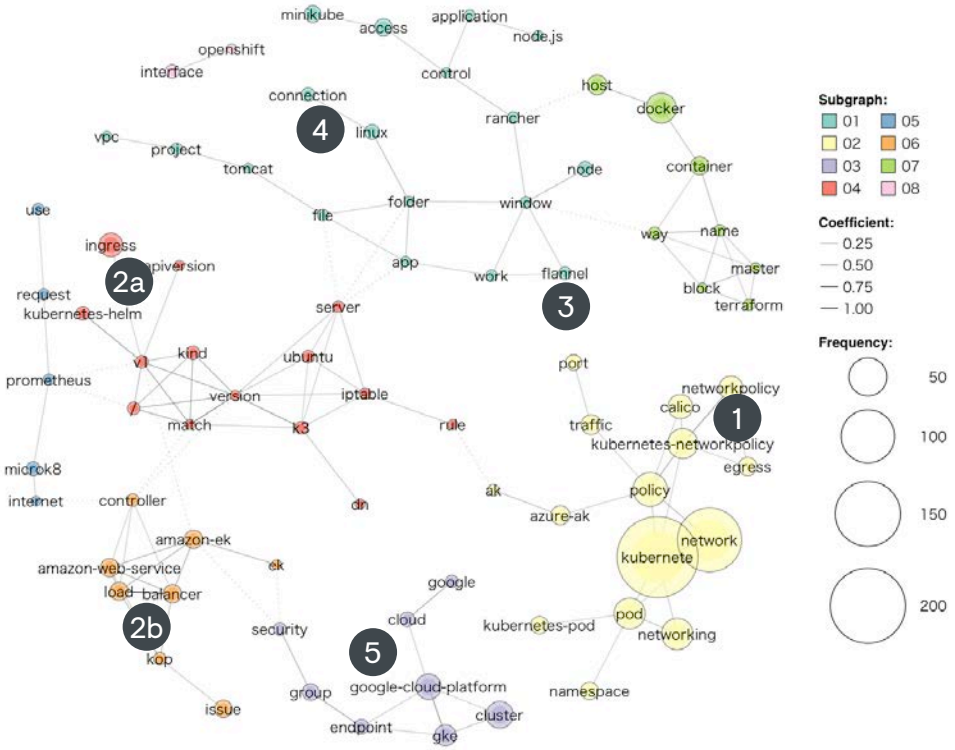
#2 Challenge: Kubernetes Networking

5 Key Topics

1. **Network Policies and Traffic Control:** This category includes issues related to Kubernetes network policies, ingress and egress traffic control, Calico network policies, and controlling network access between different Kubernetes resources.
2. **Load Balancers and Ingress Issues:** These posts discuss issues and questions related to load balancers, such as AWS Network Load Balancer and GCP Network Load Balancer, as well as Kubernetes Ingress and ingress controllers like NGINX.
3. **Networking Setup and Configuration:** This category contains posts about setting up and configuring Kubernetes networking, including topics like Flannel, kops, Minikube, and network interfaces.
4. **Network Errors and Troubleshooting:** These posts involve network-related errors and troubleshooting in Kubernetes, such as connection issues, unreachable errors, and network plugin initialization problems.
5. **Private and Internal Networking:** This category includes posts discussing private or internal networking in Kubernetes, such as GKE private clusters, network access between resources within a cluster or VPC, and private service connections.

EMA Perspective

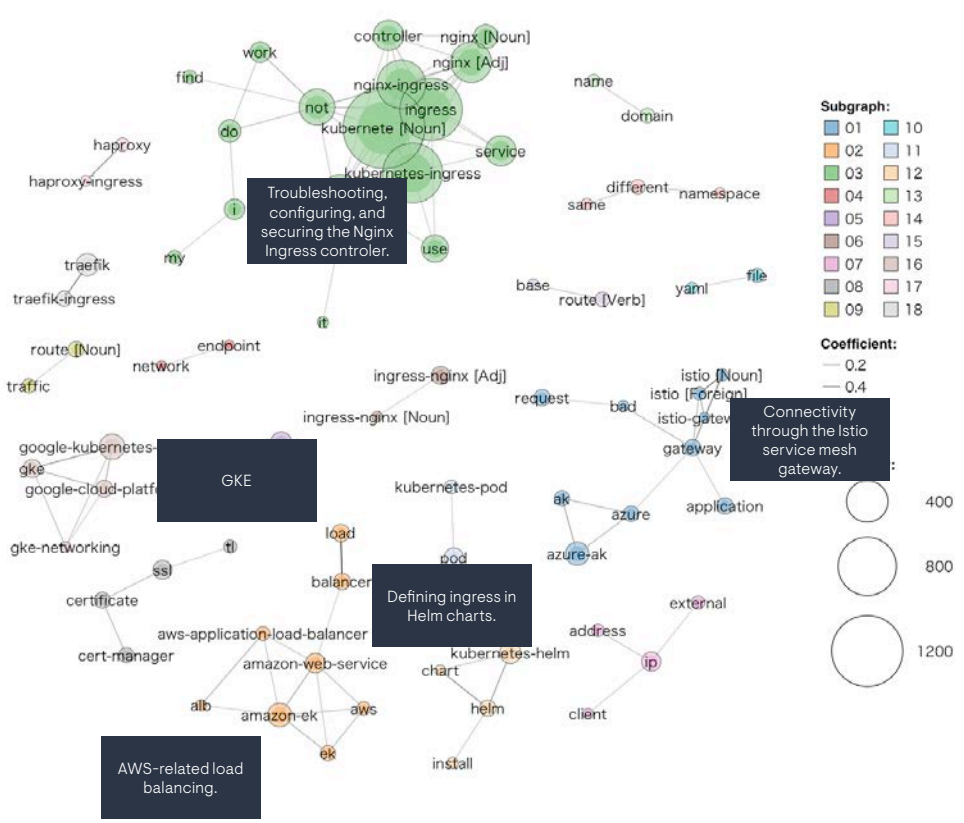
It is evident that Kubernetes networking is a vital aspect of managing containerized applications. The discussion categories highlight the importance of understanding and implementing network policies, traffic control, load balancing, ingress management, and network setup and configuration. Additionally, network errors and troubleshooting, as well as private and internal networking, play a crucial role in ensuring the efficient operation of Kubernetes clusters. By investing in resources and expertise to address these areas, businesses can optimize the performance, security, and reliability of their Kubernetes-based applications, thus enhancing their overall infrastructure and service delivery.



Source: StackOverflow

Network Problem Area: Ingress Control

Ingress control topics include troubleshooting issues with Kubernetes ingress, configuring ingress, and SSL certificate issues. Troubleshooting includes 404 errors with the NGINX Ingress Controller, issues with the Ingress host URL and Ingress NGINX, and problems with Ingress on Minikube and K3s. Configuring ingress covers topics such as exposing the Traefik v2 dashboard and building an ingress controller with central certificate and authentication management. Issues with SSL certificates include problems with cert manager, AWS EKS, and Google-managed SSL certificates. There are also issues with load balancers and ingress controllers, such as problems with external IPs and network load balancers, and failed health checks with GKE Ingress.



Source: StackOverflow

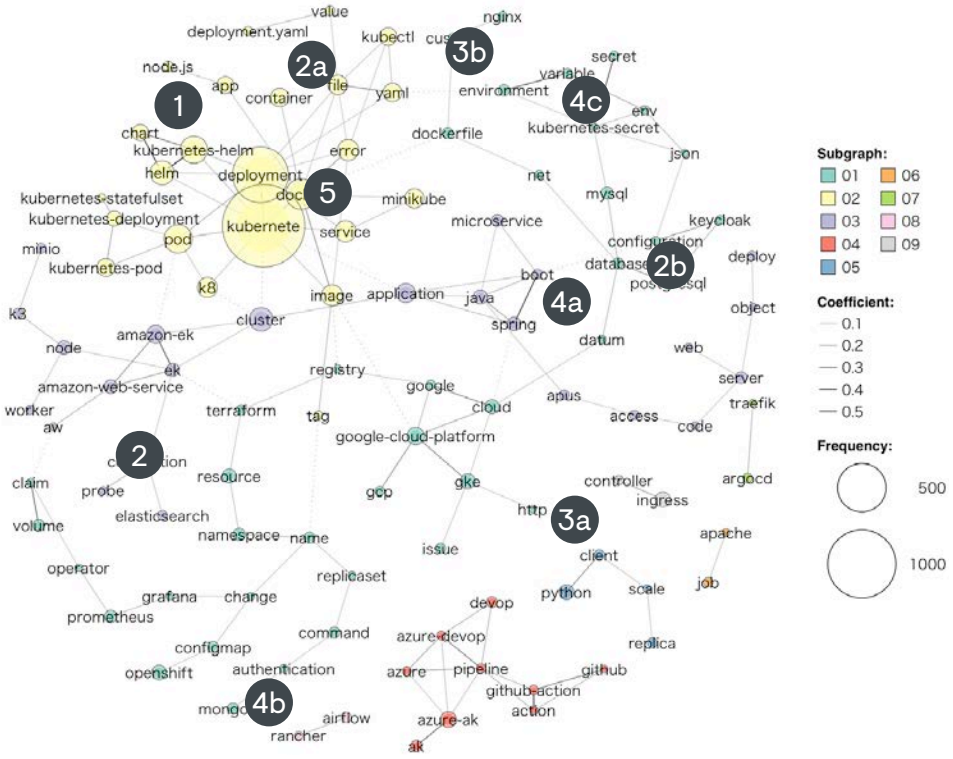
#3 Challenge: Cluster and Application Deployment

5 Key Topics

1. **Argo CD and Helm-Related Issues:** This category covers posts related to Argo CD, a declarative GitOps tool for Kubernetes, and Helm, a package manager for Kubernetes. Topics include configuring Argo CD with TLS, using Argo CD in CI pipelines, and managing Helm chart deployments.
2. **Cluster Deployment and Configuration:** These posts discuss various aspects of deploying and configuring applications on Kubernetes, such as creating and updating deployments, attaching ConfigMaps, setting environment variables, and using kubectl commands to manage deployments.
3. **Networking and Ingress Issues:** This category includes posts about networking and ingress-related problems in Kubernetes clusters, such as deploying and exposing services, configuring ingress, and troubleshooting connectivity issues.
4. **Storage and Database Deployments:** Posts in this category focus on deploying and managing databases and storage systems in Kubernetes, including topics like deploying MongoDB, PostgreSQL, and SQL Server, as well as managing persistent volumes and storage classes.
5. **Cluster and Pod Management:** This category covers posts related to managing Kubernetes clusters and pods, such as spreading pods across nodes, using pod anti-affinity, performing rolling restarts, and monitoring deployments with client-go and the Python Kubernetes client.

EMA Perspective

From a business perspective, these points emphasize the need for effective management of Kubernetes-based applications and infrastructure. The focus areas include efficient deployment and management of applications and packages, successful configuration of applications within clusters, addressing networking and ingress issues, implementing and managing storage and database solutions, and optimizing cluster and pod management. By addressing these aspects, businesses can enhance critical success factors, such as release speed, quality, scalability, and reliability.



Source: StackOverflow

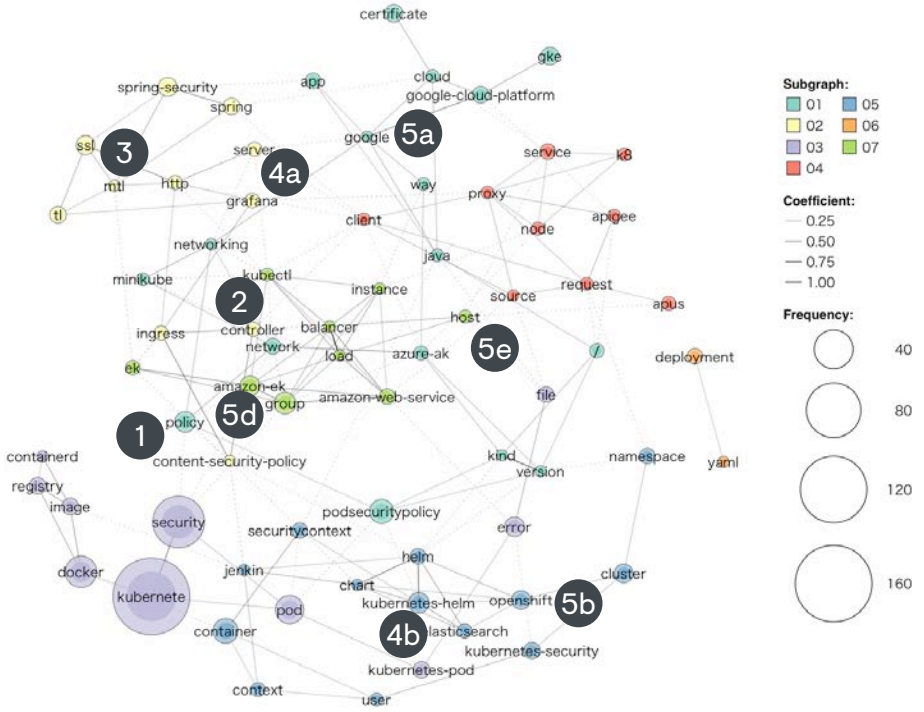
#4 Challenge: Security

5 Key Topics

1. **Kubernetes Security and Policies:** This category includes topics related to securing Kubernetes clusters, configuring security policies, and network policies. Posts discuss issues like applying Linux policies, using Calico and pod security groups together, and setting up authentication and authorization.
2. **Ingress, Load Balancers, and Networking:** This category covers security questions and issues related to ingress controllers, load balancers, network security groups, and networking configurations in Kubernetes. Topics include configuring content security policies, using ingress with TLS termination, and working with AWS Load Balancer Controller.
3. **TLS, Certificates, and Authentication:** This category addresses concerns about using TLS, SSL certificates, and authentication mechanisms within Kubernetes clusters. Discussions include enabling HTTPS traffic, mutual TLS, and working with insecure registries or self-signed certificates.
4. **Running and Securing Applications in Kubernetes:** This category encompasses posts about deploying and securing applications on Kubernetes, focusing on aspects like container security, securing application communication, and managing access to external APIs. Examples include setting up secure communication with Istio and securing Grafana ingress.
5. **Platform-Specific Security Issues:** This category deals with security-related questions and issues specific to platforms like GKE, EKS, AKS, OpenShift, and DigitalOcean Kubernetes. Topics include patching GKE-managed instance groups for package security updates, disabling user impersonation, and setting unsafe sysctl during pod deployments.

EMA Perspective

These discussions revolve around enhancing Kubernetes security via various mechanisms, including proper configuration of security policies, application of SELinux policies, and usage of Calico alongside pod security groups. They also delve into networking concerns, like ingress controllers, load balancers, and network configurations, managing TLS and certificates, ensuring secure application deployment and communication, and addressing platform-specific security issues in GKE, EKS, AKS, OpenShift, and DigitalOcean Kubernetes.



Source: StackOverflow

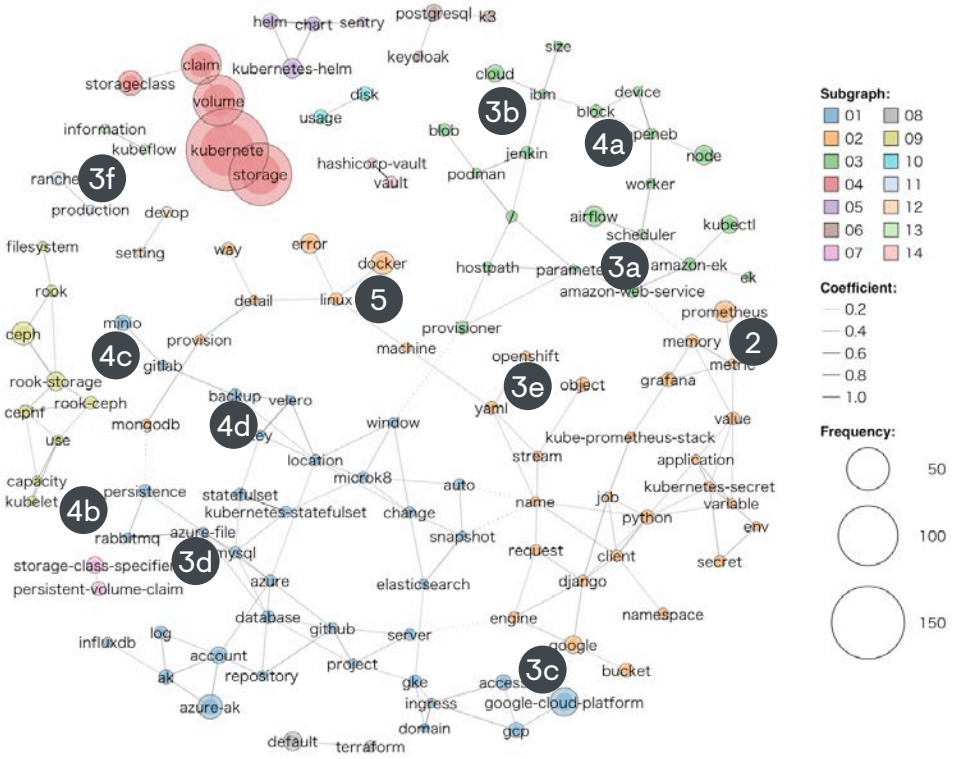
#5 Challenge: Storage

5 Key Topics

1. **Kubernetes Storage in General:** This category includes posts about problems and questions related to storage in Kubernetes, such as persistent volumes, storage classes, and PVCs. Topics include resizing PVCs, configuring storage retention, and dealing with issues related to specific storage solutions like Ceph, Gluster, and NFS.
2. **Ephemeral Storage:** These posts involve ephemeral storage in Kubernetes, discussing issues such as how to monitor disk usage, how to configure limits, and how to troubleshoot problems related to ephemeral storage, including eviction and low-resource scenarios.
3. **Cloud Provider-Specific Storage:** These posts address storage issues and questions specific to various cloud providers, including AWS EKS, Azure AKS, and GCP GKE. Topics include configuring storage classes and persistent volumes, using specific storage solutions like Azure Blob Storage and Google Cloud Storage, and migrating storage classes.
4. **Third-Party Tools and Integrations:** This category includes posts about integrating third-party tools and services with Kubernetes storage, such as MinIO, Loki, Argo, and Airflow. Discussions involve setting up storage for these tools, monitoring disk usage, and configuring remote logging.
5. **Miscellaneous Storage Questions and Issues:** These posts cover a variety of storage-related questions and issues not fitting neatly into the other categories, such as configuring remote logging for Airflow, using local storage with Rancher, and using host machine local hard drive storage in Minikube.

EMA Perspective

The challenges organizations face in Kubernetes storage management span general, ephemeral, and cloud provider-specific storage, as well as third-party tool integration and other miscellaneous issues. Addressing these challenges is crucial to improving storage infrastructure, ensuring better app efficiency and performance across the organization.



Source: StackOverflow

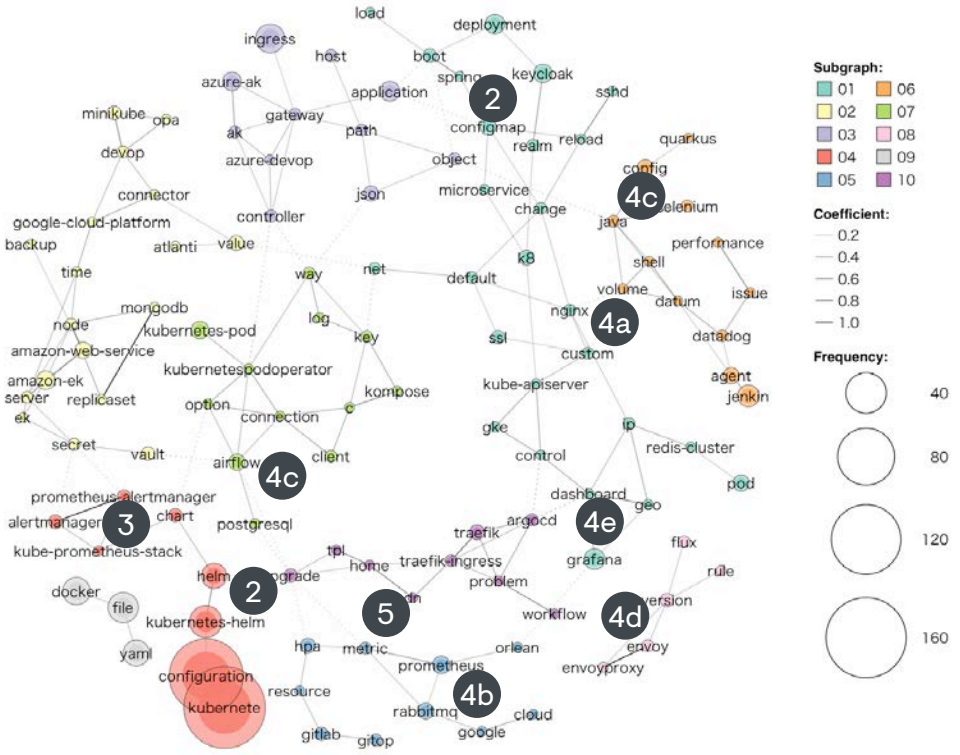
#6 Challenge: Configuration

5 Key Topics

1. **Cluster Configuration (33):** This category includes questions and discussions related to various configuration problems encountered in Kubernetes environments, such as ingress, networking, and deployment issues. It also covers specific component configurations like NGINX, CoreDNS, and Envoy.
2. **Helm & Chart Configuration (14):** This category covers issues and questions related to configuring and managing Helm charts and their associated YAML files. Topics include dynamic ConfigMaps, upgrading charts, and customizing chart configurations.
3. **Monitoring & Alerting Configuration (10):** This category pertains to configuring monitoring and alerting tools in Kubernetes, such as Prometheus, Alertmanager, and Grafana. It includes topics like setting up service monitors, routing alerts to Slack, and configuring Grafana dashboards.
4. **Application & Service Configuration (10):** This category is about configuring applications and services within Kubernetes, such as Airflow, Keycloak, and RabbitMQ. Discussions include setting up readiness probes, managing secrets, and handling application-specific errors.
5. **Configuration Management & Best Practices (9):** This category focuses on questions and discussions around managing and maintaining Kubernetes configurations, as well as sharing best practices for configuration management. Topics include configuration patterns, version control for configurations, and using development clusters for testing.

EMA Perspective

Organizations face crucial challenges in Kubernetes configuration, encompassing cluster configuration, management of application and cluster templates, monitoring and alerting, application and service configuration, and overall configuration management best practices. Successfully addressing these challenges will lead to a more robust, efficient, and reliable infrastructure, allowing for improved organizational performance.



Source: StackOverflow

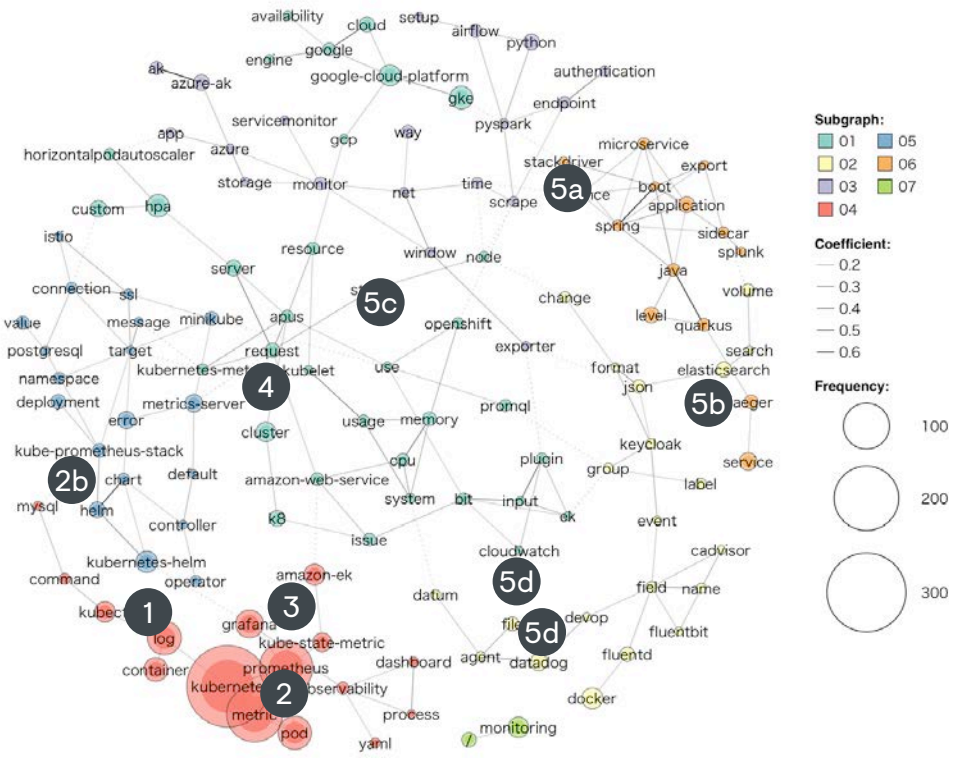
#7 Challenge: Observability

5 Key Topics

1. **Logging and Log Management:** This category consists of questions and issues related to logging and log management in Kubernetes, such as configuring logging tools like Fluentd or Fluent Bit, ingesting logs into external systems like Datadog or Loki, and accessing logs from various sources, like GKE or AKS.
2. **Prometheus Metrics and Monitoring:** This category covers topics related to Prometheus metrics and monitoring in Kubernetes, including configuring and scraping metrics, issues with metric collection or display, and working with specific exporters or service monitors.
3. **Grafana and Visualization:** These posts focus on Grafana and visualization of Kubernetes metrics, including configuring Grafana alerts, accessing logs in GKE's Logs Explorer, and working with Grafana dashboards.
4. **Metrics Server and API Issues:** This category contains questions and issues related to the Kubernetes Metrics Server and custom metrics API, such as problems with starting the Metrics Server, fetching metrics, or handling API requests.
5. **Other Monitoring and Metrics Tools:** This category includes discussions on other monitoring and metrics tools used with Kubernetes, such as AWS Cloudwatch, GCP Stackdriver, OpenTelemetry, and Zabbix.

EMA Perspective

Kubernetes observability challenges, including logging and log management, Prometheus metrics, Grafana visualization, Metrics Server and API issues, and the integration of multiple monitoring tools constitute key observability-related challenges. Overcoming these challenges is essential for maintaining a reliable and efficient infrastructure that supports data-driven decision-making and system optimization.



Source: StackOverflow

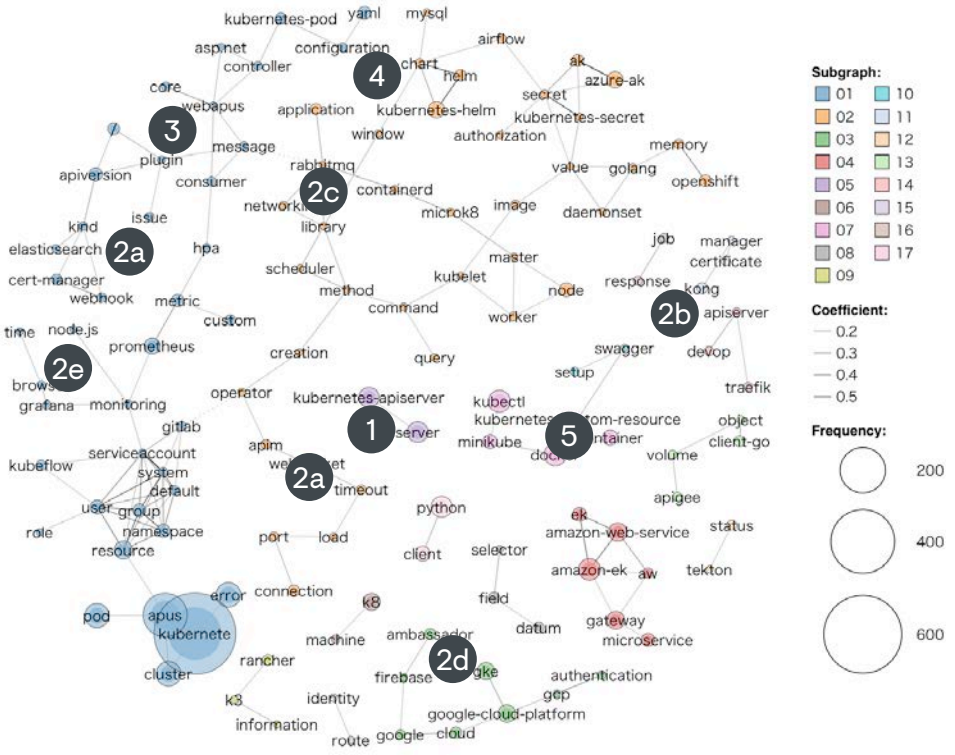
#8 Challenge: API

5 Key Topics

1. **Cluster API Issues (17):** These posts are about issues related to the Kubernetes API, such as error messages, authentication, and access control. Users are struggling with authorization errors, deprecated API calls, and connecting to remote clusters.
2. **Tools and Services Integration (13):** These posts deal with integrating various tools and services, such as Grafana, Datadog, Prometheus, and Kafka, into Kubernetes. Users are experiencing issues with API keys and error messages, and configuring these tools within their clusters.
3. **API Interaction and Development (11):** These posts involve the use and development of APIs within a Kubernetes context, such as creating and accessing API resources, querying the status of pods, and working with custom metrics APIs. Users are looking for guidance on REST API requests and error handling.
4. **Application Deployment and Configuration (7):** These posts focus on deploying and configuring applications, such as FastAPI, Strapi, and SpringBoot, within Kubernetes. Users are encountering issues with performance, URL accessibility, and error pages when deploying their applications.
5. **Kubernetes Objects and CRDs (6):** These posts concern Kubernetes objects and custom resource definitions (CRDs), including issues with deprecated API versions, Helm chart updates, and OpenAPI schema. Users are looking for solutions to unknown fields, updating ingress versions, and understanding capabilities.

EMA Perspective

API issues include tools and services integration, API interaction and development, application deployment and configuration, and Kubernetes objects and CRDs. Successfully navigating these challenges will contribute to a more efficient and seamless integration of services and applications, ultimately enhancing infrastructure and organizational performance.



Source: StackOverflow

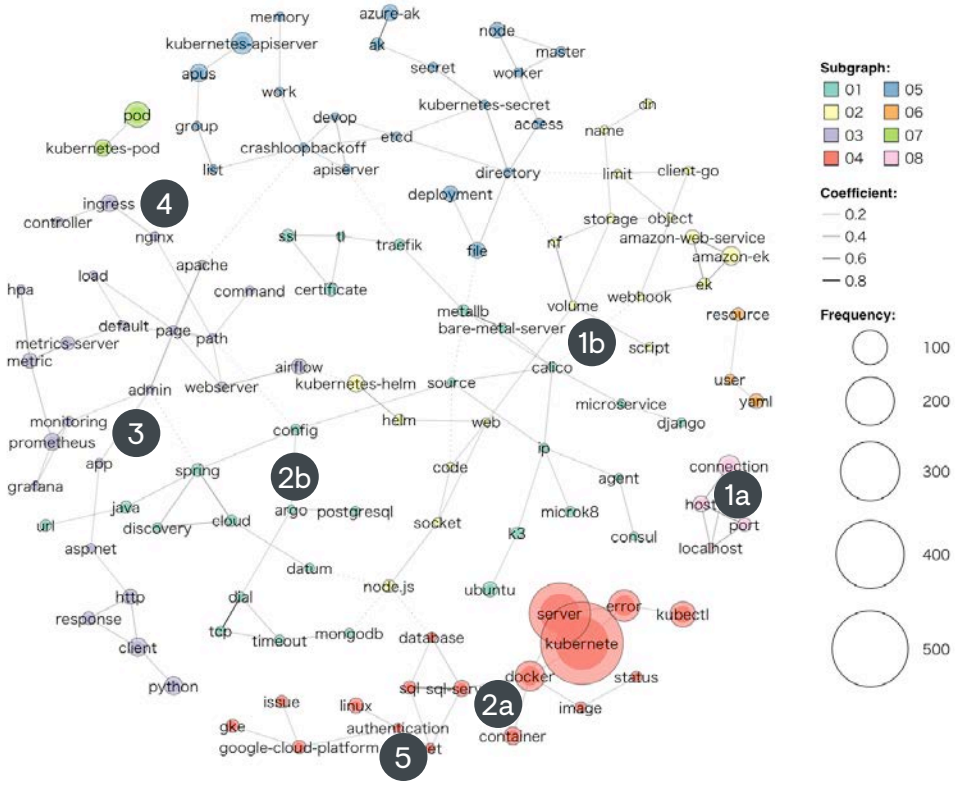
#9 Challenge: Server Infrastructure

5 Key Topics

1. **Kubernetes Connection and Configuration Issues:** These posts discuss issues related to connecting to or configuring Kubernetes clusters, including errors with API servers, refused connections, and server version mismatches.
2. **Deployments, Pods, and Containers:** These posts involve the deployment of applications, management of pods and containers, and issues with running specific software in a Kubernetes environment, such as SQL Server, MinIO, and ArgoCD.
3. **Metrics, Monitoring, and Logging:** These posts pertain to gathering metrics, monitoring, and logging within Kubernetes clusters, including issues with Metrics Server, Prometheus, Grafana, and Zabbix.
4. **Networking and Ingress:** These posts focus on networking and ingress issues in Kubernetes, including problems with K3s, NGINX ingress controllers, and networking configurations for different Kubernetes resources.
5. **Authentication and Authorization:** These posts deal with authentication and authorization problems in Kubernetes, such as errors with client credentials, Kerberos authentication, and access to resources within the cluster.

EMA Perspective

Server issues include connection and configuration problems, deployments and container management, metrics and monitoring, networking and ingress, and authentication and authorization. Addressing these challenges is vital for infrastructure optimization and for ensuring seamless integration of applications and services, improving overall business performance.



Source: StackOverflow

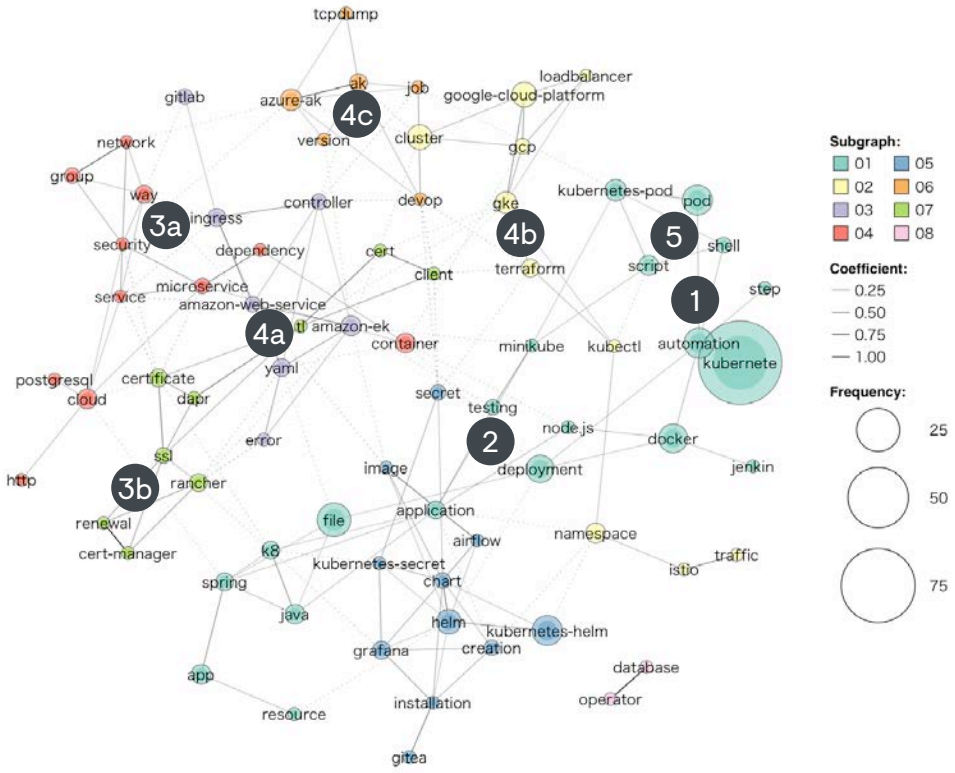
#10 Challenge: Automation

5 Key Topics

1. **Kubernetes and Automation:** These posts discuss automating various Kubernetes tasks, such as deploying applications, managing secrets, and scaling resources. Topics include automated deployments, managing secrets, and automatic restarts.
2. **CI/CD, Testing, and Monitoring:** These questions involve automating and managing CI/CD pipelines, testing, and monitoring in Kubernetes environments. They cover Jenkins, Grafana, Git, and other tools for deployment and monitoring.
3. **Configuration and Security:** This category includes questions about Kubernetes configuration, security, and certificates management. Discussions focus on automating the creation and management of Kubernetes resources, certificates, and firewall rules.
4. **Cloud Platforms and Tools Integration:** These posts cover integrating Kubernetes with various cloud platforms (GCP, Azure, AWS, Digital Ocean) and tools (Terraform, ArgoCD, Istio). Topics include automating GKE node migration, automating AKS logs uploading, and automatic traffic shifting using Istio.
5. **Miscellaneous Automation and Scripting:** These questions cover a range of automation and scripting tasks not strictly related to Kubernetes. Examples include automating tcpdump capturing, creating serverless functions, and using Go to merge Kubernetes YAML files.

EMA Perspective

Automation challenges are related to deployment and resource management, CI/CD pipelines and monitoring, configuration and security, cloud platform and tool integration, and various miscellaneous automation tasks. Successfully tackling these challenges will lead to a more efficient, secure, and streamlined infrastructure, ultimately enhancing organizational performance and agility.

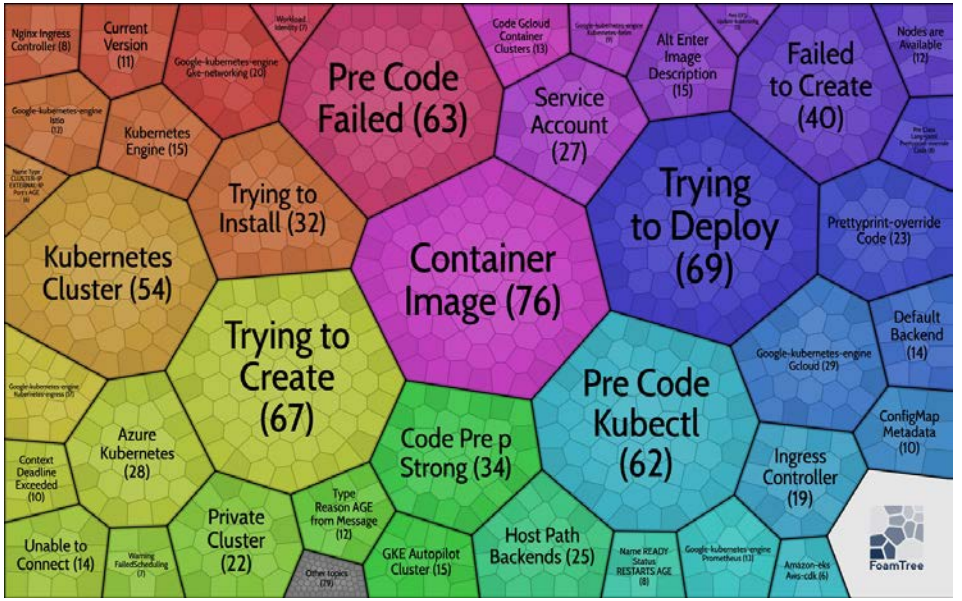


Source: StackOverflow

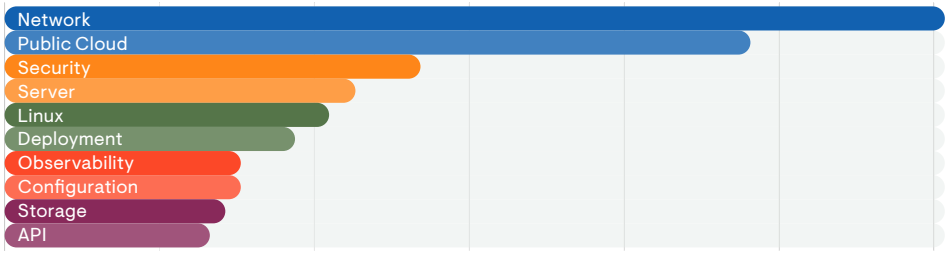
Top 10 Critical Kubernetes-Related Challenges for Operators

Learning from real-life challenges

1. **Networking:** Ensuring efficient and secure communication among various cluster components and handling service discovery, load balancing, and ingress/egress control.
2. **Public Cloud-Managed Kubernetes:** Managing and optimizing cloud-based Kubernetes offerings (Amazon EKS, Azure AKS, Google GKE), ensuring seamless integration and efficient resource utilization.
3. **Security:** Implementing stringent access controls, network policies, and encryption to safeguard Kubernetes clusters and maintain application integrity.
4. **Server:** Ensuring uptime and resource optimization in Kubernetes servers, as well as efficient management of worker nodes.
5. **Linux:** Leveraging Linux-based distributions and tools to ensure compatibility, stability, and performance in Kubernetes deployments.
6. **Deployment:** Streamlining application deployment and scaling to maximize application uptime, minimize downtime, and maintain optimal performance levels.
7. **Observability:** Implementing comprehensive monitoring, logging, and tracing for continuous insight into cluster performance and health.
8. **Configuration:** Ensuring consistent configuration management to enhance stability and maintainability across the Kubernetes deployments.
9. **Storage:** Managing persistent storage options, like volumes and persistent volume claims, to provide reliable and flexible storage solutions for stateful applications.
10. **API:** Securing and streamlining the interaction of API servers with the etcd data store, managing resources, and various Kubernetes components.



Source: Serverfault



Based on the number of operator questions submitted on the Serverfault operator forum

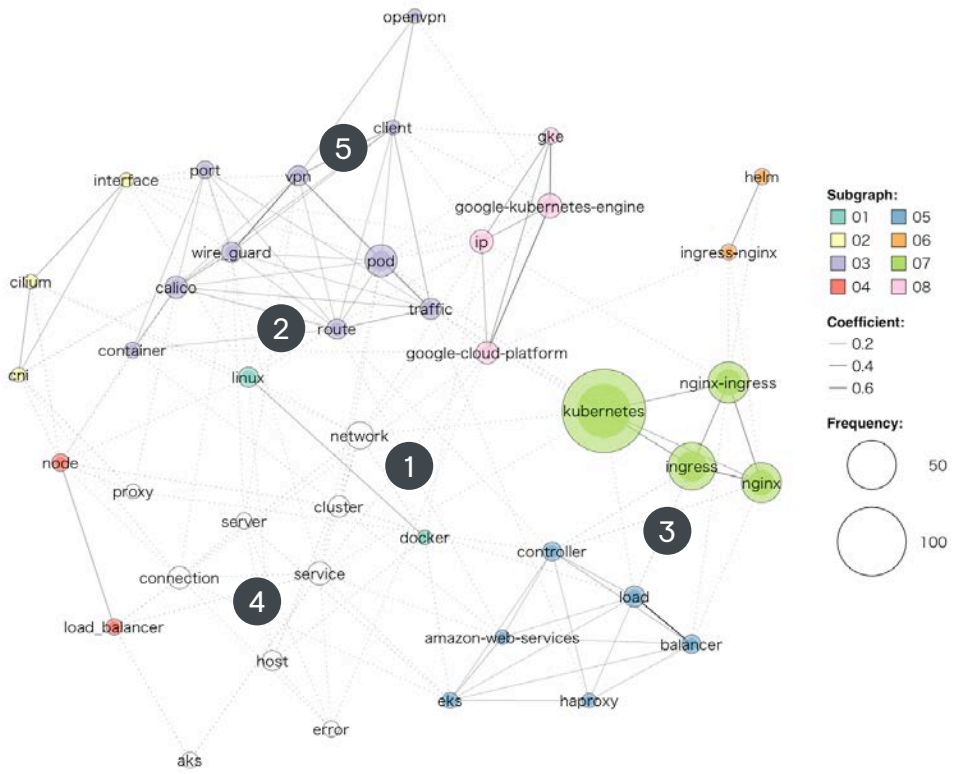
#1 Challenge: Networking

5 Key Topics

1. **General Networking Issues:** Operators encounter various networking issues in Kubernetes clusters, such as pod communication failures, unreachable nodes, and routing problems.
2. **Network Policies and Configuration:** Operators face challenges in configuring network policies and CNI plugins, as well as integrating with external network components.
3. **Ingress and Load Balancers:** Operators often struggle with setting up and managing ingress controllers and load balancers in Kubernetes environments.
4. **Performance and Optimization:** Operators experience performance concerns and seek ways to optimize Kubernetes networking for high-speed environments and specific use cases.
5. **Specialized Networking Use Cases:** Operators explore the implementation of specialized networking scenarios like VPN connections, multi-cast support, and network migration in Kubernetes.

EMA Perspective

Kubernetes networking challenges reach from basic communication issues to more advanced use cases. Operators struggle to configure and optimize network policies, ingress controllers, and load balancers. Additionally, they look for solutions to improve network performance and address specialized networking scenarios.



Source: Serverfault

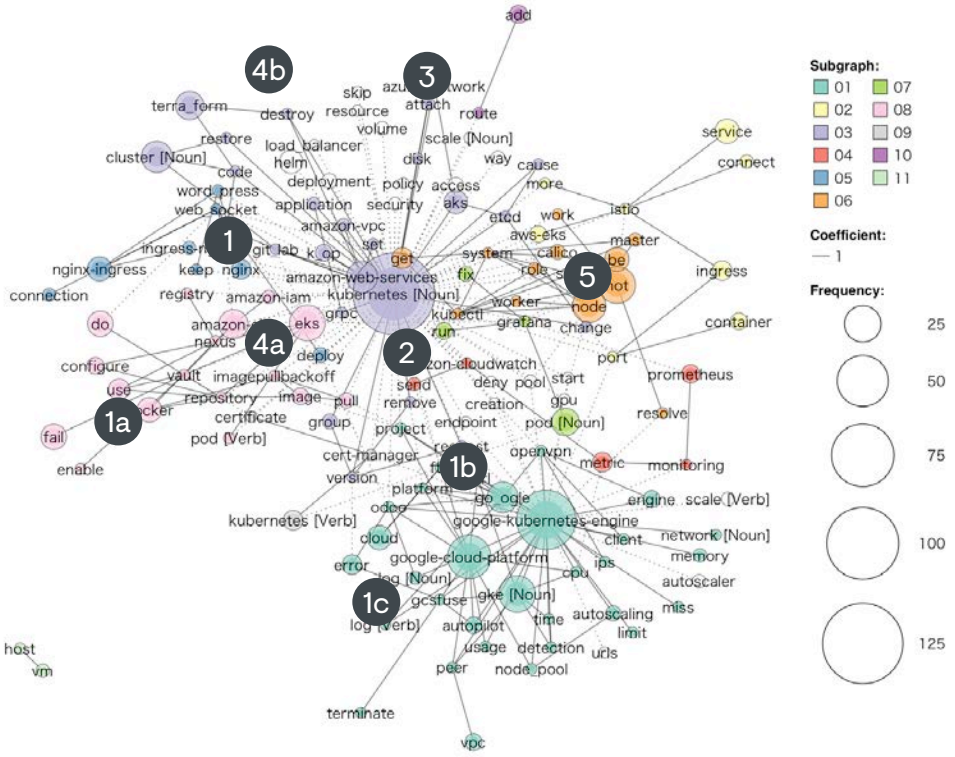
#2 Challenge: Kubernetes on EKS, AKS, and GKE

5 Key Topics

1. **General Kubernetes Issues:** IT operators face challenges such as pod creation order, managing security context, preserving client IPs, and troubleshooting intermittent node/kubelet reboots.
2. **Managed Kubernetes Services:** IT operators experience problems related to autoscaling, networking, and upgrading versions.
3. **Networking and Ingress:** Operators encounter networking and ingress issues, including NGINX configurations, GRPC call errors, and ingress updates causing problems with WordPress.
4. **Cloud Provider Integrations:** Operators face challenges in integrating Kubernetes with cloud providers like AWS, GCP, and Azure, including image pulling from ECR, connecting GCP SQL instances from GKE, and Terraform issues.
5. **Certificates and Security:** Operators deal with problems related to certificates and security, such as cert manager expiring certificates, issues with private Microsoft PKI, and configuring Kubernetes to use different Docker image repositories.

EMA Perspective

Finding the most popular managed Kubernetes services (EKS, AKS, and GKE) at the top of the list of IT operator challenges is not surprising. These services only focus on a small sliver of operating Kubernetes in production. Real production challenges lie in a wide range of nitty-gritty issues, such as configuring ingress control, load balancing, image repositories, certificate management, autoscaling, and helm templates. At the same time, operators struggle with numerous integration requirements with databases, automation platforms, and many more neighboring tools. This illustrates that EKS, AKS, and GKE are not turnkey solutions but take care of a number of operations tasks, such as managing the Kubernetes control plane, provisioning, updating, and scaling worker nodes, monitoring node health and managing identity and access management. Other tasks, such as cluster scaling, cost optimization, and network and security configuration, are left to corporate IT.



Source: Serverfault

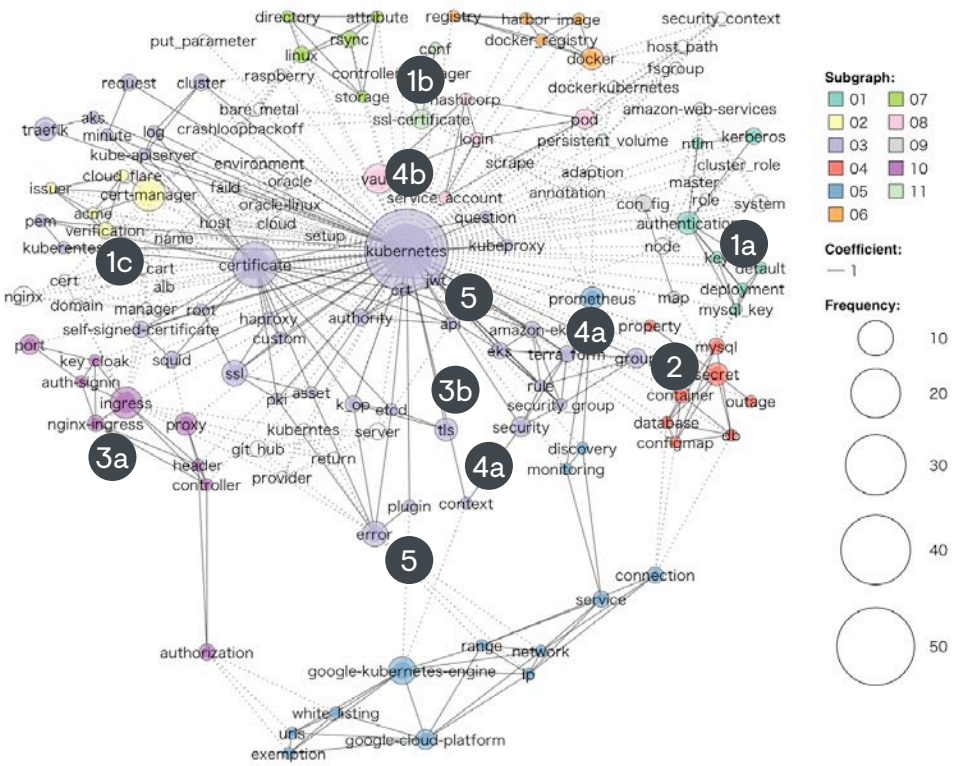
#3 Challenge: Security

5 Key Topics

1. **Certificate Management and Authentication:** IT operations engineers face challenges related to managing certificates, configuring authentication mechanisms, and handling expired certificates in Kubernetes environments.
2. **Security and Authorization:** Engineers need guidance on implementing security best practices, managing secrets, and configuring access control policies within Kubernetes clusters.
3. **Ingress and TLS Configuration:** IT operations engineers encounter issues with ingress controllers, integrating with certificate managers, and enabling or disabling TLS in Kubernetes deployments.
4. **Infrastructure and Tool Integration:** Engineers work to integrate Kubernetes with various tools and platforms, such as Terraform, HashiCorp Vault, and Prometheus, while overcoming associated challenges.
5. **Kubernetes Cluster Configuration:** IT operations engineers need assistance in configuring and troubleshooting Kubernetes clusters, including working with the kube-apiserver, kubeadm, and addressing issues related to API server registration.

EMA Perspective

The fact that IT operators still struggle with standard tasks, such as certification management, authentication, authorization ingress control, and encryption, illustrates that Kubernetes management still poses a challenge to many organizations. Taking these seemingly simple challenges off the operators' plates will be key to making room for these operators to focus on higher-level automation tasks as the foundation for scalability.



Source: Serverfault

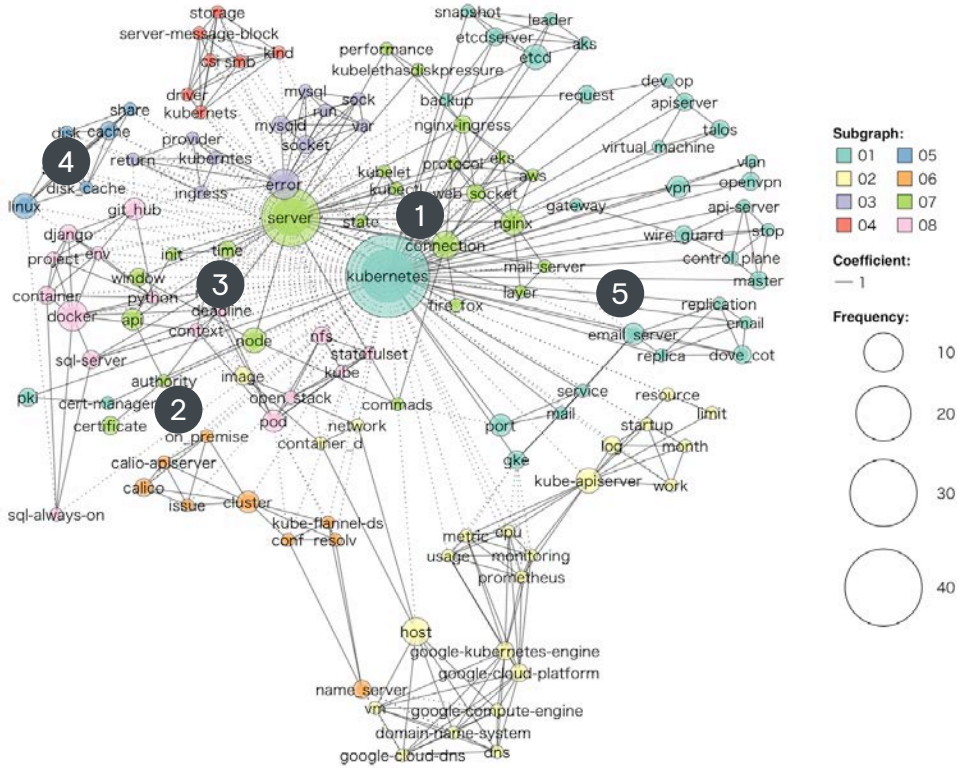
#4 Challenge: Server

5 Key Topics

1. **Server Connectivity and Configuration:** IT operations engineers face challenges related to server connectivity, such as configuring resolv.conf in kube-flannel-ds pods, managing connections to API servers, and addressing timeouts during kubeadm initialization.
2. **Certificate and Authentication Issues:** Engineers must deal with challenges involving certificates and authentication, such as expired certificates, registering nodes with API servers using certificates signed by unknown authorities, and configuring OAuth2-proxy with GitHub providers.
3. **Application Deployment and Management:** Challenges arise when deploying and managing applications on Kubernetes, such as setting up SQL Always On in Kubernetes Linux containers, running NFS servers on kube statefulset pods, and handling SignalR server deployments on AWS EKS behind NGINX.
4. **Resource Management and Limitations:** IT operations engineers need to manage resource limitations on servers, such as fixing disk pressure issues, setting resource limits on the Kubernetes API server, and sharing disk cache between multiple servers.
5. **Networking and Service Exposure:** Challenges related to networking and service exposure include exposing mail services on port 25 using GKE, running a mail server on an NGINX layer 4 server, and setting up VPN gateways on the same server with Kubernetes.

EMA Perspective

IT operations engineers face various challenges related to the underlying Kubernetes server infrastructure. These include connectivity problems, configuration issues, certificate and authentication issues, application deployment and management problems, resource management limitations, and networking and service exposure. Addressing these challenges is crucial for maintaining reliable and efficient Kubernetes environments.



Source: Serverfault

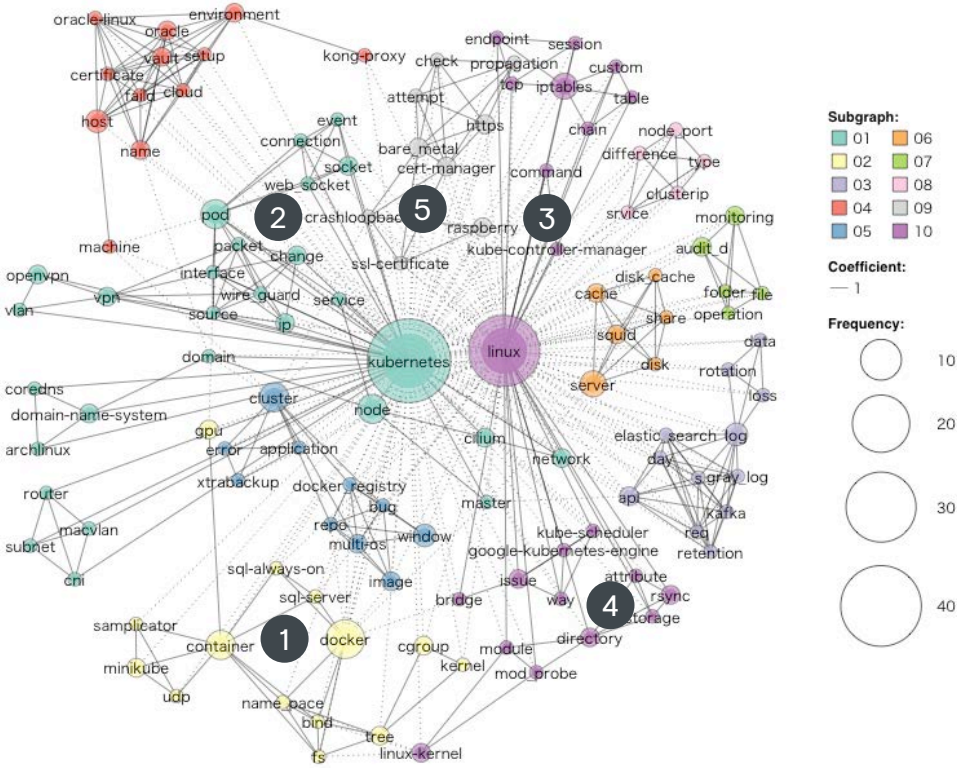
#5 Challenge: Linux

5 Key Topics

1. **Node Assignment and Pod Management:** IT operations engineers face challenges in understanding the underlying mechanisms for assigning pods to nodes and managing pod resources in a Kubernetes environment.
2. **Networking and Connectivity Issues:** Engineers need to address various networking challenges, such as dealing with bridge networking, VPN interfaces, and Cilium, to ensure proper communication within a Kubernetes cluster.
3. **Cluster and Component Configuration:** IT operations engineers must manage the configuration and troubleshooting of different Kubernetes components, such as CoreDNS, kube-scheduler, and kube-controller-manager.
4. **Storage and Data Management:** Engineers face challenges in ensuring proper data management, such as avoiding data loss during log rotation, managing disk caches, and handling mounted folders in a Linux-based Kubernetes environment.
5. **Security and Certificates:** Ensuring secure communication within the cluster and managing certificates with tools like cert manager and Vault is crucial for maintaining a stable and secure Kubernetes environment.

EMA Perspective

Node assignment and pod management, networking and connectivity issues, cluster and component configuration, storage and data management, and security certificates are some critical challenges illustrating the importance of the underlying . Addressing these challenges is essential for maintaining stable and secure Kubernetes environments.



Source: Serverfault

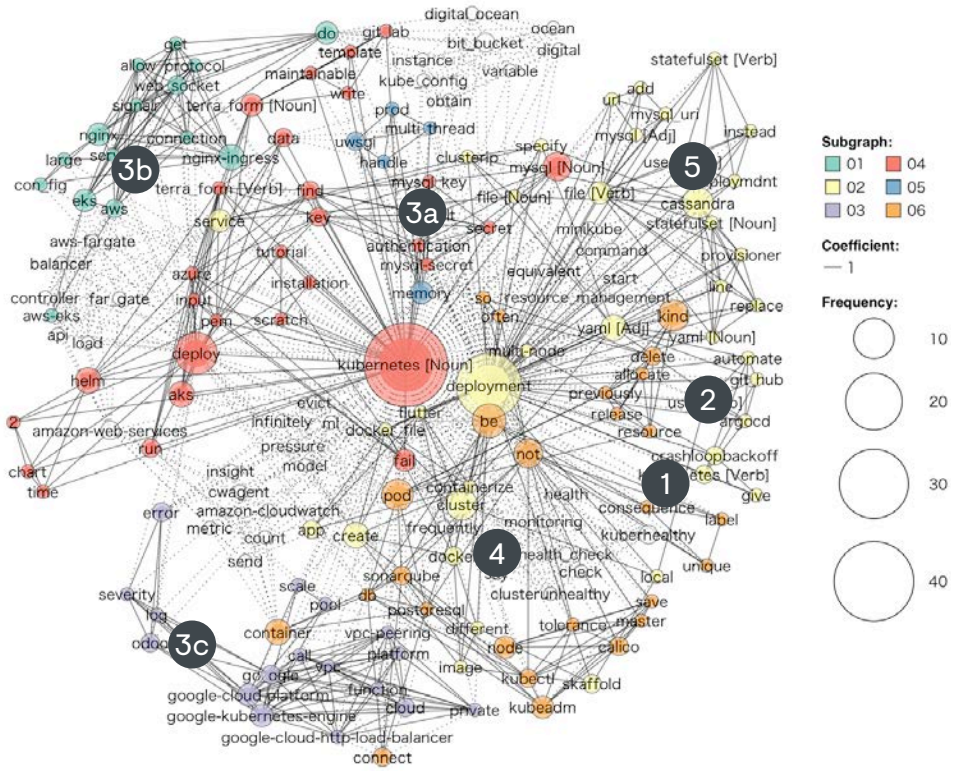
#6 Challenge: Deployment of Cluster and Applications

5 Key Topics

1. **Deployment Errors and Troubleshooting:** Operators experience various issues during Kubernetes deployment, such as crashloopbackoff errors, pod communication failures, and resource allocation problems.
2. **Deployment Automation and Orchestration:** Operators seek to automate Kubernetes deployments using tools like ArgoCD and Terraform.
3. **Kubernetes Application Deployment:** Operators face challenges in deploying specific applications, such as MySQL, NGINX, Odoo, and SonarQube on Kubernetes clusters.
4. **Kubernetes Cluster Management:** Operators need assistance in managing Kubernetes clusters, including handling resource scaling, health checks, and cluster configuration.
5. **Kubernetes Deployment Best Practices:** Operators want to understand best practices for Kubernetes deployments, such as when to use StatefulSets, how to manage deployment labels, and how to create maintainable Terraform templates.

EMA Perspective

Deployment-related operator challenges run the gamut between basic troubleshooting, managing multiple Kubernetes clusters, and leveraging advanced capabilities, such as StatefulSets and Terraform templates, to deploy hybrid applications. Operators look for automation and orchestration tools to streamline the end-to-end deployment process and the deployment of standard apps, such as databases, business apps, ingress controllers, and internal developer platforms.



Source: Serverfault

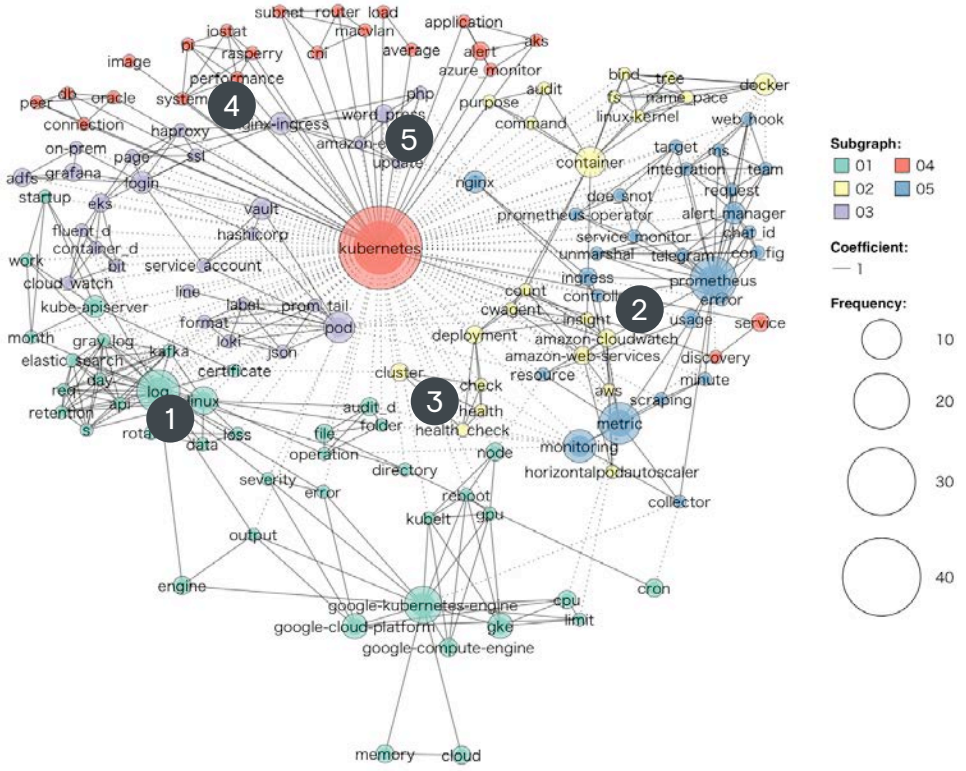
#7 Challenge: Observability

5 Key Topics

1. **Log Management:** IT operations engineers face challenges in aggregating, managing, and analyzing log files from various sources in a Kubernetes cluster.
2. **Monitoring and Metrics:** Engineers need to collect, analyze, and understand various metrics related to cluster and application performance, such as CPU usage, resource utilization, and pod count metrics.
3. **Alerting and Health Checks:** IT operations engineers deal with issues related to setting up alerts and health checks for applications and cluster components, ensuring proper monitoring and prompt response to issues.
4. **Performance Optimization:** Engineers encounter challenges in optimizing system performance within Kubernetes clusters, including addressing resource limitations, network bottlenecks, and high CPU usage.
5. **Integration with External Tools:** IT operations engineers need to integrate Kubernetes with external monitoring and alerting tools, such as Prometheus, Alertmanager, and webhook integrations with communication platforms like Microsoft Teams.

EMA Perspective

Core observability functions, such as log management, monitoring and metrics, alerting and health checks, performance optimization, and integration with external tools, still pose significant challenges for many organizations. Kubernetes management platforms need to offer simple solutions for these issues to ensure operational efficiency, better resource utilization, and higher overall customer satisfaction.



Source: Serverfault

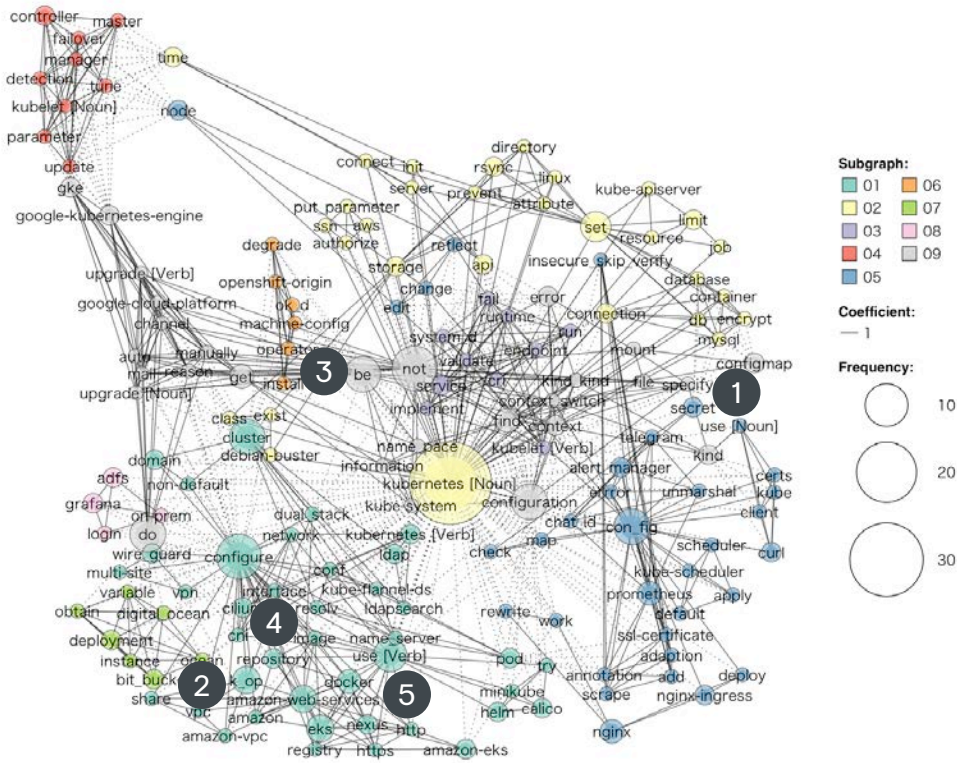
#8 Challenge: Configuration

5 Key Topics

1. **Configuration Management:** IT operations engineers encounter challenges when configuring and managing Kubernetes resources, such as ConfigMaps, secrets, and storage classes.
2. **Cluster Configuration:** Engineers face issues related to cluster configurations, including working with shared VPCs, enabling dual-stack services, and addressing degraded cluster operator states.
3. **External Service Integration:** IT operations engineers need to integrate various external services into Kubernetes, such as on-prem ADFS, LDAP, and WireGuard, and often require additional configuration and troubleshooting.
4. **CNI and Network Configuration:** Engineers deal with network-related configuration challenges, including configuring CNI plugins like Calico and Cilium, setting up kube-flannel-ds pods, and managing NGINX configurations.
5. **Repository and Scheduler Configuration:** IT operations engineers need to configure repositories for image storage, such as Nexus private Docker registry, and make adjustments to kube-scheduler configurations for optimal performance.

EMA Perspective

Rapidly expanding complexity and the dynamic character of distributed application environments contributed to the increased importance of configuration management within and across Kubernetes clusters.



Source: Serverfault

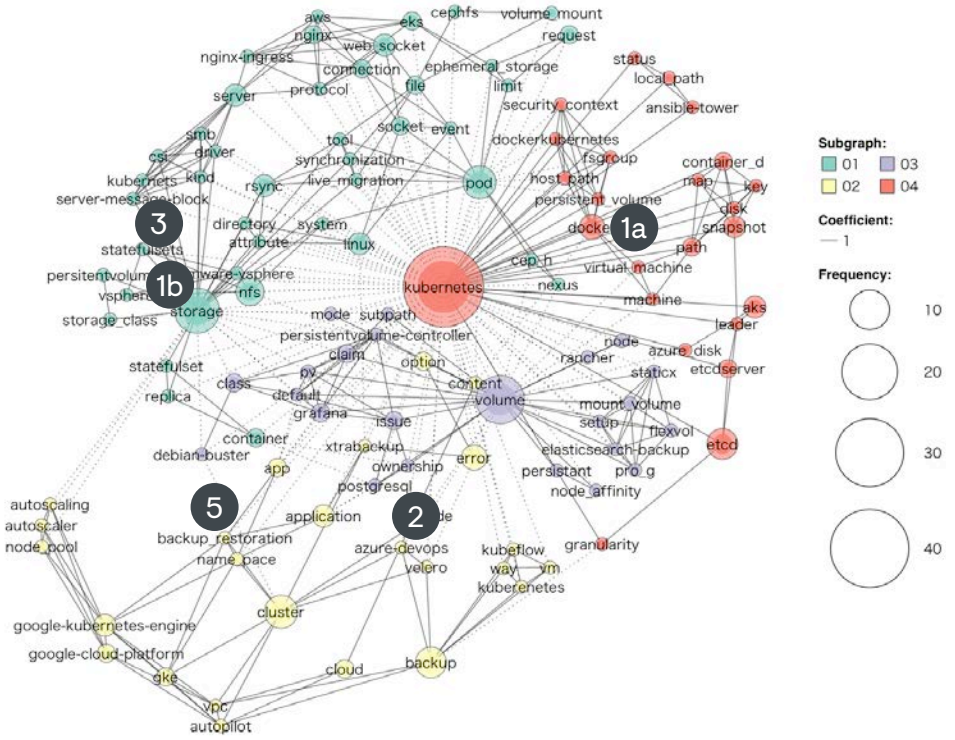
#9 Challenge: Storage

5 Key Topics

1. **Certificate Management and Authentication:** IT operations engineers face challenges related to managing certificates, configuring authentication mechanisms, and handling expired certificates in Kubernetes environments.
2. **Security and Authorization:** Engineers need guidance on implementing security best practices, managing secrets, and configuring access control policies within Kubernetes clusters.
3. **Ingress and TLS Configuration:** IT operations engineers encounter issues with ingress controllers, integrating with certificate managers, and enabling or disabling TLS in Kubernetes deployments.
4. **Infrastructure and Tool Integration:** Engineers work to integrate Kubernetes with various tools and platforms, such as Terraform, HashiCorp Vault, and Prometheus while overcoming associated challenges.
5. **Kubernetes Cluster Configuration:** IT operations engineers need assistance in configuring and troubleshooting Kubernetes clusters, including working with the kube-apiserver, kubeadm, and addressing issues related to API server registration.

EMA Perspective

The ephemeral character of Kubernetes environments still causes operators to struggle to maintain key enterprise-grade storage capabilities, including volume mapping, backup, and resource sharing across pods. Addressing these challenges is essential for ensuring the stability and reliability of applications running on Kubernetes.



Source: Serverfault

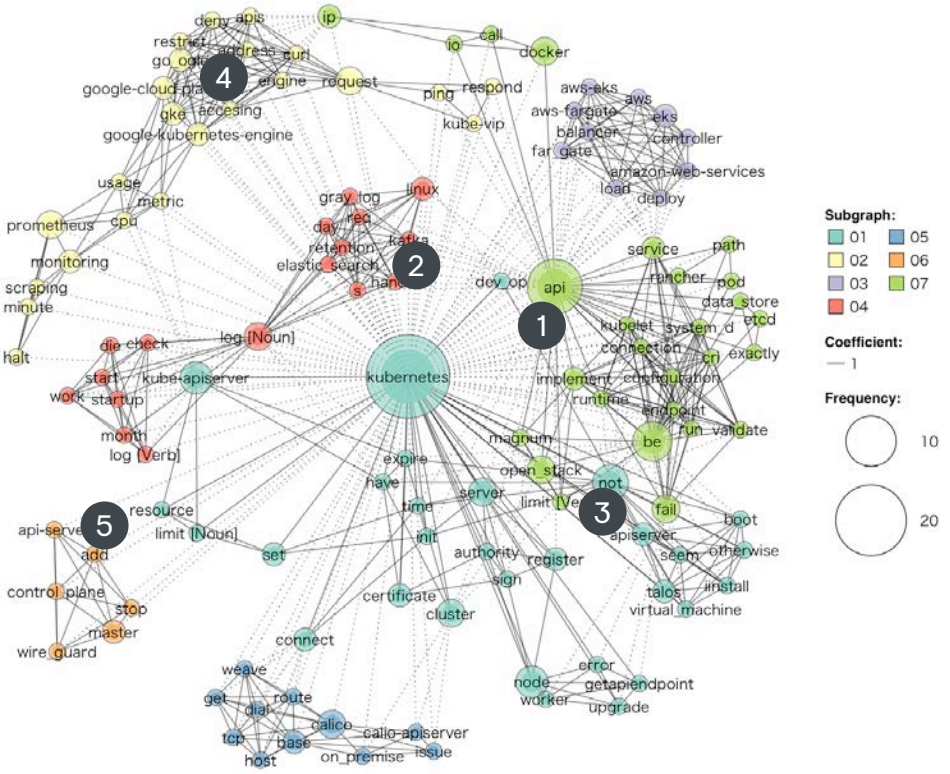
#10 Challenge: API

5 Key Topics

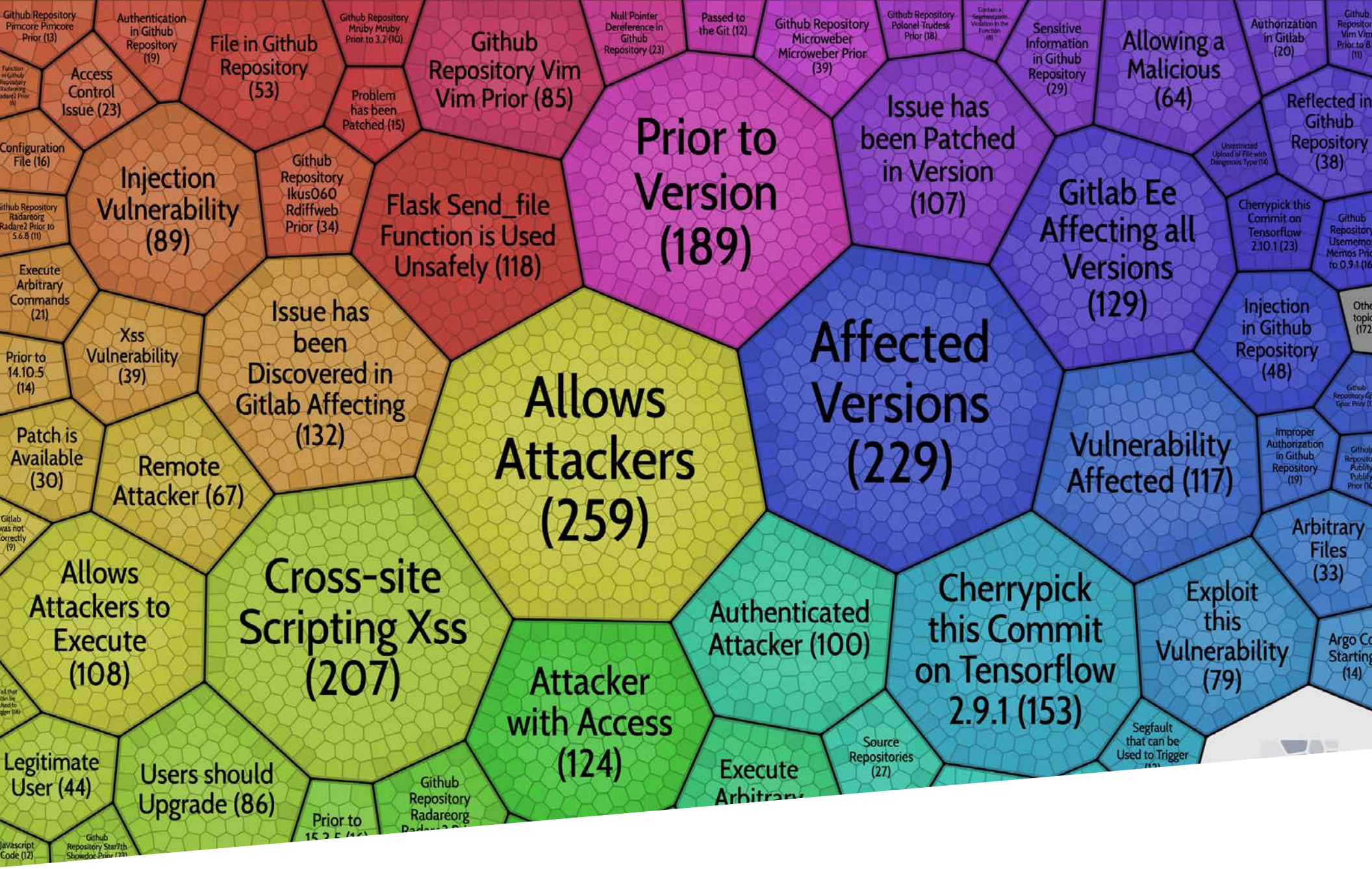
1. **Cluster API Issues:** These posts are about issues related to the Kubernetes API, such as error messages, authentication, and access control. Users are struggling with authorization errors, deprecated API calls, and connecting to remote clusters.
2. **Tools and Services Integration:** These posts deal with integrating various tools and services, such as Grafana, Datadog, Prometheus, and Kafka, into Kubernetes. Users are experiencing issues with API keys, error messages, and configuring these tools within their clusters.
3. **API Interaction and Development:** These posts involve the use and development of APIs within a Kubernetes context, such as creating and accessing API resources, querying the status of pods, and working with custom metrics APIs. Users are looking for guidance on REST API requests and error handling.
4. **Application Deployment and Configuration:** These posts focus on deploying and configuring applications, such as FastAPI, Strapi, and SpringBoot, within Kubernetes. Users are encountering issues with performance, URL accessibility, and error pages when deploying their applications.
5. **Kubernetes Objects and CRDs:** These posts concern Kubernetes objects and custom resource definitions (CRDs), including issues with deprecated API versions, Helm chart updates, and OpenAPI schema. Users are looking for solutions to unknown fields, updating ingress versions, and understanding capabilities.

EMA Perspective

API issues include tools and services integration, API interaction and development, application deployment and configuration, and Kubernetes objects and CRDs. Successfully navigating these challenges will contribute to a more efficient and seamless integration of services and applications, ultimately enhancing infrastructure and organizational performance.



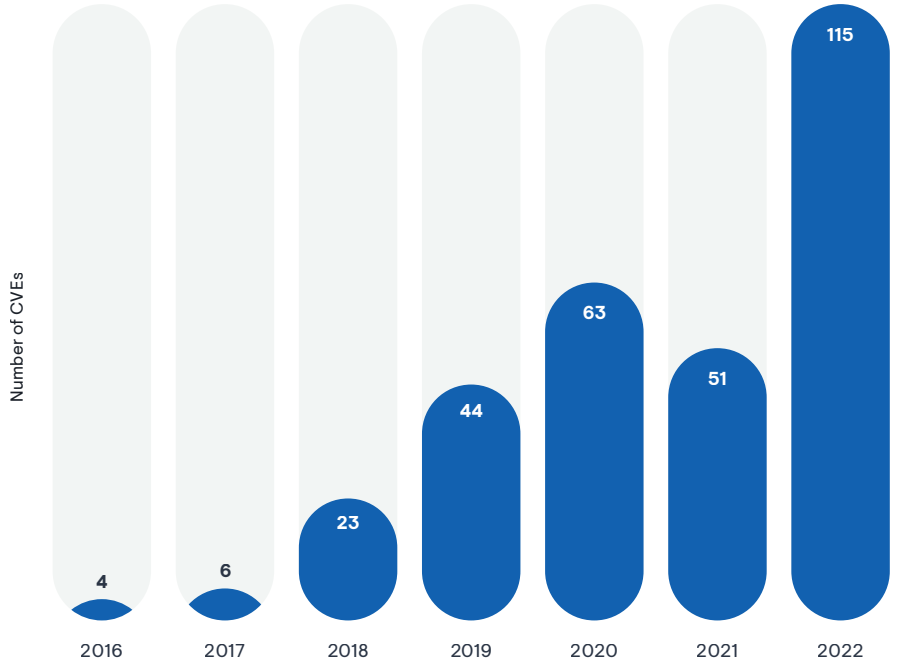
Source: Serverfault



Part 5: Kubernetes Security Challenges

The fact that the number of Kubernetes-related CVEs doubled over the past year indicates the need to take Kubernetes security seriously. This increase is a direct result of broader and deeper adoption of the Kubernetes platform in combination with the rapidly increasing complexity of the Kubernetes ecosystem.

ANNUAL NUMBER OF KUBERNETES-RELATED CVES



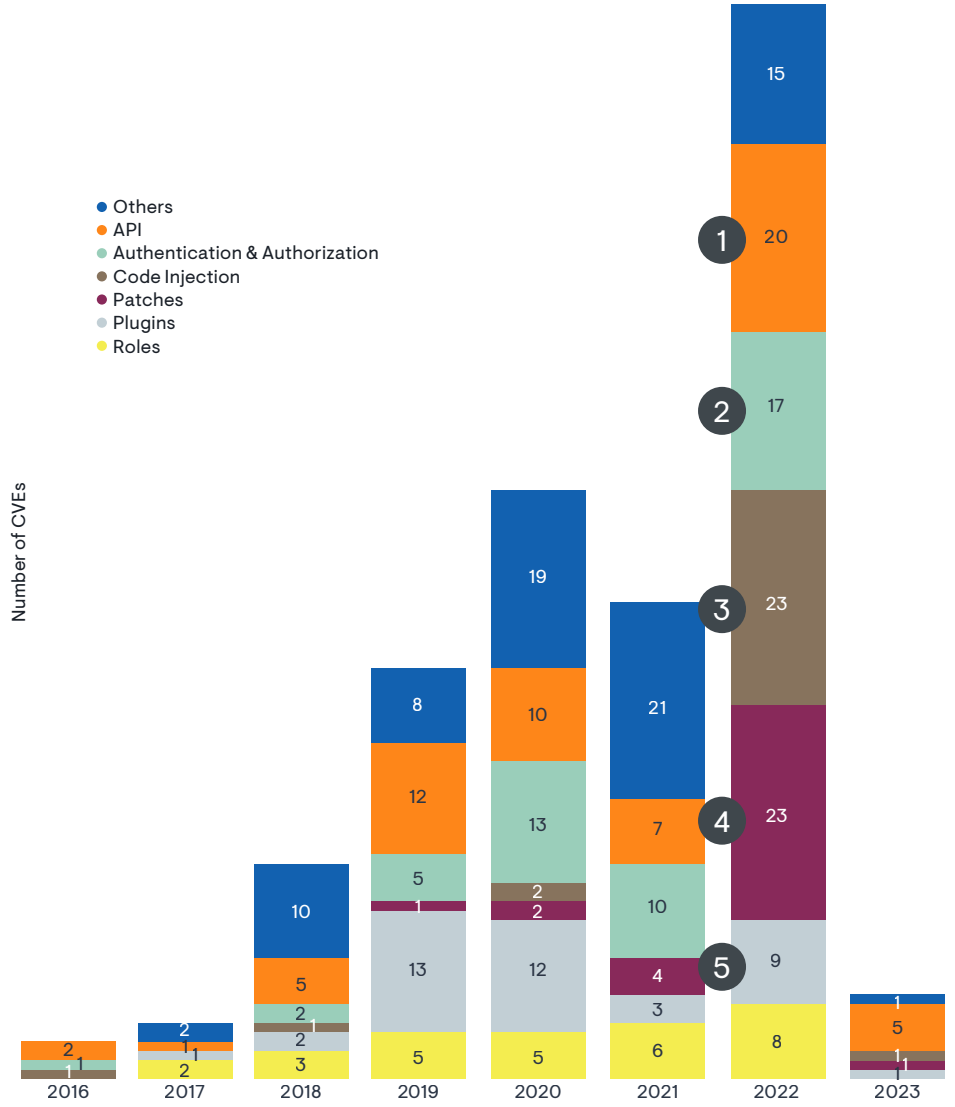
Source: MITRE Corporation

Five Core Types of Kubernetes Vulnerabilities

The distributed character of Kubernetes apps and the large ecosystem of Kubernetes-related software projects, by definition, provides a much wider attack surface than during the old days of mostly monolithic applications. This leads to five core types of Kubernetes vulnerabilities.

1. **APIs:** The distributed, open, and often dynamic character of Kubernetes applications leads to an increase in the number of APIs that often act as interfaces to valuable data and services. This increases the risk and impact of code bugs, configuration mistakes, and late patches.
2. **Authentication, Authorization, and RBAC:** In a Kubernetes environment that reaches across numerous microservices, nodes, and sometimes clusters, security gaps are more difficult to find and the risk of misconfigurations increases.
3. **Code Injection:** Code injection can allow attackers to execute malicious code within a specific application context to then gain unauthorized access to the entire cluster.
4. **Plugins:** Plugins for Kubernetes and related software tools can provide unauthenticated or unauthorized entry to the Kubernetes cluster.
5. **Late Patches:** The omission of timely patching of any component of the overall Kubernetes application stack can leave the door open for attackers to penetrate Kubernetes clusters.

Next, we will dive deeper into each one of these vulnerability types to demonstrate the different aspects connected with each vulnerability.



Source: MITRE Corporation

1. API-Related Vulnerabilities

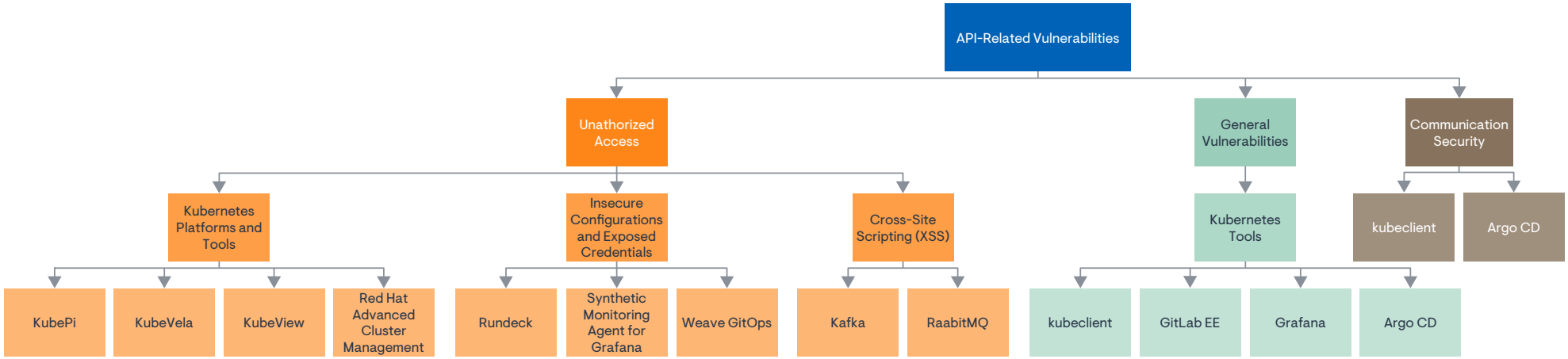
Unauthorized Access: Security vulnerabilities in Kubernetes platforms and tools, including KubePi, KubeVela, KubeView, and Red Hat Advanced Cluster Management, expose APIs to unauthorized access, sensitive information leakage, and unauthenticated SSRF attacks.

Insecure configurations and exposed credentials in automation and monitoring tools, such as Rundeck, Synthetic Monitoring Agent for Grafana, and Weave GitOps, can result in unauthorized access to APIs, sensitive information leakage, and compromise of API credentials.

Cross-site scripting (XSS) vulnerabilities in messaging applications, such as Kafka or RabbitMQ, can allow attackers to inject and execute arbitrary HTML and JavaScript code within APIs, leading to unauthorized access to sensitive data.

General Vulnerabilities: Multiple vulnerabilities across Kubernetes tools, such as kubeclient, GitLab EE, Grafana, and Argo CD, expose sensitive information and allow unauthorized access to APIs, potentially leading to escalated privileges, unauthorized deployments, or leaking of repository access credentials.

Communication Security: Flaws in kubeclient and Argo CD, including man in the middle (MITM) attacks and trusting malicious OpenID Connect providers, highlight the importance of properly validating certificates and handling secure communications in APIs within Kubernetes applications.



Source: MITRE Corporation

2. Authentication, Authorization, and RBAC

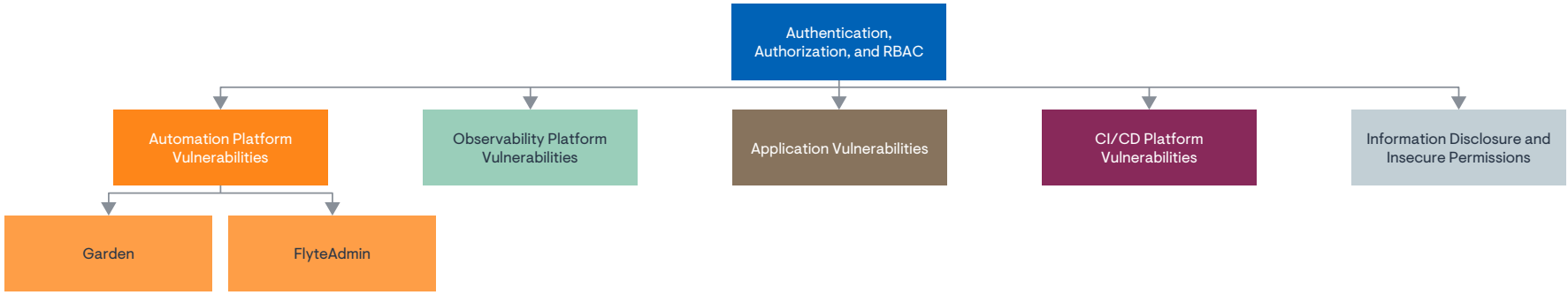
Automation Platform Vulnerabilities: Security flaws in automation platforms, like Garden and FlyteAdmin, can allow attackers to gain unauthorized access; compromise credentials, secrets, or environment variables; and gain remote code execution capability on the server, posing a risk to the organization.

Observability Platform Vulnerabilities: Vulnerabilities in observability platforms can allow attackers to leak authentication cookies, mount cross-origin attacks, and elevate their privileges, potentially leading to unauthorized access and data breaches.

Application Vulnerabilities: When applications running on Kubernetes lack proper authentication or authorization controls they can expose internal data structures, allow the unauthorized execution of commands, or gain access to Kubernetes cluster resources.

CI/CD Platform Vulnerabilities: CI/CD platforms can have improper access controls that allow malicious users to elevate their privileges in order to be able to deploy apps outside of the allowed namespaces, bypass logging security, or view sensitive cluster configurations.

Information Disclosure and Insecure Permissions: Applications can expose sensitive information to unauthorized actors or allow unauthorized access to files. Upgrading to secure versions, implementing proper permissions, or adjusting log levels is necessary to protect sensitive data.



Source: MITRE Corporation

3. Code Injection Vulnerabilities

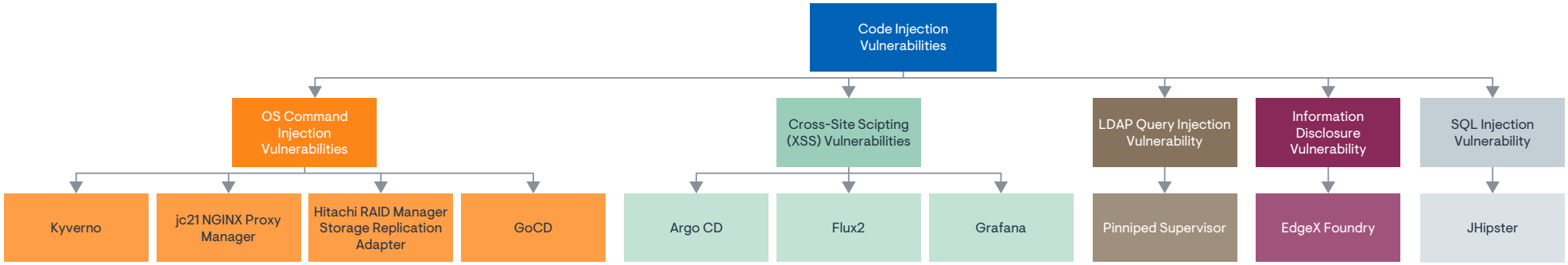
OS Command Injection Vulnerabilities: Kyverno, jc21 NGINX Proxy Manager, Hitachi RAID Manager Storage Replication Adapter, and GoCD are all affected by OS command injection vulnerabilities, which could allow remote attackers to execute arbitrary commands.

Cross-Site Scripting (XSS) Vulnerabilities: Argo CD versions prior to 2.5.17, Flux2 versions between 0.1.0 and 0.29.0, and Grafana versions prior to 8.5.16 and 9.2.8 have XSS vulnerabilities that could allow attackers to inject arbitrary JavaScript code into a victim's browser, potentially giving them administrative access to the system.

LDAP Query Injection Vulnerability: An issue was discovered in the Pinniped Supervisor that could allow attackers to change their user entry on the LDAP or AD server to include special characters, which could be used to perform LDAP query injection on the Supervisor's LDAP query to determine their Kubernetes group membership.

Information Disclosure Vulnerability: Prior to version 2.1.1, EdgeX Foundry exposes message bus credentials to local unauthenticated users, bypassing access controls when running in security-enabled mode. Attackers could intercept data or inject fake data into the EdgeX message bus, requiring users to upgrade to EdgeXFoundry Kamakura release (2.2.0) or to the June 2022 EdgeXFoundry LTS Jakarta release (2.1.1) to receive a patch.

SQL Injection Vulnerability: JHipster generated web applications using reactive Spring WebFlux and an SQL database using r2dbc, which are vulnerable to SQL injection in the findAllBy (Pageable pageable, Criteria criteria) method of an entity repository class. The issue was patched in v7.8.1, and users unable to upgrade should be careful when combining criteria and conditions.



Source: MITRE Corporation

4. Late Patching

Unauthorized Access and Privilege Escalation: Attackers may exploit vulnerabilities to gain unauthorized access to systems or escalate their privileges, allowing them to perform actions they shouldn't be allowed to, such as manipulating or deleting resources, forging HTTP requests, and impersonating other users or roles.

Data Leakage and Breaches: Vulnerabilities can lead to unauthorized parties accessing sensitive data or leaking it, potentially compromising the confidentiality and integrity of the affected system. This can include sensitive files, encrypted data, or repository access credentials.

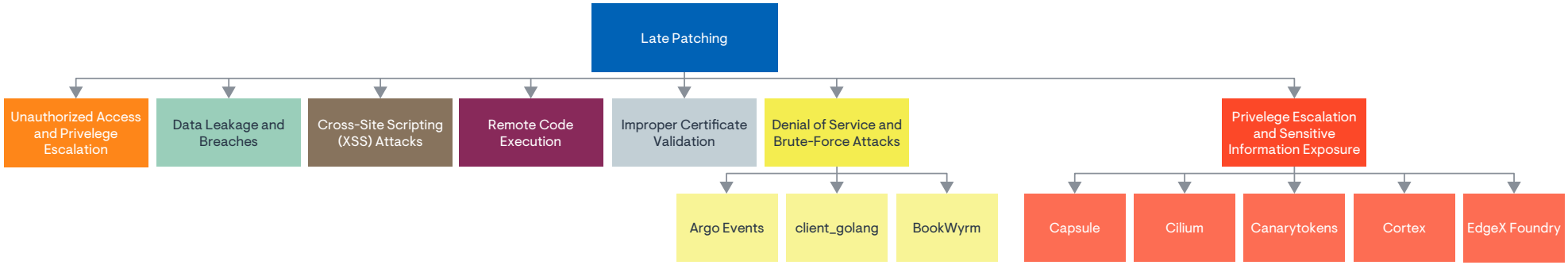
Cross-Site Scripting (XSS) Attacks: Attackers may exploit XSS vulnerabilities to inject malicious scripts into web pages, potentially compromising the security of the affected system and its users. These scripts can be used to steal user data, manipulate system settings, or perform other malicious actions.

Remote Code Execution: Attackers may exploit certain vulnerabilities to execute arbitrary code on the affected system, potentially compromising the security and integrity of the system and its data.

Privilege Escalation and Sensitive information Exposure: Issues in Capsule, Cilium, Canarytokens, Cortex, and EdgeX Foundry enable attackers to gain higher privileges, access sensitive information, or exploit cross-site scripting. Updating to patched versions helps secure systems against unauthorized access and data breaches.

Improper Certificate Validation: Vulnerabilities related to improper certificate validation can result in trusting malicious or untrustworthy providers, potentially compromising the security of the affected system.

Denial of Service and Brute-Force Attacks: Vulnerabilities in Argo Events, client_golang, and BookWyrm can lead to server crashes, memory exhaustion, or brute-force attacks on user accounts. Patching these vulnerabilities helps prevent service disruptions and unauthorized access attempts.



Source: MITRE Corporation

5. Plugins

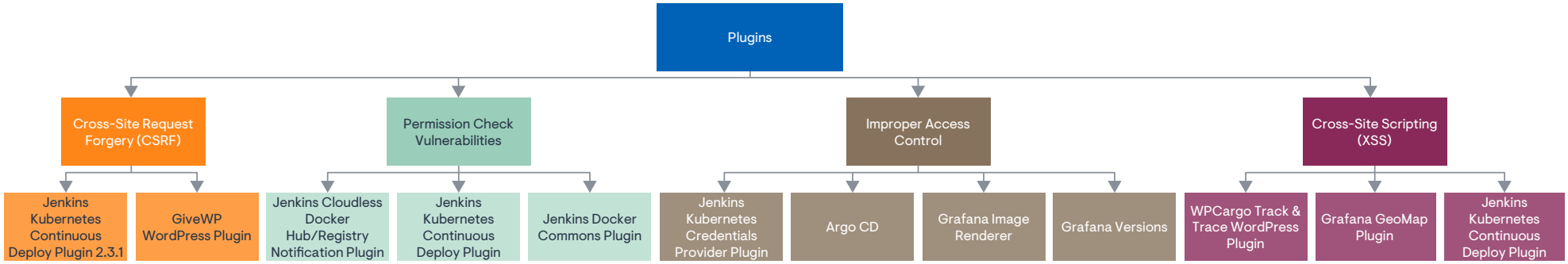
Cross-Site Request Forgery (CSRF): Jenkins Kubernetes Continuous Deploy plugin 2.3.1 and GiveWP WordPress plugin are vulnerable to CSRF attacks, where attackers can exploit users' authenticated sessions or cause denial of service attacks.

Permission Check Vulnerabilities: Jenkins CloudBees Docker Hub/Registry Notification plugin, Jenkins Kubernetes Continuous Deploy plugin, and Jenkins Docker Commons plugin suffer from missing or incorrect permission checks, which can result in unauthorized access and unintended actions, such as OS command execution.

Improper Access Control: Jenkins Kubernetes Credentials Provider plugin, Argo CD, Grafana Image Renderer, and Grafana versions have improper access control vulnerabilities, which may lead to sensitive information leakage or unauthorized actions.

Cross-Site Scripting (XSS): WPCargo Track & Trace WordPress plugin, Grafana GeoMap plugin, and Jenkins Kubernetes Continuous Deploy plugin are susceptible to XSS attacks, where attackers can inject malicious scripts and compromise user accounts.

Cross-Site Scripting (XSS): WPCargo Track & Trace WordPress plugin, Grafana GeoMap plugin, and Jenkins Kubernetes Continuous Deploy plugin are susceptible to XSS attacks, where attackers can inject malicious scripts and compromise user accounts.



Source: MITRE Corporation

Vulnerabilities by Product

Spotlight on Argo, NGINX, Grafana, Helm, and Git

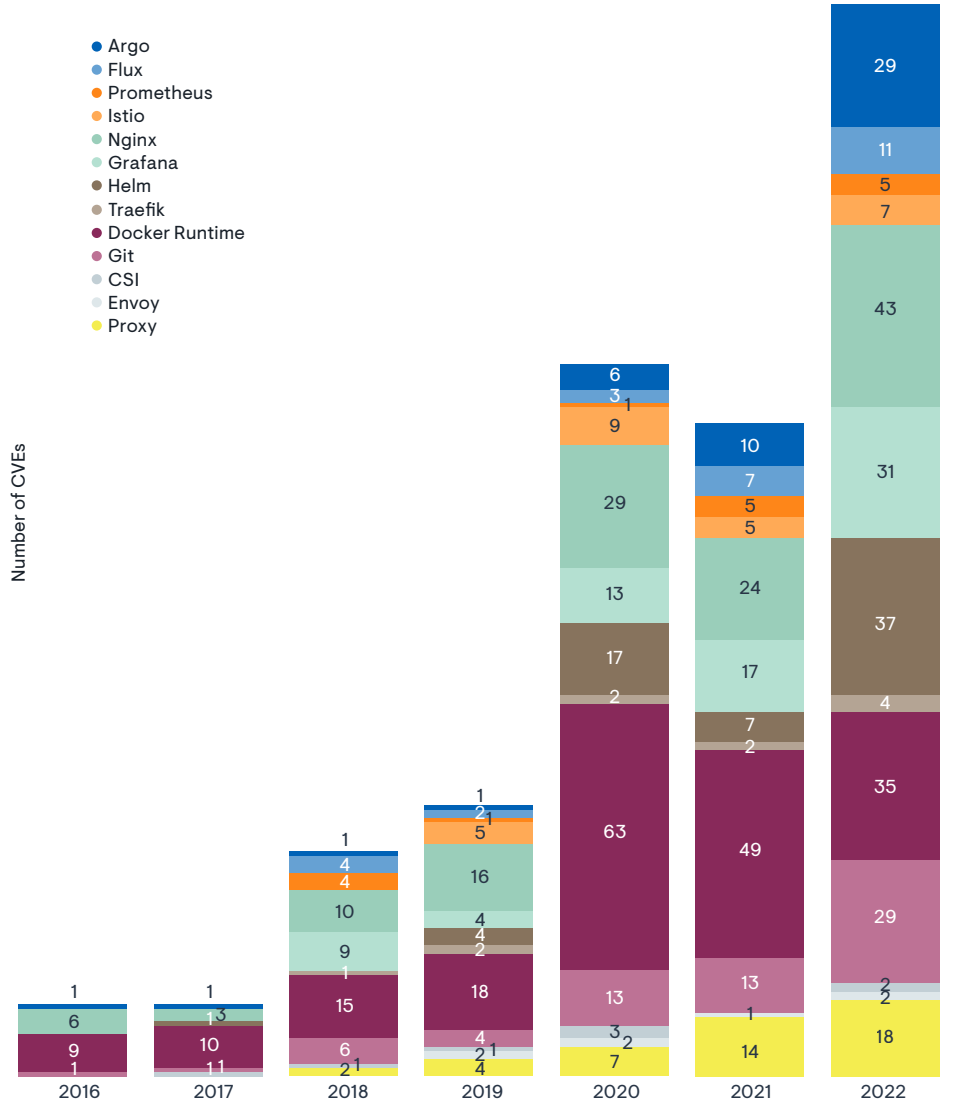
The products appearing in the largest number of CVEs directly correspond with the types of vulnerabilities previously identified.

Argo: The vulnerabilities in Argo CD could lead to unauthorized access, privilege escalation, sensitive data leaks, and denial of service, potentially disrupting business operations and causing damage to a company's reputation. Organizations using Argo CD should prioritize patching and implementing mitigation measures to prevent severe security incidents and protect critical infrastructure.

Example: A large online retailer uses Argo CD for their continuous delivery needs. However, an unpatched vulnerability in Argo CD allowed an attacker to escalate their privileges and gain unauthorized access to the deployment configurations. This led to a significant service disruption during a peak shopping season, causing a loss of revenue and a hit to the company's reputation. Following the incident, the company prioritized the patching of Argo CD and implemented additional security measures to prevent such incidents in the future.

NGINX: The analyzed CVEs reveal multiple security issues in NGINX, including segmentation violations, denial of service, command injection, and path traversal vulnerabilities, which could lead to unauthorized access, data leaks, and service disruptions. Organizations should prioritize patching these vulnerabilities and implement mitigation measures to ensure the security of their infrastructure and protect sensitive data.

Example: A popular social media platform relies on NGINX to handle their web traffic. A path traversal vulnerability in an unpatched version of NGINX allowed an attacker to access sensitive server files, resulting in a major data leak. This incident led to users' personal data being exposed, causing a significant disruption in service and harm to the company's reputation. In response, the company prioritized patching their NGINX servers and bolstered their security protocols to prevent future attacks.



Source: MITRE Corporation

Grafana: Grafana showed vulnerabilities related to access control, code execution, configuration issues, and denial of service. These vulnerabilities could potentially lead to unauthorized access, system instability, or loss of sensitive information, impacting businesses relying on Grafana for monitoring and data visualization.

Example: A multinational corporation uses Grafana for monitoring their IT infrastructure. An exploited vulnerability related to access control in their Grafana setup led to unauthorized code execution. The attacker was able to manipulate the displayed data, causing system instability and misleading the IT team, which delayed the detection and response to another ongoing attack. The company suffered financial losses as a result of these attacks and had to invest heavily in incident response and mitigation. After the incident, the company prioritized patching Grafana and enhancing its security configuration to protect against similar incidents.

Helm: Forms of attack include remote code execution, directory traversal, improper access control, privilege escalation, symlink following, and denial of service. The vulnerabilities can lead to sensitive data leakage, disclosure of confidential information, and unauthorized access to resources. The vulnerabilities impact several tools, Argo CD, Flux2, FlyteAdmin, and GiveWP WordPress plugin, among others. The affected versions have patches or mitigations available and users are advised to update their software as soon as possible.

Example: An international airline uses Helm to manage their Kubernetes applications, including their ticketing and customer service systems. However, an attacker exploited a directory traversal vulnerability in Helm to gain unauthorized access to resources. The attacker was able to access sensitive data, including flight schedules and customer information, leading to a data breach. The incident affected the company's operations and led to a loss of trust among customers. The airline company was also using the affected Argo CD and Flux2 tools, further exacerbating the impact of the breach. In response, they prioritized applying patches and mitigations for the affected Helm versions and increased their focus on securing their Kubernetes environment.

Git: Several GitOps tools, including Argo CD, Cargo, and GitHub Actions Runner, were found to have vulnerabilities that could allow attackers to perform various malicious activities, such as unauthorized access, remote code execution, and session hijacking. Patches were released for these vulnerabilities and users are advised to update their software as soon as possible.

Example: A major online gaming company uses GitOps tools, including Argo CD and GitHub Actions Runner, for managing their development and deployment processes. A vulnerability in these tools allowed an attacker to hijack a session, gaining access to the game's source code. The attacker then exploited a remote code execution vulnerability to disrupt the game services, causing a significant outage. This led to a loss of revenue and damage to the company's reputation among its player base. Following the incident, the gaming company promptly applied the released patches for the vulnerabilities and reviewed their overall security posture to prevent such attacks in the future.

Developer Perspective: Five Categories of Security-Related Pain Points

Compliance and Regulatory Concerns: Implementing security controls in Kubernetes environments is crucial for meeting regulatory requirements and avoiding costly non-compliance penalties.

Example: A health care company using Kubernetes for managing patient data must adhere to Health Insurance Portability and Accountability Act (HIPAA) standards. They implement role-based access control (RBAC) in their Kubernetes clusters to control who can access specific resources and ensure confidentiality of the patient's data. Furthermore, they use encryption for data at rest and in transit to meet the regulatory requirements.

Business Continuity and Reputation: A lack of security in Kubernetes can result in service disruptions, data breaches, and unauthorized access to sensitive information, potentially damaging a company's reputation and leading to financial losses.

Example: A global e-commerce company suffered a significant data breach when their Kubernetes environment was exploited due to an insecure configuration. This led to a disruption in their services for several hours, loss of customer data, and substantial harm to their reputation. The financial impact was significant in terms of immediate loss of sales and longer-term loss of customer trust.

Source: StackOverflow

Resource Management and Optimization: Properly configuring security policies and settings in Kubernetes can help optimize resource utilization and reduce the risk of resource exhaustion, leading to improved operational efficiency and cost savings.

Example: A software development company uses Kubernetes to manage its microservices. By implementing proper security policies and settings, they prevent unauthorized pods from consuming excessive resources. This helps to optimize resource utilization, prevent outages due to resource exhaustion, and ultimately generate significant cost savings and improved operational efficiency.

Third-Party Integration: Integrating third-party services with Kubernetes environments requires careful consideration of security implications to ensure the confidentiality, integrity, and availability of data and resources.

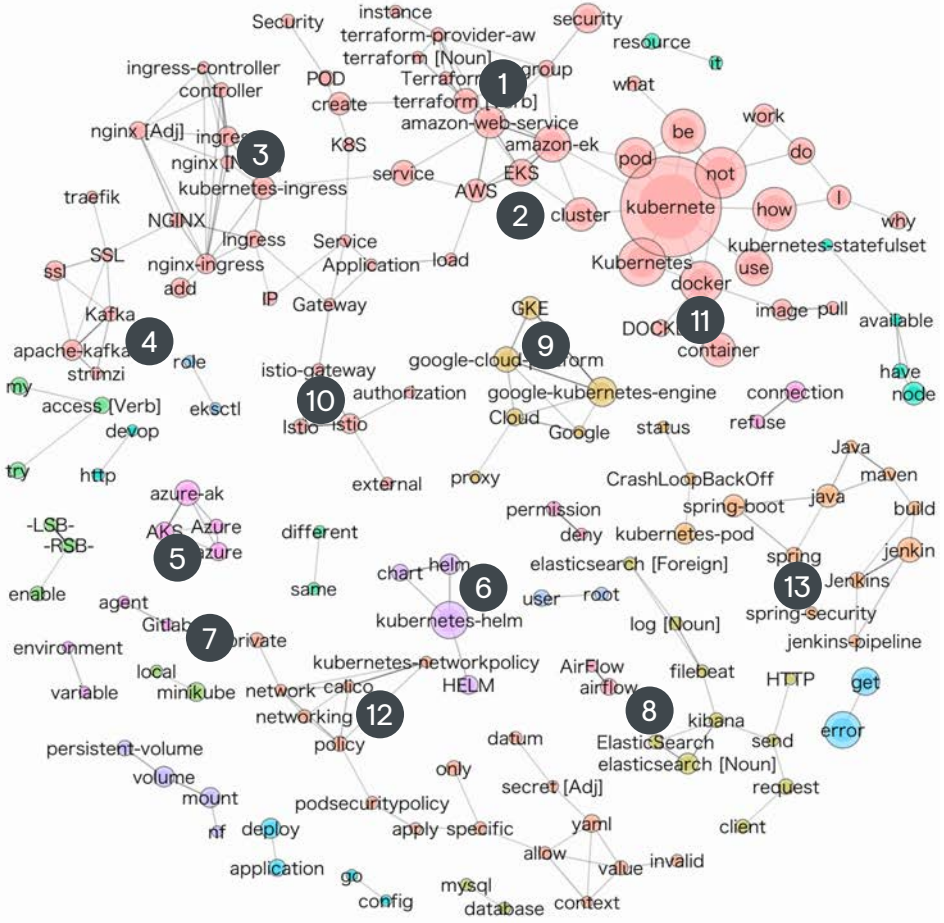
Example: A financial services company integrates a third-party analytics service into their Kubernetes environment. They perform a thorough security review of the third-party service, taking into account how it interacts with their Kubernetes resources and ensuring that data confidentiality, integrity, and availability are maintained. They also implement network policies to restrict the traffic between their pods and the third-party service to reduce the risk of a potential security breach.

Employee Training and Awareness: Employee training and awareness on Kubernetes security best practices is essential for preventing human errors and security incidents caused by improper configuration or misuse of Kubernetes resources.

Example: A tech startup running several critical applications on Kubernetes invests in comprehensive training for their developers and IT staff. The training includes best practices for Kubernetes security, such as how to configure network policies, manage secrets, and implement access controls. This investment significantly reduces the risk of security incidents caused by human error or misuse of Kubernetes resources and helps the company maintain a high standard of security hygiene.

Product-Related Developer Security Challenges

1. **Terraform:** Users are facing challenges with various aspects of Terraform, including security group management, Pod Security Policy configuration, and Load Balancer Controller sync issues. These issues can result in service disruptions, unauthorized access, and data breaches.
2. **EKS:** Users are encountering issues with EKS, such as multiple tagged security groups found for instances, syncing errors with Load Balancer Controller, and Pod Security Policy configuration problems. These issues can lead to security breaches, service disruptions, and compliance violations.
3. **NGINX:** Users are experiencing issues with NGINX, including segmentation violations, command injection, path traversal vulnerabilities, and denial of service attacks. These issues can result in unauthorized access, data leaks, and service disruptions, potentially impacting business operations.
4. **Kafka:** Users are facing challenges with Kafka, such as security verification on Fluent-bit, connecting to AKS Kafka Cluster, and updating package security in GKE Managed Instance Groups. These issues can result in unauthorized access, data breaches, and service disruptions, potentially impacting business operations.
5. **AKS:** Users are encountering issues with AKS, such as configuring security policies for NGINX ingress controller, setting securityContext/allowPrivilegeEscalation in Spring Cloud Data Flow, and connecting to ActiveMQ Artemis. These issues can lead to security breaches, service disruptions, and compliance violations, potentially impacting business operations.
6. **Helm:** Users are experiencing issues with Helm, including failures to install tekton and enabling security on Kibana after installation. These issues can result in unauthorized access, data leaks, and service disruptions, potentially impacting business operations.



Source: StackOverflow

7. **Git:** Users are facing challenges with GitOps tools, such as Argo CD, Cargo, and GitHub Actions Runner. These issues can result in unauthorized access, remote code execution, and session hijacking, potentially impacting business operations.
8. **Elastic:** Users are encountering issues with Elastic, including misconfigurations and vulnerabilities related to access control, code execution, and denial of service attacks. These issues can lead to unauthorized access, data breaches, and service disruptions, potentially impacting business operations.
9. **GKE:** Users are concerned with the security posture of their GKE clusters and are seeking ways to evaluate their security posture via API. Additionally, they face challenges in deploying applications securely, such as mounting security certificates and ensuring proper security contexts for pods.
10. **Istio:** Users are experiencing issues with Istio, including pod security on untrusted nodes and ensuring proper security policies are configured. These issues can lead to security breaches, service disruptions, and compliance violations, potentially impacting business operations.
11. **Docker:** Users are facing challenges with Docker, such as creating sample security issues and ensuring proper security context for containers, which can result in unauthorized access and data breaches.
12. **Calico:** Users are concerned with the interaction between Calico network policies and pod security groups and how they can use both simultaneously for a comprehensive security approach. Additionally, they face challenges in configuring security policies for Calico and Kubernetes, which can result in unauthorized access and data breaches.
13. **Spring:** Users are encountering issues with Spring, such as setting securityContext/allowPrivilegeEscalation while deploying a stream and Spring Security Okta Registration Redirect Uri. These issues can lead to security breaches, service disruptions, and compliance violations, potentially impacting business operations.

Operator Perspective: Five Categories of Security-Related Pain Points

Access and Authentication: These pain points focus on how to securely access Kubernetes from remote servers or clients, how to authenticate users, and how to manage access to Kubernetes resources.

Example: A large financial institution uses Kubernetes for managing their applications. They are facing challenges with implementing robust access controls and user authentication. They decide to integrate an identity provider (IdP) with their Kubernetes environment, enabling them to leverage federated identity management and single sign-on (SSO). This allows them to manage access to Kubernetes resources more effectively and securely.

Pod Security: These pain points address issues related to securing pods, including restricted PodSecurityPolicy, using pod security context, and ensuring that authenticated users cannot create pods.

Example: A cloud-based software company uses Kubernetes to manage their microservices. However, they discover that authenticated users are able to create pods, potentially leading to unauthorized resource consumption or malicious activity. They implement a PodSecurityPolicy that restricts the creation of pods to certain privileged service accounts, thereby enhancing the security of their Kubernetes environment.

Network Security: These pain points deal with securing network traffic and communications between Kubernetes resources, such as securing ingress to backends and adding firewall rules.

Example: An ecommerce company uses Kubernetes for their backend services. They are facing challenges in securing network traffic between their Kubernetes resources. They decide to implement network policies that restrict ingress and egress traffic to their backends, ensuring that only legitimate traffic is allowed. They also add firewall rules to further secure their Kubernetes environment.

Source: Serverfault

Container Security: These pain points focus on securing containers, including securing Docker hosts to prevent rooting, mounting NFS servers securely, and granting security capabilities to containers.

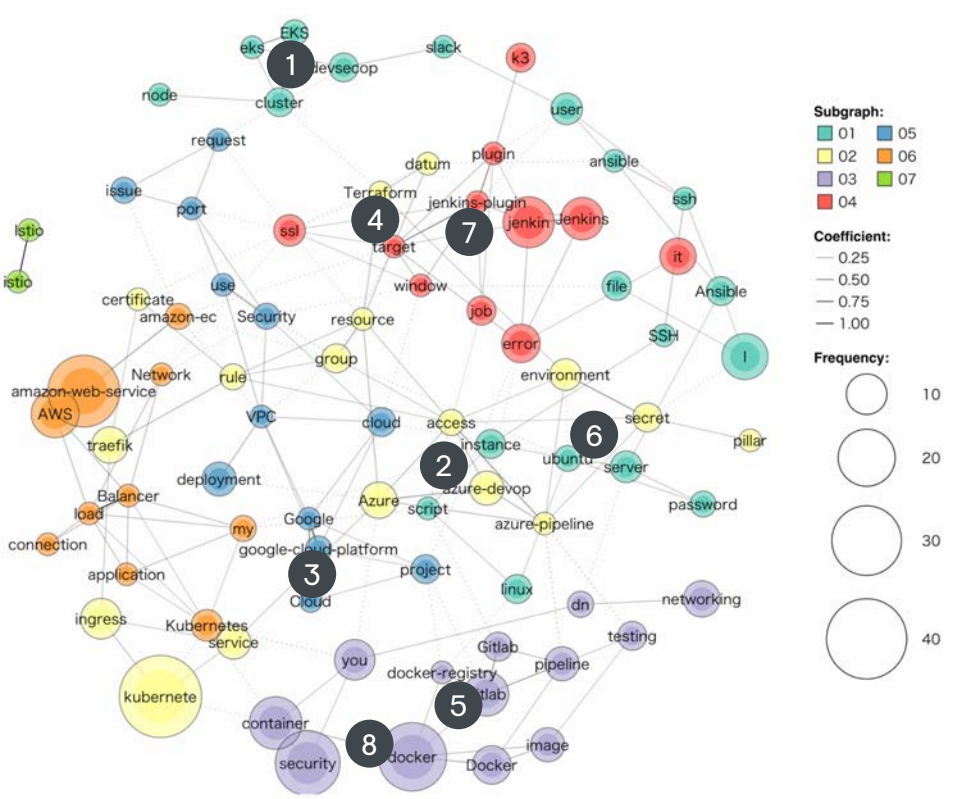
Example: A tech startup uses Docker containers in their Kubernetes environment. They are facing issues with securing their Docker hosts and NFS servers. They decide to implement strict security configurations for their Docker hosts and they mount their NFS servers securely to prevent unauthorized access. They also grant only necessary security capabilities to their containers, minimizing the potential attack surface.

Monitoring and Patching: These pain points address how to secure Kubernetes clusters and resources over time, such as applying system security patches, securing Prometheus monitoring, and securing the Kubernetes API port.

Example: A global telecommunications company uses Kubernetes for managing their IT infrastructure. They need to ensure that their Kubernetes clusters and resources remain secure over time. They implement a regular patching schedule for applying system security patches and they secure their Prometheus monitoring and the Kubernetes API port to prevent unauthorized access. They also deploy a Kubernetes-native security solution that provides real-time monitoring and automated patching capabilities, improving their overall security posture.

Security Pain Points for Operators

1. **EKS:** These challenges focus on various aspects of Amazon Elastic Kubernetes Service (EKS), such as handling authentication, integrating with other AWS services, and configuring cluster settings. Common concerns include managing spot instances for worker nodes, comparing EKS to other AWS services, and facilitating continuous delivery with tools like Helm.
2. **Azure DevOps:** These pain points center around challenges encountered when working with Azure Kubernetes Service (AKS), particularly in areas such as networking, permissions, and configuration. Common concerns include configuring health probes, implementing network policies, managing AKS ingress routing, and maintaining a balanced distribution of pods across nodes.
3. **GCP:** These problems focus on managing Google Kubernetes Engine (GKE) clusters, with emphasis on connectivity, deployment, and scaling. Common concerns include configuring Argo workflows, integrating GKE with Jenkins, troubleshooting issues with persistent volumes and GPUs, and assessing the value of GKE for specific use cases, like infrequent program execution.
4. **Terraform:** These topics address using Terraform for Kubernetes cluster management and automation, focusing on topics like deploying clusters on different platforms, automating Helm's Tiller installation, and working with Kubernetes manifests. Other concerns include installing Kubeflow, integrating with Azure Key Vault for TLS secrets, and handling specific Terraform issues related to data sources and provider blocks.
5. **Git:** These CVEs focus on security aspects of deploying and managing Kubernetes clusters using GitLab, Helm, and other tools for GitOps. Topics include access control when pulling private registry images, securing secrets with SealedSecrets, and configuring GitLab runners for CI/CD. Additionally, concerns arise about best practices for branch-to-namespace associations, automating Helm deployments, and managing short-lived feature branches in Openshift/Kubernetes.



6. **Ubuntu:** These challenges focus on the configuration and setup of Kubernetes with Docker containers on Ubuntu VMs, as well as using Minikube on Ubuntu 20.04 WSL. They also discuss the security implications of choosing between a generic Ubuntu base image and application-specific images, such as Node or Python, for containerized applications.
7. **Jenkins:** These problems include various aspects of Jenkins integration with Kubernetes, including connecting external Jenkins instances to Kubernetes clusters, deploying Helm charts from Jenkins, managing authentication and access control, handling private container registries, and ensuring secure communication in Jenkins X. They also discuss issues related to plugin compatibility, persistent storage, and running Docker-in-Docker containers alongside Jenkins agents.
8. **Docker:** Building and deploying Docker images, connecting Jenkins to Kubernetes, working with persistent storage in Docker, and using Ansible to manage Kubernetes. From a security perspective, these posts highlight potential vulnerabilities related to container image compatibility, registry authentication, and permission management.

Part 6: Public Cloud-Managed Kubernetes Alone is Not the Answer

10 Key Business Challenges of EKS, GKE, and AKS

From a business perspective, public cloud-managed Kubernetes alone is not the answer. Cloud-managed platforms such as EKS, AKS, and GKE are always embedded within the much larger context of the respective cloud vendor. While these offerings take care of managing the Kubernetes control plane, they still require enterprises to address many Kubernetes-related topics, such as cost control, security and data protection, scalability and performance, service reliability and availability, integration and interoperability, control and flexibility, technology lock-in, monitoring and analytics, and support and training.

Complexity and Learning Curve: While advertised as managed Kubernetes services, EKS, AKS, and GKE bring a tremendous amount of complexity into the organization. All three take care of the provisioning, upgrading, and maintaining of the Kubernetes control plane, but leave customers with a wide range of responsibilities.

Example: A leading ecommerce company decided to leverage the power of GKE for managing its microservices. While Google manages the Kubernetes control plane, the company soon realizes the breadth of responsibilities they still need to take on: creating and managing deployment scripts, maintaining pods, managing networking policies, or even tuning the performance of the underlying storage and compute. These are all tasks that are not directly handled by GKE and require deep Kubernetes expertise, contributing to the complexity and steep learning curve.

Cost Control: EKS, GKE, and AKS require continuous cost management and optimization to minimize overspending by paying for overprovisioned and unused resources and services.

Example: A media streaming startup is using AKS to handle their global customer base. The business initially overprovisioned their clusters to handle expected traffic. However, they soon notice that during off-peak hours, they're incurring unnecessary costs from the unused resources. They must implement tools and practices to track resource usage and adjust their provisioning in real time to avoid paying for resources they don't need.

"Leveraging EKS, AKS, and GKE is not an end in itself. It is the starting point to address the multifaceted challenges businesses face in cloud environments." - Director Platform Engineering, Multinational Insurance Firm

Security and Data Protection: Businesses are responsible for protecting sensitive data and maintaining compliance with regulatory requirements.

Example: Consider a health tech company that utilizes EKS for deploying its sensitive applications. While AWS takes care of the Kubernetes security at the control plane level, the company is responsible for the security of their applications running on the clusters. This includes securing the application secrets, implementing network policies, and ensuring data encryption. With sensitive patient data at stake, maintaining compliance with regulations like HIPAA becomes a critical task for the company.

Scalability and Performance: Ensuring the seamless operation of these services quickly becomes a challenging task, especially when dealing with complex hybrid and distributed workloads with high traffic volumes and scalability requirements, and many external integration points.

Example: A popular gaming company uses managed Kubernetes services to deploy its new online multiplayer game. As the game quickly gains popularity, the company struggles to handle the sudden increase in traffic, leading to slow game performance and dissatisfied users. This highlights the need for the company to effectively manage the scalability of their services, optimize resources, and ensure high performance for their users across different regions and at different load levels. This is a challenging task that requires a sophisticated understanding of how to scale and optimize Kubernetes clusters efficiently.

Service Reliability and Availability: Downtime and service disruptions can easily occur through poorly designed application architectures; incorrectly configured deployments, services, ingress, or network policies; insufficient CPU, memory, or storage resources; misconfigured autoscaling rules and policies; and developers or operators making changes to resource configuration or even deleting resources.

Example: A fintech company is using AKS to handle its digital banking application. One day, the banking app goes down due to misconfigured deployment settings made by one of the operators. The customers start facing issues like being unable to access their accounts, leading to a rise in customer complaints and reputation damage. The incident shows how crucial it is to maintain service reliability and availability and how a single misconfiguration can disrupt the entire service.

Integration and Interoperability: Integrating with existing systems and applications can be challenging, and businesses must ensure that these cloud services work seamlessly with other systems and applications.

Example: A multinational corporation is utilizing GKE for its data analytics applications. However, the company is facing challenges in integrating these applications with their legacy on-premises database systems. This is leading to inefficient data exchange and loss in overall productivity. The company now has to invest in building custom adapters and make several changes to ensure seamless interoperability, which adds to the complexity of the deployment.

Lack of Control and Flexibility: Businesses may feel like they have limited control over these cloud services and may struggle with customization and flexibility.

Example: Consider a health care technology firm that is using EKS to manage its health record systems. They find that while they can outsource the management of the Kubernetes control plane to AWS, they are restricted in their ability to customize some settings to fit their specific use case. For instance, they can't apply custom kernel settings or install specific networking plugins to enhance performance due to the managed nature of EKS. This lack of control and flexibility limits their ability to tailor the Kubernetes environment to their specific needs and constraints.

Vendor Lock-In: Businesses may find it challenging to switch to a different cloud provider, resulting in dependence on a single provider.

Example: An online retail company decided to move their infrastructure to GKE to take advantage of Google's strong machine learning offerings. However, as the business grows, they realize they also need to leverage AWS's advanced data analytics services. The company finds it challenging to switch or even distribute workloads across cloud providers due to the lack of interoperability between GKE and AWS. This reliance on a single provider, GKE in this case, represents vendor lock-in.

Monitoring and Analytics: Monitoring and analyzing cloud utilization and cost can be challenging and require specialized tools and expertise.

Example: A tech startup uses EKS for its SaaS product. Despite having the service up and running, they struggle to get clear insights into the system's performance, utilization, and cost. Although AWS provides some monitoring tools, the team lacks the expertise to interpret the data effectively, indicating a need for more specialized tools and training.

Support and Training: Businesses may require significant support and training to effectively operate and manage these cloud services.

Example: A manufacturing company decides to modernize its legacy systems and moves toward using AKS for managing their applications. However, the existing IT team, skilled primarily in traditional systems, struggles to navigate the complexities of Kubernetes. They encounter problems from deployment to scaling, indicating a need for extensive support and training in order to operate these services effectively.

The Impact of Human Mistakes

Human mistakes can cause a variety of challenges in a cloud-managed and traditional Kubernetes environments alike.

Human Mistake	Example
Misconfiguration: Errors in configuring Kubernetes resources, such as deployments, services, or ingress controllers, could lead to application failures, downtime, or incorrect routing of traffic.	An ecommerce platform is using GKE for its operations. A configuration error in the ingress controller mistakenly routes users' checkout requests to the product browsing service, resulting in incomplete transactions and frustrated customers.
Resource Limitations: Improper allocation of CPU and memory to pods or nodes might lead to performance issues or even crashes if Kubernetes cannot gracefully handle excessive resource usage.	A digital marketing agency uses EKS to manage its advertising platform. Due to improper allocation of CPU and memory resources to the pods, the application starts lagging during peak user activity, affecting campaign effectiveness and client satisfaction.
Insufficient Access Controls: Failing to set up proper RBAC can lead to unauthorized access or accidental changes to the cluster configuration by users or services.	An admin user unintentionally grants a "read/write" role to a new user, allowing them to make changes to the cluster configuration.
Wrong Container Image: Using incorrect or outdated container images might result in deploying applications with bugs, security vulnerabilities, or incompatible dependencies.	A developer deploys an application using a deprecated Python 3.5 Docker image, resulting in security vulnerabilities.
Inadequate Monitoring and Logging: Misconfiguration of monitoring and logging tools can make it difficult to identify and troubleshoot issues in the cluster.	The monitoring system is misconfigured to only log errors every 24 hours, making it difficult to track real-time issues in the cluster.
Inadequate Scaling and Performance Planning: Failing to plan for application scaling needs or performance optimization can lead to resource issues and service degradation, especially during periods of high load.	The ecommerce application isn't designed to auto-scale, causing it to crash during a sudden spike in user traffic on Black Friday.
Lack of Redundancy and Disaster Recovery Strategy: If a proper disaster recovery strategy is not in place or there isn't enough redundancy, an unexpected failure or human error can have severe consequences that impact the entire cluster or application.	In the absence of a robust disaster recovery plan, a power outage at the data center leads to prolonged downtime for the cluster.
Poor Data Management: Human errors in managing persistent data, such as misconfiguring storage or not creating proper backup/restore strategies, can result in data loss or corruption.	A database administrator forgets to configure a backup strategy for a critical PostgreSQL database, leading to irreversible data loss after an unexpected disk failure.
Networking Issues: Misconfiguration of network policies or incorrect network settings can lead to communication failures between pods and services or expose services to unauthorized access.	An incorrect network policy configuration blocks inter-pod communication, preventing services from functioning correctly.
Failure in Applying Updates and Patches: Delayed or incomplete application of security patches or software updates can expose the cluster to various security risks or stability issues.	A critical Kubernetes security patch is overlooked and not applied, exposing the cluster to a known exploit.

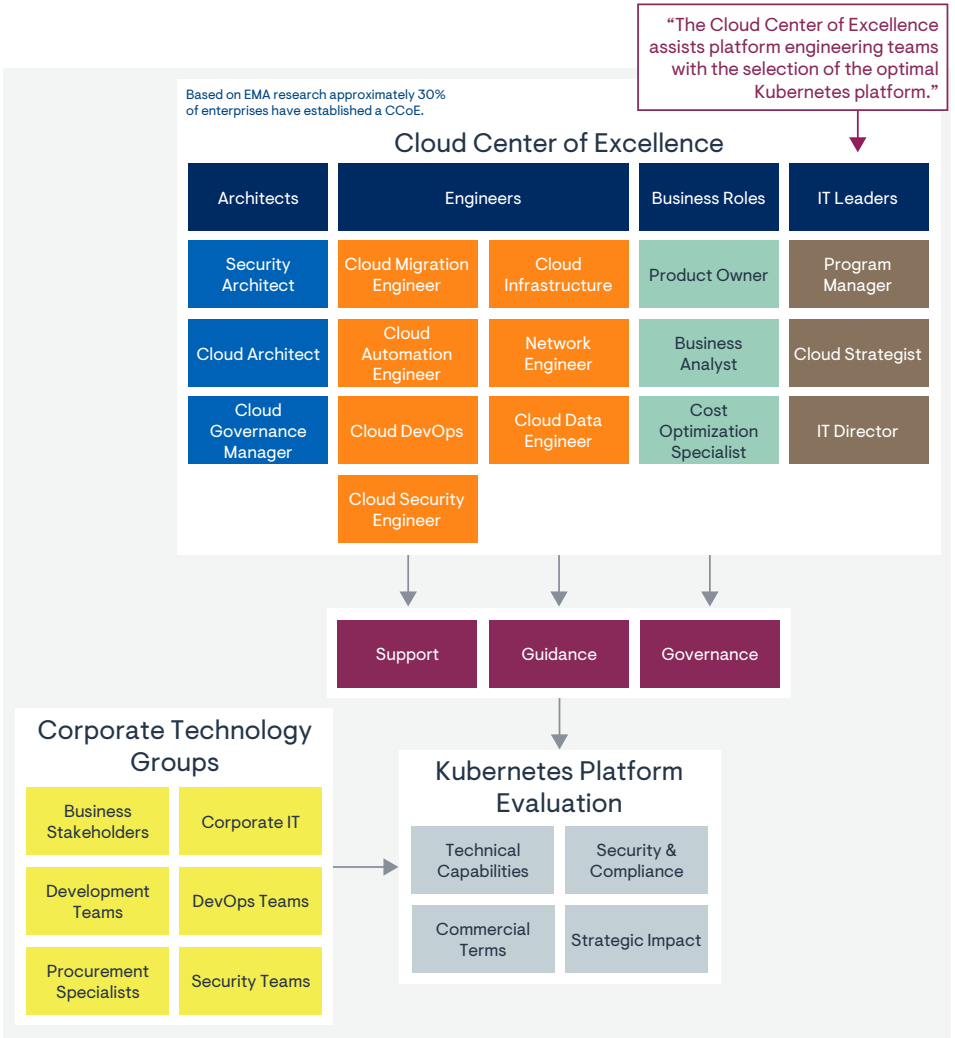
Empowering Cloud Innovation and Governance: The Pivotal Role of Cloud Centers of Excellence in Enterprise Success

Cloud Centers of Excellence (CCoE) typically consist of architects, engineers, IT leaders, and business personas focused on improving cloud governance and security, driving cloud adoption and innovation, reducing cloud cost, and improving cloud management and operations. They provide a central point of expertise and help organizations plan and implement cloud strategies. Within this capacity, the CCoE assists platform engineering teams with the election of the best possible Kubernetes platform based on technical capabilities, financial terms, security and compliance, and expected strategic impact of the new platform.

Furthermore, the CCoE acts as a catalyst for cloud innovation within the organization. They keep a pulse on emerging cloud technologies, trends, and advancements and evaluate their potential impact on the business. Through research, proof-of-concept projects, and pilot programs, the CCoE identifies opportunities to leverage cutting-edge cloud capabilities to drive business growth, enhance customer experiences, and improve operational efficiencies. By actively exploring new possibilities, the CCoE fosters a culture of experimentation and continuous improvement, enabling the organization to stay ahead in the rapidly evolving cloud landscape.

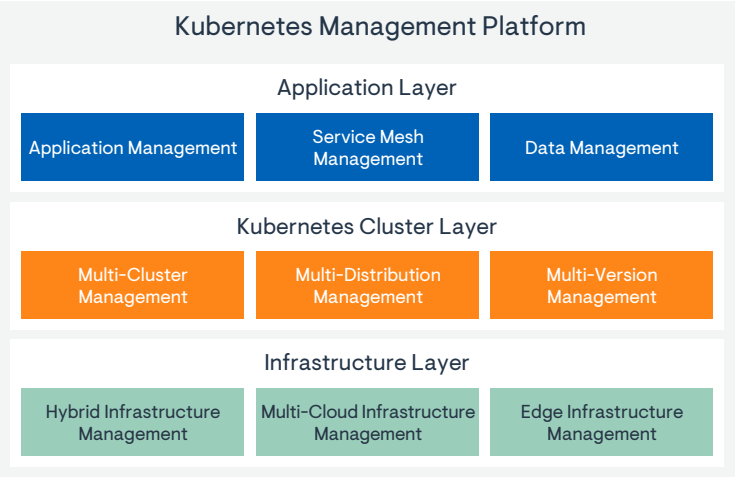
By providing centralized expertise and guidance, enforcing governance policies, promoting best practices, and driving innovation, the Cloud Center of Excellence empowers organizations to harness the full potential of cloud computing. Through their collaborative efforts, the CCoE helps create a well-architected, secure, and cost-effective cloud environment that accelerates digital transformation, fuels innovation, and positions the organization for long-term success in the cloud era.

Based on EMA research, approximately 30% of enterprises have established a CCoE, with a significant percentage of additional organizations having implemented variations of the CCoE, such as cloud enablement teams, distributed cloud champions, or cloud governance committees.



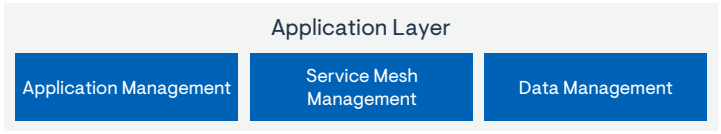
Source: EMA Poll

Part 7: The Importance of a Unified Container Management Platform



A unified container management platform can provide business-driven management of applications, their underlying Kubernetes clusters, and the infrastructure on which these clusters rely.

ADDRESSING THE CHALLENGES AT THE APPLICATION LAYER



A unified management framework for Kubernetes not only helps manage the Kubernetes clusters themselves, but also aids in managing applications, service mesh, and data across clusters. Here's how the framework can assist with each aspect.

Application Management:

- Consistent way to deploy, manage, and scale applications across different Kubernetes clusters and cloud providers.
- Enable seamless application migration between clusters and cloud providers, reducing the risk of downtime while optimizing cloud resource cost.
- Centralized monitoring and logging for applications helps track performance and troubleshoot issues.
- Include CI/CD pipelines and other DevOps tools for consistent and automated application deployment and management.

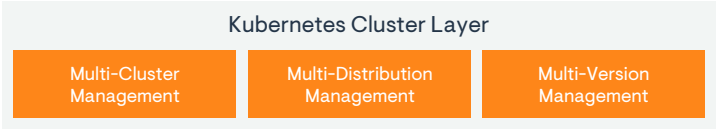
Service Mesh Management:

- Simplify the deployment and management of service meshes like Istio, Linkerd, or Consul across multiple Kubernetes clusters.
- Centralized control over service mesh configurations, allowing you to manage routing, load balancing, and security policies consistently across different clusters and cloud providers.
- Offer observability tools to monitor and analyze service mesh performance and troubleshoot issues.
- By integrating with popular service mesh solutions, the framework can help reduce the learning curve and complexity associated with managing service meshes.

Data Management:

- Help manage stateful applications and their associated data across different Kubernetes clusters and cloud providers.
- Facilitate the deployment and management of storage solutions like persistent volumes, StatefulSets, and distributed databases.
- Help ensure data consistency, durability, and high availability by managing replication, backup, and recovery mechanisms.
- It can provide tools to monitor and optimize data storage usage, performance, and cost.

ADDRESSING THE CHALLENGES AT THE CLUSTER LEVEL



A unified management framework for Kubernetes can address the challenges of using managed Kubernetes services like EKS, GKE, AKS, and any other Kubernetes distribution by providing a single, consistent platform for managing all aspects of the Kubernetes clusters across different cloud providers.

- **Complexity and Learning Curve:** Simplify the administration of Kubernetes clusters by offering a consistent user interface, standardizing workflows, and abstracting the underlying cloud provider's complexities.
- **Cost Control:** Provide cost optimization tools and insights to identify underutilized resources, automate rightsizing, and manage resource scaling to minimize overspending.
- **Security and Data Protection:** Deliver a centralized security policy management system, ensuring that consistent security policies are applied across all clusters, regardless of the cloud provider. Additionally, it can provide tools to help businesses maintain compliance with regulatory requirements.
- **Scalability and Performance:** Provide automatic scaling, load balancing, and resource optimization to help manage complex workloads and ensure that the system operates smoothly under high-traffic conditions.
- **Service Reliability and Availability:** Offer tools for managing deployments, services, and network policies, as well as identifying and mitigating potential risks and bottlenecks that may lead to downtime.
- **Integration and Interoperability:** Provide a consistent API and a set of connectors to easily integrate with existing systems and applications, ensuring seamless interoperability.
- **Lack of Control and Flexibility:** Give businesses more control over their Kubernetes clusters by providing a consistent way to manage, customize, and extend the platform across different cloud providers.
- **Vendor Lock-In:** Reduce vendor lock-in by offering a consistent platform to manage Kubernetes clusters across different cloud providers, making it easier to switch or use multiple providers simultaneously.
- **Monitoring and Analytics:** Provide centralized monitoring and analytics tools, making it easier to track and analyze cloud utilization, performance, and costs across different cloud providers.
- **Support and Training:** Offer businesses consistent documentation, training materials, and support services, reducing the learning curve and enabling them to effectively operate and manage their Kubernetes clusters across different cloud providers.

ADDRESSING THE CHALLENGES AT THE INFRASTRUCTURE LEVEL

A unified container management framework can provide significant benefits for hybrid cloud management, multi-cloud management, and edge management. Here's how the framework can assist with each aspect.

Hybrid Cloud Management:

- The framework can provide a consistent way to manage Kubernetes clusters running in both on-premises data centers and public cloud environments.
- It can simplify the deployment, scaling, and migration of applications and data between on-premises and cloud environments.
- Centralized monitoring, logging, and analytics can help you manage performance and troubleshoot issues across your hybrid cloud infrastructure.
- The framework can help enforce consistent security and compliance policies across on-premises and cloud environments.

Multi-Cloud Management:

- The unified management framework can offer a single control plane for managing Kubernetes clusters across multiple cloud providers, such as AWS, Google Cloud, and Azure.
- It can simplify the deployment and scaling of applications and data across multiple cloud environments, allowing you to leverage the unique benefits of each cloud provider.
- By providing a consistent API and set of connectors, the framework can help ensure interoperability between different cloud providers.
- The framework can offer centralized cost management and optimization tools to help you minimize cloud spending across multiple providers.

Edge Management:

- The framework can help manage Kubernetes clusters running on edge devices and remote locations, ensuring a consistent management experience across your entire infrastructure.
- It can provide tools for deploying, managing, and scaling applications and data at the edge, taking into account the unique constraints of edge environments, such as limited compute and storage resources.
- The framework can offer centralized monitoring and analytics for edge environments, allowing you to manage performance and troubleshoot issues from a single control plane.
- By helping enforce consistent security and compliance policies across edge and centralized environments, the framework can ensure data protection and regulatory compliance across your entire infrastructure.
- By providing a consistent, unified approach to managing Kubernetes clusters across hybrid cloud, multi-cloud, and edge environments, a unified container management framework can help businesses optimize their operations, reduce complexity, and ensure optimal performance and reliability across their infrastructure.
- By providing a consistent, unified approach to managing Kubernetes clusters across hybrid cloud, multi-cloud, and edge environments, a unified container management framework can help businesses optimize their operations, reduce complexity, and ensure optimal performance and reliability across their infrastructure.

Impact of a Unified Management Framework on Developer Productivity

A unified management framework enhances developer productivity by streamlining the development, deployment, and management of applications across multiple Kubernetes clusters and environments. It brings consistency, simplification, and automation to the process, allowing developers to focus on writing and deploying code rather than managing infrastructure.

By addressing these aspects, a unified management framework helps developers concentrate on their core tasks of building, testing, and deploying applications, resulting in increased productivity and faster time-to-market. For example, a team can test their new Go application on a local Kubernetes cluster, then deploy it to a production cluster on Azure with minimal changes and without downtime, thanks to the migration capabilities of the framework.

Benefit	Description	Example
Consistent Workflows	The framework provides a standardized way to deploy and manage applications across different Kubernetes clusters and environments, reducing the learning curve and complexity for developers.	With this framework, developers can deploy a Node.js application to any Kubernetes cluster regardless of whether it's hosted on AWS, GCP, or on-premises, without having to learn new deployment procedures for each environment.
Simplified Management	A unified management framework abstracts the underlying infrastructure complexity, enabling developers to manage applications without needing deep expertise in Kubernetes or cloud-specific services.	A developer without deep knowledge in Kubernetes or AWS services can use this framework to deploy and manage an application without dealing with the intricacies of Kubernetes manifests or AWS EC2 instances.
Integrated DevOps Tools	The framework often includes CI/CD pipelines and other DevOps tools, which facilitate automated and consistent application deployment and management, speeding up development cycles and reducing errors.	The framework includes Jenkins for CI/CD, allowing developers to consistently build, test, and deploy a Python web application whenever code changes are pushed to the repository, minimizing manual error.
Enhanced Collaboration	By providing a single control plane, the framework encourages collaboration among developers, operations, and security teams, fostering faster feedback loops and more efficient development processes.	Developers, operations, and security teams can work together using the single control plane provided by the framework, allowing for faster feedback loops and more efficient troubleshooting during the development of a Java microservices application.
Centralized Monitoring and Logging	Unified monitoring and logging tools help developers quickly identify and troubleshoot issues, reducing the time spent on debugging and maintenance tasks.	The framework includes integrated Grafana dashboards and ELK stack for logging, helping developers quickly identify issues in a deployed Ruby application, reducing debugging and maintenance time.
Interoperability	The framework supports smooth application migration between clusters and environments, allowing developers to rapidly test and deploy their applications in different settings without downtime.	A team can test their new Go application on a local Kubernetes cluster, then deploy it to a production cluster on Azure with minimal changes and without downtime thanks to the migration capabilities of the framework.

Impact of a Unified Management Framework on Business Metrics

Based on the advantages of unified Kubernetes management frameworks, the following key business metrics can be derived.

Benefit	Description	Example
Time-to-Market	The reduced learning curve, integrated DevOps tools, and streamlined workflows can result in faster development and deployment of applications, leading to shorter time-to-market.	A startup was able to deploy their MVP to market in three months instead of six thanks to the framework's streamlined workflows and integrated DevOps tools.
Deployment Frequency	With simplified management and automated deployment processes, developers can deploy new features and updates more frequently, leading to continuous improvement and innovation.	A web service team was able to increase their deployment frequency from once a month to once a week, allowing for faster iteration and innovation.
Mean Time to Resolution (MTTR)	Centralized monitoring and logging, as well as enhanced collaboration, can help developers quickly identify and troubleshoot issues, reducing the time it takes to resolve problems and minimize their impact on the business.	An ecommerce platform reduced their MTTR from four hours to one hour due to the framework's centralized monitoring and enhanced collaboration features.
Cost Savings	By simplifying management and optimizing resource utilization, a unified management framework can help businesses reduce infrastructure and operational costs.	A tech company was able to save $1M annually in infrastructure costs by utilizing the framework's simplified management and resource optimization capabilities.
Developer Productivity	Increased productivity can be measured by the amount of code produced, features deployed, or projects developers complete within a given time frame.	With the unified management framework, a team of developers was able to deliver twice the number of features in the same amount of time as before.
Team Collaboration Efficiency	Enhanced collaboration can lead to more efficient decision-making, faster feedback loops, and better alignment of goals and priorities, resulting in improved overall team efficiency.	The development and operations teams were able to achieve better alignment and faster decision-making, reducing project delivery time by 30%.
Application Performance and Reliability	The streamlined deployment and management of applications across various Kubernetes clusters can lead to better application performance and higher reliability, which can be measured through metrics like uptime, response time, and error rates.	An online video streaming service noticed a 50% reduction in error rates and an increase in uptime after adopting the framework for managing their Kubernetes clusters.

By monitoring these key business metrics, organizations can evaluate the impact of unified Kubernetes management frameworks on their operations and make data-driven decisions to optimize their processes further.

Key Takeaways

1. Developers need to be able to create, deploy, run, manage, and continuously enhance their code without spending time on learning, implementing, and debugging basic topics related to the setup, configuration and management of compute, network, storage, security, and observability.
2. Enterprises need to focus on achieving the ultimate productivity goal of enabling the same application code to run on all target clusters without code modifications. This allows platform engineers to deploy, scale, run, manage, debug, and upgrade applications without requiring help from the development team.
3. Policy-driven unified automation and management of the deployment, management, operations, and upgrade of Kubernetes clusters across clouds, data centers, and edge location is the precondition for the large-scale adoption of Kubernetes.
4. Public cloud-managed Kubernetes clusters alone are not the answer. They leave a large number of deployment, management, and integration tasks to the customer. Amazon EC2 Kubernetes Services (EKS), Azure Kubernetes Services (AKS), and Google Kubernetes Engine (GKE) come with significant cost pitfalls because they require customers to assign and scale network storage and server resources for their Kubernetes clusters.
5. Enterprises adopt multiple Kubernetes distributions based on individual team and project requirements, regulatory compliance, workload requirements, cost, and flexibility. This leads to an increasing number of corporate Kubernetes clusters that can be located in on-premises, the cloud, or edge cloud locations.
6. A unified Kubernetes management platform is critical to accelerate time-to-market, deployment frequency, and mean time to repair by helping to maximize developer productivity, collaboration, and application performance and reliability.
7. While organizations have quickly adopted public cloud-managed Kubernetes, there is an estimated split of 50/50 between public and private cloud Kubernetes.
8. Kubernetes special interest groups (SIG) and working groups (wg) continuously guide and channel the progress of the upstream Kubernetes codebase. APIs, storage, security, and observability are some of today's key topics.

About Enterprise Management Associates, Inc.

Founded in 1996, Enterprise Management Associates (EMA) is a leading industry analyst firm that provides deep insight across the full spectrum of IT and data management technologies. EMA analysts leverage a unique combination of practical experience, insight into industry best practices, and in-depth knowledge of current and planned vendor solutions to help EMA's clients achieve their goals. Learn more about EMA research, analysis, and consulting services for enterprise line of business users, IT professionals, and IT vendors at **www.enterprisemanagement.com**. You can also follow EMA on **Twitter** or **LinkedIn**.

This report, in whole or in part, may not be duplicated, reproduced, stored in a retrieval system or retransmitted without prior written permission of Enterprise Management Associates, Inc. All opinions and estimates herein constitute our judgement as of this date and are subject to change without notice. Product names mentioned herein may be trademarks and/or registered trademarks of their respective companies. "EMA" and "Enterprise Management Associates" are trademarks of Enterprise Management Associates, Inc. in the United States and other countries.

©2023 Enterprise Management Associates, Inc. All Rights Reserved. EMA™, ENTERPRISE MANAGEMENT ASSOCIATES®, and the mobius symbol are registered trademarks or common law trademarks of Enterprise Management Associates, Inc.

2770 Arapahoe Road, Ste 132 - 544, Lafayette, CO 80026 +1 303.543.9500 **www.enterprisemanagement.com**

4303.062823