THENEWSTACK

USE CASES FOR KUBERNETES

The New Stack:
Use Cases for Kubernetes

Alex Williams, Founder & Editor-in-Chief
Benjamin Ball, Technical Editor & Producer
Gabriel Hoang Dinh, Creative Director
Lawrence Hecht, Data Research Director

Contributors:
Judy Williams, Copy Editor
Norris Deajon, Audio Engineer

TABLE OF CONTENTS

USE CASES FOR KUBERNETES

KUBERNETES SOLUTIONS DIRECTORY

SPONSOR

We are grateful for the support of Intel.

ABOUT THE AUTHOR

Janakiram MSV is the Principal Analyst at Janakiram & Associates and an adjunct faculty member at the International Institute of Information Technology. He is also a Google Qualified Cloud Developer, an Amazon Certified Solution Architect, an Amazon Certified Developer, an Amazon Certified SysOps Administrator, and a Microsoft Certified Azure Professional. His previous experience includes Microsoft, AWS, Gigaom Research, and Alcatel-Lucent.

OVERVIEW OF THE KUBERNETES PLATFORM

by **JANAKIRAM MSV**

Kubernetes is a container management platform designed to run enterprise-class, cloud-enabled and web-scalable IT workloads. It is built upon the foundation laid by Google based on 15 years of experience in running containerized applications. The objective of this ebook is to highlight how Kubernetes is being deployed by early adopters. It touches upon the usage patterns and key deployment scenarios of customers using Kubernetes in production. We'll also take a look at companies, such as Huawei, IBM, Intel and Red Hat, working to push Kubernetes forward.

The Rise of Container Orchestration

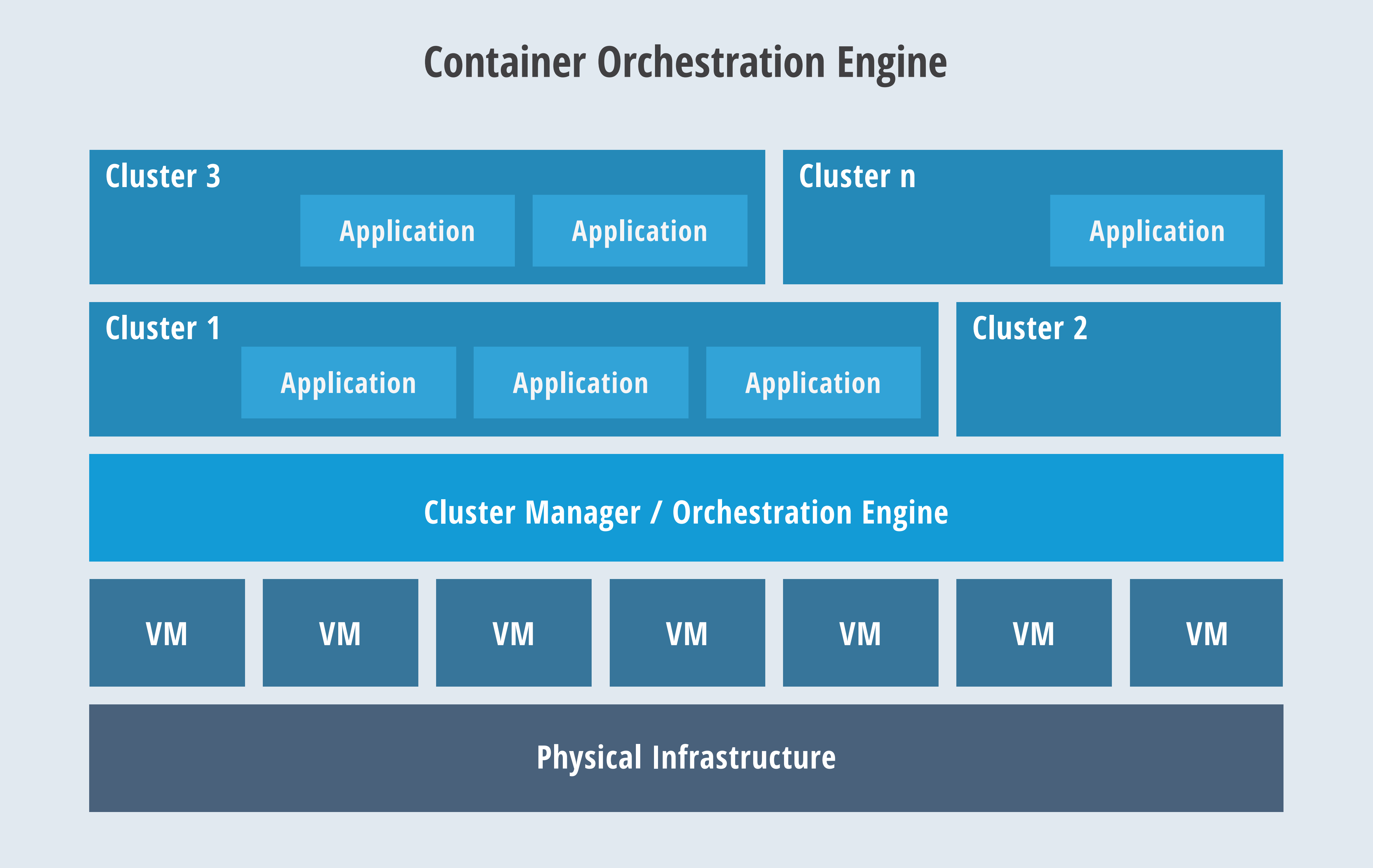
The concept of containers has existed for over a decade. Mainstream Unix-based operating systems (OS), such as Solaris, FreeBSD and Linux, had built-in support for containers, but it was Docker that truly democratized containers by making them manageable and accessible to both the development and IT operations teams. Docker has demonstrated that containerization can drive the scalability and portability of applications.

Developers and IT operations are turning to containers for packaging code and dependencies written in a variety of languages. Containers are also playing a crucial role in DevOps processes. They have become an integral part of build automation and continuous integration and continuous deployment (CI/CD) pipelines.

The interest in containers led to the formation of the Open Container Initiative (OCI) to define the standards of container runtime and image formats. The industry is also witnessing various implementations of containers, such as LXD by Canonical, rkt by CoreOS, Windows Containers by Microsoft, CRI-O — being reviewed through the Kubernetes Incubator, and vSphere Integrated Containers by VMware.

While core implementations center around the life cycle of individual containers, production applications typically deal with workloads that

FIG 1: *High-level architecture of a container orchestration engine.*



Source: Janakiram MSV

have dozens of containers running across multiple hosts. The complex architecture dealing with multiple hosts and containers running in production environments demands a new set of management tools. Some of the popular solutions include Docker Datacenter, Kubernetes, and Mesosphere DC/OS.

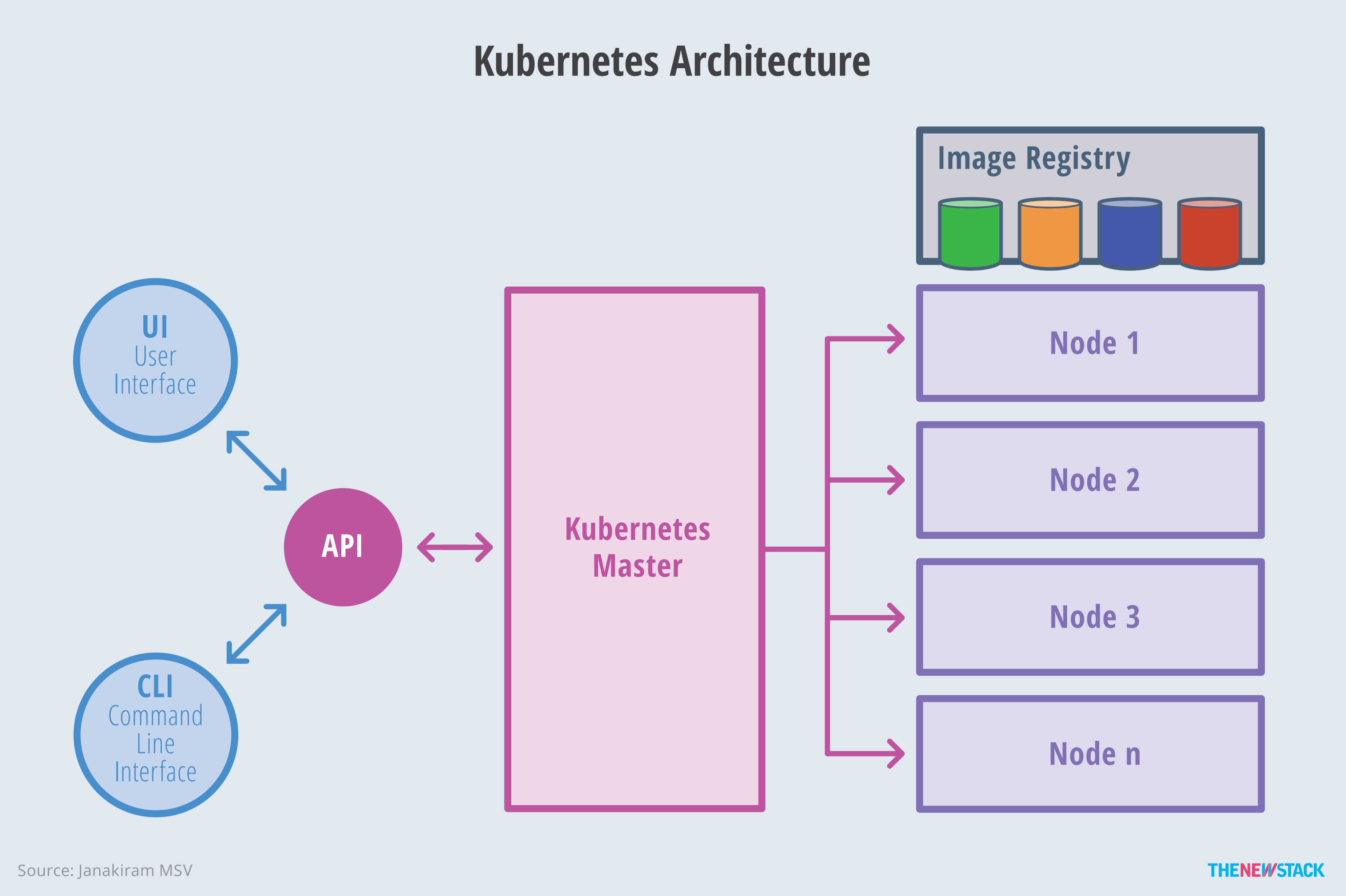
Container orchestration has influenced traditional Platform as a Service (PaaS) architecture by providing an open and efficient model for packaging, deployment, isolation, service discovery, scaling and rolling upgrades. Most mainstream PaaS solutions have embraced containers, and there are new PaaS implementations that are built on top of container orchestration and management platforms. Customers have the choice of either deploying core container orchestration tools that are more aligned with IT operations, or utilizing a PaaS implementation that targets developers.

The key takeaway is that container orchestration has impacted every aspect of modern software development and deployment. Kubernetes will play a crucial role in driving the adoption of containers in both enterprises and emerging startups.

Kubernetes Architecture

Like most distributed computing platforms, a Kubernetes cluster consists of at least one master and multiple compute nodes. The master is responsible for exposing the application program interface (API), scheduling the deployments and managing the overall cluster.

Each node runs a container runtime, such as Docker or rkt, along with an agent that communicates with the master. The node also runs additional components for logging, monitoring, service discovery and optional add-ons. Nodes are the workhorses of a Kubernetes cluster. They expose



Source: Janakiram MSV

FIG 2: *Kubernetes breaks down into multiple architectural components.*

compute, networking and storage resources to applications. Nodes can be virtual machines (VMs) in a cloud or bare metal servers in a datacenter.

A pod is a collection of one or more containers. The pod serves as Kubernetes' core unit of management. Pods act as the logical boundary for containers sharing the same context and resources. The grouping mechanism of pods make up for the differences between containerization and virtualization by making it possible to run multiple dependent processes together. At runtime, pods can be scaled by creating replica sets, which ensure that the deployment always runs the desired number of pods.

Replica sets deliver the required scale and availability by maintaining a pre-defined set of pods at all times. A single pod or a replica set can be exposed to the internal or external consumers via services. Services

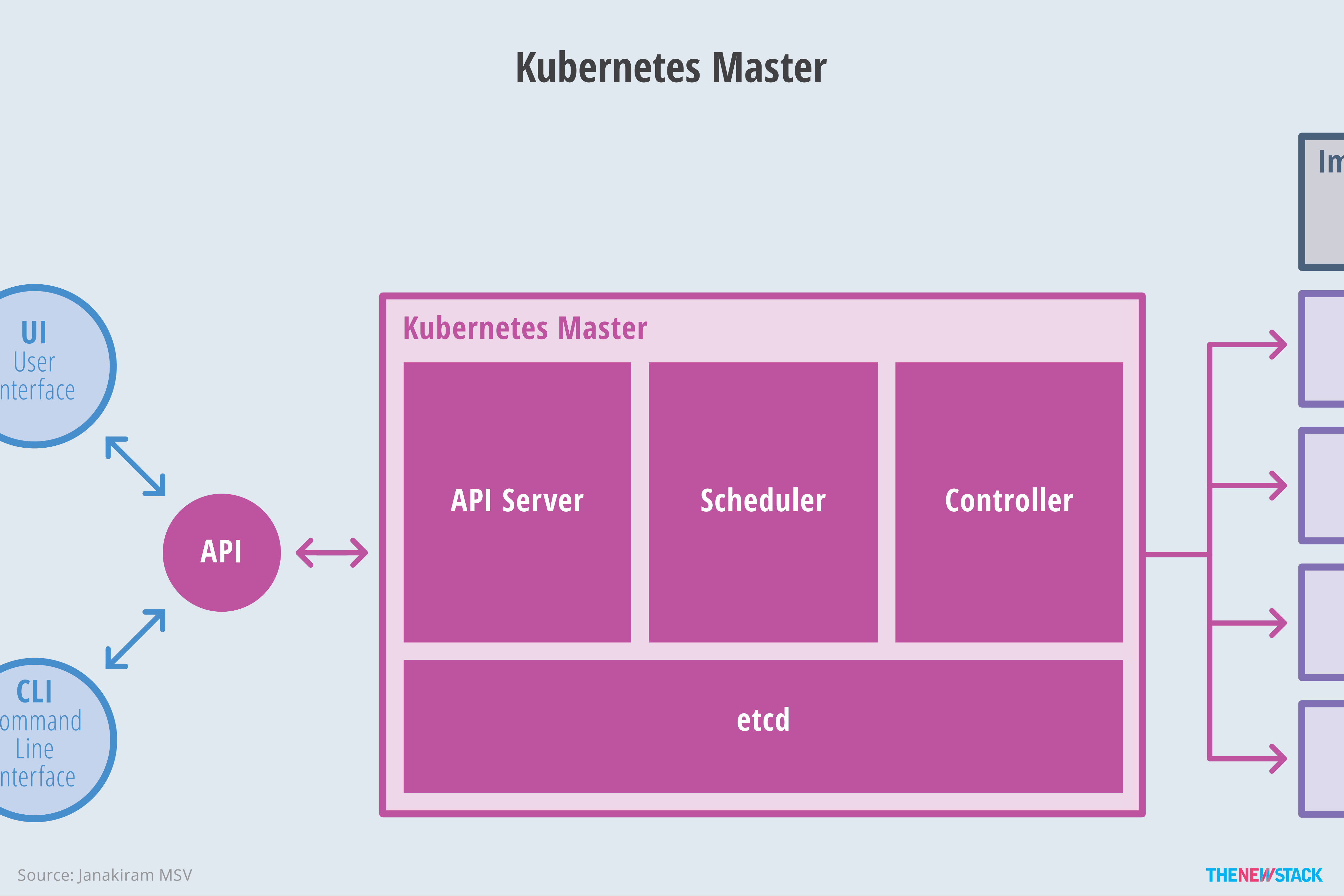
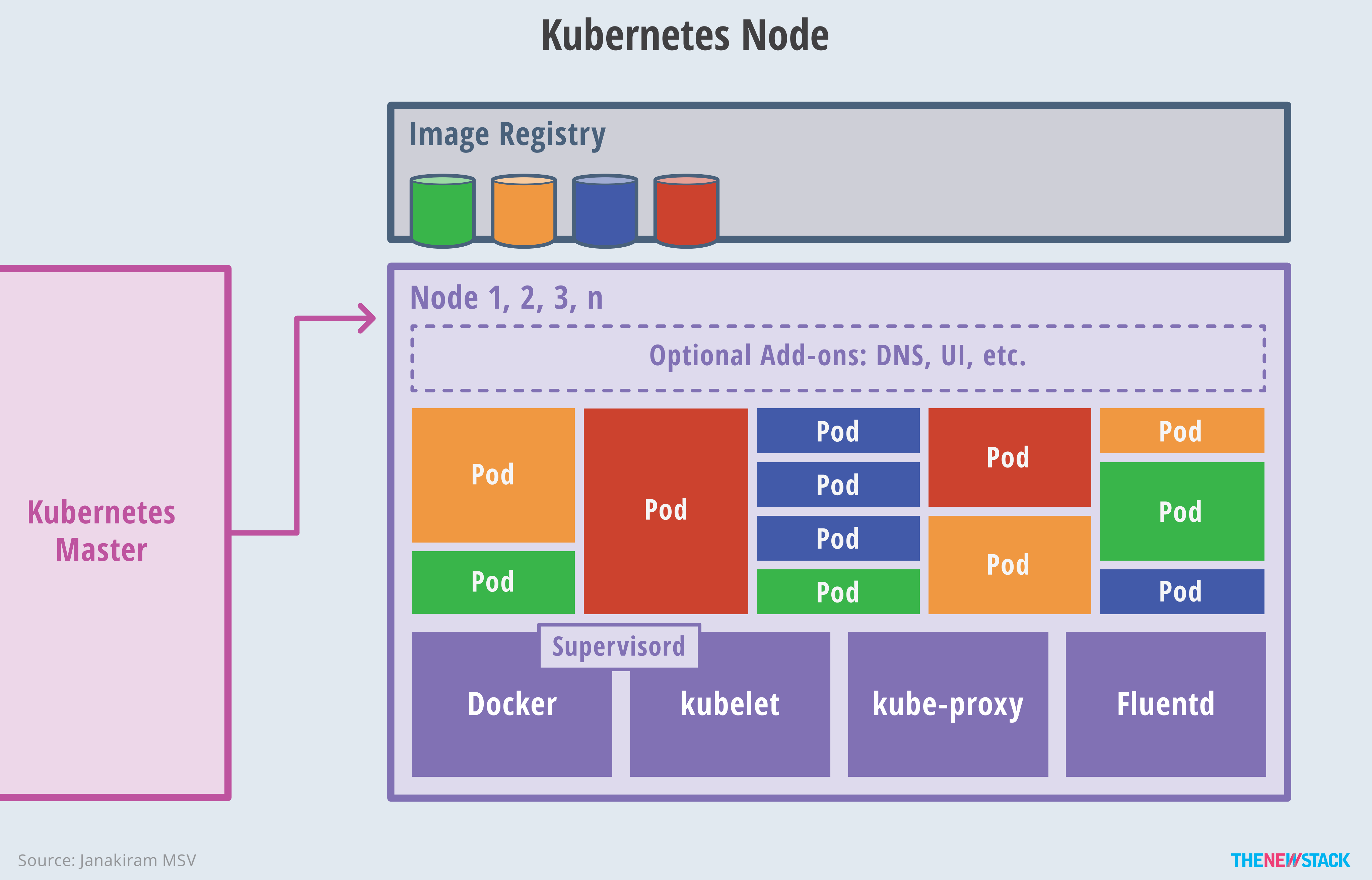


FIG 3: *The master is responsible for exposing the API, scheduling the deployments and managing the overall cluster.*

enable the discovery of pods by associating a set of pods to a specific criterion. Pods are associated to services through key-value pairs called labels and selectors. Any new pod with labels that match the selector will automatically be discovered by the service.

The definition of Kubernetes objects, such as pods, replica sets and services, are submitted to the master. Based on the defined requirements and availability of resources, the master schedules the pod on a specific node. The node pulls the images from the container image registry and coordinates with the local container runtime to launch the container.

etcd is an open source, distributed key-value database from CoreOS, which acts as the single source of truth (SSOT) for all components of the Kubernetes cluster. The master queries etcd to retrieve various



Source: Janakiram MSV

FIG 4: *Nodes expose compute, networking and storage resources to applications.*

parameters of the state of the nodes, pods and containers. This architecture of Kubernetes makes it modular and scalable by creating an abstraction between the applications and the underlying infrastructure.

Key Design Principles

Kubernetes is designed on the principles of scalability, availability, security and portability. It optimizes the cost of infrastructure by efficiently distributing the workload across available resources. This section will highlight some of the key attributes of Kubernetes.

Workload Scalability

Applications deployed in Kubernetes are packaged as microservices. These microservices are composed of multiple containers grouped as pods. Each container is designed to perform only one task. Pods can be composed of stateless containers or stateful containers. Stateless pods can easily be scaled on-demand or through dynamic auto-scaling. Kubernetes 1.4 supports horizontal pod auto-scaling, which automatically scales the number of pods in a replication controller based on CPU utilization. Future versions will support custom metrics for defining the auto-scale rules and thresholds.

Hosted Kubernetes running on Google Cloud also supports cluster auto-scaling. When pods are scaled across all available nodes, Kubernetes coordinates with the underlying infrastructure to add additional nodes to the cluster.

An application that is architected on microservices, packaged as containers and deployed as pods can take advantage of the extreme scaling capabilities of Kubernetes. Though this is mostly applicable to stateless pods, Kubernetes is adding support for persistent workloads, such as NoSQL databases and relational database management systems (RDBMS), through pet sets; this will enable scaling stateless applications such as Cassandra clusters and MongoDB replica sets. This capability will bring elastic, stateless web tiers and persistent, stateful databases together to run on the same infrastructure.

High Availability

Contemporary workloads demand availability at both the infrastructure and application levels. In clusters at scale, everything is prone to failure, which makes high availability for production workloads strictly necessary. While most container orchestration engines and PaaS offerings deliver application availability, Kubernetes is designed to tackle the availability of both infrastructure and applications.

On the application front, Kubernetes ensures high availability by means of replica sets, replication controllers and pet sets. Operators can declare the minimum number of pods that need to run at any given point of time. If a container or pod crashes due to an error, the declarative policy can bring back the deployment to the desired configuration. Stateful workloads can be configured for high availability through pet sets.

For infrastructure availability, Kubernetes has support for a wide range of storage backends, coming from distributed file systems such as network file system (NFS) and GlusterFS, block storage devices such as Amazon Elastic Block Store (EBS) and Google Compute Engine persistent disk, and specialized container storage plugins such as Flocker. Adding a reliable, available storage layer to Kubernetes ensures high availability of stateful workloads.

Each component of a Kubernetes cluster — etcd, API server, nodes – can be configured for high availability. Applications can take advantage of load balancers and health checks to ensure availability.

Security

Security in Kubernetes is configured at multiple levels. The API endpoints are secured through transport layer security (TLS), which ensures the user is authenticated using the most secure mechanism available. Kubernetes clusters have two categories of users — service accounts managed directly by Kubernetes, and normal users assumed to be managed by an independent service. Service accounts managed by the Kubernetes API are created automatically by the API server. Every operation that manages a process running within the cluster must be initiated by an authenticated user; this mechanism ensures the security of the cluster.

Applications deployed within a Kubernetes cluster can leverage the concept of secrets to securely access data. A secret is a Kubernetes object that contains a small amount of sensitive data, such as a password, token or key, which reduces the risk of accidental exposure of data. Usernames and passwords are encoded in base64 before storing them within a Kubernetes cluster. Pods can access the secret at runtime through the mounted volumes or environment variables. The caveat is that the secret is available to all the users of the same cluster namespace.

To allow or restrict network traffic to pods, network policies can be applied to the deployment. A network policy in Kubernetes is a specification of how selections of pods are allowed to communicate with each other and with other network endpoints. This is useful to obscure pods in a multi-tier deployment that shouldn't be exposed to other applications.

Portability

Kubernetes is designed to offer freedom of choice when choosing operating systems, container runtimes, processor architectures, cloud platforms and PaaS.

A Kubernetes cluster can be configured on mainstream Linux distributions, including CentOS, CoreOS, Debian, Fedora, Red Hat Linux and Ubuntu. It can be deployed to run on local development machines; cloud platforms such as AWS, Azure and Google Cloud; virtualization environments based on KVM, vSphere and libvirt; and bare metal. Users can launch containers that run on Docker or rkt runtimes, and new container runtimes can be accommodated in the future.

Through federation, it's also possible to mix and match clusters running across multiple cloud providers and on-premises. This brings the hybrid cloud capabilities to containerized workloads. Customers can seamlessly move workloads from one deployment target to the other. We will discuss the hybrid architecture in the next section.

DEPLOYMENT TARGETS IN THE ENTERPRISE

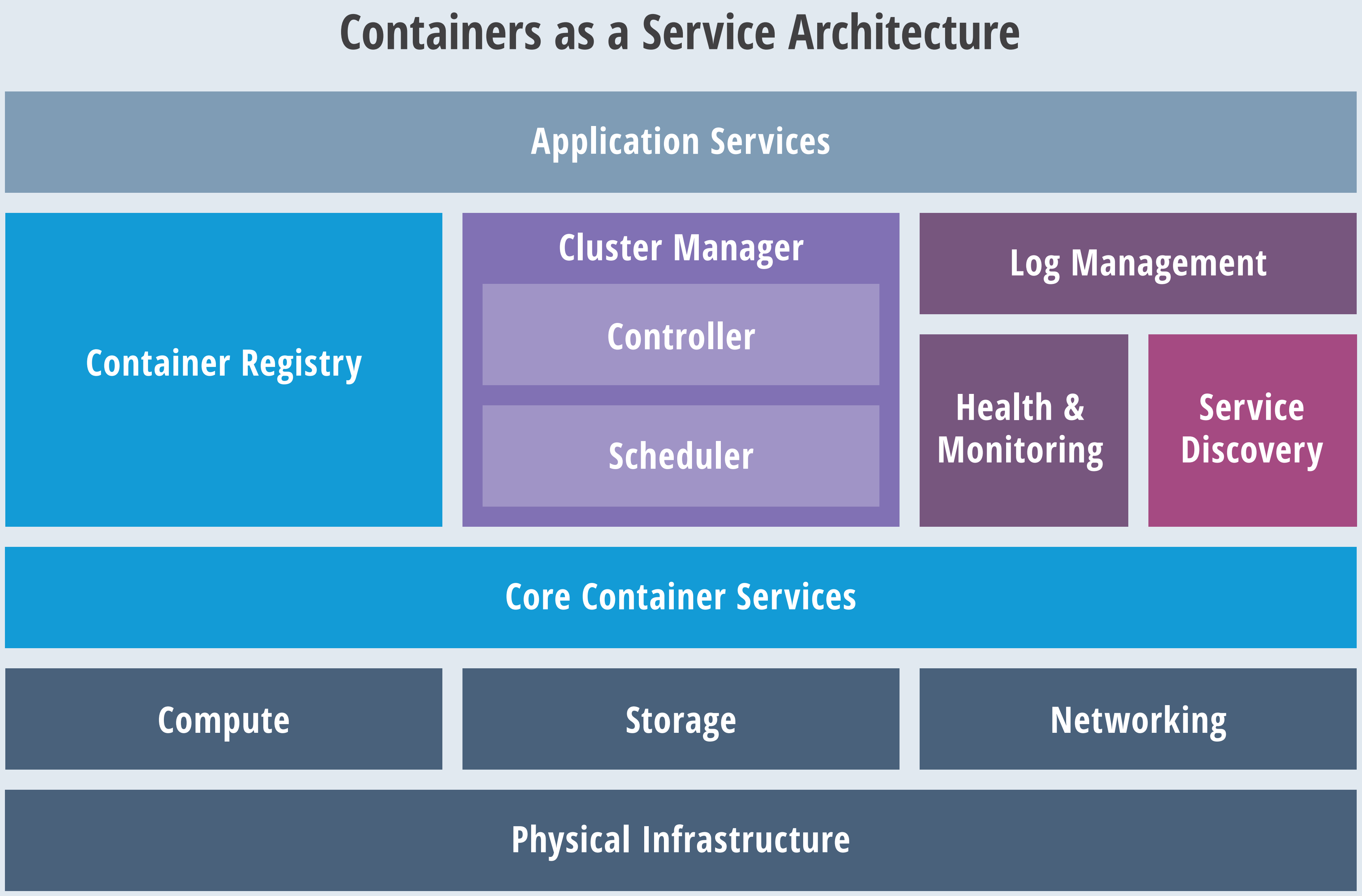
by **JANAKIRAM MSV**

Kubernetes may be deployed in a variety of environments utilizing different deployment patterns. As more teams and organizations are using, it's important to identify the implementations that they are most likely to adopt. There are several deployment models for running production workloads that vendors have already targeted with a suite of products and services. This section highlights these models, including:

- Managed Kubernetes and Containers as a Service.
- Public Cloud and Infrastructure as a Service.
- On-premises and data centers.
- Hybrid deployments.

Managed Kubernetes and Containers as a Service

A managed Kubernetes cluster is hosted, maintained and managed by a commercial vendor. Based on PaaS concepts, this delivery model is called



Source: Janakiram MSV

FIG 1: *The components that make up the architecture of a CaaS solution.*

Containers as a Service (CaaS). It offers service-level agreement (SLA)-driven high availability and scalability of infrastructure. CaaS goes beyond exposing the basic cluster to users. Google Container Engine (GKE), Amazon EC2 Container Service, Azure Container Service and Rackspace Carina are some examples of CaaS in the public cloud. Often, it comes with other essential services such as image registry, L4 and L7 load balancers, persistent block storage, health monitoring, managed databases, integrated logging and monitoring, auto scaling of pods and nodes, and support for end-to-end application lifecycle management.

Apart from CaaS, Kubernetes is also offered as a service via the public cloud. Many customers get started with Kubernetes through Google Container Engine, as it was one of the first commercially managed Kubernetes offerings in the public cloud. Managed and maintained by Google, GKE delivers automated container management and integration

Container Management Solutions

Cloud Container Engine (Huawei) A scalable, high-performance container service based on Kubernetes.

CloudStack Container Service (ShapeBlue) A Container as a Service solution that combines the power of Apache CloudStack and Kubernetes. It uses Kubernetes to provide the underlying platform for automating deployment, scaling and operation of application containers across clusters of hosts in the service provider environment.

Google Container Engine (Google) Google Container Engine is a cluster management and orchestration system that lets users run containers on the Google Cloud Platform.

Kubernetes as a Service on Photon Platform (VMware) Photon is an open source platform that runs on top of VMware's NSX, ESXi and Virtual SAN. The Kubernetes as a Service feature will be available at the end of 2016.

with other Google Cloud services such as logging, monitoring, container registry, persistent disks, load balancing and VPN.

GKE only exposes the nodes of the Kubernetes cluster to customers, while managing the master and etcd database itself. This allows users to focus on the applications and avoid the burden of infrastructure maintenance. With GKE, scaling out a cluster by adding new nodes can be done within minutes. GKE can also upgrade the cluster to the latest version of Kubernetes, ensuring that the infrastructure is up-to-date. Customers can point tools such as kubectl to an existing GKE cluster to manage application deployments.

Other hosted Kubernetes offerings include AppsCode, KCluster and StackPointCloud. AppsCode delivers complete application lifecycle management of Kubernetes applications, while providing the choice of running the cluster on AWS or Google Cloud. KCluster is one of the emerging players in the hosted Kubernetes space. It runs on AWS, with other cloud platform support planned for the future. StackPointCloud focuses on rapid provisioning of clusters on AWS, Packet, DigitalOcean, Azure and Google Cloud. It is built on open source tools including Tectonic, Prometheus, Deis, fabric8 and Sysdig.

Hosted Kubernetes and PaaS Solutions

AppsCode (AppsCode) Integrated platform for collaborative coding, testing and deploying of containerized apps. Support is provided for deploying containers to AWS and Google Cloud Platform.

Deis Workflow (Engine Yard) A Kubernetes-native PaaS focused on developer self-service and operational flexibility. Deis Workflow helps teams quickly get up and running with Kubernetes on any public cloud, private cloud or bare metal cluster.

Eldarion Cloud (Eldarion) DevOps services and development consulting, packaged with a PaaS powered by Kubernetes, CoreOS and Docker. It includes Kel, a layer of open source tools and components for managing web application deployment and hosting.

Giant Swarm (Giant Swarm) A hosted container solution to build, deploy and manage containerized services with Kubernetes as a core component. It offers customers fully-managed private Kubernetes clusters — including management of master and nodes. It is offered "as a service" or can be deployed and managed on-premises by Giant Swarm.

Hasura Platform (34 Cross Systems) A platform for creating and deploying microservices. This emerging company's infrastructure is built using Docker and Kubernetes.

Hypernetes (HyperHQ) A multi-tenant Kubernetes distribution. It combines the orchestration power of Kubernetes and the runtime isolation of Hyper to build a secure multi-tenant CaaS platform.

KCluster (KCluster) A hosted Kubernetes service that assists with automatic deployment of highly available and scalable production-ready Kubernetes clusters. It also hosts the Kubernetes master components.

OpenShift Container Platform (Red Hat) A container application platform that can span across multiple infrastructure footprints. It is built using Docker and Kubernetes technology.

OpenShift Online (Red Hat) Red Hat's hosted version of OpenShift, a container application platform that can span across multiple infrastructure footprints. It is built using Docker and Kubernetes technology.

Platform9 Managed Kubernetes for Docker (Platform9) Kubernetes offered as a managed service. Customers can utilize Platform9's single pane of glass, allowing users to orchestrate and manage containers alongside virtual machines. In other words, you can orchestrate VMs using OpenStack and/or Kubernetes.

StackPointCloud (StackPointCloud) Allows users to easily create, scale and manage Kubernetes clusters of any size with the cloud provider of their choice. Its goal is to be a universal control plane for Kubernetes clouds.

Public Cloud and Infrastructure as a Service

Apart from signing up for a managed Kubernetes CaaS running in the public cloud, customers can also deploy and configure the cluster themselves. They can choose a specific version of Kubernetes distribution and integrate it with the native features of Infrastructure as a Service (IaaS).

CoreOS is one of the most preferred operating systems used to run Kubernetes in the public cloud. The company has comprehensive documentation and step-by-step guides for deploying Kubernetes in a variety of environments.

With each release, Kubernetes is becoming easier to install. In version 1.4, a new tool called kubeadm attempts to simplify the installation on machines running CentOS and Ubuntu. Customers can get a fully functional cluster running in just four steps.

On-Premises and Data Centers

Enterprise customers often prefer to run Kubernetes clusters inside a data center. This gives them the choice of using either virtualized infrastructure or bare metal servers to deliver better performance. When deploying Kubernetes behind a firewall, it is also important to consider the need and requirements for image registry, image scanning, monitoring and logging components.

There are commercial enterprise offerings that provide these necessary components, available in two different forms: commercial distributions and enterprise PaaS based on Kubernetes. Tectonic from CoreOS and the Canonical Distribution of Kubernetes are examples of commercial distributions that are customized for enterprise deployments. Red Hat OpenShift and Apprenda are application platforms built on top of Kubernetes. They provide end-to-end application lifecycle management capabilities for cloud-native applications.

Hybrid Deployments

A hybrid deployment is an evolving use case with customers utilizing a Kubernetes deployment that spans the on-premises datacenter and public cloud. It leverages the virtual networking layer provided by the cloud vendor and the concept of federated clusters in Kubernetes.

Kubernetes cluster federation can integrate clusters running across multiple zones and regions, or even clusters that are deployed on different cloud platforms. Federation creates a mechanism for multi-cluster geographical replication, which keeps the most critical services running even in the face of regional connectivity or data center failures.

The hybrid architecture, based on federated clusters, enables organizations to run sensitive workloads within the datacenter while moving the internet-facing, elastic workloads to the public cloud. With federation, customers are able to efficiently manage their deployments across multiple deployment targets, and save on operating costs by dynamically optimizing deployments across multiple providers and regions.

KUBERNETES IN A MULTI-CLOUD WORLD

In a discussion with Jonathan Donaldson of Intel, we talk about Intel's motivations for taking an active and invested role in the Kubernetes ecosystem. Kubernetes' capabilities as a platform for automating deployments and orchestrating applications is a game-changer for both multi-cloud service providers and customers. Kubernetes is also able to combine with solutions such as OpenStack to address the need for health monitoring, making services highly available, and scaling services as needed. This kind of flexibility helps solve many of the operational concerns with maintaining infrastructure. Overall, Intel sees Kubernetes as a way to expand the flexibility of multi-cloud environments, which makes it a technology worth investing in and encouraging further adoption by end users. **Listen on SoundCloud**

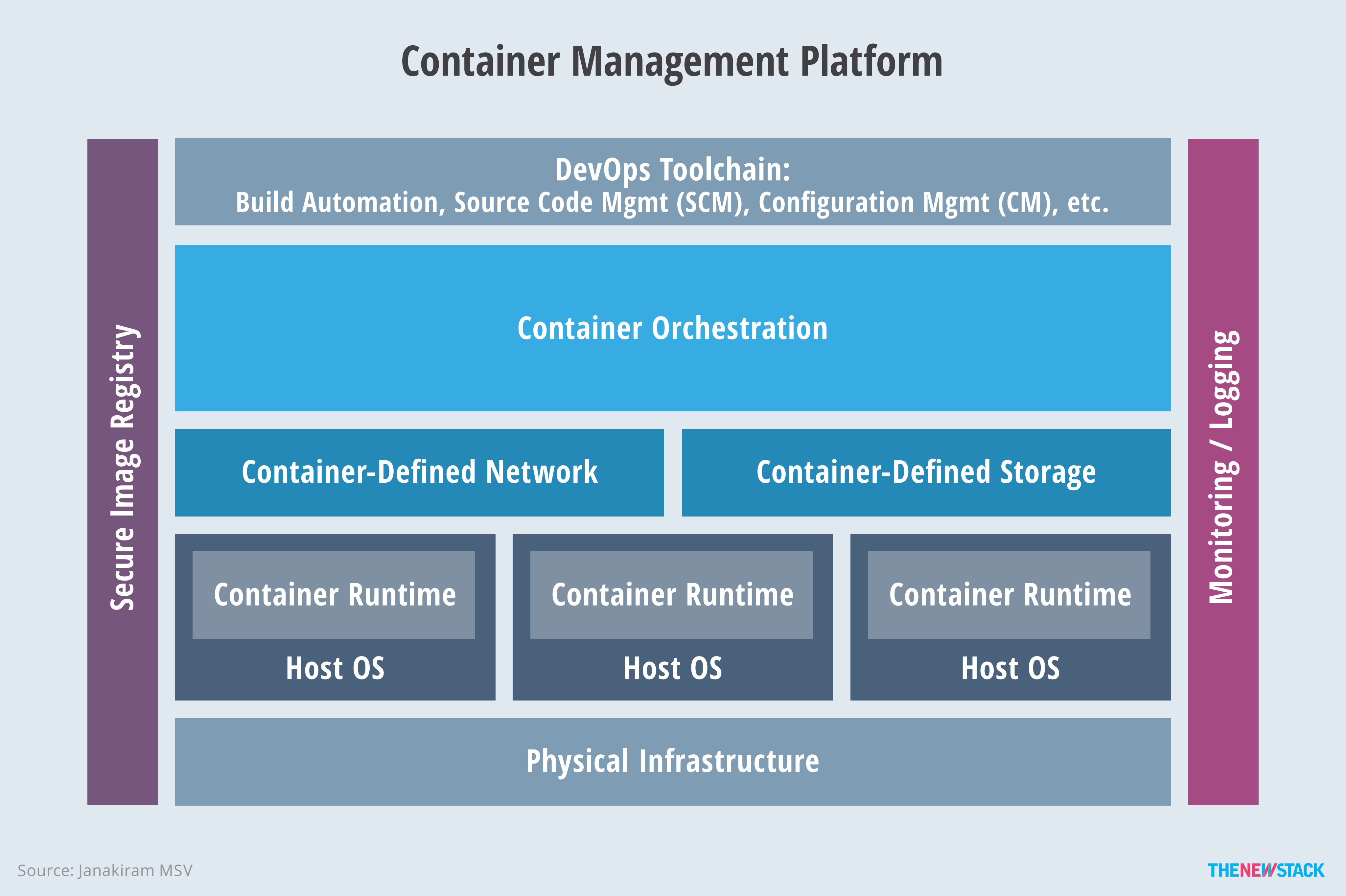
Jonathan Donaldson *is the vice president of the Data Center Group and general manager of the Software Defined Infrastructure Group at Intel Corporation. He leads the team responsible for defining and carrying out Intel's strategy for private, hybrid and public cloud automation. Donaldson previously worked at companies such as VCE, EMC, NetApp and Cisco, holding various leadership and technical roles that encompassed enterprise solution engineering, internal startups, field sales, federal programs, marketing and emerging technologies.*

KEY DEPLOYMENT SCENARIOS

by **JANAKIRAM MSV**

Kubernetes is deployed in production environments as a container orchestration engine, PaaS, and as core infrastructure for managing cloud-native applications. These use cases are not mutually exclusive. It is possible for DevOps to delegate complete application lifecycle management (ALM) to a PaaS layer based on Kubernetes. They may also use a standalone Kubernetes deployment to manage applications deployed using the existing CI/CD toolchain. Customers building greenfield applications can leverage Kubernetes for managing the new breed of microservices-based cloud-native applications through advanced scenarios such as rolling upgrades and canary deployments.

This section looks to capture the top customer use cases involving Kubernetes. Before highlighting the key deployment scenarios of Kubernetes, let's take a closer look at the essential components of an enterprise container management platform.



Source: Janakiram MSV

FIG 1: *A closer look at the essential components of an enterprise container management platform.*

The Building Blocks of an Enterprise Container Management Platform

Customers need to assemble a set of tools to efficiently manage the life cycle of containerized applications. They form the core building blocks of the container management platform. This model of platform is becoming increasingly prevalent, and is critical for deploying and managing containers in production.

Operating System

Containers reduce the dependency of applications on the underlying operating system. Lightweight operating systems, like CoreOS and Red Hat Atomic Host, are preferred to run containerized workloads. The reduced footprint results in lower management costs of infrastructure.

Container Engine

Container engines manage the life cycle of each container running on a specific host or node. Orchestration tools interact with the container engine to schedule containers across the nodes of a cluster. Docker and rkt are two examples of container engines.

Image Registry

The image registry acts as the central repository for container images. It provides secure access to images that are used by the orchestration engine at runtime. Docker Trusted Registry, CoreOS Quay Enterprise and JFrog Artifactory are some of the available choices.

Image Security

Since images are critical to the foundation of developing containerized applications, they need to be scanned for vulnerabilities and potential threats. CoreOS Clair, Twistlock, and OpenSCAP can be used for scanning images.

Container Orchestration

This is the most important element of managing containerized workloads. It provides distributed cluster management and container scheduling services. Kubernetes, Docker native orchestration and DC/OS deliver container orchestration and management.

Distributed Storage

Containers demand a new breed of distributed storage to manage stateful workloads. Products such as ClusterHQ, Portworx, Joyent Manta and Diamanti offer container-optimized storage.

Monitoring

Production workloads need constant visibility into the status and health of applications. The monitoring solution needs to span infrastructures and the containers deployed in it. Datadog, Sysdig and Prometheus are examples of container monitoring services.

Logging

Analyzing logs provides insight into the performance, stability, and reliability of containers and their hosts. As with any production workload, logging is a critical component. Splunk, Loggly and Logentries provide logging services for containers.

Source Control Management

Though source code management (SCM) is typically used for maintaining revisions of source code, it also plays an important role in versioning artifacts of containerized workloads, such as image definitions and Kubernetes object declarations. Existing SCM solutions, such as GitHub, Bitbucket and GitLab, are used for managing both code and artifacts.

Build Automation

Container images are built as a part of CI/CD pipelines. By utilizing existing workflows and build automation pipelines, customers can achieve automated deployments of containerized applications. Specialized CI/CD tools, such as Shippable, or existing tools, like Jenkins, can be extended for automating deployments.

Configuration Management

Traditional configuration management tools are extended to support containers. Customers who mix and match virtualization with containerization may want to use a unified toolchain to provision, configure, deploy and manage applications. Chef, Puppet, Ansible and SaltStack have added support for containers, making it possible to utilize their existing playbooks and recipes for the new breed of containerized applications.

Kubernetes as Container Orchestration and Management Tool

One of the first and most common usages of Kubernetes is managing containerized applications in production. Customers deploy an open source distribution of the Kubernetes cluster or subscribe to a commercial offering from a vendor. The deployment target could be either on-premises, public or hybrid cloud.

In this scenario, customers have a mix of DevOps tools for managing existing applications and contemporary applications. They use a set of heterogeneous tools for managing the image registry, security scanning, storage, networking and build automation. Kubernetes plays the critical role of container orchestration and management by integrating with existing tools. Customers can choose a commercial Kubernetes offering, such as Tectonic, or the enterprise distribution from Canonical.

CoreOS Tectonic

Coming from CoreOS, a company that was born in the container-era, Tectonic is one of the first Kubernetes-based, end-to-end commercial container orchestration engines for the enterprise. It is a CoreOS stack combined with Kubernetes and security enhancements. Subscribers will get the latest updates to keep their infrastructure up to date.

The key differentiating factor of Tectonic, when compared to the stock distribution of Kubernetes, is the security aspect. Distributed Trusted Computing (DTC) enables customers to cryptographically verify the integrity of their entire environment, from hardware to distributed clusters. It can validate trusted individual nodes before they join the cluster. This is important when enterprises run Kubernetes in a remote, co-located infrastructure. DTC promises cluster integrity, even in potentially compromised data center conditions.

Tectonic can integrate with Quay Enterprise, the private image registry from CoreOS, to build, store and distribute containers.

Canonical Distribution of Kubernetes

Canonical, riding high on its distribution of Ubuntu, has entered the crowded market of container orchestration with its commercial Kubernetes offering. Canonical's distribution provides customers access to stable upstream Kubernetes releases, as well as access to early builds of the upstream Kubernetes development branch. The master nodes can be scaled independent of the worker nodes. Workloads are automatically portable between public clouds, private clouds and bare metal with an easy onramp to Google Container Engine. Life cycle features come with the ability to easily create and remove user access, and a maintenance mode with a supported upgrade path to the latest version.

The Canonical distribution of Kubernetes includes the following components: Kubernetes dashboard; a Prometheus-based monitoring system to collect and process system metrics; log monitoring and visualization based on Elasticsearch, Logstash and Kibana; Flannel for container-optimized networking; and optional Ceph integration for distributed storage.

Companies Offering Support for On-Premises Kubernetes

Canonical Kubernetes (Canonical) Canonical's distribution provides customers access to stable upstream Kubernetes releases, as well as access to early builds of the upstream Kubernetes development branch. Canonical has optimized Kubernetes to run with its existing infrastructure and DevOps tools, but it also works across all major public clouds and private infrastructure.

Deis Professional Services (Engine Yard) Deis provides commercial support of Helm, which is a Kubernetes-native package manager. It works with enterprises at all stages of Kubernetes adoption, including proof of concept, production cluster standup, application migration, and cluster certification as well training. In addition, Deis provides 24/7 operational support for Kubernetes clusters they manage.

Kubernetes Support (Apprenda) Professional support for Kubernetes to handle both original implementation and ongoing operations. Apprenda offers three tiers of support, including pay per incident.

Tectonic (CoreOS) A commercial distribution of combined Kubernetes and CoreOS stacks. Tectonic is a Kubernetes-based, end-to-end commercial container orchestration engine for the enterprise, with an added focus on security.

Canonical has optimized Kubernetes to run with its existing infrastructure and DevOps tools. Customers who run Canonical's OpenStack, Metal as a Service (MAAS) for bare metal infrastructure, and Juju for DevOps can easily integrate with the Kubernetes distribution.

Kubernetes as a Private PaaS

Customers deploy PaaS primarily to standardize their development and deployment environments. By using a Kubernetes-based PaaS, they will be able to bring traditional line-of-business applications and contemporary containerized applications onto the same platform. Kubernetes has been adopted by traditional PaaS vendors to deliver end-to-end platform to enterprise customers. Built on top of core container orchestration capabilities, these PaaS offerings deliver complete life cycle management for containerized applications.

PaaS vendors add additional capabilities — such as application staging, message routing, service brokers, capacity planning, integrated logging, and monitoring — to Kubernetes. This presents a unified platform view to developers without exposing the nuts and bolts of underlying infrastructure. The PaaS layer attempts to abstract the complexity of dealing with Kubernetes by simplifying the workflow.

Apprenda and Red Hat OpenShift are prominent players in the space of the Kubernetes-based PaaS market. Both vendors offer commercial support for Kubernetes. Other Kubernetes-based PaaS players, including AppsCode and Eldarion Cloud, are delivered via the public cloud.

Apprenda

Apprenda is an enterprise PaaS targeted at customers with both Microsoft .NET and Java application portfolios to deliver an efficient application deployment architecture. The platform was designed with the goal of bringing a PaaS-like experience to both development and operations teams dealing with existing enterprise applications running on-premises.

Recently, Apprenda embraced Docker and Kubernetes with the goal of bridging the gap between traditional and contemporary applications. Customers will be able to mix and match DevOps processes that target both .NET and Java applications as well as Docker-based containerized applications.

Developers and Operations teams can use existing CI/CD tools to maintain old applications as well as Kubernetes-based applications. Though Apprenda PaaS is not open source, it enables integration to its platform through hooks and the REST API. Applications deployed in Kubernetes can talk to traditional .NET or Java applications using standard protocols.

Apprenda is a policy-driven PaaS, which enables the operations team to define granular policies aligning with application onboarding, monitoring, compliance and security. This approach meets the goal of enterprise customers that have an existing application portfolio, but actively consider the inclusion of containerized applications. It allows deployment, security and compliance policies to be centrally defined and administered, while allowing development teams to remain focused on targeting applications onto the scalable platform.

Red Hat OpenShift

OpenShift from Red Hat is a leading open source enterprise PaaS based on Kubernetes. After Google, OpenShift engineers at Red Hat are the most active contributors to the Kubernetes project.

Similar to its competitors, Red Hat built the initial versions of OpenShift on a platform-specific implementation. With the acceptance of Docker and Kubernetes among developers, Red Hat shifted its focus to making OpenShift the best PaaS built on top of the Kubernetes engine.

OpenShift aims to offer a developer-centric experience while abstracting the Kubernetes infrastructure. It comes with a set of user interfaces that connect Git, proprietary Red Hat Enterprise Linux (RHEL) tools, Docker and Kubernetes. Developers interact with the platform through existing tools that work with Git and through the integrated image registry that comes with the OpenShift platform. The platform includes a Source-to-Image (S2I) tool that converts developers' source code into a running Docker container, while preserving the layered image structure of the Docker format.

Red Hat built on top of Kubernetes to add enterprise-centric features — such as projects, users, and groups — that enable multi-tenant capabilities. Multiple departments or business units can deploy different applications that are isolated but belonging to the same cluster. OpenShift also provides a comprehensive view of your application logs, including runtime logs, build logs and deployment logs.

DevOps teams can work with familiar Docker and Kubernetes tools to manage the platform. OpenShift comes with an in-built policy management tool that focuses on user-authentication for jobs and management of internal Docker Registry artifacts.

Kubernetes as an Infrastructure for Cloud-Native Apps

While there is no exact consensus among industry players on the definition of cloud-native applications, most agree they can be defined as contemporary applications packaged in containers, deployed as microservices, running on elastic infrastructure, and managed through agile DevOps processes. Cloud-native applications are closely aligned with the principles of Twelve-Factor Apps.

It’s important to understand that not every containerized application is a cloud-native application. While Kubernetes can certainly be used to deploy and manage containerized applications that are essentially refactored from existing virtual machines, Kubernetes shines in the area of cloud-native application life cycle management. Customers designing, developing, testing, deploying and managing cloud-native applications choose Kubernetes as the preferred deployment platform.

These applications target Kubernetes clusters deployed on existing infrastructure or PaaS. While there is an overlap with the above defined scenarios, this use case is often seen in organizations that are building applications from the ground up that don’t have to be interoperable with existing legacy applications.

Cloud-native applications take advantage of advanced features of Kubernetes, such as rolling updates, canary deploys, horizontal pod autoscaling and cluster autoscaling. And while Kubernetes as an infrastructure for cloud-native applications is still an evolving use case, it’s worth pointing out that this is a value strategy for many users and organizations.

Kubernetes as an infrastructure for cloud-native applications is a model at work in implementations such as the Stackanetes project from Intel and CoreOS. Stackanetes is OpenStack on top of Kubernetes; it provides high availability with scaling, self-healing, and the full complement of OpenStack IaaS features – all while being deployed and managed with Kubernetes automation. A technical preview for the project was recently released at OpenStack Barcelona.

WHAT TO KNOW WHEN USING KUBERNETES

by **JANAKIRAM MSV**

Kubernetes is gaining ground in the container orchestration and cloud-native application management segment. While there are options available to customers in the form of other orchestration engines, PaaS and hosted solutions, the community and ecosystem built around Kubernetes makes it a top contender.

This section touches upon the key factors of Kubernetes that customers may consider before adopting it.

Strengths

- **Kubernetes has a clear governance model** managed by the Linux Foundation. Google is actively driving the product features and roadmap, while allowing the rest of the ecosystem to participate.

- **A growing and vibrant Kubernetes ecosystem** provides confidence to enterprises about its long-term viability. Huawei, IBM, Intel and Red Hat are some of the companies making prominent contributions to the project.

- **The commercial viability of Kubernetes makes it an interesting choice for vendors**. We expect to see new offerings announced over the next several months.

- **Despite the expected growth in commercial distributions, Kubernetes avoids dependency and vendor lock-in** through active community participation and ecosystem support.

- **Kubernetes supports a wide range of deployment options.** Customers can choose between bare metal, virtualization, private, public and hybrid cloud deployments. It enjoys a wide range of delivery models across on-premises and cloud-based services.

- **The design of Kubernetes is more operations-centric** than developer-oriented, which makes it the first choice of DevOps teams.

- **Customers with PaaS-specific requirements can choose from commercial enterprise private PaaS offerings** such as OpenShift and Apprenda.

- **Kubernetes is less prescriptive than some other PaaS offerings.** The flexible service discovery and integration model makes it easy for developers to package existing applications for Kubernetes.

Limitations

- **Kubernetes' support for stateful applications is still evolving.** In its current version 1.4, running transactional databases and big data workloads is not recommended.

- **Lack of support for Microsoft Windows** is another major gap in the Kubernetes ecosystem. There are no vendors offering integration with Windows Containers and Hyper-V Containers running within the Microsoft environment.

- **Kubernetes is still a relatively young project**, and there are some concerns about its use in production. However, there are many examples of those who have been successful. For users new to Kubernetes, you don't need to rush to use it in production environments.

- **As of November 2016, Kubernetes doesn't support true multi-tenancy.** Customers will need to find workarounds to deploy workloads that need strong isolation and independent governance models.

- **Fragmentation of plugins and add-ons will be a challenge for customers.** From SDN to storage and logging, there are dozens of tools created for Kubernetes. Some of them are actively maintained, while a few are discontinued. It's not an easy task for customers to identify the best set of tools and plugins for production use.

Roadmap

The Kubernetes ecosystem is working on filling the gaps in the system. Based on the usage patterns and customer feedback, the immediate priority is stateful applications. Upcoming releases of Kubernetes will support pet sets, a feature that makes it possible to run highly available stateful workloads such as Cassandra Clusters and MySQL deployments.

Kubernetes will focus on supporting Microsoft Windows in the future. Given Microsoft's investment in Azure Container Service, and a strong partnership with Docker, it makes sense to bridge these two ecosystems. A shrinking gap between Linux and Windows containers would make it possible for orchestration engines to seamlessly orchestrate workloads running in both environments. There is a Kubernetes Special Interest Group (SIG) dedicated to bringing support to Windows.

As enterprise adoption of containers grows, there will be a need to support hybrid deployments. Kubernetes' concept of federated clusters is gearing up to deploy containerized workloads in hybrid environments. Customers will be able to move applications across clusters deployed both on-premises and in the public cloud. Federated clusters also enable application portability across hosted Kubernetes platforms and internal clusters managed by IT teams.

Going forward, Kubernetes will include core primitives that are available as add-ons and optional open source projects. These include monitoring, logging, user interface and automation. Kubernetes 1.4 includes a rich dashboard UI that supports almost all the tasks performed through command-line interface (CLI). Similarly, strong integration with build automation and CI/CD tools will enable customers to extend continuous delivery to Kubernetes workloads.

Because Kubernetes has attention and support from the open source ecosystem, users will enjoy greater choice of container runtimes, network plugins, storage plugins, monitoring, logging and frontend tooling.

KUBERNETES SOLUTIONS DIRECTORY

COMMERCIAL DISTRIBUTIONS & OTHER COMMERCIAL SUPPORT FOR ON-PREMISES KUBERNETES

Although Kubernetes is open source, companies charge for support, often times only for their own distribution. This section does not include Kubernetes that is bundled into a larger container management or PaaS offerings.

Product/Project (Company or Supporting Org.)
Canonical Kubernetes (Canonical) Canonical's distribution provides customers access to stable upstream Kubernetes releases, as well as access to early builds of the upstream Kubernetes development branch. Canonical has optimized Kubernetes to run with its existing infrastructure and DevOps tools, but it also works across all major public clouds and private infrastructure.
Deis Professional Services (Engine Yard) Deis provides commercial support of Helm, which is a Kubernetes-native package manager. It works with enterprises at all stages of Kubernetes adoption, including proof of concept, production cluster standup, application migration, and cluster certification as well training. In addition, Deis provides 24/7 operational support for Kubernetes clusters they manage.
Kubernetes Support (Apprenda) Professional support for Kubernetes to handle both original implementation and ongoing operations. Apprenda offers three tiers of support, including pay per incident.
Tectonic (CoreOS) A commercial distribution of combined Kubernetes and CoreOS stacks. Tectonic is a Kubernetes-based, end-to-end commercial container orchestration engine for the enterprise, with an added focus on security.

CONTAINER MANAGEMENT, HOSTED SOLUTIONS AND PAAS

These offerings include hosted Kubernetes, Containers as a Service, as well as container management and PaaS built with Kubernetes.

Product/Project (Company or Supporting Org.)	Deployment Type
AppsCode **(AppsCode)** Integrated platform for collaborative coding, testing and deploying of containerized apps. Support is provided for deploying containers to AWS and Google Cloud Platform.	Hosted, PaaS
Giant Swarm **(Giant Swarm)** A hosted container solution to build, deploy and manage containerized services with Kubernetes as a core component. It offers customers fully-managed private Kubernetes clusters — including management of master and nodes. It is offered "as a service" or can be deployed and managed on-premises by Giant Swarm.	Hosted, On-premises
Cloud Container Engine **(Huawei)** A scalable, high-performance container service based on Kubernetes.	CaaS
CloudStack Container Service **(ShapeBlue)** A Container as a Service solution that combines the power of Apache CloudStack and Kubernetes. It uses Kubernetes to provide the underlying platform for automating deployment, scaling and operation of application containers across clusters of hosts in the service provider environment.	CaaS, On-premises
Deis Workflow **(Engine Yard)** A Kubernetes-native PaaS focused on developer self-service and operational flexibility. Deis Workflow helps teams quickly get up and running with Kubernetes on any public cloud, private cloud or bare metal cluster.	PaaS, On-premises
Eldarion Cloud **(Eldarion)** DevOps services and development consulting, packaged with a PaaS powered by Kubernetes, CoreOS and Docker. It includes Kel, a layer of open source tools and components for managing web application deployment and hosting.	PaaS
Google Container Engine **(Google)** Google Container Engine is a cluster management and orchestration system that lets users run containers on the Google Cloud Platform.	CaaS
Hasura Platform **(34 Cross Systems)** A platform for creating and deploying microservices. This emerging company's infrastructure is built using Docker and Kubernetes.	PaaS

Product/Project (Company or Supporting Org.)	Deployment Type
Hypernetes **(HyperHQ)** A multi-tenant Kubernetes distribution. It combines the orchestration power of Kubernetes and the runtime isolation of Hyper to build a secure multi-tenant CaaS platform.	On-premises
KCluster **(KCluster)** A hosted Kubernetes service that assists with automatic deployment of highly available and scalable production-ready Kubernetes clusters. It also hosts the Kubernetes master components.	Hosted
Kubernetes as a Service on Photon Platform **(VMware)** Photon is an open source platform that runs on top of VMware's NSX, ESXi and Virtual SAN. The Kubernetes as a Service feature will be available at the end of 2016.	On-premises
OpenShift Container Platform **(Red Hat)** A container application platform that can span across multiple infrastructure footprints. It is built using Docker and Kubernetes technology.	PaaS, On-premises
OpenShift Online **(Red Hat)** Red Hat's hosted version of OpenShift, a container application platform that can span across multiple infrastructure footprints. It is built using Docker and Kubernetes technology.	Hosted
Platform9 Managed Kubernetes for Docker **(Platform9)** Kubernetes offered as a managed service. Customers can utilize Platform9's single pane of glass, allowing users to orchestrate and manage containers alongside virtual machines. In other words, you can orchestrate VMs using OpenStack and/or Kubernetes.	Hosted, On-premises
StackPointCloud **(StackPointCloud)** Allows users to easily create, scale and manage Kubernetes clusters of any size with the cloud provider of their choice. Its goal is to be a universal control plane for Kubernetes clouds.	Hosted

TOOLS TO DEPLOY AND MONITOR KUBERNETES CLUSTERS

These tools help build, deploy, manage and monitor containers in Kubernetes clusters. Many tools that are not specifically made for Kubernetes are not included.

Product/Project (Company or Supporting Org.)	Sub-category
AppFormix (AppFormix) Provides metrics and analytics for containers organized with the Kubernetes architecture. Users can view analysis and make alarm or orchestration rules for specific or all namespaces, hosts, pods and containers. The dashboard can be further customized to map to application projects, host clusters, or the Kubernetes daemon sets.	Monitoring
Bootkube (CoreOS) A helper tool for launching self-hosted Kubernetes clusters.	Deploy
Cabin (Skippbox) An iOS application for managing Kubernetes applications. An Android application is being developed.	Deploy
ElasticBox ElasticKube (CenturyLink) A service for connecting CI/CD pipelines, configuration management tools, and deploying cloud applications. It includes ElasticKube, an open source management platform for Kubernetes that promotes self-service for containerized applications.	Deploy
Fuel CCP (OpenStack Foundation) CCP stands for containerized control plane. It is part of the Fuel project, and its goal is to make deploying Kubernetes easier for operators.	Deploy
Heapster (Cloud Native Computing Foundation) Enables analysis of compute resource usage and monitoring of container clusters. Heapster currently supports Kubernetes and CoreOS natively.	Monitor
Helm (Cloud Native Computing Foundation) A Kubernetes-native package manager that helps operators declare and manage complex, multi-part applications.	Deploy
Kargo by Kubespray (N/A) Kargo is the main component of this open source project, which allows users to deploy a Kubernetes cluster on bare metal, AWS, Google Cloud and/or OpenStack.	Deploy

Product/Project (Company or Supporting Org.)	Sub-category
kaws **(N/A)** A tool for creating and managing Kubernetes clusters on AWS using Terraform.	Depoy
Kolla-Kubernetes **(OpenStack Foundation)** The project provides Docker containers and Ansible playbooks to deploy Kubernetes on OpenStack.	Depoy
Kompose **(Skippbox)** Helps Docker users embrace Kubernetes by taking a Docker-compose file or Docker bundle and automatically transforming it into Kubernetes resource manifests. Kompose is a Kubernetes incubated project.	
Kraken **(Samsung)** Enables deployment of a Kubernetes cluster using Terraform and Ansible on top of CoreOS.	
Kubernetes Anywhere **(Cloud Native Computing Foundation)** An automated solution that will eventually allow users to deploy Kubernetes clusters across multiple clouds.	Deploy
Kubernetes Dashboard **(Cloud Native Computing Foundation)** A general purpose, web-based UI for Kubernetes clusters. It allows users to manage applications running in the cluster and troubleshoot them, as well as manage the cluster itself.	Monitor
Kubernetes Operations (Kops) **(Cloud Native Computing Foundation)** Production grade Kubernetes installation, upgrades and management.	Deploy
Minikube **(Cloud Native Computing Foundation)** Minikube is a tool that makes it easy to run Kubernetes locally. Minikube runs a single node Kubernetes cluster inside a VM on your laptop. For users looking to try out Kubernetes or develop with it day-to-day.	Deploy
Navops Launch **(Univa)** Navops Launch allows users to build container clusters using Kubernetes.	Deploy
Stackanetes **(N/A)** Uses Kubernetes to deploy standard OpenStack services in containers.	Deploy
Supergiant Support **(Qbox)** Supergiant is an open source framework that runs Docker containers. It hosts stateful, clustered applications utilizing Kubernetes under the hood. It uses its own structures and code for persistent storage and external load balancing. Qbox, the creator of Supergiant, provides commercial support.	Deploy
Sysdig Cloud **(Sysdig)** Natively monitors Kubernetes clusters.	Monitor

INTEGRATIONS

These tools help build, deploy, manage and monitor containers in Kubernetes clusters. Many tools that are not specifically made for Kubernetes are not included.

Product/Project (Company or Supporting Org.)
Apprenda (Apprenda) Apprenda provides private PaaS for enterprises that support the hosting of containers.
Crunchy PostgreSQL Container Suite (Crunchy Data) A set of prepackaged Docker containers and Kubernetes services, the Crunchy Container Suite allows teams to run and manage PostgreSQL in Kubernetes-based environments.
Datadog-Kubernetes Integration (Datadog) Collects and monitors metrics Kubelets in real time. It is deployed as a Docker container alongside existing workloads.
fabric8 (Red Hat) An open source DevOps and integration platform that is built as a set of microservices that run on top of Kubernetes and OpenShift V3. Its continuous delivery is based on Jenkins, Nexus and SonarQube.
Joyent Triton ContainerPilot (Joyent) Works in conjunction with other schedulers — letting them start and stop containers — with ContainerPilot orchestrating the rest. Applications orchestrated by ContainerPilot are portable from one scheduler to another.
Kubeflix (Red Hat) Kubernetes integration with Netflix open source components such as Hystrix, Turbine and Ribbon.
NetScaler CPX (Citrix) Docker containerized load balancer that can be supported on-premises and in multi-cloud environments. NetScaler CPX integrates with a Kubernetes deployment to load balance containerized applications in a cluster environment. In a Kubernetes environment, NetScaler CPX replaces kube-proxy on the minions and balances the load across the containers in a pod.
Wercker Workflows (Wercker) Creates custom pipelines to build, push to a container registry and notify a scheduler like Kubernetes.

DISCLOSURES

The following companies mentioned in this ebook are sponsors of The New Stack: CNCF, CoreOS, DigitalOcean, Docker, IBM, Intel, Joyent, Mesosphere, Red Hat OpenShift, Sysdig, Twistlock, Wercker.

thenewstack.io