Best Practices for Running Containers and Kubernetes in Production

Published 4 August 2020 - ID G00730344 - 12 min read

By Analysts Arun Chandrasekaran

The container ecosystem is immature and lacks operational best practices; however, the adoption of containers and Kubernetes is increasing for legacy modernization and cloud-native applications. Infrastructure and operations leaders must accelerate container deployment in production environments.

Overview

Key Challenges

- Most organizations face challenges in determining the right time to scale pilot projects into production deployments, given the steep learning curve, lack of a DevOps culture and an unclear ROI.
- Container usage for production deployments in enterprises is still constrained by concerns around operational complexity in the areas of security, monitoring, data management and networking.
- Cloud-native applications require a high degree of infrastructure automation and specialized operations skills, which are not commonly found in enterprise IT organizations.
- Identifying the right operational model for Kubernetes deployments is hard, due to unclear responsibility and accountability matrices across developers, infrastructure and operations and security teams.

Recommendations

I&O leaders responsible for the data center should:

- Determine whether they have a clear ROI and robust workload assessment model for Kubernetes that is supported by a strong DevOps culture to ensure seamless move to production.
- Select a Kubernetes platform that aligns with their application architecture and multienvironment deployment goals.

- Develop a Kubernetes strategy that applies best practices across security, governance monitoring, storage, networking, container life cycle management and platform selection.
- Integrate the Kubernetes platform with continuous integration/continuous delivery, security and operational tools, and if needed, augment it with best-of-breed tooling that enables I&O to meet business SLAs and simplify the developer workflow.
- Create a platform ops team that works with application developers for platform selection and operations and focuses on continuous improvement to meet the required SLAs of production applications.

Strategic Planning Assumptions

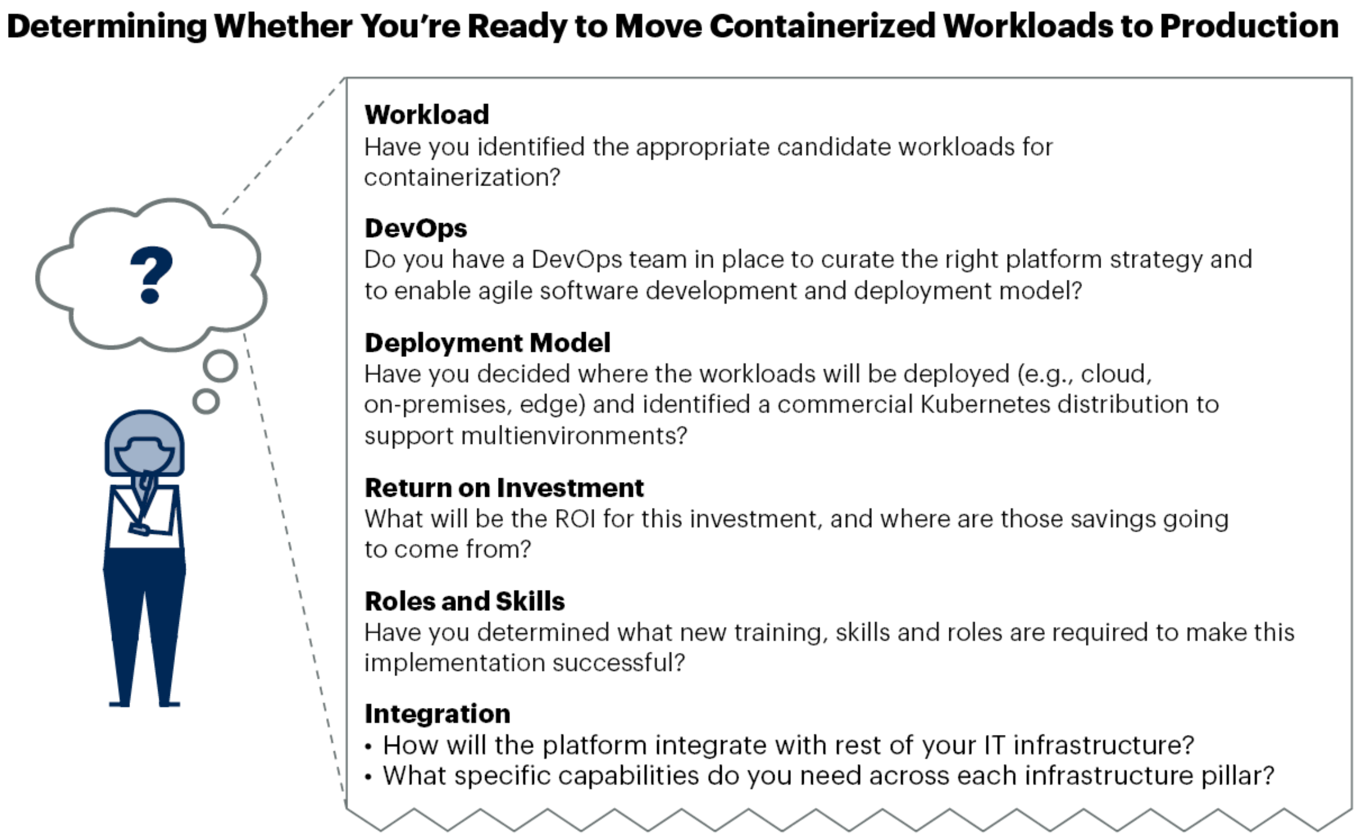
By 2025, more than 50% of enterprises will adopt a centralized platform engineering and operations approach to facilitate DevOps self-service and scaling, which is a significant increase from less than 20% in 2020.

By 2025, more than 85% of global organizations will be running containerized applications in production, which is a significant increase from fewer than 35% in 2019.

Introduction

Although there is growing interest in and rapid adoption of containers, running them in production requires a steep learning curve, due to technological immaturity and a lack of operational know-how. Hence, organizations should take a realistic view of the business requirements that demand containerization of production workloads. IT leaders should determine whether they have the right skill sets to forge ahead, given the steep learning curve. The questions shown in Figure 1 are a good starting point to determine whether IT leaders are ready to deploy containers in production.

Figure 1: Determining Whether You're Ready to Move Containerized Workloads to Production



Determining Whether You're Ready to Move Containerized Workloads to Production

Workload
Have you identified the appropriate candidate workloads for containerization?

DevOps
Do you have a DevOps team in place to curate the right platform strategy and to enable agile software development and deployment model?

Deployment Model
Have you decided where the workloads will be deployed (e.g., cloud, on-premises, edge) and identified a commercial Kubernetes distribution to support multienvironments?

Return on Investment
What will be the ROI for this investment, and where are those savings going to come from?

Roles and Skills
Have you determined what new training, skills and roles are required to make this implementation successful?

Integration
- How will the platform integrate with rest of your IT infrastructure?
- What specific capabilities do you need across each infrastructure pillar?

Source: Gartner (2020)
730344_C

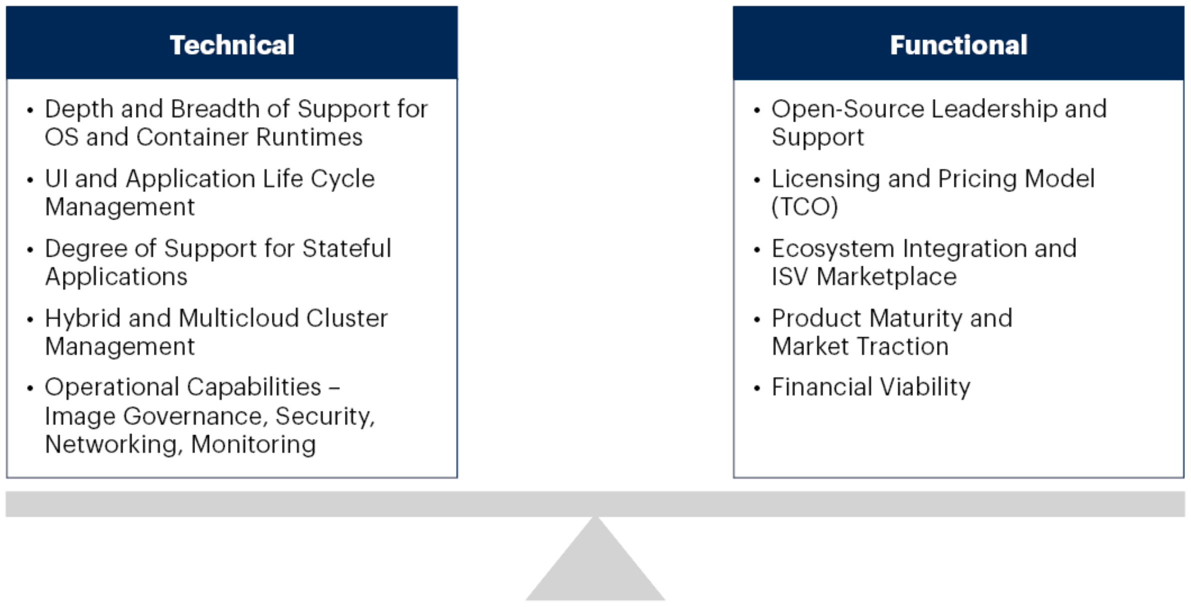
Analysis

Select a Kubernetes Platform That Aligns With Your Application Architecture and Deployment Goals

Kubernetes has emerged as the de facto standard for container orchestration, with a vibrant community and support from most of the leading commercial vendors. Customers deciding between various Kubernetes distributions need to identify the right product by comparing them across the technical and functional factors shown in Figure 2.

Figure 2: Kubernetes Platform Selection Factors

Kubernetes Platform Selection Factors


Technical
• Depth and Breadth of Support for OS and Container Runtimes
• UI and Application Life Cycle Management
• Degree of Support for Stateful Applications
• Hybrid and Multicloud Cluster Management
• Operational Capabilities – Image Governance, Security, Networking, Monitoring
Functional
• Open-Source Leadership and Support
• Licensing and Pricing Model (TCO)
• Ecosystem Integration and ISV Marketplace
• Product Maturity and Market Traction
• Financial Viability

Source: Gartner (2020)
730344_C

Kubernetes deployment models can be categorized into the three patterns:·

- **Do It Yourself (DIY) With Upstream Version —** A deployment model in which users build and manage Kubernetes clusters by using the open-source Kubernetes upstream version from GitHub repositories. This has been a common deployment model among early adopters that prefer customizability. However, it is complex to build and operate for most enterprises.
- **Public Cloud Container Services —** A deployment model in which users consume a managed Kubernetes service offered by a public cloud infrastructure as a service (IaaS) provider. Examples are Amazon Elastic Kubernetes Service (EKS), Microsoft Azure Kubernetes Service (AKS), Google Kubernetes Engine (GKE), IBM Cloud Kubernetes Service and Alibaba container service for Kubernetes. Some public cloud container services can run on-premises through distributed cloud products (e.g., Amazon Web Services (AWS) Outposts, Google Anthos and AKS engine on Azure Stack). Cloud services offer operational simplicity and faster time to production.
- **Container Management Software —** A deployment model in which users build 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/networking integration. The software vendors can be categorized as:

- **Operations-Focused —** This approach focuses on simplifying container operations management, providing more vendor neutrality in DevOps toolchains and application infrastructure software. Examples include Mirantis Docker Kubernetes Service, Rancher Kubernetes Engine, and VMware Tanzu Kubernetes Grid. In addition, managed service providers (MSPs), such as Platform9 and Giant Swarm, provide Kubernetes as a managed service across on-premises, edge and multicloud environments, with the potential for simplified operations.

- **Developer-Focused —** This approach provides a more comprehensive DevOps and microservices 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 Service (TAS for Kubernetes is currently in beta).

Recommendations:

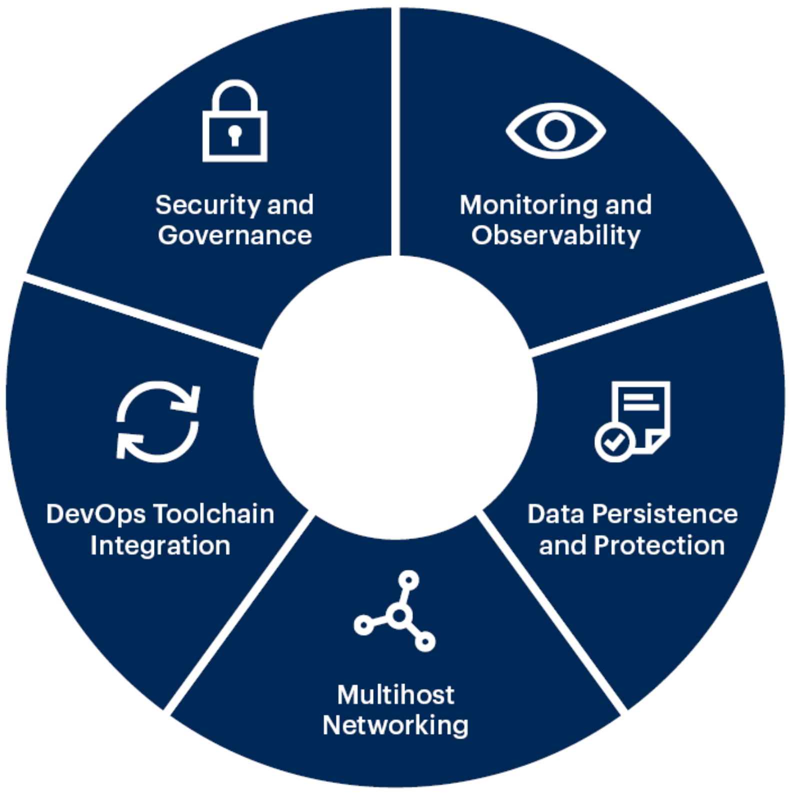
- Capture your container orchestration requirements across the factors mentioned above and choose the tool that best matches your requirements and use cases.
- Use Gartner research (see "CTO's Guide to Containers & Kubernetes — Answering the Top 10 Frequently Asked Questions [FAQs]") to understand the trade-offs between different Kubernetes deployment models, and choose the one that is best-suited to your environment.
- Prioritize vendors that can provide hybrid orchestration for container workloads across multiple environments, with tight integration to back ends, federated management planes and consistent pricing models.

Use Gartner's Best Practices to Operationalize Containers and Kubernetes

Organizations that are deploying containers in production need to create a strategy to operationalize Kubernetes that encompasses the elements shown in Figure 3.

Figure 3: Kubernetes Operational Elements

Kubernetes Operational Elements


Security and Governance
Monitoring and Observability
Data Persistence and Protection
Multihost Networking
DevOps Toolchain Integration

Source: Gartner (2020)
730344_C

Security and Governance Best Practices

Security can't be an afterthought. It needs to be embedded in the DevOps process, which Gartner refers to as "DevSecOps" (see "12 Things to Get Right for Successful DevSecOps"). Organizations need to plan for securing the containerized environment across the entire life cycle, which includes the build, deploy and run phases of an application.

Recommendations:

- Integrate an image-scanning and signing process to prevent vulnerabilities as part of an enterprise's continuous integration/continuous delivery (CI/CD) process. Here applications are scanned during the build and run phases of the software development life cycle. Emphasize the scanning and identification of open-source components, libraries and frameworks.
- Harden the configurations by using Center for Internet Security (CIS) benchmarks, which are available for Kubernetes, to enable a strong security posture.

- Set up mandatory access controls, ensure separation of duties and institute a secrets management policy. Sensitive information, such as Secure Sockets Layer (SSL) keys or database credentials, should be encrypted by the orchestrator or a third-party management service, and be provisioned at runtime.
- Deploy security products that provide whitelisting, behavioral monitoring and anomaly detection to prevent malicious activity.
- "Shift left" by implementing strong version control for code and components and by training the developers, QA teams on secure coding practices.

Monitoring and Observability Best Practices

Developers are more focused on the functional aspects of these application containers than on the operational requirements of monitoring them. Historically, monitoring tools have focused on host-level metrics, such as CPU utilization, memory utilization, input output (I/O) per second, latency and network bandwidth. Although these metrics are still important for operations teams, by themselves, they won't paint a full picture, because they lack granular detail at the container or service level. Developers should instead focus on instrumenting their applications to enable observability. This will involve deploying tools for additional logging, metrics collections and distributed tracing.

Recommendations:

- Focus on monitoring at a container granularity and across containers at a service level, so that you are monitoring "apps," rather than just physical hosts.
- Prioritize tools and vendors that offer deep integration with the Kubernetes distribution vendor that you have chosen.
- Favor tools that have automated service discovery, perform rich application monitoring, do distributed tracing, integrate with Prometheus and can provide action-oriented recommendations in real time, using analytics and/or machine learning.

Storage Best Practices

As container adoption for stateful workloads grows, customers need to consider data persistence beyond the host, as well as the need to protect that data. If your primary use case is "lift and shift" of legacy applications or stateless use cases, there may be little change in storage needs. However, if the application will be refactored significantly, or if this will be a new, microservices-oriented, stateful application, then infrastructure and operations (I&O) leaders need a storage platform that can maximize the availability, agility and performance of that workload.

Recommendations:

- Select storage solutions that are aligned with microservices architecture principles, adhere to the requirements of container-native data services, exhibit hardware-agnosticism, are API-driven, have distributed architectures, and support on-premises, edge and public cloud deployments.
- Thoroughly vet the performance and stability of the storage product and ensure that it integrates with your Kubernetes distribution and supports container storage interfaces (CSI).

Networking Best Practices

Agility and portability are the developers' top concerns, and they expect their applications to be portable across the software development life cycle. Portability must be enabled from a developer's laptop to software testing to production. This can span on-premises and public cloud infrastructures. The traditional enterprise network model, in which IT creates network environments for development, testing, quality assurance and production for every project, is not necessarily well-aligned with this workflow. Moreover, container networking spans several layers, including the physical switching and routing infrastructure to intrahost and interhost networking through container runtime overlays. Networking solutions need to be tightly integrated with Kubernetes' primitives and policy engine. IT leaders should strive for a high degree of network automation and provide developers with proper tools and sufficient flexibility.

Recommendations:

- Determine whether your Kubernetes distribution or software-defined networking (SDN) solution supports Kubernetes networking. If this is absent or insufficient, then choose a container networking interface (CNI) integrated network overlay and policy enforcement engine.
- Ensure that the Kubernetes distribution being selected provides ingress controller support for load balancing across hosts in the cluster. If that is lacking, then consider third-party proxies or service mesh technologies.
- Train your network engineers on Linux networking and network automation tools to bridge the skills gap and enhance agility.

Integrate Kubernetes With the DevOps Toolchain

For a highly automated and seamless application delivery pipeline, organizations need to complement container orchestration with other automation tools, such as code repositories, CI/CD pipeline, and infrastructure-as-code (IaC) products. The I&O organization should deploy infrastructure automation tools to ensure that infrastructure provisioning and management is automated and streamlined to account for the dynamic nature of containerized workloads. Containers also present the potential for sprawl similar to what existed with virtual machine (VM) deployments; thus, I&O must have tooling and processes for life cycle management of containers.

Containers enable development teams to define the runtime environment of applications, thus reducing unpredictability between production and preproduction environments. Development teams can build and test applications in productionlike environments by integrating DevOps toolchains with Kubernetes-based container orchestration tools. This consistency between different environments improves reliability in production and improves collaboration between development and operations teams.

Recommendations:

- Automate I&O tasks around infrastructure provisioning and management using infrastructure automation tools.
- Use container-aware configuration management systems to manage the life cycle of container images, which are often constructed and layered on top of existing base images in private or public repositories.
- Integrate the Kubernetes platform (see Note 1) with CI/CD tools, particularly if you have already invested in them, to automate container image building, testing and deployment to production.

Create a Platform Ops Team to Scale Your DevOps

The traditional operating model in which the development team writes code and the quality assurance (QA) team tests software applications and hands them over to the operations organization for day-to-day management is a recipe for failure in a cloud-native world. Not only is the software development and release velocity high, but the platform itself needs to be treated as a product, because it’s dynamic and constantly evolving in terms of capabilities and scale. Rather than having a siloed IT operations team, organizations need to create a collaborative “platform ops” team, which consists of team members with both software engineering and operations skills. The aim of this team should be to foster operational best practices and create standardized platforms that are automated, scalable and resilient. The key responsibilities of I&O as part of a platform ops team should include:

- Automating infrastructure provisioning and management
- Installation and management of the Kubernetes software distribution
- Capacity management
- Workload isolation and ensuring high platform availability
- Establishing image standards and governance
- Monitoring and managing application behavior (often with site reliability engineering)

Recommendations:

- Create a cross-functional platform ops team that includes DevOps and I&O
- Provide the team with training, and empower it to choose a centralized Kubernetes platform and make decisions on platform operations.
- Mandate regular information sharing by the platform ops team with developers on the causes of failures and operational best practices that can enable DevOps scaling.

Evidence

This research is based on more than 400 client interactions on this topic handled by the author during the past 12 months.

Note 1: Kubernetes Platform Vendors

Several types of CaaS and PaaS products can run on-premises and in public IaaS:

CaaS Vendors (IT-Ops-Focused):

- Canonical MicroK8S
- Cisco Container Platform
- D2IQ Konvoy and Mesosphere Kubernetes Engine (MKE)
- Diamanti Spektra
- Hewlett Packard Enterprise (HPE) Container Platform
- Mirantis Docker Enterprise
- Rancher Kubernetes Engine
- Red Hat OpenShift Kubernetes Engine
- Robin Hyperconverged Kubernetes
- SUSE CaaS Platform
- VMware Tanzu Kubernetes Grid

CaaS Delivered as SaaS:

- Platform9

- Giant Swarm
- Rafay Systems

PaaS Vendors:

- Alauda Container Platform
- Red Hat OpenShift Container Platform
- VMware Tanzu Application Services (previously Pivotal Cloud Foundry)

In addition to the Kubernetes platform vendors, there are several established vendors and startups that provide products to enable production ready deployments across various technology areas. Some of these container infrastructure ecosystem vendors are shown in Table 1.

Table 1: Container Infrastructure Ecosystem Vendors

Technology	***Sample List of Vendors***
Infrastructure Automation	Red Hat (Ansible), Chef, Hashicorp (Terraform), Pulumi, Puppet
Monitoring	Datadog, Dynatrace, Instana, New Relic, Sensu, Splunk, Sysdig, Turbonomic
Networking	Cisco, Juniper Networks, Tigera, VMware
Security	Alcide, Anchore, Aqua Security, Qualys, Palo Alto Networks, NeuVector, Portshift, StackRox, Snyk, Styra, Sysdig, Tigera
Service Mesh	Aspen Mesh, AWS (App Mesh), Buoyant (Linkerd), Decipher Technology Studios, Hashicorp (Consul), Tetrate.io, VMware (NSX Service Mesh)
Storage	Arrikto, Diamanti, Kasten, NetApp, Portworx, Robin.io, StorageOS

Source: Gartner (August 2020)