Ovum
TMT intelligence | informa

Ovum Market Radar: Container Management Platforms 2018/19

Understanding the container management platform market

Publication Date: 09 Nov 2018 | Product code: INT003-000218

Roy Illsley

Summary

Catalyst

The growth in cloud-native (built to execute in the cloud independent of the underlying infrastructure) applications is synonymous with the use of containers. However, containers as a technology introduce some specific challenges for CIOs, not least of which is the lack of skills when it comes to the management of large-scale container environments. It is important to understand from a management perspective what the key challenges are with adopting containers at scale and to discover some of the different vendor approaches to mitigating these challenges. Ovum has focused on Kubernetes and Container Initiative (OCI)-compliant container-based solutions because these have become the de facto standard approach used in the market, with 45% of respondents to the OpenStack user survey 2017 using Kubernetes, and 65% using the Docker or the OCI-compliant container format for OpenStack services. However, these are not the only approaches for managing containers.

Ovum view

The shift from a dominant VM-based world to a more mixed cloud-native and VM-based world is the new reality. Over the next five years, the dominance of VMs will be eroded as more workloads are developed and deployed in container-based environments. While Ovum does not believe that all workloads will become container-based, we expect this to represent a significant proportion of the workloads by 2023. The developer community has been quick to see the potential of containers and has embraced the technology. However, this shift requires an equally big cultural and skills shift to occur within operations and security teams, and this is one of the forces that is holding back the faster adoption of containers.

Key messages

- Lifecycle management of applications using Kubernetes is the most popular starting point for many organizations, although the definition of what is included in a managed service differs between vendors.
- All container management platforms provide 24/7 monitoring and proactive feedback for the attainment of operational assurance.
- A single-pane-of-glass approach across different clouds, availability zones, clusters, containers/applications, and environments differs by vendor, but the direction of travel is toward a full hybrid and cross-cloud capability.
- Support for organizational financial objectives by presenting show that back and chargeback metrics is a key area of differentiation. While chargeback is important, the cost savings that the correct platform can generate will also help with the ROI calculation.

Recommendations

Recommendations for enterprises

The need to change the organizational structure, skills, processes, and culture are all aspects that must be addressed by CIOs as they look to adopt containers at scale. However, the amount of change required depends on the platform and approach selected. The developer community is already advanced in its understanding of the new roles and responsibilities, but the challenge is to translate this to the other teams involved in delivering IT services. The use of a container management platform that can simplify and automate some of the manual activities involved in managing container environments is beginning to transform operational teams' roles and is helping them acquire the skills required to operate containers at scale.

Recommendations for vendors

Organizations looking to adopt containers at scale will also have VMs at scale. This translates to the need for changes to IT teams. These teams may need to manage both environments, although the formation of new teams with the new capabilities is also possible. Ovum also expects container platforms on bare-metal and public cloud environments that are not VM based to gain in popularity. Accepting that the world will be a hybrid mixture, for most organizations anyway, of containers and VMs for the foreseeable future is an important capability requirement to remember. Organizations are struggling to retrain existing IT administrators in the different approaches required when using containers. This is where the vendors with these platforms can help, but only if they recognize that simplification does not mean abdication of responsibility.

Recommendations for service providers

The challenge for many enterprise customers is how to use the same team for the management of cloud-native solutions based on containers and the management of existing legacy and VM-based workloads. The container management platforms offer a solution that many are adopting, but these are application-related compared to VMs that are infrastructure-related, so different skills and disciplines are required. However, the key capabilities looked for by enterprise customers are:

- How to secure and govern the new container-based workloads.
- How to use the platform to retrain existing staff and move from a fully managed starting position to a more hybrid approach.
- To use a standard and open technology to ensure the workloads are portable.
- To integrate the management of containers and VMs in a way that demonstrates how these technologies can work together. This is important not only in terms of technology but also in terms of being under the auspice of the operations team. This means decisions on workload target platforms can be evaluated based on a holistic knowledge of all workloads.

Defining and exploring container management platforms

Definition and characteristics

Operational management of a container-based environment is fundamentally different to that currently deployed for a VM environment and is more analogous to a DevOps approach. This means that not only do the roles and responsibilities need to change, but also the tools and wider ecosystem need to evolve to support it. The transition to the new structures and work practices required will take time to become ingrained in organizational culture, but platforms that can help with the management of container-based environments will accelerate this migration. Ovum expects these platform approaches to proliferate over the next few years and to increasingly become more comprehensive in terms of the capabilities offered.

Key capabilities

The important elements of any container management platform do differ significantly from those of a traditional VM management solution. Customers are looking to build high-scale services consisting of microservices and this differs from a traditional VM management solution, because containers are application-based not machine-based. Because container management is more application-based, the high-level capabilities required in any management solution are the same, but the detailed approach taken by container management platforms does differ. Container platforms take into account that environments are distributed and applications move and are ephemeral, unlike infrastructure which is more static and long-lived. Ovum believes seven capabilities are key to any container management platform: security, ease of use, network management, resource and service management, automation, flexibility, and standardization and interoperability.

Security: The security requirements in a container environment are different from those in a VM environment and any platform must deal with the specific security requirements of applications as well as more traditional infrastructure security concerns.

Ease of use: One of the main reasons for the popularity of container management platforms is because of a skills shortage and the need for existing IT staff to be able to manage containers with as much familiarity as possible. This is particularly important from a developer perspective because the correct platform will empower the developers and not get in the way.

Network management: Like security, network challenges differ in a container-based environment. Any platform must therefore support the container-native network management approaches and technologies.

Resource and service management: Managing a container differs from managing a VM. A container requires the workload to be managed separately from the host, VM, itself. This means that in a container-based environment the containers are more portable and significantly greater in number. Therefore, resource and service management must operate at the very granular resource level, the core infrastructure level (Kubernetes elements such as Pods, Worker, and Master nodes), and the service level (the combination of many different containers).

Automation: Managing any container deployment at scale requires the underlying nodes to be managed correctly. The nodes must also remain up to date with current patch levels. Automation plays a big role in simplifying the management tasks, whether for the customer or for being delivered as a managed service, it is key to being able to operationally support containers at scale.

Flexibility: The big challenge is for organizations to be able to manage the container environments irrespective of where the containers are executing (on-premises or in a cloud).

Standardization and interoperability: The containers market is a nascent market and as such has a number of competing technology standards. The key element is that any platform can manage the most popular, currently OCI-compliant, containers such as Docker Engine.

Business value and applications

The traditional data center is a complex environment to manage because it typically consists of multiple different technologies. However, because of the time it takes to add new services, its costs are well known. By comparison, IaaS involves more variable costs because the environments can be created and used within a short time, but it still involves a level of management complexity to deal with the different environment types and resources. PaaS is "easier" because much of the underlying infrastructure and connective tissue is abstracted away, but often these environments are designed only for very specific application types and components. Also, the costs in PaaS are more predictable due to the approach to licensing and usage. The rise of containers will require a PaaS-like approach because containers can be more volatile (they are started and stopped more often) than VMs, which can lead to more variable costs. Organizations looking to deploy container technology, based on the ability to be agile and move the workload to wherever it is best to be run, will need platforms that can deal with the complexity of the technology, cost, and people. The business value from these container management platforms will be in their ability to manage the services being delivered at a more granular level in terms of performance, availability, scale, and cost, and to manage the service levels independently from the infrastructure.

Market landscape and participants

Market origin and dynamics

Containerization dates to the 1970s, when the technology was initially used solely for isolating application code. The technology initially lacked the usability and portability benefits that users have now come to expect since the launch of Docker and the introduction of the container engine in 2013. The containerization movement then formed around the open source project, which was developed to address the common pain point of the “dependency hell”, the challenge of making an ever-increasing set of applications, language, frameworks, and more interact properly with an ever-increasing set of hardware environments. The origins of Kubernetes has its roots in work done by Google when it was launched at DockerCon in 2014. Google had been running containerized workloads in production for more than a decade, and through an internal project named Borg it developed an orchestration engine. Kubernetes traces its lineage directly from Borg. Many of the developers at Google working on Kubernetes were formerly developers on the Borg project. However, other approaches were also

being developed, most notably Service Fabric, which is Microsoft's container orchestrator solution for deploying microservices across a cluster of machines.

Key trends in the container platform market

The container market is dynamic and evolving rapidly, and therefore the container management platform market is equally dynamic. The market is exhibiting great momentum for containers and equal momentum to drive standardization to accelerate adoption and industry growth. Kubernetes is currently the de facto standard orchestration engine, and Docker Engine (OCI-compliant) is the standard for containers, but other orchestrators such as Docker Swarm, CloudFoundry, Mesos, and AWS ECS are also used. However, Kubernetes and other orchestrators and concepts are often alien to IT administrators, so the direct use of Kubernetes to orchestrate a large deployment would be a challenge. The rise of the container management platform has emerged to create a bridge between these two worlds and to enable organizations to transition at a pace that fits with business demand and IT capability.

Future market development

The evolution of the container management market will almost certainly be led by the platform solutions, but these solutions will be augmented by new technologies in the storage, security, and network spaces. Current platforms take the Kubernetes orchestration and wrap other management functions around it, making it more of an enterprise solution. However, as the use of these platforms expands, more gaps in capability will be discovered, and the platforms must be flexible enough to accommodate these changes. The design criteria of many of the platforms is to be "open" and extensible to support the addition of vendor-specific hardware and new operating models. Ovum expects the market to rapidly expand, with every management and monitoring vendor producing solutions. However, ultimately the market will coalesce on a smaller number of vendor solutions, almost exclusively built using "open source" as the design criteria.

Contain management platform services

Container management platform services are typically a cloud-based managed service offering where the master nodes, and in some cases the worker nodes, are fully managed by the solution provider. These solutions typically are cloud-specific in terms of where the management layer and the containers can execute. However, some solutions are now extending this approach, so only the management plane is cloud-based and the containers can be executed in any location.

Amazon Web Services (AWS)

Ovum view

AWS has two different approaches to delivering container management: a native suite of solutions and an open source, Kubernetes-based solution. The value of these two options enables customers to select the approach that meets their specific needs. The native container management solution has deep integrations with AWS services and gives customers the ability to take full advantage of the entire AWS cloud platform. AWS has a long history of supporting open source solutions. According to

the Cloud-native Computing Foundation (CNCF), more than half of Kubernetes is run on AWS, so AWS having a Kubernetes solution seems a natural progression.

Solutions overview

The native solution from AWS has three main offerings that contribute to providing the user with choice: Amazon Elastic Container Service (ECS), AWS Fargate, and Amazon Elastic Container Registry (ECR).

Amazon Elastic Container Service (ECS)

Amazon Elastic Container Service (ECS) is a native AWS solution designed to work with the entire AWS platform. Many AWS services, including Amazon SageMaker and AWS Batch, run on Amazon ECS. ECS supports open container initiative (OCI)-compliant containers such as Docker and is deeply integrated with other AWS services including Amazon EC2, Elastic Load Balancing, Amazon VPC, AWS Identity and Access Management (IAM), Auto Scaling, Amazon CloudWatch, and the AWS CLI. Amazon ECS fully manages the control plane and supports auto-scaling across availability zones in a region. ECS uses a declarative approach (task definition) based on json files that describe the details about the workload and how it should be executed, including defining networking, storage, and deployment options. Customers that use Amazon ECS have the option to use Amazon EC2 virtual machines or the AWS Fargate serverless compute engine for running containers. While customers need to choose to run each service using Fargate or EC2, both compute types can be used together in a single ECS cluster. Using EC2 with ECS, the worker nodes are the responsibility of the customer. The management console has built-in integrations that can start and perform basic EC2 management functions, including provisioning EC2 Spot instances and informing users which worker nodes can be taken offline for any update or patching to render the process non-disruptive. The ECS control plane is fully managed by AWS and updates to the control plane are non-disruptive. Ovum believes that this solution represents a viable option for any organization that wants to keep costs low, has familiarity with AWS APIs, or does not want to manage the complexities of maintaining a container management control plane. AWS does not charge for ECS; the customer pays only for the EC2 instances or Fargate compute hours they provision to run the workloads. AWS is continuing to develop the solution and recently announced the ability to assign an IP address directly to a running group of containers (task), time (cron) and event-based scheduling, Docker volumes and volume plugins, daemon container scheduling, and integrated DNS-based service discovery that can be enabled automatically in the ECS control plane by customers.

AWS Fargate

Managing VMs to run containers gives flexibility and lots of options for configuration, but many organizations are looking for a less hands-on approach to managing their container infrastructure. AWS Fargate is a serverless compute engine that is built for containers and currently supports ECS, with plans to expand this to other orchestrators. The main difference between using EC2 and Fargate to run containers is that with Fargate, all the compute instances that run containers are managed by AWS. The customer describes the initial settings in terms of memory and CPU in the task definition, and the required compute resources are automatically launched and scaled. Billing is based on the CPU and memory required for the task and is charged by the second until the task terminates. This allows more granular resource utilization than for VMs and removes the need to manage container placement constraints or monitor cluster utilization.

Amazon Elastic Container Registry (ECR)

Amazon Elastic Container Registry (ECR) is a fully managed cloud-based OCI-compliant container image registry where customers can store and manage all their images. These images can be copied from the public Docker registry (Docker Hub) or imported from a development environment. The images stored in Amazon ECR are encrypted at rest and access is integrated with AWS IAM solutions. The current ECR offering does not perform automatic image scanning and threat identification, but AWS says this is on the roadmap. Customers pay only for the storage space being used to keep their images, and ECR compresses images when they are imported and allows customers to store images in the AWS region where their containers are being deployed. This makes containers pulling from ECR very fast to start. The customer is responsible for managing the contents of the registry, and ECR provides several lifecycle tools to help customers search for and automatically clean up unused images in the registry. This is a useful feature for large teams with multiple services and lots of image versions.

Amazon Elastic Container Service for Kubernetes (EKS)

Amazon Elastic Container Service for Kubernetes (EKS) provides an open source solution for container management on AWS to complement the native ECS solution. Amazon EKS is a managed control plane for Kubernetes on AWS. EKS manages the master node and the etcd database for the customer, while the customer is responsible for the worker nodes (like with ECS on EC2). The worker nodes must be housed in EC2, and multi-availability zone support is provided to deliver high availability. Ovum believes that EKS meets the demands of the market for a Kubernetes-based solution for AWS. AWS made sure that before it released EKS, it was fully certified as a Kubernetes provider. The EKS control plane runs upstream Kubernetes to maintain compatibility with open source and third-party tooling as well as existing Kubernetes implementations. The pricing is based on two factors. First, the customer pays for the EC2 instances they use for the worker nodes. Second, AWS charges $0.20 per control plane cluster per hour. The EKS solution is currently available in the AWS US west coast and east coast regions and in the Ireland region.

Google Kubernetes Engine

Summary

Google is the biggest contributor to the open source Kubernetes project, making approximately 50% of all contributions over the last couple of years. This is not unexpected because Kubernetes came from Google's own work on project Borg, an internal development to manage its cloud platform. Google Kubernetes Engine, which has been available since August 2015, is one of the most mature platforms on the market and has just been extended to cover on-premises as well as cloud-based workloads.

Solution overview

Google Kubernetes Engine (GKE) is a production-ready solution that covers both cloud and on-premises (GKE On-Prem was announced in July 2018) as a fully managed service. Ovum believes that the new on-premises solution extends the reach of the Google platform, and that the managed service approach will appeal to most organizations because it is enterprise-grade and gives users the flexibility to manage the worker nodes to fit with any organizational process or procedure. The key capabilities of GKE are: automation, multiple zones and regional clusters, security, monitoring, networking, visualization, and container registry and creation.

Automation

As part of its managed service, Google provides three levels of automation: Auto-repair, Auto-upgrade, and Auto-scaling.

Auto-repair: When GKE detects that a node is failing some of the health checks, it initiates a repair process. The health check covers nodes that are unreachable or unresponsive, as well as nodes that have been in a not-ready state for a prolonged period, and nodes whose boot disks are out of space. The repair process drains the pods before recreating the node.

Auto-upgrade: The second automation capability is to ensure that the master and nodes are maintained at the latest stable release level for Kubernetes. GKE will automatically upgrade the master nodes in a non-disruptive way, while the worker nodes are the user's responsibility. GKE provides options for how the worker nodes can be upgraded (automatically, scheduled automatically, or manually).

Auto-scaling: The third automation capability manages the scaling of any container cluster. GKE manages scaling according to the specific needs of the workload.

Horizontal Pod Autoscaler is used in conjunction with Stackdriver and will add or remove pods based on set metrics, such as CPU utilization, the number of requests per second, or any custom metrics provided by the user, such as the number of tasks in the task queue.

Cluster Autoscaler increases the number of nodes based on pending pod requests. This is driven by the user's scheduling of pods, and if the cluster does not have sufficient resources (CPU and memory) to meet the pod request, GKE will automatically add a worker. GKE clusters are made of nodepools (groups of identical nodes). Each cluster can contain several nodepools. Cluster Autoscaler automatically decides which of the nodepools should get additional nodes to maintain the best cost/performance ratio. Ovum notes that GKE Cluster Autoscaler does not operate on a pod resource utilization approach because this can be variable but is instead designed to ensure the demand for pod creation can be met.

Cluster Autoscaler Balancing Across Zones enables GKE to manage how scale-up applications are distributed across multiple zones in a region. GKE attempts to balance the size of each node pool in each zone so that the traffic is distributed evenly.

Multiple zones and regional clusters

GKE provides a high-availability capability through its ability to operate in multiple zones and to make these operate as a regional cluster. The concept of regional clusters supports the ability to perform zero-downtime upgrades and improves service availability by delivering an uptime of 99.95% through the use of multiple master nodes in any given region.

Security

Security in a containerized environment differs from that in a VM-based environment in several subtle ways. First, the platform is a shared kernel model that brings unique challenges. Second, the validation of the software supply chain in a continuous delivery environment requires strong change tracking. Finally, the Linux runtime environment with its use of daemons and "root" privilege add an extra challenge. GKE addresses these challenges with different approaches that include:

- Full at rest encryption of all the components, this includes the location of secrets in etcd (a value store for all cluster data).

- Transport Layer Security (TLS) for master-to-master and worker node-to-master communication.
- Easy to use network policy enforcement using Calico to control pod-to-pod communication.
- Metadata concealment to prevent user pods from accessing certain VM metadata from cluster nodes, such as Kubelet credentials and VM instance information. Specifically, metadata concealment protects access to kube-env, which contains Kubelet credentials, and the VM's instance identity token.
- Private clusters enable organizations to deploy clusters privately as part of the Google Virtual Private Cloud (VPC). This provides an isolated boundary where applications can be securely deployed. The reality of a Private Cluster is that it can only be accessed from within the trusted VPC. An extra security measure provided in Private Clusters involves blocking master nodes from being accessed from the internet by default.

Monitoring

GKE uses Stackdriver to monitor the cluster from a resource consumption perspective and monitor against metrics, which can be standard or user-defined. Stackdriver is also used to perform log consolidation (collect, store, and process container and system log files). Stackdriver's big advantage is that it works not only for Google Cloud Platform, but also for AWS cloud.

Networking

GKE is fully integrated with Google Cloud Platform networking, which enables a layer-7 load balancer (or HTTP) to be provisioned for ingress, and a layer-4 network load balancer (or TCP) for internal load balancing.

Visualization

GKE uses the Google Cloud Console to provide visualization of the workloads, configuration, load balancers, and storage volumes across all the clusters.

Container registry and creation

GKE provides a private container registry that holds all an organization's container images, which are automatically scanned for vulnerabilities. The registry also integrates with many other Google cloud services, including Google Cloud Container Builder, enabling developers to import from a range of different repositories such as GitHub and BitBucket.

IBM Cloud Kubernetes Service

Summary

IBM has two main container offerings: a customer managed solution that runs on IBM Cloud Private for organizations that want to operate and manage the environment themselves, and IBM Cloud Kubernetes Service (IKS), a fully managed service offering. IBM Cloud Private is less a managed container platform than a private cloud that provides a self-managed container platform. The focus of this report is IKS.

Solution overview

With IKS, IBM provides a flexible approach to managing container environments based on IBM Cloud. The core concept is not only to simplify the creation and management of secure Kubernetes clusters,

but also to offer an environment where these container applications can be extended by using the wealth of services IBM and its partners provide.

Simplified cluster management

The major issue for most IT operational teams is that managing containers and the environment is fundamentally different to that required for VMs. With the IKS service, IBM provides fully managed, dedicated Kubernetes clusters by managing the master and worker nodes, while customers are responsible for deploying and managing the Kubernetes resources within the cluster. The whole service is based on the IBM Cloud, so the service is available globally in six IBM cloud regions and more than 20 data centers. IKS automatically applies master patch levels and offers flexibility when it comes to the worker nodes, automating the worker's update to the latest version aligned with the master Kubernetes level based on when the customer wishes for the update to take place. IKS uses advanced analytics to evaluate the cost of any update, and using customer configurations, a "disruption budget" is calculated to decide how many worker nodes can be updated simultaneously without causing a service disruption for the application.

With its global reach, IBM includes in IKS the concept of multizone clusters, where worker pools are used to provision worker nodes across multiple zones within a region. This capability improves the reliability of the solution as well as making it optimized in terms of performance by enabling "pods" to be evenly distributed across worker nodes and zones. IKS offers an automated worker node recovery capability where unhealthy worker nodes can be identified and automatically recovered.

Simplification of the management task is the number-one reason why container management platforms are becoming popular, and with IKS, IBM has other capabilities worth mentioning. First, IKS supports bare-metal worker nodes and includes support for GPUs. This enables organizations to fully isolate its worker nodes from physical systems without any additional virtualization overhead. Second, IKS provides fully managed persistent storage in support of Kubernetes Persistent Volume Claims, and in the latest release includes Kubernetes persistent storage options. Finally, IKS enables integrated load balancer and Ingress load balancing. Ingress is a Kubernetes resource that routes and balances URL requests to service endpoints in a cluster. IKS has support for both public and private load balancers and Ingress. Ingress can therefore be used to expose multiple application services to public or private networks by using a unique public or private route.

Security and network management

IKS includes the management of security and the network as part of its offering, something that is increasingly requiring different skills than those needed for VM security. IKS supports Linux Unified Key Setup (LUKS) encryption for the local storage of each worker node in a cluster with its own encryption keys. The container images can be stored in private repositories within the IBM Cloud Container Registry. As part of the container registry, IKS provides a secure service that scans container images in the registry, and the images can be scanned as part of the build process prior to being stored in the registry. The container registry supports Docker Notary to allow for cryptographically signed images. IKS has an image enforcer that enables customers to restrict the deployment of images that have vulnerabilities or that are missing required image keys. Ovum believes that this level of secure control over the images that can be deployed provides the confidence needed for those new to container application workloads. IKS adds a trusted compute layer available with bare-metal worker nodes to the security it provides. Trusted compute provides an addition level of assurance because the hardware trust is transparent to Kubernetes so that an

application can have confidence that the integrity of the BIOS, boot layer, and kernel have not been compromised.

In terms of the network, IKS provides a VPN capability that allows applications in the cluster to access resources outside the cluster through a secure tunnel. IKS also offers Istio service mesh networking, which is specifically designed to connect and manage microservices architectures. With the use of a service mesh, policy can be enforced at the container level and the entire deployment becomes fully observable, enabling proactive management of the service delivery.

Additional services

IKS uses some of the associated capabilities that IBM Cloud offers, as well as access to other services IBM has developed, such as those in the IoT and Watson practice areas. There are nearly 200 IBM and third-party services that can be used on IKS. Ovum believes that having services such as weather, natural language processing, MongoDB, and blockchain can provide a great degree of added value for developers. The big benefit for organizations is that using these services enables existing applications to be modernized and extended quickly and easily.

Microsoft Azure Kubernetes Service

Summary

Microsoft has its own container technology, Windows Containers, but Azure Kubernetes Service (AKS) is a hosted managed solution for OCI-compliant containers such as Docker. As with many other cloud service providers' solutions, the use of the service is free and the customer pays for the resources consumed, in this case by the worker nodes on the Azure platform. AKS assumes responsibility for the master nodes and provides monitoring and management services for the entire cluster. The monitoring solutions provide metric-driven analysis of the cluster so that customers can quickly understand if any deployment is performing abnormally, or can understand the resource consumption and estimate the future cost. The use of monitoring metrics requires agents to be deployed on the Linux images to provide the metric-level information required.

Solution overview

AKS is a fully managed Kubernetes service that supports zero-cost managed master nodes, offering the latest Kubernetes releases (e.g. 1.11.x) and end-to-end developer experience through differentiated add-ons, such as Virtual Network, Azure Dev Spaces, and Azure Monitor. The open source Virtual Kubelet project allows a unique serverless Kubernetes experience with AKS and Azure Container Instances (ACI)-backed worker nodes, enabling instant scale and per-second billing. There are three main ways customers can interact with Azure: through a portal, via a command line interface, or via resource templates defined in Resource Manager or Teraform. Ovum believes that the use of these templates makes deployment much simpler, because AKS manages the entire deployment of a cluster, including master and worker nodes. AKS also differs from other container management platforms in how it manages cluster upgrades. AKS enables the customer to select and apply upgrades as they become available through the portal or command line interface. It manages the upgrade process automatically and non-disruptively by draining down nodes before upgrade and rehydrating them post upgrade in a controlled approach.

AKS offers CPU and GPU options in its resources to execute clusters, and will auto-scale up and down cluster to meet demand. Ovum particularly likes the use of HTTP application routing used by AKS so that an Ingress Controller can be configured automatically, and this also enables publicly

accessible domain name services (DNS) to be configured automatically. To ensure secure access, AKS supports role-based access control that is integrated with Azure Active Directory. In AKS, a role contains rules that represent a set of permissions. These permissions are purely additive. In AKS there is no concept of a deny rule stopping a role from accessing an operation. A role can be defined within a namespace using the Role command, or cluster-wide with a ClusterRole command. If the Role command is used, it can only be used to grant access to resources within a single namespace. Ovum believes this approach provides an added level of security, because the namespace is in effect how different environments are isolated. The Role command can therefore be seen as a local privilege, while ClusterRole is wider in its reach.

AKS caters for developers as well as operations teams by providing an environment where developers can use a wide variety of tools, such as Helm, Draft, and the Kubernetes extension for Visual Studio Code. These tools are all designed to work seamlessly with AKS, and work in conjunction with Azure Dev Spaces and the Azure DevOps project to link to the Git repository to provide an environment for continuous integration and delivery. The Azure Container Registry (ACR) is a Docker 2.0-compliant registry that can include both Windows and Linux images. Customers control image names for all container deployments. ACR also uses standard Docker commands to push images into a repository, or to pull an image from a repository.

AKS supports two types of networking (a basic configuration and advanced configuration) but does not support the service mesh networking concepts and therefore requires a bridge to translate the IP address for workloads. The basic networking is the default setting and means that AKS manages the Kubernetes pods, and the customer does not have control over the IP address configuration for the subnets or the cluster. The advanced networking option places the Kubernetes pods in the Azure virtual network (AVnet). The advantage of this is that pods are assigned an IP address and can therefore communicate with other pods in the cluster. Another advantage is that pods can connect to other services in a peered VNet, and to on-premises networks over ExpressRoute and site-to-site (S2S) VPN connections.

AKS supports persistent storage requirements of a container through Azure files and Azure disks. Each AKS cluster includes two pre-created storage classes, both configured to work with Azure disks:

- The default storage class provisions a standard Azure disk. This standard storage is backed by HDDs, and is aimed at cost-effective storage while still delivering a good level of performance. Standard disks are ideal for dev and test workload.
- The managed premium storage class provisions a premium Azure disk. Premium disks are based on SSDs and offer a high-performance, low-latency disk.

Other Microsoft offerings

ACI previewed in July 2017 and was on general availability in April 2018. It is a secure multitenant containers-as-a-service offering that supports Linux and Windows containers at per-second pricing. Microsoft designed ACI for "bursty" container workloads without any VM or cluster management overhead, and it offers the customer a serverless container experience.

OpenShift on Azure is a fully managed OpenShift service that is jointly operated by Microsoft and RedHat. Currently in private preview, it is the first container service to support RHEL containers and Windows containers with a unified billing and support experience.

Azure Service Fabric is a microservices platform that supports Linux/Windows containers, as well as guest executables for large-scale stateful and stateless microservices. Optimized for .NET workloads, it also supports a variety of programming models, including the Actor pattern. Service Fabric Mesh (currently in preview) is a fully managed serverless option for Service Fabric that requires no VM or orchestrator management.

Azure App Service is a platform-as-a-service (PaaS) offering that supports Linux and Windows containers (also in preview) and is a developer-optimized platform for containerized web applications.

Azure Container Service is a service that supports automated cluster deployment and management for open source orchestrators such as Docker Swarm and DC/OS. This service also supports Kubernetes, but with the emergence of AKS, customers have a better option.

Azure Marketplace offers partner-supported container solutions such as Pivotal Cloud Foundry, Mesosphere DC/OS, and Docker Datacenter.

In the quest to enable containerization from cloud to edge, Azure Stack (an extension of Azure-enabling support for hybrid applications on-premises) currently supports Service Fabric and Kubernetes with support for other Azure container services in the roadmap. Azure IoT Edge runtime supports container-based deployments on supported devices. Azure Functions runtime supports container deployment on various orchestration engines, including Kubernetes enabling serverless compute on edge through containers.

Oracle Container Native Service

Ovum view

Oracle has taken an open and flexible approach to its container management services, which is based on two core offerings; Oracle Container Engine for Kubernetes (OCEK) and Oracle Private Registry. The ethos of Oracle is to enable the DevOps process of build, deploy, and operate containers in a way that supports the multi-cloud strategy that enterprise customers have adopted. The other key driver behind OCEK is to simplify the management challenges that organizations are experiencing by reducing the administrative burden that Kubernetes-based environments demand, such as maintaining the data plane (master and worker node patching and upgrading), overlaying persistent storage, and managing the control plane (API Server, etcd, scheduler, etc.).

Solution overview

The two offerings that Oracle provides are best when used together, but fitting with the open ethos customers can use the OCEK with any open container initiative (OCI)-compliant container registry, and likewise Oracle's Private Registry can be used with any container management platform that support OCI-compliant containers.

Oracle Container Engine for Kubernetes

OCEK uses a certified cloud-native computing foundation (CNCF) conformant version of the standard upstream Kubernetes as its orchestration layer. This adherence to the CNCF conformance and OCI compliance on the container images means Oracle supports the full portability of containers across platforms. This is demonstrated by the way OCEK is deployed, which can be on bare metal or VMs. While the worker nodes run on the customer's own tenancy so they are close to the applications. This approach enable the customer to be in control of the infrastructure its worker nodes are running on, and while these worker nodes are not automatically upgraded and patched, unlike the master nodes

that are automatically and non-disruptively upgraded and patched, Oracle provides a one-click approach to enabling the customer to do this themselves.

Like all the container management services OCEK is run on the vendor's cloud, in this case the Oracle Cloud, which enables it to take advantage of added value services such as, load balancers, high availability, persistent volume claim, and persistent volume native integration. The high availability is provided both at the managed control plane and the availability domain levels. This provides customers with options in terms of how to construct the services in terms of ensuring reliability. Oracle looks after the control plane so the customer does not need to be concerned it its availability, and by using the availability domains and the ability to create clusters that span availability domains the reliability of services can be delivered to match that needed by the customer. Ovum considers this approach to provide real choice, but customers need to understand that as more availability domains are used the cost also increases, so the level of reliability can be directly linked to the spend, which is a positive alignment from the business perspective.

Oracle also provides several additional services to support OCEK and make its use more DevOps friendly:

- Operational efficiency as It provides a capability to self-heal clusters and node-pools so that operational teams do not have constantly monitor the health of the deployment and can focus on other activities.
- Team based access controls enable permissions to be managed at a per cluster level allowing role based access to Kubernetes using native Oracle Cloud Infrastructure Identity and Access Management.
- Developer friendly access with single click cluster creation and management, full REST API and CLI, use of familiar Docker based run time, and the use of open tools such as Helm.

Oracle Functions is a newly-announced Functions-as-a-service (FaaS) offering based on the open source Fn Project, allowing customers to quickly build, deploy, and scale functions on the Oracle Cloud. This is a move towards releasing more developer-facing services for organizations going cloud-native.

Oracle Private Registry

Oracle's registry is a standard Docker v2 compliant repository that allows access via a CLI. The images are encrypted once they have been imported into the registry and while full signature security and vulnerability scanning is on the road map, it supports the ability to use partner solutions to deliver a secure registry. Oracle Private Registry only charges for the storage and network used by the customer, not based on some arbitrary size of the registry. Ovum particularly like the self-cleaning capability in the registry where it can tidy up the images to ensure more efficient use of the registry as well as keeping the images generated in-line with corporate policies on retention/deletion.

Platform9 Managed Kubernetes

Summary

Platform9 has developed an open and vendor agnostic approach to dealing with the complexities of managing a multicloud environment. Ovum particularly like Platform9's capabilities for supporting both cloud-native and legacy workload management. This hybrid approach to cloud workloads is going to be a reality for many organizations in the next five to 10 years. The other key observation is that

Platform9 supports the most popular tools for containers and serverless computing, and builds on these to make management a simple and consistent experience from an operations perspective.

Solution overview

Platform9 Managed Kubernetes (PMK) is a SaaS solution that provides a simple way for organizations to deploy, monitor, and manage upgrades, and troubleshoot Kubernetes-based container deployments. PMK makes it possible to run Platform9's Kubernetes on any underlying infrastructure. PMK includes a minimum-disruption, fully managed upgrade service that starts with upgrading the master nodes and then continues with the worker nodes. SLAs are provided for customers, and Platform9 says it has a 99.9% availability uptime record. Another key aspect of managing Kubernetes-based container deployments at scale is the ability to remediate failing or failed containers. PMK performs 24/7 monitoring of the Kubernetes environment and will if possible auto-repair any containers it finds as unresponsive or exhibiting characteristics consistent with a failing container. If an auto-repair is not possible, PMK will alert the nominated IT administrators so that action can be taken.

PMK provides a single dashboard experience that enables a single administrator to have visibility of all the clusters, workloads, and resources across different public cloud regions and availability zones. The PMK solution enables highly available Kubernetes clusters to be built and deployed. It also enables high availability by using the availability zones in the public cloud providers, and by building Kubernetes clusters that span availability zones. Another key aspect of PMK is that it has been designed with multitenant use cases in mind. For example, resource quotas can be defined and used on a per-tenant basis. Ovum believes this is an excellent approach to managing container deployments because different workloads can be allocated different quota to ensure that any expansion of the platform is managed to corporate governance policies.

Security in PMK is not linked to any specific vendor solution, it supports single sign on (SSO) that includes Active Directory, Okta, LDAP, and others. In the recent Ovum Decision Matrix on multicloud management, Platform9 recorded a score of 8.67 out of 10 in the financial management category. Platform9 provides a good range of financial control over a wide range of technologies, from VMs to cloud-native. This financial control includes showback and chargeback, which means it can be deployed to match organizations' levels of financial maturity. Platform9 supports a wide variety of environments where its solution can be deployed, including on-premises, public clouds, or as a SaaS solution. This enables organizations to select the environment that best fits their current and future objectives. Ovum believes that this degree of flexibility in terms of where a solution can be run aligns with the new thinking that as organizations adopt newer forms of computing, they would prefer to move existing familiar toolsets if possible.

VMware Cloud PKS

Ovum view

VMware entered the Kubernetes container management space with its VMware Kubernetes Engine (VKE) solution, which has now been rebranded to VMware Cloud PKS and went on public beta on June 26, 2018 and is expected to be on general availability sometime in late 2018. VMware's approach is to provide a range of solutions designed to meet the differing needs of the different personas and capabilities operating in the cloud-native space. VMware provides managed services

for Kubernetes with VMware Cloud PKS, while also partnering with its Dell Technologies sister company, Pivotal, on a more self-managed solution.

Solution offering

VMware Cloud PKS is a fully managed Kubernetes service that is hosted on an AWS shadow account that is totally owned and managed by VMware. The term managed service when used in conjunction with Kubernetes can be thought of as two levels of service that align with the control plane (master node) and the worker nodes. VMware Cloud PKS provides a fully managed service across both these, unlike some solutions that manage the master node (control plane), but the customer must manage the worker nodes and the associated complexities of dealing with node-to-node communications.

One of the core value propositions behind VMware's strategy is to simplify the experience of being able to use, develop, and manage containers. The GUI used by VMware Cloud PKS does not require any technical knowledge of the resources required currently or in the future. The user starts by selecting the type of cluster they need. VMware Cloud PKS offers two basic types of configuration, a development cluster and a production cluster. The key difference between these configurations is that a production cluster consists of three master nodes spread across three different AWS availability zones where the development cluster has a single master node. The selection is based on a simple choice of cost versus availability, with the development cluster being cheaper to deploy, but less resilient and requiring nominated downtime so that any Kubernetes updates can be applied. VMware Cloud PKS automatically updates the cluster with security patches, but if customers want to upgrade the Kubernetes version, they must select the version and let VMware know. This upgrade is performed by VMware, but the level of disruption depends on the application architecture selected. The master nodes in a production cluster can be upgraded non-disruptively, whereas the development cluster requires the master node to be offline. However, the worker nodes are where the application architecture impacts how disruptive any upgrade will be.

Another key value proposition of VMware Cloud PKS is that it fully manages the scaling of the cluster. This is achieved through the use of VMware's Smart Cluster technology. VMware Smart Cluster automates the selection of compute resources to constantly optimize resource usage, provide high availability, and reduce cost. Smart Cluster works according to the service intents the customer enters, and then automatically configures the cluster to meet these intents. The Smart Cluster continually monitors use of the cluster and adjusts the resources as needed to ensure each cluster is only allocated just enough to deliver the required service intent. Another key benefit of using Smart Cluster is that it automatically configures high availability and deals with the complexity of cross-availability zone communication. Moreover, VMware Cloud PKS continuously monitors the health of Smart Clusters and automatically remediates any potential issues.

VMware has addressed some of the security concerns of running and managing container environments. The OCI-compliant container image works on a shared kernel principle, and security is a different challenge to running VMs. One of the big risks is that of containment failures. This requires the namespace to ensure that it separates the "root" inside the container from "root" outside the container. This also introduces the concept of privileged OCI-compliant containers, a privileged container has access to hard disks on the compute host that may contain data and applications. If the privileged container (pod) is a third-party container, this information is at risk. VMware Cloud PKS splits any privileged pods so that only a small subset of the commands requires elevated privileges.

The running of OCI-compliant containers requires the OCI-compliant daemon to be executed, and this means having "root" privileges. "Root" is a very powerful privilege and access to run the OCI-compliant daemon should be restricted. Access to the OCI-compliant daemon has some significant security implications and VMware Cloud PKS uses system namespace and daemon sets to reduce the impact of this exposure.

While VMware Cloud PKS is only hosted on AWS, VMware's vision is to have it available on multiple public clouds to make it a multicloud Kubernetes-as-a-service offering, which will be a key differentiator of VMware Cloud PKS when competing with other public cloud Kubernetes offerings. Azure was announced to be the next cloud supported by VMware Cloud PKS.

Container management platform solutions

Container management platform solutions offer a flexible approach to where the solution is executed (hosted) as well as providing different management options from fully managed to self-managed. These solutions can run in a public cloud or on-premises, the key difference is they tend to me more platform agnostic.

Docker Enterprise

Summary

Docker has developed the capabilities of Docker Engine and its Docker Enterprise container platform solution to ensure it remains both true to its open source heritage and is capable of supporting the wide range of application use cases in the market. Docker provides developers and IT operations the freedom to build, secure, and manage applications efficiently without technology or infrastructure lock-in. Central to Docker's vision is that its solution is multicloud, multi-operating system (Windows and Linux), and multi-Linux, which translates to no vendor lock-in regarding where the containers can run and be managed. Docker offers: Moby project, Docker Engine - Community, Docker Engine -Enterprise, Docker Desktop, and Docker Enterprise.

Moby project: Docker is the primary sponsor of this open source project that provides components and a framework for assembling specialized container systems intended for open source contributors and container ecosystem developers.

Docker Engine - Community: This is a free to use container runtime for Linux that comes with community-based support. Powered by containerd, the core container runtime used by millions of users, Docker Engine-Community has a client-side command-line interface (CLI) that enables users to interact with the daemon through the Docker Engine API. The intended audience includes development and testing providers and organizations that are comfortable with managing and supporting open source software.

Docker Engine - Enterprise: Powered by containerd, the enterprise version is available via subscription and includes a validated container runtime with enterprise-class software maintenance/patches, support with defined SLAs and private support channels and certified for enterprise operating systems and infrastructure. Also available is an ecosystem of certified network and storage plugins and containers with ISV software with collaborative support. Docker Engine – Enterprise forms the foundation for an enterprise container platform.

Docker Desktop: Used by developers worldwide, Docker Desktop is an easy to install application for Mac or Windows environments that enables users to start coding and containerizing in minutes. Docker Desktop enables developers to build applications locally with either Docker Swarm or Kubernetes and deploy them to production to Docker Enterprise environments. Docker enables users to maintain a consistent developer-to-operator workflow with the added value of Docker Desktop, which includes everything users need to start building containerized applications.

Docker Enterprise: This is an enterprise-grade container platform that comes with integrated orchestration, security, and management capabilities that enables organizations to build, secure, and manage applications across the entire application lifecycle with a full support package. It is a subscription-based solution. Docker Enterprise is used by more than 650 enterprises including ADP, Liberty Mutual, and Visa.

Solution overview

When it developed its Docker Enterprise container platform, Docker focused on how to solve the challenges around the disconnected development-to-operations experience that organizations commonly face. From Docker's perspective, the competition was heavyweight platforms that focus mostly on one end of the application pipeline. Some were forcing IT operations to completely renew process and procedures in addition to learning complex orchestration tools directly, while others were forcing developers to write applications only in a specific architecture. Docker focused on freedom of choice, agility, and integrated security as the core pillars of its enterprise container platform.

Docker Enterprise is designed to be open and can run Kubernetes and Swarm as orchestration engines that can coexist so both can be managed at the same time. Docker Enterprise can also support many different Linux distributions, Ubuntu, SUSE, CentOS, RHEL, and Oracle Linux, as well as Windows Server 2016 operating systems and Windows containers. In addition, Docker Enterprise supports X86, IBM Power, and IBM Z platforms, and provides Docker Certified Infrastructure for best practices-based deployment into VMware vSphere, AWS, and Microsoft Azure environments. Docker Enterprise provides APIs and open interfaces to integrate with a range of enterprise IT systems at both legacy and modern vendors.

Docker also recently announced federated application management for Docker Enterprise that provides a single management plane to automate the management and security of containerized applications on premises and across hosted Kubernetes-based cloud services, including Azure AKS, AWS EKS, and Google GKE. Docker Enterprise customers can migrate applications to and between clouds as their needs change, enabling organizations to achieve multicloud portability.

From a simplicity perspective, Docker Enterprise is designed so that IT operational teams do not need Kubernetes experience to manage, deploy, or secure containers. Docker Enterprise makes use of automation to perform many of the activities from development to deployment. Docker Compose enables developers to define the containers and environment requirements for an application and provide this information to Docker Enterprise to drive much of the task-level automation. When considering scaling out container platforms to host a larger number of applications, different platforms provide different approaches. The ability to manager cluster sprawl is a little understood challenge, as VM sprawl was when it emerged at the start of the virtualization era. Docker Enterprise's focus on the ability to use a permission-based model and secure application zones helps create the conditions where scale can be achieved without creating an explosion in the cluster estate.

An aspect of security that Docker Enterprise provides is focused on ensuring that the container supply chain is secure and tamper-proof. With Docker Enterprise, both Kubernetes and Swarm users can take advantage of a policy-driven secure supply chain that is designed to provide governance and oversight over the entire container lifecycle. Organizations can set policies to automate the process of moving an application through test, QA, staging, and production, and users can enforce the rules around which applications are allowed to be deployed. Because these are automated processes that enforce governance without adding any manual bottlenecks to the delivery process, safer applications can be delivered without getting slowed down.

Docker Enterprise comes with Docker Trusted Registry, a local registry that performs automated vulnerability scanning and image isolation if any known vulnerabilities are found. Docker Enterprise uses the concept of "content trust", where organizations create secure checkpoints throughout the application lifecycle, so only the people authorized have been involved with the application. Docker Enterprise automatically detects if anybody else has touched the application, and if so then the application gets blocked.

In addition to the security aspects, the Docker Enterprise registry provides some other useful benefits. For example, organizations can share images from one repository to another, which helps global teams. The content is centralized but users can have a repository at each site if required. The images can be mirrored so that local registries can be provided for improved performance, with the images kept synchronized. Docker Enterprise also support image caching, which can extend a registry to a local cache but retains the access control via the Transport Layer Security (TLS) protocol for encryption and data integrity.

For integrated container networking, Docker Enterprise works with Calico networking to provide a secure networking solution within the platform. Calico creates and manages a flat layer-3 network, assigning each workload a fully routable IP address. This means that workloads can communicate without IP encapsulation or network address translation, resulting in improved performance, easier troubleshooting, and better interoperability. In environments that require an overlay, Calico uses IP-in-IP tunneling or can work with other overlay networking such as flannel. Calico also provides dynamic enforcement of network security rules. Using Calico’s simple policy language, users can achieve fine-grained control over communications between containers, virtual machine workloads, and bare-metal host endpoints.

To help organizations adopt containerization, Docker developed its Modernize Traditional Applications (MTA) solution, which provides a path to cutting operational costs and gaining agility. Most enterprise applications are legacy, and by containerizing these applications, they become more secure, cost-efficient, and portable to hybrid cloud environments, without needing to touch the code. Docker’s MTA solution includes the Docker Enterprise platform, custom-built tools such as the Docker Application Converter, Docker Certified Infrastructure, and help from the Docker Professional Services team. With MTA, organizations can create the governance and application pipelines necessary to continue modernization for these applications and pave the way for microservices and continued innovation.

HashiCorp

Summary

HashiCorp believes that as organizations transition from a predominantly VM-based deployment model to a cloud-based container model, the core tasks – development, security, operations, and

networking – all require modernization. However, HashiCorp considers these four key disciplines will transition at different speeds and will be driven by different forces. Therefore, it offers a suite of solutions (Nomad, Vault, Terraform, and Consul), designed to operate individually or as a collective, to match the needs of the organization. Ovum considers that taking this modular approach, and being an open source-based project, makes HashiCorp's offering a vendor-neutral solution; it can integrate easily with existing tools from other vendors, making the deployment non-disruptive.

Solution overview

The four core products from HashiCorp are designed to deliver the solution to the four key tasks: development, security, operations, and networking. All these solutions come in three standard packages; Open Source, Pro, and Premium, with the Open Source flavor being free to download and use, while Pro and Premium are enterprise products.

Nomad

Nomad is aimed at the developer, enabling both containerized and legacy application deployment to be decoupled from the infrastructure lifecycle. It achieves this through the use of a declarative approach, where the developer declares what they want to run (such as Docker image, resources, priority, or constraints) and Nomad handles where it should run, on what cluster and node, and how to run it, how many instances it needs, etc. While Nomad supports Docker as a first-class citizen, it is not strictly required. Users can run legacy applications directly using Nomad's "exec" driver. The modeling within Nomad uses the concept of regions (which can be a collection of data centers). The scheduling happens at this regional level and therefore supports cross-data-center scheduling of jobs. Regions can be easily federated together with a single CLI command to support unified deployments across private infrastructure and one or more public cloud providers. One of the key aims of Nomad is simplicity, which is achieved in terms of deployment by using a single binary, with no dependencies on external services for coordination or storage functions. Simplicity in terms of usability is delivered by Nomad, enabling it to run VMs, containers, or application runtimes like Java. Nomad has also been proven to scale single clusters to more than 10,000 nodes in a production setting.

Vault

Aimed at the security aspects of containers, HashiCorp Vault is an identity-based security solution for securely managing and tightly controlling access to applications, systems, and secrets. A secret can be anything that requires tightly controlled access such as passwords, encryption keys, etc. One of the key features of Vault is the concept of a lease. All secrets in Vault have a lease, and at the end of the lease, Vault revokes that secret. The ability to revoke secrets can be applied to all secrets of a particular type, or to all secrets that relate to a specific user. Another key feature HashiCorp provides with Vault is the generation of dynamic credentials. This feature provides a number of improvements over the use of credential rotation. For example, when using dynamic secrets, there is no need to accommodate a deadlock period during credential rotation (which places a burden on software developers to incorporate application logic to deal with rotation schedules). This can lead to unplanned service downtime if not implemented properly. With dynamic credentials, a unique set of credentials is used per client. If a single set of credentials is exposed, the point of compromise can be quickly determined, and the credentials can be revoked. The big advantage of the HashiCorp approach is it provides a robust infrastructure – built on dynamic secrets – that allows for aggressive rotation, so that credentials are only good for short time periods, improving the threat model of dynamic environments. HashiCorp has built Vault for the cloud and container era, and includes the

ability to dynamically authenticate different trusted identity providers or containers, and service accounts to generate dynamic secrets on demand to meet the requirements of certain cloud and container services, with direct native integration to Kubernetes. These dynamic secrets are automatically revoked when the lease expires. Vault provides encryption as a service with centralized key management to simplify encrypting data across containers or systems, so secrets being used are encrypted during transit or at rest before any operation to write them to persistent storage.

Terraform

Terraform is aimed at operations; it provides a workflow engine that can be used to codify and provision infrastructure, making infrastructure-as-code a reality. Terraform is designed to manage the infrastructure at the level the customer wants to operate at; it can go down to compute and storage, or to higher levels such as SaaS. Terraform supports two basic entry formats; both are text descriptors of the infrastructure. The native format is HashiCorp Configuration Language (HCL), which is the domain specific language (DSL) used with a .tf (Terraform) file. It allows comments, and is designed to be easier to understand with a more human-readable approach. Terraform also supports a JSON format, which is designed for machine-generated descriptions. To use Terraform Enterprise, a user must have a Terraform Enterprise account and have set up an organization. Next, access to the organization's version-control service must be enabled. Finally, a workspace must be created. In Terraform, the workspace is where the infrastructure is organized. A typical workspace consists of a collection of Terraform configurations (these are retrieved from a version-control repository), the value of any variables the configurations require, persistent stored state for the resources it manages, and historical state and run logs. Terraform then runs the workspace; this enforces Terraform's division between plan and apply operations. Terraform will always execute a "plan" first; the output is saved, then used as the input for the "apply" stage. In the default configuration, user approval is required after the "plan" before running an "apply." However, this can be configured to automatically apply successful plans.

Consul

Consul is a service mesh to connect, secure, and configure services as a dedicated infrastructure layer. Consul has a distributed server-client architecture, where clients are used to collect location and health information on services and the nodes they are connected to. Clients also integrate with sidecar proxies of services to establish secure communication between services.

Red Hat OpenShift

Ovum view

Red Hat with OpenShift, which is based on Kubernetes orchestration, has taken a balanced perspective on the needs of its customers when it comes to the requirements from any container management platform. Red Hat offer customers a choice of how they wish to engage with and use OpenShift, Red Hat OpenShift Container Platform, Red Hat OpenShift Dedicated, and Red Hat OpenShift Online. The platform that underpins these is the same, with consistency of experience a key design theme. Coming from an open source heritage, Red Hat understands that by making its APIs open and extensible, a wider ecosystem can evolve. Red Hat has also understood that many organizations are moving existing workloads to a containerized environment as part of the application modernization approach. This acceptance that containers are not just for new cloud-native workloads means Red Hat understands that some workloads will be stateful and others stateless, but they need

to be managed in the same way. Red Hat is also leading the open source KubeVirt project for container-native virtualization (CNV). This enables organizations to move existing legacy workloads into the cloud. Most organizations today opt to encapsulate a workload running on top of a virtual machine to lift it into the cloud, but with KubeVirt it's easier to lift both the workload and the virtual machine platform on which it runs into the cloud using containers.

Solution overview

The three main solution offerings from Red Hat enable customers to select the approach that best matches their experience. Ovum believes that as organizations become familiar with the technology, these services will probably change to reflect the maturity of the market. However, today the market is nascent and as such most organizations are looking for pathways to adoption of containers at scale. The starting point is typically developers that want access to environments to develop and test new solutions, which then need to be made production-ready and run at scale. These will then need to be extended so that they become integrated with other enterprise services.

Red Hat OpenShift Container Platform

This offering enables an organization to build its own clusters and manage these based on its own requirements. Customers can opt to use current storage and networking infrastructure, which is supported through comprehensive APIs with the leading vendors and is aimed predominantly at building an on-premises capability. It can, however, be run in any environment that supports RHEL on-premises or in public cloud such as AWS, Azure, GCP, or Alibaba. This offering is aimed at organizations that have experience in both development and management of containers. These organizations typically have the skills needed for the technical management of Kubernetes and the other associated infrastructure elements, such as storage, network, and monitoring. In this offering the customer has full control of the platform.

Red Hat OpenShift Dedicated

If the customer has no or very little experience with Kubernetes and containers in general, the Red Hat OpenShift Dedicated offering is the most suitable. This approach provides a single consistent way to implement and use the platform, with Red Hat assuming responsibility for maintaining the platform. Red Hat OpenShift Dedicated is available on AWS and GCP (the Dedicated offering on Google Cloud is being revamped and will be available in early 2019) as a single-tenant solution running on shared infrastructure instances, with managed OpenShift on Azure was announced at RH Summit 2018. The big advantage of the Dedicated offering is that it enables the customer to focus on application lifecycle management, leaving Red Hat to manage the Kubernetes container orchestration layer and the RHEL host layer. The base package consists of an HA cluster with a minimum of five master and infrastructure nodes, four application nodes that consist of 100GB of SSD persistent storage, 4 vCPUs, 16 GB RAM, and 12TB/yr per application node network I/O. The other major advantage with the Dedicated offering is that it comes with a 24/7 comprehensive support agreement, where Red Hat assumes responsibility for the entire platform.

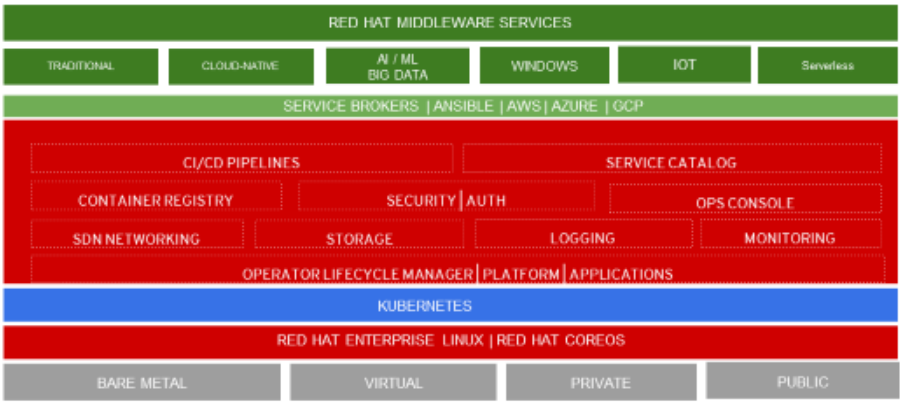
Red Hat OpenShift Online

The multitenant hosted public cloud solution of Red Hat OpenShift Online is aimed at individual developers working on container-based projects who require access to an environment that will enable the development, testing, and execution of these workloads. This is a fully managed PaaS offering where the developer can access templates of application images, database images, middleware images, and other services in a self-service approach. The platform supports a range of

the most popular development languages, including Java, Node.js, .NET, Ruby, Python, and PHP,and makes it simple for developers to "code and push". For some organizations, the Online offering will be used to develop and run workloads, but for most it is more of a starting position and many will move to the Dedicated offering.

Architecture

Figure 1: OpenShift architecture



Source: Red Hat

Figure 1 shows the architecture of Red Hat OpenShift, which for consistency only runs the Kubernetes orchestrator that enables Red Hat to ensure that the different components, such as storage and networks, can be treated consistently on the platform. Ovum is particularly supportive of the Operator Framework, which is an approach to open the OpenShift platform to other applications such as middleware and databases that will allow these applications to be consumed and managed natively through OpenShift. The other significant feature of OpenShift's architecture is the concept of the service broker. This is a very important feature because this is how organizations can use OpenShift in a typical hybrid/mixed environment where legacy workloads can communicate with new container-based workloads, all managed in a single platform. The architecture enables, through the Custom Resource Definitions concept, an extensible API that will allow organizations and the wider ecosystem to develop solutions that are industry/market-specific, without turning the OpenShift core in to a large and complex organism.

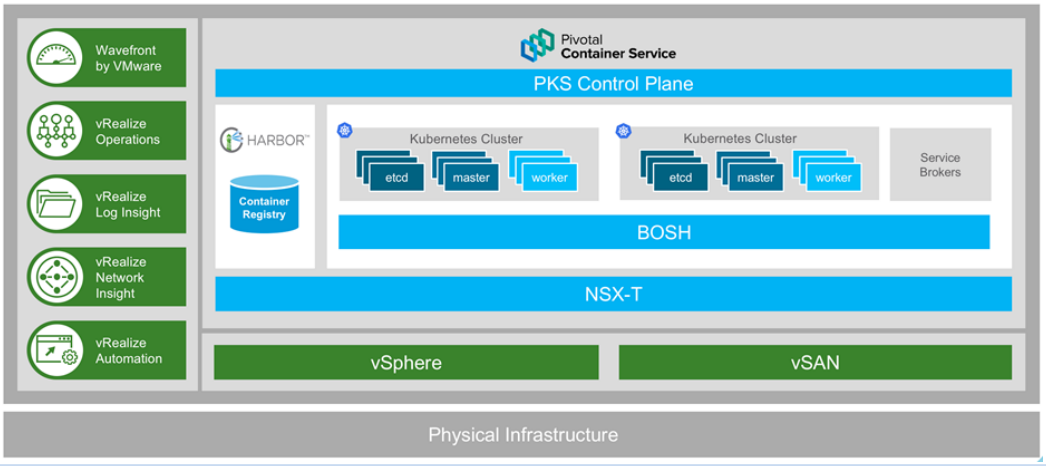
VMware PKS

Ovum view

VMware PKS is a self-managed solution designed for both on-premises and public cloud deployment, see Figure 2. The initial release offers support for vSphere and GCP, with AWS supported in VMware PKS 1.2, and Azure support on the roadmap. The key value proposition of VMware PKS is that it provides a simplified experience for deploying and managing Kubernetes on vSphere and a known and familiar management experience (using VMware products such as VRA, VROps, VRLi, and Wavefront) from making use of existing investments in VMware solutions. However, VMware PKS is

not limited to VMware environments and tooling. Most organizations are running VMware, so by supporting it, VMware PKS can provide a flexible and cost-effective way for organizations to adopt cloud-native.

Figure 2: VMware PKS



Source: Pivotal

Solution overview

Pivotal has invested significant time and effort in developing the open source BOSH infrastructure orchestration layer (it is a key contributor) that controls and manages the Kubernetes application orchestration layer above it. Kubernetes is considered by many to be a single orchestration layer for container workloads that span the application and infrastructure layer, but Kubernetes does not deliver all the infrastructure orchestration needed. For example, Kubernetes will move application workloads from failing nodes and ensure the service is maintained, but it will not restart the node that failed. BOSH was therefore originally developed at VMware for use with Cloud Foundry and was subsequently open-sourced. Pivotal customers have been working with BOSH for more than five years so that the underlying infrastructure can be managed effectively, leaving Kubernetes to manage the application layer. The BOSH layer enables VMware PKS to offer the automated repair, rotate, and repave approach to securely maintaining the infrastructure. BOSH provides a level of software-defined infrastructure automation driven by desired-state declarations. BOSH is synonymous with the rotate, repair, and repave concept to automated infrastructure security. Regularly repaving VMs to a known good state limits the time window for any bad actor to do permanent damage to a system. Rotating credentials minimizes the exposure of any leakage, and repairing OS images with the latest CVEs minimizes the opportunity for newly discovered exploits to be used. This automated approach can be used to good effect in rotating any infrastructure so that it is maintained at its optimal effectiveness from a security and compliance perspective.

Another element of the management of containers at scale is the approach taken to how the workloads are deployed and how the network is managed. VMware PKS uses the NSX-T (VMware PKS includes the license for NSX-T) from VMware, a software-defined networking approach to apply the policies required to ensure the containers behave as intended. NSX-T works by implementing three separate but integrated planes: management, control, and data. The three planes are implemented as a set of processes, modules, and agents residing on three types of nodes: manager, controller, and transport. Every node hosts a management plane agent, and the NSX manager node

hosts the API services. Each NSX-T installation can support a single NSX manager node, but this does not support an NSX manager cluster. The NSX controller nodes host the central control plane cluster daemons, while the NSX manager and NSX controller nodes may be co-hosted on the same physical serve, and the transport nodes host local control plane daemons and forwarding engines. Using NSX-T enables VMware PKS to segment at the pod level, where it can enforce different policies according to granular requirements, increasing the level of management control over workloads.

VMware PKS also includes Harbor, an open source image registry that originated from VMware and was later donated to CNCF. Harbor has a number of key features that some other registries do not offer.

First, it supports user account and authentication (UAA), which is an open source identity server project under the Cloud Foundry (CF) foundation. Harbor can share UAA authentication with both pivotal application service (PAS) and VMware PKS. Harbor also offers role-based access control and can segment the access based on a project approach, where a user can have different privileges based on the images in the different project folders. Ovum thinks this level of control enables organizations to use elevated privileges more precisely.

Second, the images can be synchronized between multiple registry instances with auto-retry on errors. This feature enables organizations to provide load balancing, high availability, multi-datacenter, hybrid, and multicloud capabilities.

Third, Harbor uses Clair technology to scan images regularly and compare these scans against six different common vulnerabilities and exposure (CVE) databases: Debian Sec Bug Tracker, Ubuntu CVE Tracker, Red Hat Security Data, Oracle Linux Sec Data, Alpine SecDB, and NIST NVD. Clair warns users of vulnerabilities it has discovered and provides remediation advice. Harbor also performs image signatures so that the provenance of images can be assured in the registry.

Finally, Harbor integrates with an existing enterprise LDAP/AD configuration for user authentication and management.

Emerging technologies related to container management platforms

In this section, emerging areas of container management platform technologies are highlighted. These technologies are not comprehensive container management platforms but are providing a new approach to dealing with some of the challenges containers are experiencing now that they are being deployed at scale.

Kata Containers

Summary

Kata Containers is an open source project that is designed to address the security concerns associated with containers (see Figure 3), namely the shared kernel technology, and how different workloads or environments can be isolated. Kata Containers combines technology from Intel Clear Containers and Hyper runV. The code is hosted on GitHub under the Apache 2 license and the project is managed by the OpenStack Foundation. Ovum believes that interest in Kata Containers will

accelerate as the ability to isolate container workloads successfully to the same degree that virtual machines (VMs) can be isolated will become an issue for some market segments.

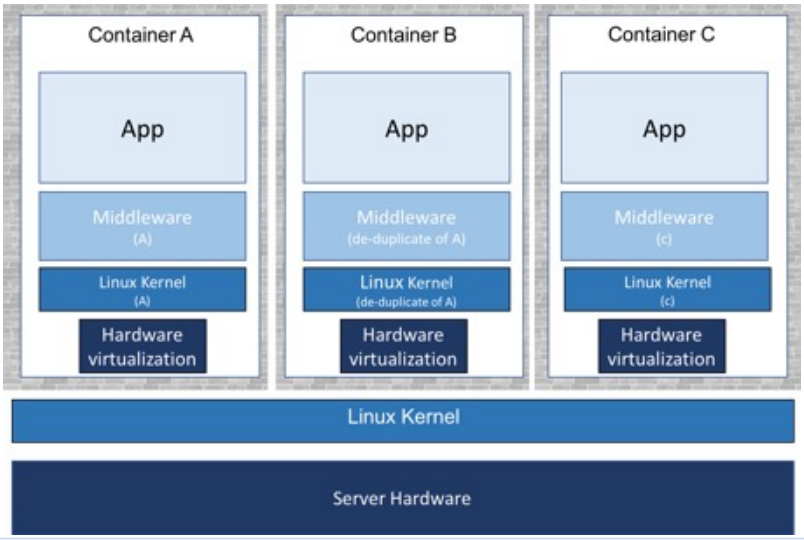
Solution overview

Kata Containers consists of six components: Agent, Runtime, Proxy, Shim, Kernel, and QEMU.

- **Agent:** The Agent interfaces with the Runtime, and runs inside the VM, where it supports the spawning of processes and containers.
- **Runtime:** Runtime is the command-line interface access to Kata Containers and manages the host environment. The runtime is OCI-compatible, allowing it to work seamlessly with both Docker and Kubernetes.
- **Proxy:** A Kata Proxy instance is launched for each VM to handle multiplexing and de-multiplexing commands and streams.
- **Shim:** The Shim manages the communications between the container and the Agent. It uses Google's Remote Protocol Call (gRPC) to make direct calls on the application, and does this on different machines, while still making it look like a local object. The Shim is needed because it is not possible to monitor the container process directly from the host operating system, so the Shim acts as the container process, and the container process reaper then monitors this.
- **Kernel:** The lightweight VM created by Kata Containers requires a guest operating system and a guest kernel to create and boot the container inside the guest operating system.
- **QEMU:** QEMU is a full virtualization open source solution for Linux.

The concept behind Kata Containers is to build an OCI-compliant, lightweight VM that operates and behaves like a container. Using this approach, Kata Containers can offer a higher level of workload isolation beyond using namespaces that current OCI-compliant containers use. Kata Containers provides this isolation at different levels depending on the technology being used. For example, for Docker environments the VM isolation is at the container level. However, for Kubernetes the VM isolation is provided at the pod level.

Figure 3: Kata containers architecture



Source: Kata containers

Scality MetalK8s

Summary

One of the biggest challenges for organizations with skills and expertise in container technology is in deploying a Kubernetes-based environment on-premises. The many different container management platforms are nearly all focused on cloud deployments, and include all the different management tooling needed. However, an on-premises deployment typically requires the customer to select, assemble, integrate, and install all the different tools required to make the Kubernetes cluster operational, particularly if they want to run it on bare metal. Scality noticed that there was a gap in the market for on-premises bare metal deployment options for a container management platform. Deploying Kubernetes on-premises is useful for stateful applications that require direct I/O to a local disk, something that the standard Kubernetes solutions find difficult to support. Kubernetes, by default, looks to use SAN or cloud-based storage, which increases the cost and decreases performance. Ovum accepts that the use of direct attached local storage is a specialist use case for containers, and that the performance and cost benefits are only noticeable for those stateful applications that organizations consider not to be good candidates to migrate to the cloud.

Solution overview

Scality has released MetalK8s under the Apache Software License v2.0 on GitHub, and it is based on the open source Kubespray "playbook" project. The base Kubernetes cluster is installed using Ansible, and Scality has made some definitive choices on the other projects that it would support. MetalK8s uses Calico for its container network interface (CNI) implementation, and the "ingress controller" is based on Nginx. The management tools for monitoring and metering the deployment have also been preselected by Scality, and include

- Prometheus, to handle the metering and monitoring of the cluster.
- Grafana, to provide dashboards of the different metrics and resource information.
- ElasticSearch, to collect all the log files in a cluster and make them available for reporting.
- Kibana, to enable operators to access the logs stored in ElasticSearch.
- Helm, to manage the cluster, with all the different projects being treated as Helm packages.

MetalK8s requires CentOS 7.4 as the operating environment for the cluster. With MetalK8s, Scality has created a solution that provides for bare metal on-premises Kubernetes clusters a degree of manageability similar to that afforded to the more mainstream solutions. The major difference is that MetalK8s changes the default storage to locally attached direct storage, not SAN or cloud storage.

Appendix

Methodology

- Detailed technical briefings were conducted with each vendor in the report.
- Supplemental information was obtained from vendor literature and websites, other Ovum surveys, and Ovum’s data products/market forecasts.
- The article was peer reviewed by at least two different analysts/consultants.

Further reading

Making the Transition from Managing VMs to Orchestrating Containers, INT003-000180 (June 2018)

Author

Roy Illsley, Distinguished Analyst, Infrastructure Solutions

roy.illsley@ovum.com

Ovum Consulting

We hope that this analysis will help you make informed and imaginative business decisions. If you have further requirements, Ovum's consulting team may be able to help you. For more information about Ovum's consulting capabilities, please contact us directly at consulting@ovum.com.

Copyright notice and disclaimer

Ovum
TMT intelligence | informa

CONTACT US

ovum.informa.com
askananalyst@ovum.com

INTERNATIONAL OFFICES

Beijing
Dubai
Hong Kong
Hyderabad
Johannesburg
London
Melbourne
New York
San Francisco
Sao Paulo
Tokyo