CTOs' Guide to Containers and Kubernetes — Answering the Top 10 FAQs

Published 31 May 2022 - ID G00763328 - 17 min read

By Analyst(s): Arun Chandrasekaran, Wataru Katsurashima

Initiatives: Technology Innovation; Cloud and Edge Infrastructure

Containers and Kubernetes have emerged as prominent platform technologies for building cloud-native apps and modernizing legacy workloads. We answer common client questions on containers and Kubernetes that enable enterprise architecture and technology innovation leaders to make sound decisions.

Additional Perspectives

- Summary Translation + Localization: CTOs' Guide to Containers and Kubernetes — Answering the Top 10 FAQs
(26 August 2022)

Overview

Key Findings

- Enterprise architecture and technology innovation leaders are keen to invest in container platform tools to enable software agility, developer productivity and to modernize their applications.

- Kubernetes has become a popular platform for building cloud-native applications, but the key constraints are lack of adequate skills and mature DevOps practices to both operationalize and succeed with large-scale production deployments.

- Large cloud providers and container management software vendors are competing to be the platform of choice for cloud-native apps. Multicloud management, DevOps toolchain integration, integrated security, flexible consumption models and product maturity are among the key differentiators.

- Enterprises face challenges in accurately measuring the ROI of their cloud-native investments and in creating the right organizational structure for it to flourish.

Recommendations

Enterprise architecture (EA) and technology innovation leaders driving business transformation through technology innovation should:

- Ensure that a strong business case exists, identify appropriate use cases and institute a DevOps culture before committing to and scaling Kubernetes platform environments.
- Create a platform team that curates platform selection, drives standardization and automation of DevOps functions, and collaborates with developers to foster cloud-native architectures.
- Choose packaged software distributions or cloud-managed services for production deployments that integrate different technology components, simplify the life cycle management of that stack and provide multicloud management — rather than adopting a DIY approach.
- Measure and communicate the benefits accurately, both in terms of technical metrics like software velocity, release success and operational efficiency gain — as well as through business metrics, such as top line growth and customer satisfaction.

Strategic Planning Assumption(s)

By 2027, more than 90% of global organizations will be running containerized applications in production, which is a significant increase from fewer than 40% in 2021.

By 2027, 25% of all enterprise applications will run in containers, an increase from fewer than 10% in 2021.

By 2027, more than 65% of commercial-of-the-shelf (COTS) vendors will offer their software in container format, up from less than 20% in 2021.

Introduction

Containers and Kubernetes have significantly grown in popularity and adoption over the past five years. While container technology has been in existence for more than a decade, its recent meteoric rise can be attributed to changes in software architecture and development patterns, growing adoption of DevOps and the fact that both containers and Kubernetes are open-source projects supported by a wide ecosystem of participants. In this research, we answer the most common client questions about containers and Kubernetes that will help enterprise architecture and technology innovation leaders benefit from and operationalize these technologies.

Questions and Answer Links

Analysis

1. What are some of the key benefits of containers and Kubernetes?

Containers and Kubernetes can bring several benefits to an organization. Some of the key benefits are:

- Agile application development and deployment: Containers help simplify application packaging and enable a rapid application deployment process with the ability to do frequent application builds, quick software release and granular rollbacks. The ability to develop and deploy software faster can have a significant impact on top-line growth and customer experience.
- Environmental consistency: Through tight application component packaging, containers enable platform consistency across development, testing, staging and production clusters. This is an important driver of developer productivity and service resiliency.
- Immutability: Containers should be deployed in an immutable way using declarative principles — which means that there won't be any out of process changes or patches. This makes container deployments highly repeatable, automated and secure with lower operational overheads.
- Flexibility and choice: Kubernetes is supported by a huge ecosystem of cloud providers, ISVs and IHVs. This API and cross-platform consistency, open-source innovation and industry support offers a great degree of flexibility for CTOs.

2. What limitations and challenges should we be aware of?

- Platform complexity: While containers and Kubernetes can be used for many use cases today, they don't need to be used in every case. Using Kubernetes to orchestrate static, COTS applications can be overkill due to the inherent complexity of Kubernetes that would overshadow any meaningful business benefits that may accrue. Similarly, deploying it for performance-sensitive and stateful applications requires significant know-how and operational maturity.
- Security: While there is nothing inherent in the container technology that makes it unsecure, deploying it at scale requires a mature DevSecOps process and shared responsibility between developers, platform operations, SRE and security teams.
- Automation and governance: Successful container deployments require extensive and continuous curation of technology components, consistent operations and upgrade of existing tools and processes to ensure automation and governance. This necessitates investments in new tooling and removing any constraints in enabling an agile environment — which can often lead to conflicting opinions across teams, as well as a laborious time-consuming selection and integration process.

- Culture and skills: Organizations face steep challenges in building containerized apps and operationalizing them due to a dearth of skills across development, security and operations teams. Moreover, deploying a common platform like Kubernetes requires clearly defined shared responsibilities across teams and a growth mindset that can tolerate failures and iterate accordingly.

3. What are the common use cases for containers and Kubernetes?

There are several use cases for containers and Kubernetes, as outlined below:

- **Microservices**: Microservices application architecture is characterized by independent application components that are distributed and loosely coupled. Containers and Kubernetes enable a strong foundation architecture for microservices due to their ability to orchestrate these modular services, enable scaling and self-healing of the services, and create a layer of service isolation.
- **DevOps enabler**: Containers enable a CI/CD process by isolating code into a discrete unit, which makes it easier for developers and DevOps teams to modify and update the code across the software development life cycle.
- **Application portability (or lock-in risk reduction):** The runtime parity of containers, and the ubiquitous availability of Kubernetes, enable developers to build apps that can run in a consistent way across hybrid or multicloud environments. While complete application portability is usually not possible because most applications have dependencies outside containers and Kubernetes, organizations can reduce the vendor lock-in risk.
- **Legacy app modernization:** You can take advantage of the more efficient deployment and service isolation of containers by modernizing and streamlining the life cycle management legacy applications. However, simple, lift-and-shift types of migrations are usually only feasible for lightweight workloads.

4. Can commercial off-the-shelf (COTS) applications be deployed in containers?

As of today, most container images are based on open-source software. According to the Datadog Annual Survey on containers, [1] the top six common container images are NGINX, Redis, Postgres, Elasticsearch, RabbitMQ and Calico — all of which are open-source software. Compared to open source software, where container support is already commonplace, the container support for COTS applications has been much slower and greatly varies by vendor. While some COTS ISVs strategically provide strong support for Kubernetes, such as IBM, many COTS ISVs haven't supported yet — especially in Windows-based or enterprise business applications. You should review container support strategy and roadmaps of their strategic COTS ISVs.

However, although slower than open source software, the container support of COTS is steadily increasing. For example, AWS Marketplace for Containers has a total of 524 container-related entries as of February 2022, 64% up from 320 as of February 2020. Also, more ISVs are enabling deeper integrations with containers/Kubernetes than just providing container images. You can find such examples at OperatorHub.io (backed by Red Hat, AWS, Microsoft and Google Cloud), which is a public registry for Kubernetes operators that simplify the life cycle management of containerized applications. The software operators available for Kubernetes include Apache Spark, Cassandra, CockroachDB, Couchbase, Kafka, PostgreSQL, MongoDB, NuoDB and Redis.

5. How do we determine which applications are good candidates for containers and Kubernetes?

Applications that are good initial candidates for containerization exhibit the following traits:

- Low degree of external application dependencies
- The infrastructure and platform technologies that the application runs on are already available as container images (i.e., Linux, Tomcat, NGINX, PostgreSQL, etc.)
- Preferably stateless apps for initial deployments
- Application has rapid elasticity needs and requires frequent code changes
- If you deploy a COTS application, there is a vendor supported image for it

6. How do we measure the ROI of our container deployments?

Ensuring ROI by building a thorough business case is important to validate that you aren't investing in containers and Kubernetes purely because it is a shiny new technology. Organizations need to take a realistic view of the costs incurred and potential benefits.

The potential benefits could be:

- **Developer productivity:** Organizations often measure this on a per developer basis annually.
- **Agile CI/CD environment:** This is often measured through deployment frequency, lead time for changes, release success rate, time for service restore — as well as business metrics, such as user satisfaction and top-line growth.
- **Infrastructure efficiency gain:** Because of their smaller resource footprint, containers can enable a much higher tenant density on a host. This increases utilization of host resources, as well as ISV licensing costs in scenarios where commercial off-the-shelf (COTS) software is licensed on a per host basis. This is measured based on deferred hardware upgrades, better infrastructure utilization and potential ISV savings.
- **Reduced operational overhead:** Kubernetes abstracts away a fair amount of low-level operational management, freeing up valuable IT staff time. This is measured as IT staff productivity savings.

Costs:

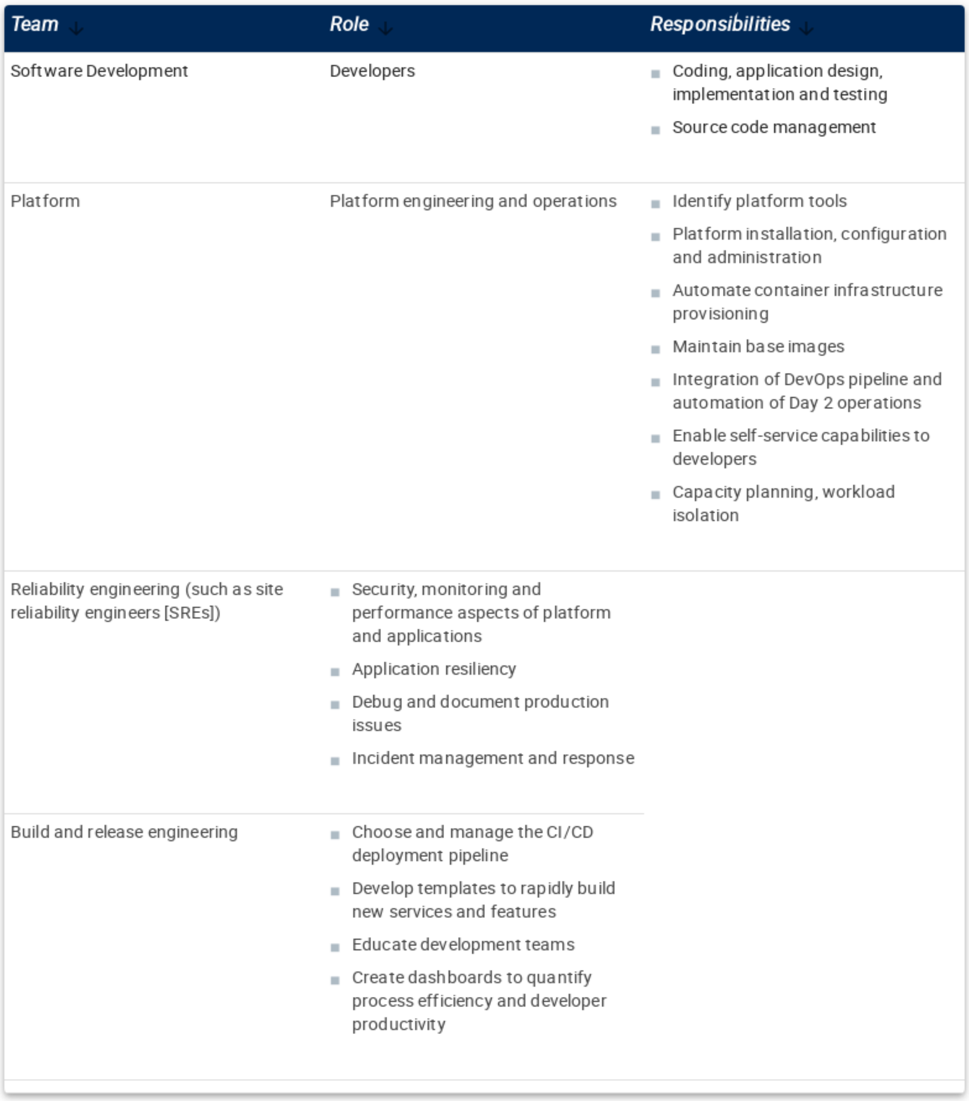
- **CaaS/PaaS subscription and other licenses:** Most organizations consume commercially supported downstream Kubernetes software distribution or cloud services, which involves paying for annual software subscription fees or for cluster management (cloud services). In addition, optional software licensing fees may include security, monitoring and automation tools.
- **Infrastructure acquisition and upgrades:** These would include investments in compute, storage and networking infrastructure (software and hardware), and cloud-based IaaS required to support containerized applications.
- **Staff training and new hires:** Container and Kubernetes investments often entail training programs to upskill existing employees across software development, operations and security. They also demand money to hire new employees to fill gaps in any of the aforementioned areas — or in newer practice areas, such as release engineering, automation engineering or SRE.
- **Rollout and implementation costs:** These are the initial costs of professional and implementation services to get the environment up and running.

7. What skills and roles do we need to succeed with our Kubernetes deployment?

Success with containers and Kubernetes requires a variety of core and auxiliary roles, depending on the use cases and maturity of the organization. While the security and platform ops roles are critical across most enterprises, software development use cases certainly create the need for developer, platform, build, release and reliability engineering teams. Table 1 shows the required key skills and roles.

Table 1: Containers and Kubernetes — Key Roles and Responsibilities

(Enlarged table in Appendix)



Team	Role	Responsibilities
Software Development	Developers	■ Coding, application design, implementation and testing ■ Source code management
Platform	Platform engineering and operations	■ Identify platform tools ■ Platform installation, configuration and administration ■ Automate container infrastructure provisioning ■ Maintain base images ■ Integration of DevOps pipeline and automation of Day 2 operations ■ Enable self-service capabilities to developers ■ Capacity planning, workload isolation
Reliability engineering (such as site reliability engineers [SREs])	■ Security, monitoring and performance aspects of platform and applications ■ Application resiliency ■ Debug and document production issues ■ Incident management and response	
Build and release engineering	■ Choose and manage the CI/CD deployment pipeline ■ Develop templates to rapidly build new services and features ■ Educate development teams ■ Create dashboards to quantify process efficiency and developer productivity	

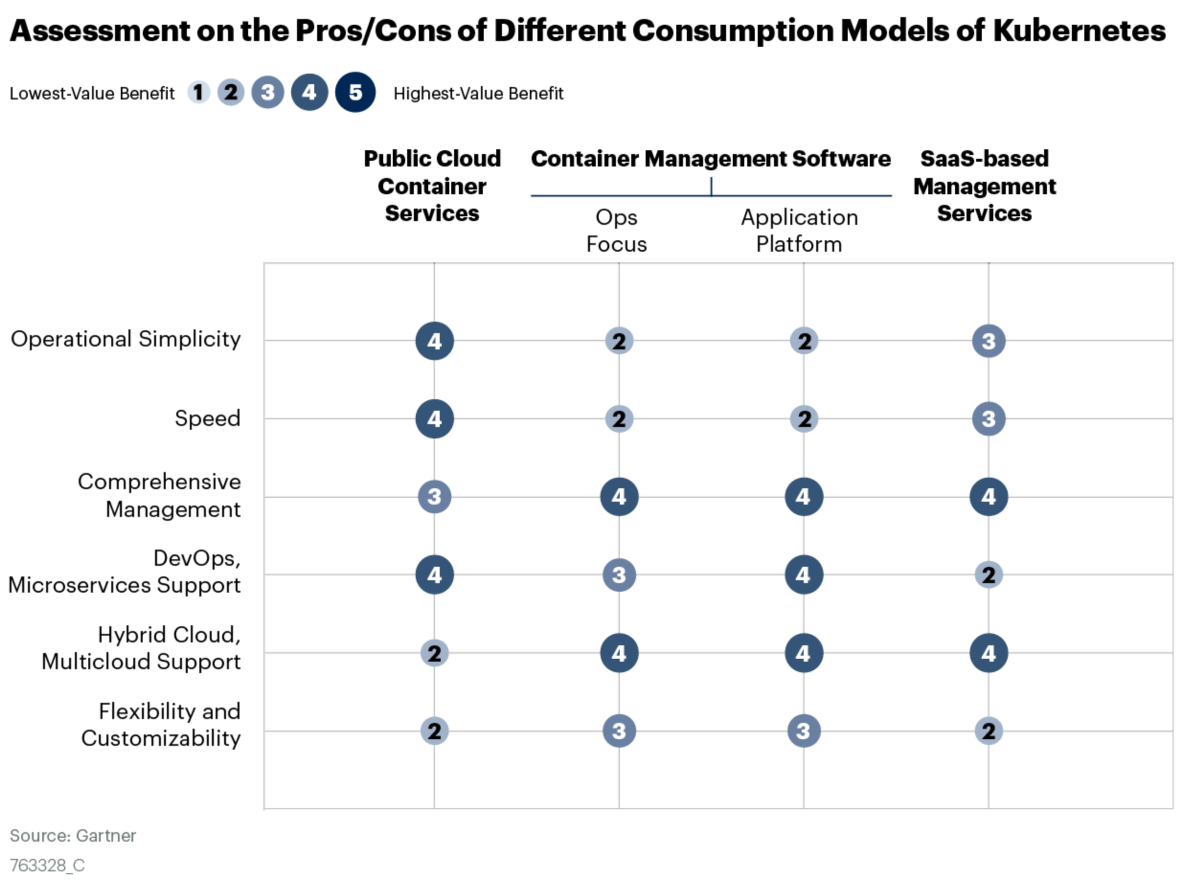
Source: Gartner (May 2022)

8. How do we deploy Kubernetes? What are the pros/cons of various deployment models?

Kubernetes deployment models can be categorized into three patterns (see Figure 1):

- **Public cloud container services**: A deployment model where users consume a managed Kubernetes service offered by a public cloud IaaS provider. Examples are AWS EKS, Azure Kubernetes Service (AKS) and Google Kubernetes Engine (GKE). Users have lesser management overheads; some providers allow their services to run on-premises through distributed cloud products.
- **Container management software**: A deployment model where users operate and manage Kubernetes clusters on-premises and/or off-premises by using a packaged software solution. The software solutions combine a Kubernetes distribution with other management capabilities, such as security, monitoring and storage integration.
    - **Container-operations-focused**: This approach focuses on simplifying container operations management, providing more vendor neutrality in DevOps toolchains and application infrastructure software. Examples include D2IQ Kubernetes Platform, Mirantis Docker Enterprise, HPE Ezmeral Container Platform, Rancher Kubernetes Engine, Red Hat OpenShift Kubernetes Engine, VMware Tanzu for Kubernetes Operations.
    - **Application platform**: This approach provides a more comprehensive DevOps and microservice development experience by including DevOps toolchains and application infrastructure software for adjacent middleware services. Examples are Red Hat OpenShift Container platform and VMware Tanzu Application Platform.
- **SaaS-based management services**: A deployment model where users consume a Kubernetes multicluster management service offered as SaaS. This model can generally cover on-premises and multicloud environments, even for clusters created by public cloud container services. Examples include Giant Swarm, Platform 9 and Rafay Systems.

Figure 1: Pros/Cons of Various Kubernetes Deployment Models



Assessment on the Pros/Cons of Different Consumption Models of Kubernetes

Lowest-Value Benefit 1 2 3 4 5 Highest-Value Benefit

	Public Cloud Container Services	Container Management Software: Ops Focus	Container Management Software: Application Platform	SaaS-based Management Services
Operational Simplicity	4	2	2	3
Speed	4	2	2	3
Comprehensive Management	3	4	4	4
DevOps, Microservices Support	4	3	4	2
Hybrid Cloud, Multicloud Support	2	4	4	4
Flexibility and Customizability	2	3	3	2

Source: Gartner
763328_C

Gartner.

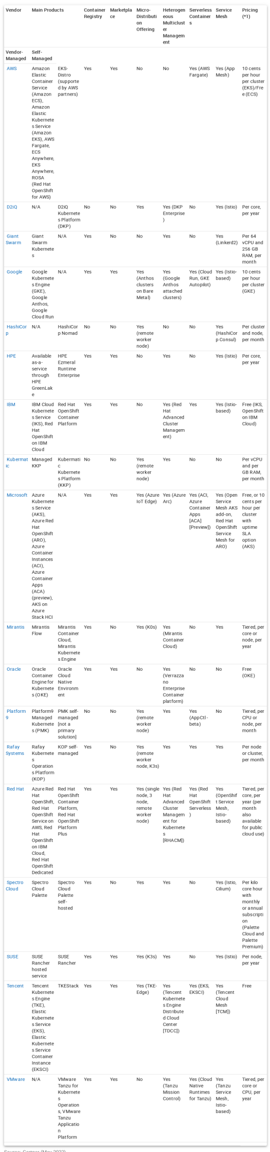
Major advantages of public cloud container services are better operational simplicity and speed. With these models, users don't have to build or manage Kubernetes master services, which simplifies operations and improves time to value. Container management software can make it easier to achieve better consistency in hybrid and multicloud environments. SaaS-based management services can enable better operational simplicity and speed than software. There is also the option of consuming from upstream open source version (not discussed above), which offers better customizability, but Gartner generally doesn't recommend enterprises use it due to its complexity.

9. Who are the key Kubernetes platform vendors and startups?

The container management market consists of different types of market participants. For a detailed overview of various projects and startups, refer to The Innovation Leader's Guide to Navigating the Cloud-Native Container Ecosystem. Table 2 lists the key container management vendors and compares their capabilities:

Table 2: Comparative Overview of Container Management Vendors

(Enlarged table in Appendix)



Vendor	Main Products		Container Registry	Marketplace	Micro-Distribution Offering	Heterogeneous Multicluster Management	Serverless Containers	Service Mesh	Pricing (*1)
	Vendor-Managed	Self-Managed							
AWS	Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), AWS Fargate, ECS Anywhere, EKS Anywhere, ROSA (Red Hat OpenShift for AWS)	EKS-Distro (supported by AWS partners)	Yes	Yes	No	No	Yes (AWS Fargate)	Yes (App Mesh)	10 cents per hour per cluster (EKS)/Free (ECS)
D2iQ	N/A	D2iQ Kubernetes Platform (DKP)	No	No	Yes	Yes (DKP Enterprise)	No	Yes (Istio)	Per core, per year
Giant Swarm	Giant Swarm Kubernetes	N/A	Yes	No	No	Yes	No	Yes (Linkerd2)	Per 64 vCPU and 256 GB RAM, per month
Google	Google Kubernetes Engine (GKE), Google Anthos, Google Cloud Run	N/A	Yes	Yes	Yes (Anthos clusters on Bare Metal)	Yes (Google Anthos attached clusters)	Yes (Cloud Run, GKE Autopilot)	Yes (Istio-based)	10 cents per hour per cluster (GKE)
HashiCorp	N/A	HashiCorp Nomad	No	No	Yes (remote worker node)	No	No	Yes (HashiCorp Consul)	Per cluster and node, per month
HPE	Available as-a-service through HPE GreenLake	HPE Ezmeral Runtime Enterprise	Yes	Yes	No	Yes	No	Yes (Istio)	Per core, per year
IBM	IBM Cloud Kubernetes Service (IKS), Red Hat OpenShift on IBM Cloud	Red Hat OpenShift Container Platform	Yes	Yes	No	Yes (Red Hat Advanced Cluster Management)	Yes	Yes (Istio-based)	Free (IKS, OpenShift on IBM Cloud)
Kubermatic	Managed KKP	Kubermatic Kubernetes Platform (KKP)	No	No	Yes (remote worker node)	Yes	No	No	Per vCPU and per GB RAM, per month
Microsoft	Azure Kubernetes Service (AKS), Azure Red Hat OpenShift (ARO), Azure Container Instances (ACI), Azure Container Apps (ACA) (preview), AKS on Azure Stack HCI	N/A	Yes	Yes	Yes (Azure IoT Edge)	Yes (Azure Arc)	Yes (ACI, Azure Container Apps [ACA] [Preview])	Yes (Open Service Mesh AKS add-on, Red Hat OpenShift Service Mesh for ARO)	Free, or 10 cents per hour per cluster with uptime SLA option (AKS)
Mirantis	Mirantis Flow	Mirantis Container Cloud, Mirantis Kubernetes Engine	Yes	No	Yes (K0s)	Yes (Mirantis Container Cloud)	No	Yes	Tiered, per core or node, per year
Oracle	Oracle Container Engine for Kubernetes (OKE)	Oracle Cloud Native Environment	Yes	Yes	No	Yes (Verrazzano Enterprise Container platform)	No	No	Free (OKE)
Platform9	Platform9 Managed Kubernetes (PMK)	PMK self-managed [not a primary solution]	No	No	Yes (remote worker node)	Yes	Yes (AppCtl-beta)	No	Tiered, per CPU or node, per month
Rafay Systems	Rafay Kubernetes Operations Platform (KOP)	KOP self-managed	Yes	No	Yes (remote worker node, K3s)	Yes	Yes	Yes	Per node or cluster, per month
Red Hat	Azure Red Hat OpenShift, Red Hat OpenShift Service on AWS, Red Hat OpenShift on IBM Cloud, Red Hat OpenShift Dedicated	Red Hat OpenShift Container Platform, Red Hat OpenShift Platform Plus	Yes	Yes	Yes (single node, 3 node, remote worker node)	Yes (Red Hat Advanced Cluster Management for Kubernetes [RHACM])	Yes (Red Hat OpenShift Serverless)	Yes (OpenShift Service Mesh, Istio-based)	Tiered, per core, per year (per month also available for public cloud use)
Spectro Cloud	Spectro Cloud Palette	Spectro Cloud Palette self-hosted	Yes	No	Yes	Yes	No	Yes (Istio, Cilium)	Per kilo core hour with monthly or annual subscription (Palette Cloud and Palette Premium)
SUSE	SUSE Rancher hosted service	SUSE Rancher	Yes	Yes	Yes (K3s)	Yes	No	Yes (Istio)	Per node, per year
Tencent	Tencent Kubernetes Engine (TKE), Elastic Kubernetes Service (EKS), Elastic Kubernetes Service Container Instance (EKSCI)	TKEStack	Yes	Yes	Yes (TKE-Edge)	Yes (Tencent Kubernetes Engine Distributed Cloud Center [TDCC])	Yes (EKS, EKSCI)	Yes (Tencent Cloud Mesh [TCM])	Free
VMware	N/A	VMware Tanzu for Kubernetes Operations, VMware Tanzu Application Platform	Yes	Yes	No	Yes (Tanzu Mission Control)	Yes (Cloud Native Runtimes for Tanzu)	Yes (Tanzu Service Mesh, Istio-based)	Tiered, per core or CPU, per year

Source: Gartner (May 2022)

10. What are the emerging trends around containers & Kubernetes?

- **VM Convergence:** As customers deploy containerized applications alongside VMs, the need for a common management plane arises. There are several manifestations of this convergence, such as projects like Kubevirt that aim to manage VMs through Kubernetes, vSphere pod service that runs containers directly on VMs and managed by Kubernetes and open source projects such as Firecracker and Kata containers, which strive to bring MicroVMs and secure container runtimes, respectively.

- **Support for Stateful applications:** More enterprises will be running stateful workloads in containers, with Kubernetes providing stateful support via Persistent Volumes (PV) and APIs, such as StatefulSet and CSI (Container storage interface). For a detailed list of vendors enabling stateful support with data storage solutions, refer to How Do I Approach Storage Selection and Implementation for Containers and Kubernetes Deployments?.

- **Edge Computing:** The lightweight nature of containers, combined with its modular architecture, makes it appealing for edge environments. Vendors have launched distributed cloud offerings such as AWS Outposts, Azure Stack HCI, Google Anthos and Rancher K3S to cater to the needs of edge environments. Bare-metal deployments are also becoming more common in edge deployments for cost and performance reasons.

- **Serverless Convergence:** Cloud providers are increasingly delivering capabilities, where containers can be run in a serverless operating model — further abstracting and automating runtime capabilities. Examples include AWS Fargate, Microsoft Azure container instances, Google Cloud Run and IBM Cloud Code Engine. Due to growing demand from developers, Serverless function-as-a-service platforms are supporting container images, such as AWS Lambda support for container images.

- **Application workflow automation:** Operators are software extensions to Kubernetes to manage applications and their components. The Operator pattern aims to automate applications by capturing the tasks of a human operator who manages a service or set of services. In addition, newer open source projects like Kubeflow are striving to enable tighter affinity between Kubernetes and orchestrating specific application workflows, such as machine learning in case of Kubeflow. Machine learning will be one of the fastest growing workload use cases for Kubernetes in the future.

Evidence

The co-authors have spoken to most of the vendors listed in this document via briefings and inquiries. In addition, the authors have handled more than 500 client inquiries on this topic over the past 12 months.

[1] 10 Trends in Real-World Container Use, Datadog.

Document Revision History

CTO's Guide to Containers and Kubernetes — Answering the Top 10 FAQs - 27 July 2020

Recommended by the Authors

Some documents may not be available as part of your current Gartner subscription.

The Innovation Leader’s Guide to Navigating the Cloud-Native Container Ecosystem

Market Guide for Container Management

How Do I Approach Storage Selection and Implementation for Containers and Kubernetes Deployments?

Server Virtualization Isn’t Dead — But It Is Evolving

Quick Answer: What Are the Deployment Options If I Containerize My ISV Application?

Table 1: Containers and Kubernetes — Key Roles and Responsibilities

Team	Role	Responsibilities
Software Development	Developers	■ Coding, application design, implementation and testing ■ Source code management
Platform	Platform engineering and operations	■ Identify platform tools ■ Platform installation, configuration and administration ■ Automate container infrastructure provisioning ■ Maintain base images ■ Integration of DevOps pipeline and automation of Day 2 operations ■ Enable self-service capabilities to developers ■ Capacity planning, workload isolation

Team ↓	*Role* ↓	*Responsibilities* ↓
Reliability engineering (such as site reliability engineers [SREs])	▪ Security, monitoring and performance aspects of platform and applications ▪ Application resiliency ▪ Debug and document production issues ▪ Incident management and response	
Build and release engineering	▪ Choose and manage the CI/CD deployment pipeline ▪ Develop templates to rapidly build new services and features ▪ Educate development teams ▪ Create dashboards to quantify process efficiency and developer productivity	

Source: Gartner (May 2022)

Table 2: Comparative Overview of Container Management Vendors

Vendor	Main Products		Container Registry	Marketplace	Micro-Distribution Offering	Heterogeneous Multicluster Management	Serverless Containers	Service Mesh	Pricing (*1)
Vendor-Managed	**Self-Managed**								
AWS	Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), AWS Fargate, ECS Anywhere, EKS Anywhere, ROSA (Red Hat	EKS-Distro (supported by AWS partners)	Yes	Yes	No	No	Yes (AWS Fargate)	Yes (App Mesh)	10 cents per hour per cluster (EKS)/Free (ECS)

	OpenShift for AWS)								
D2iQ	N/A	D2iQ Kubernetes Platform (DKP)	No	No	Yes	Yes (DKP Enterprise)	No	Yes (Istio)	Per core, per year
Giant Swarm	Giant Swarm Kubernetes	N/A	Yes	No	No	Yes	No	Yes (Linkerd2)	Per 64 vCPU and 256 GB RAM, per month
Google	Google Kubernetes Engine (GKE), Google Anthos, Google Cloud Run	N/A	Yes	Yes	Yes (Anthos clusters on Bare Metal)	Yes (Google Anthos attached clusters)	Yes (Cloud Run, GKE Autopilot)	Yes (Istio-based)	10 cents per hour per cluster (GKE)
HashiCorp	N/A	HashiCorp Nomad	No	No	Yes (remote worker node)	No	No	Yes (HashiCorp Consul)	Per cluster and node, per month
HPE	Available as-a-service through HPE GreenLake	HPE Ezmeral Runtime Enterprise	Yes	Yes	No	Yes	No	Yes (Istio)	Per core, per year
IBM	IBM Cloud	Red Hat	Yes	Yes	No	Yes (Red Hat	Yes	Yes (Istio-	Free (IKS,

	Kubernetes Service (IKS), Red Hat OpenShift on IBM Cloud	OpenShift Container Platform				Advanced Cluster Management)		based)	OpenShift on IBM Cloud)
Kubermatic	Managed KKP	Kubermatic Kubernetes Platform (KKP)	No	No	Yes (remote worker node)	Yes	No	No	Per vCPU and per GB RAM, per month
Microsoft	Azure Kubernetes Service (AKS), Azure Red Hat OpenShift (ARO), Azure Container Instances (ACI), Azure Container Apps (ACA) (preview), AKS on Azure Stack HCI	N/A	Yes	Yes	Yes (Azure IoT Edge)	Yes (Azure Arc)	Yes (ACI, Azure Container Apps [ACA] [Preview])	Yes (Open Service Mesh AKS add-on, Red Hat OpenShift Service Mesh for ARO)	Free, or 10 cents per hour per cluster with uptime SLA option (AKS)
Mirantis	Mirantis Flow	Mirantis Container	Yes	No	Yes (K0s)	Yes (Mirantis Container	No	Yes	Tiered, per core or node,

		Cloud, Mirantis Kubernetes Engine				Cloud)			per year
Oracle	Oracle Container Engine for Kubernetes (OKE)	Oracle Cloud Native Environment	Yes	Yes	No	Yes (Verrazzano Enterprise Container platform)	No	No	Free (OKE)
Platform9	Platform9 Managed Kubernetes (PMK)	PMK self-managed [not a primary solution]	No	No	Yes (remote worker node)	Yes	Yes (AppCtl - beta)	No	Tiered, per CPU or node, per month
Rafay Systems	Rafay Kubernetes Operations Platform (KOP)	KOP self-managed	Yes	No	Yes (remote worker node, K3s)	Yes	Yes	Yes	Per node or cluster, per month
Red Hat	Azure Red Hat OpenShift, Red Hat OpenShift Service on AWS, Red Hat OpenShift on	Red Hat OpenShift Container Platform, Red Hat OpenShift Platform Plus	Yes	Yes	Yes (single node, 3 node, remote worker node)	Yes (Red Hat Advanced Cluster Management for Kubernetes [RHACM])	Yes (Red Hat OpenShift Serverless)	Yes (OpenShift Service Mesh, Istio-based)	Tiered, per core, per year (per month also available for public cloud use)

	IBM Cloud, Red Hat OpenShift Dedicated								
Spectro Cloud	Spectro Cloud Palette	Spectro Cloud Palette self-hosted	Yes	No	Yes	Yes	No	Yes (Istio, Cilium)	Per kilo core hour with monthly or annual subscription (Palette Cloud and Palette Premium)
SUSE	SUSE Rancher hosted service	SUSE Rancher	Yes	Yes	Yes (K3s)	Yes	No	Yes (Istio)	Per node, per year
Tencent	Tencent Kubernetes Engine (TKE), Elastic Kubernetes Service (EKS), Elastic Kubernetes Service	TKEStack	Yes	Yes	Yes (TKE-Edge)	Yes (Tencent Kubernetes Engine Distributed Cloud Center [TDCC])	Yes (EKS, EKSCI)	Yes (Tencent Cloud Mesh [TCM])	Free

	Container Instance (EKSCI)								
VMware	N/A	VMware Tanzu for Kubernetes Operations, VMware Tanzu Application Platform	Yes	Yes	No	Yes (Tanzu Mission Control)	Yes (Cloud Native Runtimes for Tanzu)	Yes (Tanzu Service Mesh, Istio-based)	Tiered, per core or CPU, per year

Source: Gartner (May 2022)