Best Practices for Running Containers and Kubernetes in Production

Published 25 February 2019 - ID G00385131 - 13 min read

By Analysts Arun Chandrasekaran

The container ecosystem is immature and lacks operational best practices, but adoption of containers and Kubernetes is increasing for legacy modernization and cloud-native applications. We outline best practices for I&O leaders to enable and expedite container deployment in production environments.

Overview

Key Challenges

- Container usage for production deployments in enterprises is still constrained by concerns regarding 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, creating and empowering the right team is challenging, due to legacy mindsets and operational burdens on team members.
- There is a hodgepodge of competing vendor interests, uncertain monetization strategies and varied consumption models. This makes vendor selection more difficult, poses serious questions about the commitment of large vendors and affects the viability of startups in the container space.

Recommendations

Infrastructure and operations leaders responsible for the data center should:

- Create a container platform strategy that applies best practices across security, governance monitoring, storage, networking, container life cycle management and container orchestration.
- Start with small, simple use cases; ensure that containers are stateless and immutable; and enforce standardization, automation and federation of clusters for easier management and rapid

- Integrate container as a service or platform as a service platforms with continuous integration/continuous delivery, security and operational tools; if needed, then augment it with best-of-breed tooling that enables I&O to meet business SLAs and simplify developer workflow.
- Create a platform ops team that works with application developers for platform selection and operations and is focused on continuous improvement to meet the required business SLAs of production applications.

Strategic Planning Assumption

By 2022, more than 75% of global organizations will be running containerized applications in production, which is a significant increase from fewer than 30% today.

Introduction

Although there's 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 evaluate 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 you're ready to deploy containers in production.

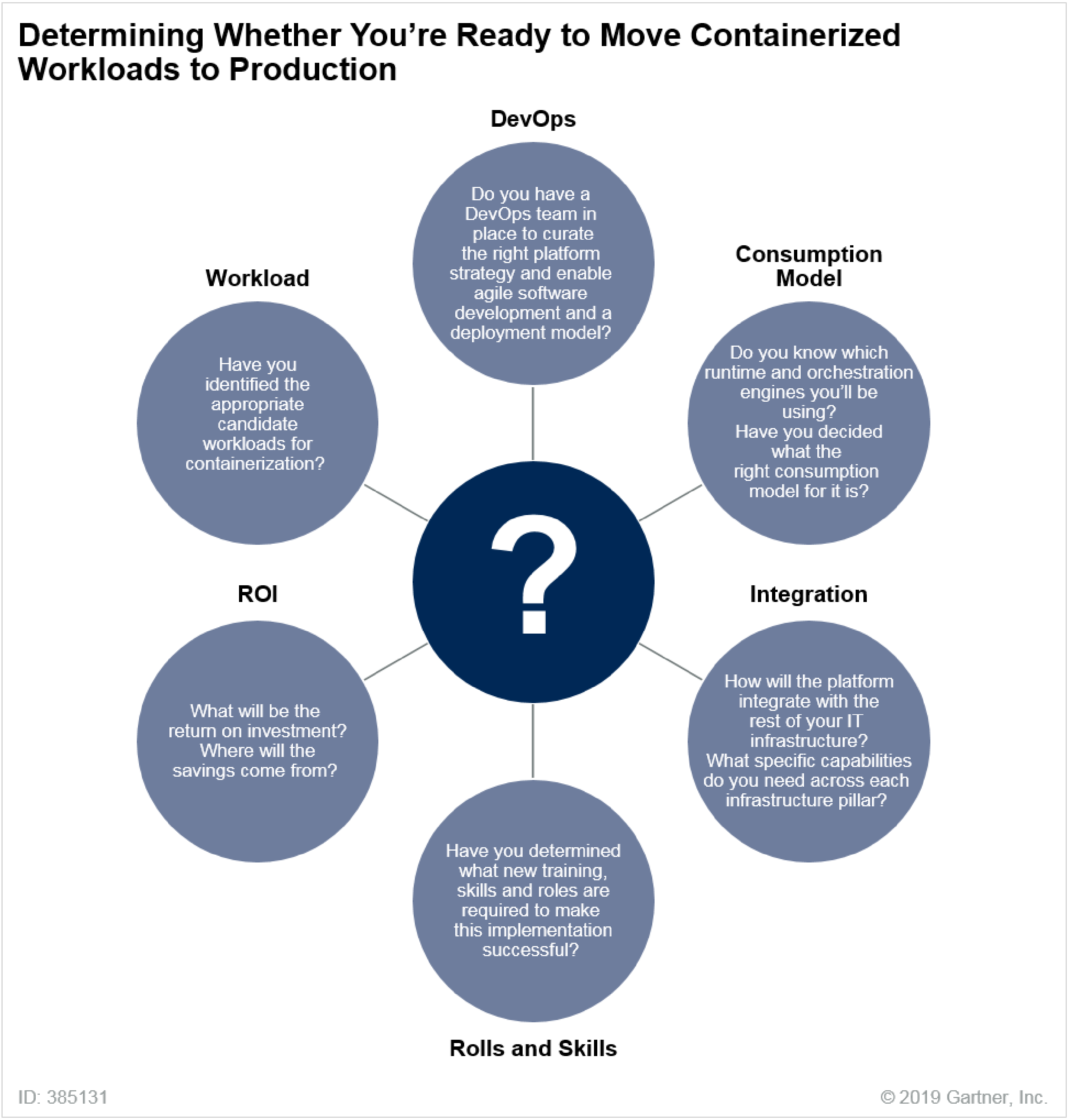
Determining Whether You're Ready to Move Containerized Workloads to Production
DevOps
Do you have a DevOps team in place to curate the right platform strategy and enable agile software development and a deployment model?
Consumption Model
Do you know which runtime and orchestration engines you'll be using? Have you decided what the right consumption model for it is?
Workload
Have you identified the appropriate candidate workloads for containerization?
?
ROI
What will be the return on investment? Where will the savings come from?
Integration
How will the platform integrate with the rest of your IT infrastructure? What specific capabilities do you need across each infrastructure pillar?
Have you determined what new training, skills and roles are required to make this implementation successful?
Rolls and Skills
ID: 385131
© 2019 Gartner, Inc.

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

Source: Gartner (February 2019)

Organizations often underestimate the effort required to operate containers in production. Gartner has observed several common mistakes in our client interactions (see Figure 2).

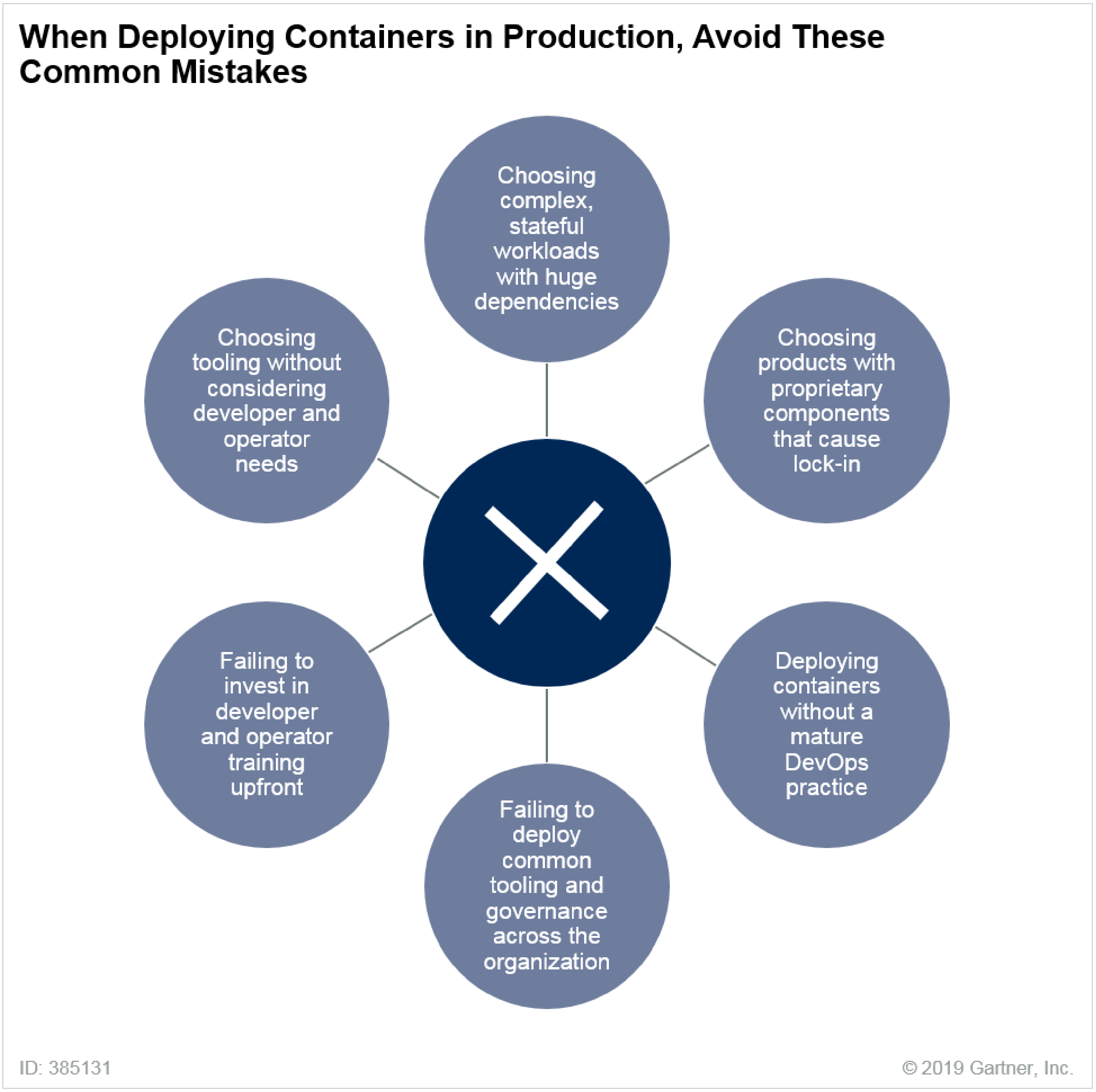
When Deploying Containers in Production, Avoid These Common Mistakes
Choosing complex, stateful workloads with huge dependencies
Choosing products with proprietary components that cause lock-in
Deploying containers without a mature DevOps practice
Failing to deploy common tooling and governance across the organization
Failing to invest in developer and operator training upfront
Choosing tooling without considering developer and operator needs
ID: 385131
© 2019 Gartner, Inc.

Figure 2. When Deploying Containers in Production, Avoid These Common Mistakes

Source: Gartner (February 2019)

Analysis

Use Gartner's Best Practices to Create a Container Platform Strategy

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

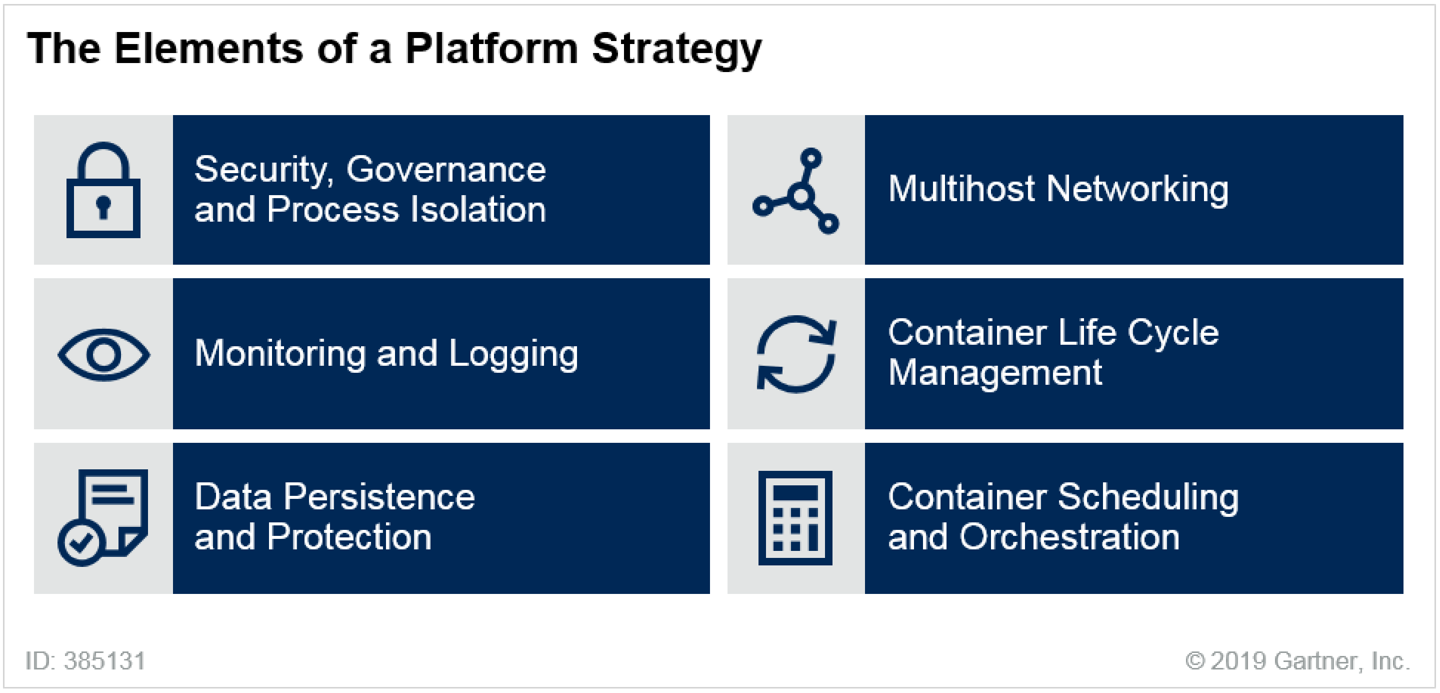
The Elements of a Platform Strategy
Security, Governance and Process Isolation
Multihost Networking
Monitoring and Logging
Container Life Cycle Management
Data Persistence and Protection
Container Scheduling and Orchestration
ID: 385131
© 2019 Gartner, Inc.

Figure 3. The Elements of a Platform Strategy

Source: Gartner (February 2019)

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 "10 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 and development process, deployment and run phase of an application.

Recommendations:

- Integrate an image-scanning process to prevent vulnerabilities as part of an enterprise's continuous integration/continuous delivery (CI/CD) process, where 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. Developer use of older, vulnerable versions is one of the leading causes of container vulnerabilities.
- Harden the configurations by using Center for Internet Security (CIS) benchmarks, which are available for Docker runtime and Kubernetes.
- 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, will be encrypted by the orchestrator or third-party management services, and will be provisioned at runtime.

- Deploy security products that provide whitelisting, behavioral monitoring and anomaly detection for preventing malicious activity.

Monitoring 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.

Recommendations:

- Focus on monitoring at a container and across containers at a service level, so that you are monitoring "apps," rather than physical hosts.
- Prioritize tools and vendors that offer deep integration with container orchestration, particularly Kubernetes.
- Favor tools that have granular logging, supply automated service discovery 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. Choose those that adhere to the requirements of container-native data services, exhibit hardware-agnosticism, are API-driven, have distributed architectures, and support on-premises and public cloud deployments.
- Avoid using proprietary plug-ins and interfaces; instead, prioritize vendors with closer integration with Kubernetes and support standard interfaces, such as container storage interfaces (CSIs).

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:

- Investigate whether your existing container as a service (CaaS) or software-defined networking (SDN) solution supports Kubernetes networking. If that is absent or insufficient, then choose a container networking interface (CNI) integrated network overlay and policy engine.
- Ensure that the CaaS or platform as a service (PaaS) tooling 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.

Container Life Cycle Management Best Practices

For a highly automated and seamless application delivery pipeline, organizations need to complement container orchestration with other automation tools, such as infrastructure-as-code (IaC) products. These include Chef, Puppet, Ansible and Terraform. They also require application release automation tools (see "Magic Quadrant for Application Release Orchestration"). Although there's overlap between these tools and the CaaS products (see Note 1), they are more complementary than competitive today. 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.

Recommendations:

- Establish standards for container base images, taking into account image size, developer flexibility to add components and licensing.

repositories.

- Integrate the CaaS platform with application automation tools, particularly if you have already invested in them, to automate the entire application workflow.

Container Orchestration Best Practices

Key functionality for container deployment is provided at the orchestration and scheduling layers. The orchestration layer interfaces with the application, keeping the containers running in the desired state and maintaining SLAs. Scheduling places the containers on the most optimal hosts in a cluster, as prescribed by the requirements of the orchestration layer. These layers could be provided by different tooling (e.g., Apache Mesos providing the scheduling, while Marathon provides the orchestration) or by a single tool (Kubernetes or Docker Swarm). 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 orchestration engines or across various Kubernetes distributions need to identify the right product by comparing them across the following factors:

- Depth and breadth of support for OS and container runtime
- Runtime stability of the overall product
- Scalability
- Degree of support for stateful applications
- Operational simplicity and quality of vendor support
- Leadership, involvement and support for open source
- Implementation and licensing costs
- Hybrid and multicloud support

Recommendations:

- Define the baseline requirements for security controls, monitoring, policy management, data persistence, networking and life cycle management of containers.
- 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 “How to Choose Your Kubernetes Deployment Model”) to understand

- 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.

Think Cloud First and Hybrid in the Long Run

There has been growing interest in deploying containers in public cloud IaaS, due to the availability of turnkey CaaS offerings, as well as the tight integration that these offerings have with other products offered by cloud providers. Cloud IaaS offers on-demand consumption of resources, rapid scalability and managed services that could obviate infrastructure integration, management and deep know-how. Most cloud providers offer a managed container service, and some offer multiple orchestrators. In addition to native services, on-premises vendors (such as Red Hat and VMware) offer a cloud-based product as a fully hosted service. Third-party managed service providers (MSPs), such as Platform9 and Giant Swarm, offer similar services on-premises and in the cloud. Table 1 shows an overview of key MSPs in the cloud.

Table 1: Managed Container Services in the Cloud

Cloud Provider	Type of Service	Product/Service
Alibaba	Native Cloud Service	Alibaba Cloud Container Service, Alibaba Cloud Container Service for Kubernetes
Amazon Web Services (AWS)	Native Cloud Service	Amazon Elastic Container Services (ECS), Amazon ECS for Kubernetes (EKS), AWS Fargate
Giant Swarm	MSP	Giant Swarm Managed Kubernetes Infrastructure
Google	Native Cloud Service	Google Container Engine (GKE)
IBM	Native Cloud Service	IBM Cloud Kubernetes Service
Microsoft	Native Cloud Service	Azure Kubernetes Service, Azure Service Fabric
Oracle	Native Cloud	OCI Container Engine for Kubernetes

Cloud Provider ↓	Type of Service ↓	Product/Service ↓
Platform9	MSP	Managed Kubernetes
Red Hat	Hosted Service	OpenShift Dedicated & Online
VMware	Hosted Service	Cloud PKS (Beta)

Source: Gartner (February 2019)

Although Docker runtime and managed Kubernetes are becoming ubiquitous across on-premises and public cloud environments, seamless hybrid environments require better federation and service brokering than is currently available. On-premises CaaS vendors, such as Docker, Mesosphere, Rancher Labs, Red Hat and VMware/Pivotal, offer cloud-based services, with varying degrees of integration and support. Public cloud providers have also released capabilities (such as AKS on Azure Stack) or made announcements of availability for on-premises products in 2019 (GKE on-premises and AWS Outposts). Hybrid and multicloud support will be an area of rapid innovation among vendors in 2019 and beyond.

Recommendations:

- Objectively evaluate your organization’s ability to deploy and manage the appropriate tooling, and strongly consider cloud container management services as an alternative.
- Choose the points of lock-in carefully; where possible, implement alternative open-source software.
- Select providers with consistent operating models across hybrid environments that offer single-pane-of-glass management of federated clusters and open service brokers that simplify IaaS self-service.

Create a Platform Ops Team to Scale Your DevOps

The traditional operating model in which the development team writes code and the QA team tests software applications and hands them over to the operations organization for day-to-day management is a recipe for failure in the 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 team members with software engineering and operations skills. The aim of the platform ops team is to foster operational best practices and create standardized platforms that are automated, scalable and resilient.

The ultimate goal of a platform ops team is to provide, as an internal service, cloudlike application infrastructure capabilities to multiple application development teams in a self-service model. These services are often beyond IaaS and include database, middleware and other platform services. The platform ops team's responsibilities include deployment and operations of CaaS/PaaS products, development and operation of custom and standardized middleware layers, automation of IaaS provisioning, deployment, enablement of security, development of observability, and simplification of developer workflow. These skills are rare and require additional training or external hires.

Recommendations:

- Create a cross-functional platform ops team that includes developers and I&O; provide them training, and empower them to 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 200 inquiries on this topic handled by the author. Detailed one-on-one interviews were conducted with more than 20 end-user organizations on their operational tooling and best practices.

Container adoption in production is becoming mainstream, as evidenced by the survey results at the author's session at the Gartner's Infrastructure, Operations & Cloud Strategies (IOCS) event in December 2018 (see Figure 4).

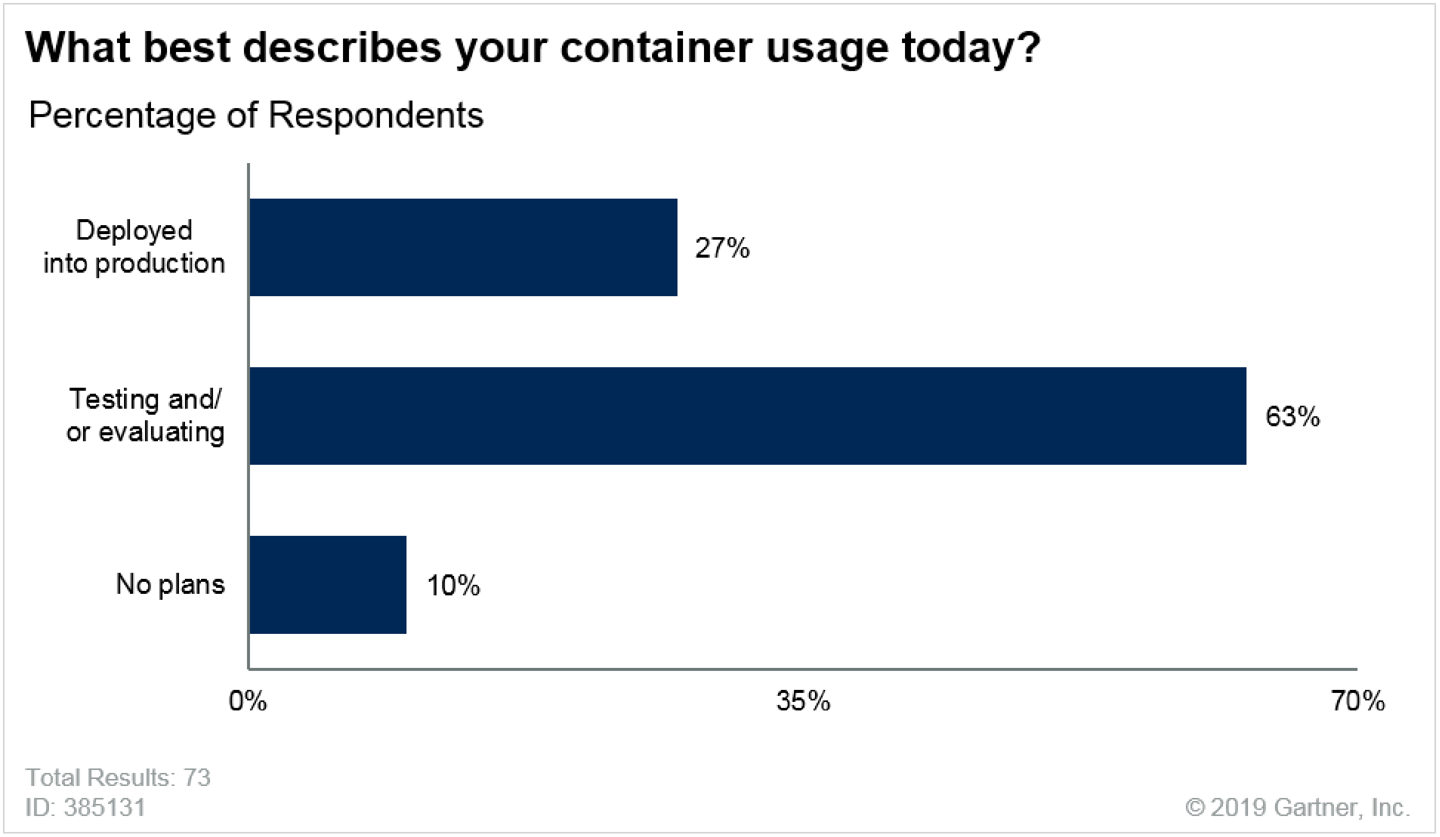

Figure 4. Container Usage

Source: Gartner (February 2019)

Note 1
CaaS Vendors

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

CaaS Vendors

- Docker Enterprise Edition (EE)
- Rancher
- Mesosphere Enterprise DC/OS
- Red Hat OpenShift Container Engine
- VMware/Pivotal — Pivotal Kubernetes Service (PKS)
- IBM Cloud Private
- Suse CaaS

- Platform9
- Giant Swarm
- VMware Cloud PKS (Beta)

PaaS Vendors

- Pivotal Application Service
- Red Hat OpenShift Container Platform

For a list of container infrastructure ecosystem vendors, see Table 2.

Table 2: Container Infrastructure Ecosystem Vendors

Technology ↓	Things to Look For ↓	Sample List of Vendors ↓
Monitoring	Service visualization, proactive alerting, compliance enforcement, auditing	Datadog, Dynatrace, Instana, Sysdig
Networking	Asset discovery, IP management for ephemeral containers, policy enforcement	Cisco, Juniper Networks, Tigera, Weaveworks
Security	OS hardening, secure runtime and orchestration, image security, traffic isolation and lockdown	Aqua Security, NeuVector, StackRox, Twistlock
Service Mesh	Service discovery, load balancing, authentication and access control, quality of service	Aspen Mesh, Avi Networks, AWS (App Mesh), Buoyant (Linkerd), Tetrate.io (Istio in Beta), VMware (NSX Service Mesh)
Storage	Container-native data services, resource coalescing, multiprotocol support	Diamanti, NetApp, Portworx, Robin Systems, StorageOS

Source: Gartner (February 2019)