AMALGAM INSIGHTS

Market Landscape

Kubernetes Evolves Into An Enterprise Platform

Author: Tom Petrocelli

October 2020

AMALGAM INSIGHTS

Market Landscape

Kubernetes Evolves Into An Enterprise Platform

EXECUTIVE SUMMARY

Key Stakeholders:

CIO, CTO, Vice President/Director/Manager of Platform Engineering, Vice President/Director/Manager of Operations, System Architect, Product Management, Product Marketing

Why It Matters:

Kubernetes is quickly becoming the base for next generation enterprise microservices[i] platforms. Understanding what is happening in the Kubernetes market now and in the near future is vital for planning, training, and design of new enterprise software platforms.

Top Takeaway:

As Kubernetes-based platforms gain traction, larger clusters that span datacenters, clouds, and geographies are becoming a feature of new software platforms. Much of the focus in the Kubernetes community is about solving problems of scale, especially management of large diverse clusters. This is why the industry is seeing technology and practices that enable the **federation**[ii] of Kubernetes and related services such as service mesh.

Vendors and Organizations Listed in This Report	
Amazon	**Lightstep**
Canonical	**Microsoft**
Cloud Foundry Foundation	**Mirantis**
Cloud Native Computing Foundation (CNCF)	**Oracle**
Datadog	**Platform9**
Google	**Rancher Labs**
Hashicorp	**Red Hat**
HPE	**Solo.io**
Humio	**Spectro Cloud**
Layer5	**VMware**

MARKET CONTEXT

The past several years has shown a radical shift in the platforms, tools, and practices that IT employs to develop software. Driven by the need to change and update software more quickly, realize higher uptime and resiliency, increase portability, and bring new levels of scalability to critical systems, new architectures have begun to emerge. Chief among those architectures are microservices-based platforms, built on container[iii] technology.

In parallel, developers want platforms that help them to focus on creating business logic and coding, according to Agile[iv] tenets. This has led to new toolchains, such as CI/CD[v], and deployment models such as Serverless[vi] computing. At the same time, Operations requires platforms that are self-healing, automated, and provide sufficient information to deal with short-term problems such as downtime to long-term issues of efficiency.

These changes in architecture, technology, and methods have resulting in an emerging back-end platform based on the Kubernetes orchestrator[vii], used to manage container clusters that implement microservices and supporting functions such as networking and logging.

In the initial years of container cluster architectures, containers were managed by hand or using simpler tools. As container architectures expanded in companies, both in size and in complexity, managing container lifecycles in this manner was impossible. Kubernetes (and other similar orchestrators[viii]) offered an opportunity to grow clusters efficiently by eliminating some of the manual steps and monitoring.

The success of Kubernetes, which was astounding to many in the IT industry, has led to more and larger clusters as many new applications, especially those that need to scale quickly, are built on this architecture. IT organizations are building out platforms based on Kubernetes filled with reusable microservices that can be updated and evolved independent of other components and applications. This architecture fits well with the continuous development and deployment philosophy of Agile.

The downside of this success is *increasing complexity*. Microservices platforms have grown from a few containers, to large clusters, to now, clusters of clusters that span data centers and public clouds. Even more complexity is derived from clusters that are geographically distant from each other that need to act as a single cluster from the point of view of an application. This *cluster sprawl* is both the root cause of much of the complexity and evolution in the Kubernetes environment.

Thankfully, the Kubernetes ecosystem and Kubernetes itself is growing to accommodate this growth and complexity. A series of new features have emerged that help to bring about better management and scalability for Kubernetes-based platforms. Amalgam Insights has identified the following aspects of the Kubernetes ecosystem as key to growing Kubernetes into a dominant platform for applications:

- Kubernetes Control Planes
- Techniques to Instantiate Containers More Quickly
- Advances in Service Mesh especially Service Mesh Management Planes
- Advances in Logging, Tracing, and Observability

All of these technologies are available today in commercial or open source form. On the horizon is something more fundamental called Kubernetes Cluster Federation, or Kubefed for short. Along with the evolution of the ecosystem, Kubernetes Cluster Federation promises to make Kubernetes viable for exceptionally large mission critical enterprise platforms.

KUBERNETES CONTROL PLANES

As Kubernetes-based platforms have evolved from small clusters to large clusters and now to many clusters that span multiple locations, clouds, and on-premises locations, vendors have responded with Kubernetes Control Planes. Ostensibly, Kubernetes Control Planes seek to enhance the ability to provision clusters from a central application. In practice, most Kubernetes Control Planes such as those from Rancher Labs, Platform9, VMware, and Red Hat have a wide range of features to manage multiple geographically and logically diverse Kubernetes installations. Common features include:

- Cluster lifecycle management especially provisioning of clusters, often from templates for common types of clusters.
- Versioning and updating including updates to Kubernetes itself.
- Cross-cluster Security and Auditing
- Cross-cluster Visibility, Monitoring, and Logging

Kubernetes Control Planes add a layer of automation as well as management, allowing for policy driven processes to take the place of manual handling. This automation allows operators to focus on governance while the control plane software performs the tasks associated with managing clusters. This helps to reduce errors while allowing for faster responses to changes or problems that may arise. The automation and visibility eliminate the need for managing many large multi-site clusters by hand, which, in turn, require large amounts of manpower and increase cost.

Most Kubernetes Control Planes provide for a graphical management interface. An interesting open source project called Crossplane adds Kubernetes control plane features to the Kubernetes command line tool, kubectl. Not only is Crossplane useful to command line users but also for scripting.

Control Plan Alternatives

There are alternatives to Kubernetes Control Planes. Depending on what an organization's needs are, Infrastructure as Code (IaC)ix systems may be able to perform some of the functions a Kubernetes Control Plane. Hashicorp's Terraform can provision clusters wherever an agent exists, including multiple clusters simultaneously. IaC used with security products may also automate cross-cluster security such as secrets management and auditing. It is important to look to existing infrastructure to see if the multi-cluster capabilities that exist in these products meet current needs. Kubernetes Control Planes may still become necessary at some scale, but existing infrastructure should be evaluated before spending money on them.

WARM STARTING AND CACHING CONTAINERS

One of the drivers for adopting Kubernetes-based platforms has been scalability. As resource need rise, typically in response to business needs, Kubernetes can instantiate more containers with necessary microservices. Instead of one service becoming overloaded or having to instantiate many copies of a service at the onset that are not used, Kubernetes can autoscale more containers, and hence more services. If the event that caused the expansion subsides, Kubernetes can spin down containers. In this way, Kubernetes creates a more efficient platform that responds more quickly to whatever the IT situation is.

It is not a perfect system though. Instantiating a container and running the code from scratch, or a cold start, takes time. If this happens rapidly and at scale, the response time can be too slow to head off system issues including crashes. The time to cold start a container also makes it difficult to accommodate event driven system models, such as Serverless computing, where an event triggers some action within the system. If the response to the event is too slow, the application generating the event may assume that there is a system failure and timeout or may simply miss the event.

There are several ways to increase Kubernetes container instantiation performance. One solution to the cold start problem is warm starting, sometimes called warm booting.

With a warm start, part of a container's lifecycle is started ahead of the need. A container may be created - that is the crgroups and namespaces are configured, the image downloaded and unpacked from the container repository, and code loaded into memory - but the code defined within the image is not run. When a new container is needed, the warm started container is ready to execute its code, significantly reducing container instantiation latency. This is a technique common in Serverless systems that need to respond quickly to events.

Kubernetes automated orchestration was a vast improvement over spinning up containers by hand. However, it is still too slow when needing to scale up in rapidly changing environments. Warm starting is a way to respond more quickly by partially instantiating the container.

Another technique has been in use, especially in cloud systems, is container caching. Container caching keeps copies of the container in memory but not running. When the need arises, the container is instantiated from the memory image, thereby avoiding some of the startup time of a container. Container caching assumes that containers that are already running or have run will be needed again later.

SERVICE MESH MANAGEMENT PLANE AND FEDERATION

A service mesh[x] has become a critical part of platforms based on Kubernetes clusters. It provides both east-west and north-south traffic management[xi], security, observability, and shaping for services implemented by the cluster and supporting components. As clusters have grown in size and platforms have become comprised of many clusters, maintaining a consistent view, management, and policies for network layer 4 – 7 traffic has become increasingly complex.

In response, a number of vendors have extended control planes with a multi-cluster management layer such that they federate and manage service meshes at scale across multiple clouds and on-premises deployments. Both Hashicorp Consul and VMware Tanzu Service Mesh built on VMware NSX extend the service mesh control plane past the local service mesh to allow a management layer over multiple clouds and clusters. The open source project Meshery is also providing a service mesh management plane as separate software that can interact using the Service Mesh Interface or through built-for-purpose adapters for a variety of existing service mesh products and projects. Solo.io's Service Mesh Hub has a similar function providing not only a management plane but federation for service meshes.

Like Kubernetes control planes, this extension of the service mesh control plane allows for consistent management and policies across clusters in different environments, a situation that is becoming more typical in enterprise IT applications.

Service mesh control plane federation is an emerging feature of service mesh products. Unlike Kubernetes control planes, it is not yet commonplace, but Amalgam Insights predicts that we will see service mesh control plane federation become a normal part of the service mesh landscape.

ADVANCES IN OBSERVABILITY

Logging, tracing, collectively called observability, are not a feature of Kubernetes. It is incredibly important for managing the containers running in the clusters and the code running in the containers. Logging creates a record of events that reported by running code. Tracing creates a record of the execution of code and typically shows internal states needed for runtime debugging. Both logging and tracing are needed for application runtime management and debugging. They are common design patterns for enterprise applications.

Over time the Kubernetes ecosystem has responded with products and open source projects to address these needs. Prometheus and FluentD (logging), and Jaeger and OpenTracing (tracing) have become common components of most commercial Kubernetes distributions as well as platforms built on vanilla Kubernetes.

As clusters have grown, and platforms built around multiple clusters have been deployed, the amount of information these logging, and tracing components generate have grown greatly. This has generated a need to manage massive amounts of logging information. Some enterprises have taken a big data approach. That is to be expected. The Hadoop big data platform was originally designed for managing large amounts of logging data.

A number of specialized solutions, such as DataDog, Humio, Chronosphere, and Lightstep, are making large amounts of logging and tracing information available in real-time. Most logging and tracing information is discarded or aggregated in favor of specific events because of the amount of data generated. New logging and tracing analysis applications make it possible to see *all* the data generated by applications, in real-time or near real-time, and provide relevant visualizations that making it easier to understand important events. This, in turn, leads to faster error detection, deeper problem analysis, and the ability to catch more esoteric issues.

In addition to processing the massive amounts of data generated by big clusters, the Thanos project, an incubating project of the CNCF, is looking to extend the reach of logging across clusters. Thanos queries Prometheus across clusters and is able to store unlimited data in object stores. In effect, Thanos creates a federated Prometheus.

It is important to note that while commercial products that manage and display observability information consume information from Kubernetes clusters, they typically process information from cloud systems as well.

SERVERLESS

Serverless computing, defined as event driven computing that abstracts the total infrastructure and only consumes resources when running code, has become widely adopted on cloud platforms. Subsequently, a Kubernetes serverless capability has become desirable for portability and for running serverless platforms on-premises. Kubernetes' advantage as a serverless platform is obvious in hybrid cloud[xii] architectures where serverless functions can act as the glue tying the cloud and on-premises environments together. Having serverless functions in containers also increases portability between cloud and on-premise platforms.

Knative had, for a time, looked to be the solution for serverless on Kubenetes. It had support from key Kubernetes vendors including IBM/Red Hat and Google. Despite technical issues such as the time it took to instantiate a serverless function in response to an event, there was no denying the interest that system architects had for Knative.

That enthusiasm has waned over the past year. It was expected that the Knative project would have been transferred to an independent not-for-profit, specifically the CNCF, in the Fall of 2019. When that did not happen, IT professionals and industry leaders became more pessimistic. Since then, there has been little major news about Knative and the focus of serverless computing has stayed with the cloud service providers.

The good news, however, is that the community continues to buildout the frameworks of a vendor neutral serverless capability. Both the CNCF CloudEvents, and Serverless Workflow Specification projects are building specifications, and some supporting technology, for cloud native Kubernetes-based serverless capabilities.

KUBERNETES CLUSTER FEDERATION

Kubernetes is that it is designed around a single cluster model. All clusters are independent of each other and do not interact directly. This isolation makes it less likely that a cluster-wide failure will affect other clusters. It also makes managing a Kubernetes-based platform or application spread across multiple clusters more difficult. While Kubernetes control planes deal with this problem, there is no *native* way to federate Kubernetes clusters.

Thankfully, there is a project within the Kubernetes community called Kubernetes Cluster Federation (https://github.com/kubernetes-sigs/kubefed), or Kubefed for short. Kubefed allows software to manage multiple cluster configurations from a single set of APIs. Operators could then be able to use these APIs to coordinate cluster operations from a single CLI or application. Kubefed also simplifies the design of cross-cluster management applications. It will not replace Kubernetes control planes. Instead, it would make it easier for a Kubenetes control plane to manage and coordinate clusters.

Kubefed is currently available as an alpha release with a beta release promised soon.

VENDORS AND PROJECTS

The Kubernetes ecosystem is one of the chief reasons that it has become a viable next generation enterprise platform. Meanwhile, the desire for Kubernetes to become an enterprise platform has helped evolve the ecosystem. This is a virtuous feedback loop. The ecosystem continues to evolve to meet the needs of the community and the community need continues to drive evolution in the ecosystem.

It is impossible to describe the entire Kubernetes or cloud native ecosystem in a short paper. The Cloud Native Computing Foundation, or CNCF, provides an excellent set of resources, including a graphical landscape diagram, that outlines much of the ecosystem. What is more interesting are the key elements of the Kubernetes landscape, crucial to its evolution into a next generation computing platform. This section will outline some of the vendors and open source projects that comprise these important components of the evolving landscape.

KUBERNETES

Everything starts with Kubernetes. There are four ways to deploy Kubernetes - pure Kubernetes from open source, a curated Kubernetes distribution, a managed Kubernetes service, and a cloud service. The first two are software that is installed either on premises or in a cloud instance. The third is a managed service typically installed in a cloud instance but managed by the service provider while the last is a cloud service installed from a cloud app store.

The current version of Kubenetes is v1.19. Most distributions and service provider instances deploy one or two versions back. Versions v1.17 and v1.16 are still commonly found in production. Deploying Kubernetes from source

code allows for the creation of a built to purpose IT platform. It also requires the most work and ongoing maintenance as it is only supported by the community.

There are advantages to developing an enterprise IT platform based on a well-known distribution such as Red Hat OpenShift or VMWare Tanzu. First, there is a curated experience where the vendor has done the difficult work of testing and integrating components. Second, these are supported ready-made platforms that require less work to deploy and hence fewer long-term personnel. Finally, these distributions can often be instantiated on popular cloud services, providing a consistent platform for hybrid cloud operations. This comes at the cost of a software license of course.

Cloud services are the easiest to deploy. Whether the Kubernetes distribution is on Amazon, Google Cloud, Oracle Cloud, or Microsoft Azure, or other cloud providers, a cloud deployment is typically fast, requiring little setup. It has the advantage of usage-based licensing and is managed by the cloud provider. The downsides are with portability – these exact Kubernetes are not able to be installed on generic on-premises infrastructure or other cloud services – and the limited control a cloud provider allows.

Finally, a new class of managed Kubernetes has emerged from providers such as Platform9 and Spectro Cloud. These services provision vanilla Kubernetes on popular cloud platforms and manage them afterwards. This provides a middle ground between the cloud service providers' nonportable offerings and install-your-own vendor distributions.

Vendor or Not-For Profit	Distribution Name	Type
Amazon	Amazon Elastic Kubernetes Service	Cloud Service
Canonical	Charmed Kubernetes, Managed Apps, MicroK8, Managed Kubernetes	Distribution, Managed Service
Cloud Foundry Foundation	KubeCF, Eirini	Open Source Project, though there are distributions especially from SUSE, VMWare Pivotal, and IBM
Cloud Native Computing Foundation (CNCF)	Kubernetes v1.18	Open Source Project
Google	Google Kubernetes Engine	Cloud Service
HPE	HPE Ezmeral Container Platform	Distribution, Managed Service[xiii]
Microsoft	Azure Kubernetes Service	Cloud Service
Mirantis	Docker Kubernetes Service	Distribution
Oracle	Oracle Cloud Infrastructure Container Engine for Kubernetes	Cloud Service
Platform9	Platform9 Managed Kubernetes	Managed Service
Rancher Labs[xiv]	Rancher K3s, Rancher RKE	Distribution
Red Hat	Red Hat OpenShift	Distribution
Spectro Cloud	Spectro Cloud	Distribution, Managed Service

VMware	VMWare Tanzu	Distribution

LOGGING AND TRACING

The number of open source projects in this category has grown dramatically in the past few years in and it is still heavy with CNCF managed projects. The market has also started to see vendor offerings that make use of the raw data collected from the open source monitoring, logging, and tracing products.

Vendor or Not-For Profit	Product or Project
Chronosphere	M3
CNCF	Prometheus
	FluentD
	OpenTrace
	Jaeger
	OpenCensus
	Thanos
DataDog	DataDog
Humio	Humio
Lightstep	Lightstep

KUBERNETES CONTROL PLANES

When discussing Kubernetes control planes, it is important to distinguish them from other types of control planes, some in closely adjacent market segments. There are service mesh control planes and cloud control planes that can be part of managing a set of Kubernetes clusters.

A Kubernetes control plane, on the other hand, is a product or part of a wider suite that is responsible for managing Kubernetes clusters. Some vendors treat their Kubernetes Control planes as named product or feature, such as Red Hat' Advanced Cluster Management for Kubernetes or VMWare Tanzu Mission Control. Others treat the Kubernetes control plane as an inherent feature as HPE does with the cluster control plane that is part of HPE Ezmeral Container Platform.

Vendor	Product
CNCF	Crossplane
Google Cloud Platform	Zonal Control Plane
HPE	HPE Ezmeral Container Platform
Platform9	SaaS Management Plane
Rancher Labs	Rancher

Red Hat	Advanced Cluster Management for Kubernetes[xv]
Spectro Cloud	SaaS Management Platform
VMWare	VMWare Tanzu Mission Control

It is worth noting that other cloud platforms such as Amazon Elastic Kubernetes Service and Microsoft Azure Kubernetes Service, have some of the functions of a Kubernetes control plane built into the overall cloud services. This is especially true of provisioning and monitoring activities.

Service Mesh Management Plane

Many existing service mesh management planes is a feature of an existing control plane. An exception to this is Solo.io's Gloo Federation and the open source Meshery project. The latter two provide federation features on top of a variety of service mesh control planes.

Vendor or Not-For-Profit[xvi]	Product or Project
Hashicorp	Consul
Layer 5[xvii]	Meshery
Solo.io	Service Mesh Hub
VMWare	VMware Tanzu Service Mesh built on VMware NSX

CONCLUSION

Kubernetes is well on the way to establishing itself as the platform of choice for microservices architectures and scalable enterprise systems. Many basic platform features – provisioning, management, restarting services, observability, storage, and networking – have been already developed. Now, the Kubernetes community is dealing with the issues of scale. The focus is shifting to managing mega clusters that span datacenters, clouds, and geographies. It is important that the community develop practices and technology that ensure that large and diverse clusters can operate without a massive burden on operations.

The clear message is that federation is key to future platforms. Features that allow for federation of management and provisioning, networking, and the Kubernetes orchestration layer itself are emerging. Whether it is a control layer on top of the platform, as is the case with Kubernetes control planes, or a project such as Kubefed that allows upper layers of a platform to coordinate activities at the Kubernetes layer, federation is the path to building large scale, heterogeneous clusters into a unified enterprise IT platform.

The community is continuing to look for more ways to create distributed versions of necessary services. Sandbox projects such as Longhorn are looking to solve difficult storage problems in distributed environments. The sheer number of projects in the CNCF portfolio, currently more than 60, is an indicator of the health and growth of the ecosystem. Many of those projects are addressing problems of scale in networking, provisioning, policy management, and observability.

AMALGAM INSIGHTS

Market Landscape

Kubernetes Evolves Into An Enterprise Platform

It is unlikely that five years ago, anyone could have predicted that Kubernetes would be the powerhouse that it has become. From its humble beginnings, solving a narrow problem, Kubernetes and its ecosystem are the underpinnings of the next generation enterprise IT platform. Continued evolution, especially service federation, is pointing to more future success.

Tom Petrocelli
Research Fellow
October 6, 2020

VENDORS AND ORGANIZATIONS LISTED IN THIS REPORT

Vendor or Organization	Website
Amazon	https://aws.amazon.com/eks/
Canonical	https://ubuntu.com/kubernetes, https://microk8s.io/
Cloud Foundry Foundation	https://www.cloudfoundry.org/
Cloud Native Computing Foundation (CNCF)	https://www.cncf.io/
Datadog	https://www.datadoghq.com/
Google	https://cloud.google.com/kubernetes-engine/
Hashicorp	https://www.hashicorp.com/
HPE	https://www.hpe.com, https://www.hpe.com/containerplatform
Humio	https://www.humio.com/
Layer5	https://layer5.io/
Lightstep	https://lightstep.com/
Microsoft	https://azure.microsoft.com/en-us/services/kubernetes-service/
Mirantis	https://www.mirantis.com/software/docker/kubernetes/
Oracle	https://www.oracle.com/cloud/compute/container-engine-kubernetes.html
Platform9	https://platform9.com/
Rancher Labs	https://www.rancher.com/
Red Hat	https://www.redhat.com/, https://www.openshift.com/
Solo.io	https://www.solo.io/
Spectro Cloud	https://www.spectrocloud.com/
VMware	https://www.vmware.com, https://tanzu.vmware.com/tanzu

OPEN SOURCE PROJECTS

Project	
Consul[xviii]	**Kubernetes Federation**
Crossplane	**Longhorn**
Eirini	**Meshery**
FluentD	**OpenCensus**
Jaeger	**OpenTrace**
Knative	**Prometheus**
KubeCF	**Service Mesh Hub**
Kubernetes	**Thanos**

ABOUT AMALGAM INSIGHTS

AMALGAM INSIGHTS

Is a leading research and advisory firm focused the financial, programmatic, and cognitive tools that multiply the value of enterprise technology including the following research practices: Technology Expense and IT Subscription Management, Accounting and Business Planning Technologies, Data Science and Machine Learning, DevOps and Open Source Development, Talent Management, Learning & Development, and Extended Reality.

TOM PETROCELLI, RESEARCH FELLOW

Tom Petrocelli is a Research Fellow with Amalgam Insights. His area of interest is developer tools, IT project efficiency, governance, and methodologies, and DevOps. He also looks at how large regulated companies, especially financial services companies, manage IT projects. Tom has over 35 years of experience in the IT industry.

Prior to Amalgam Insights, Tom:

- Worked for a large, global, banking corporation.
- Was the research director for Enterprise Social, Mobile and Cloud Applications at Neuralytix.
- Before Neuralytix, Tom was the senior analyst, Social Enterprise at Enterprise Strategy Group.
- Before becoming an analyst, Tom held various senior and executive management positions.

CONTACT AMALGAM INSIGHTS

Phone: +1 415 754-9686
Website: www.amalgaminsights.com
Twitter: @AmalgamInsights

Disclaimer: Amalgam Insights provides consulting, research and advisory services to a variety of technology consumers and vendors and may have revenue-based client relationships with companies mentioned in our research.

ENDNOTES

[i] A microservice is a small service, typically deployed in a container, that provides one or a limited number of functions to a system.
[ii] Federation in this context refers to mapping resources into a single virtual resource. It is commonly used to make database, clusters, and other software infrastructure appear as a single entity for management purposes. In this case, federation means making clusters appear unified through centralized management.
[iii] Containers are a form of lightweight virtualization that is part of the Linux operating system. They employ two Linux capabilities to create containers, cgroups and namespaces. A cgroup limits the amount of resources available to processes while a namespace provides process isolation.
[iv] Agile is a development philosophy or strategy that advocates short development cycles and integrated business-technical teams. The most common implementation of Agile is called Scrum.
[v] CI/CD collectively is the automation of the process of taking raw developer code and other build artifacts and turning them into running applications and services.
[vi] Serverless is any computer system that abstracts the infrastructure for the developer, employs an event driven model, and only consumes resources when needed.
[vii] A container orchestrator provides the basic services for a running a container cluster. The most basic of these are instantiating and shutting down containers. Other typical services including autoscaling containers, network management, and APIs for interacting with the cluster. The most common container orchestrator is Kubernetes.
[viii] Other common orchestrators were Apache Mesos, Docker Swarm, Cloud Foundry Diego, and Rancher Cattle. While all of these still exist in some fashion, Kubernetes is far and away the market leader.
[ix] Infrastructure as Code, or IaC, is the practice of writing code, usually YAML or JSON to represent the software and hardware infrastructure configuration in a system. An automation server then executes the code and creates the system configuration. IaC may use a declarative or imperative design.
[x] A service mesh provides layer 4 through 7 network services to microservices architectures. It adds additionally observability, reliability, and security to the virtual networks microservices, especially container clusters, implement for intraservice and interservice communication.
[xi] East-West traffic has traditionally been conceived of as horizontal network traffic flowing from server to server within the data center. In the context of a microservice, it can be thought of as a traffic from one microservice to another or traffic between components of the microservice that are individual processes. North-South traffic has traditionally been thought of as vertical traffic from a host outside the data center to a server within it. In the context of a microservice, it would be between any host application consuming the services of the microservice from whatever houses the microservice, VM or container.
[xii] Hybrid cloud refers to an architecture that deploys applications to both on-premises servers and cloud instances.
[xiii] HPE provides a managed Kubernetes service through the APE Greenlake product
[xiv] While this paper was being written, it was announced that SUSE, a purveyor of Linux distributions and enterprise platform products, announced their intention to acquire Rancher Labs.
[xv] Advanced Cluster Management for Kubernetes is in the process of moving from a proprietary product to open source. Red Hat expects that all the projects that comprise the product will be open source by the first half of 2021. Some portions of ACM have already been moved to open source projects.
[xvi] In this context, a vendor may refer to a commercial product, the lead commercializer in an open core project, or both.
[xvii] Meshery is currently a project under the stewardship of Layer 5. They have pledged to bring the project into the CNCF in the near future.
[xviii] Consul is both an open source project and Hashicorp product based on the open source project. This is why it is listed in both places.