Kubernetes: Integration vs Native Solution

SHADOW SOFT

Table of Contents

01

Introduction

Recent reports have shown Kubernetes controlling more than seventy percent of the container orchestration market. This far outpaced other platforms such as Swarm and Mesos. Other platforms such as Cattle, ECS, Nomad, etc. are all so relatively unpopular as to simply be grouped together into the "other" category. Due to this, almost every competitor has accepted Kubernetes as the winner and integrated it into their software stacks.

However, each integrator has incorporated Kubernetes in their own unique way. Also, some of them have offered the ability to swap between their native solutions and those offered as part of Kubernetes. The optimal solution for an enterprise company will be to decide the stack based on what works best for them given the available solutions.

kubernetes

02

DC/OS

Kubernetes Integration

DC/OS includes a web and command-line interface to facilitate remote management and monitoring of the cluster and its services. DC/OS originally provided container orchestration via Mesos/Marathon by default. However, Mesos, being only a kernel, can leverage other container orchestrators. It now officially supports Kubernetes. Another major feature of DC/OS is it has the ability to handle load balancing to all services within the cluster via Edge-LB. Edge-LB provides both North-South (external to internal) load balancing, while the Minuteman component provides East-West (internal to internal) load balancing.

DC/OS now has the ability to deploy Kubernetes clusters and was recently certified by the Cloud Native Computing Foundation (CNCF). This allows Kubernetes clusters to benefit from different features of DC/OS and the Mesos distributed systems kernel. The Mesos architecture separates the scheduling of Kubernetes from the underlying resource management. This allows resource allocation, rolling upgrades, and autoscaling features. Mesosphere DC/OS allows you to deploy a Kubernetes cluster via the catalog. This allows for easy installation and management across data centers or across a hybrid cloud. The package in the catalog is pure upstream Kubernetes and is updated concurrently with the official source code. Since it is available like any other package in DC/OS you can run multiple Kubernetes clusters alongside each other regardless of the version.

In addition, Kubernetes clusters can run alongside other frameworks, namely distributed databases, stream processing, and machine learning.

Native Solution

DC/OS is a distributed operating system based on the Apache Mesos distributed systems kernel. It enables the management of multiple machines as if they were a single compute node. It automates resource management, schedules process placement, facilitates inter-process communication, and simplifies the installation and management of distributed services.

The benefits that Mesos provides are:

- Abstract data center resources into a single pool to simplify resource allocation while providing a consistent application and operational experience across private or public clouds.
- Co-locate diverse workloads on the same infrastructure such as analytics, stateless microservices, distributed data services, and traditional apps to improve utilization and reduce cost and footprint.
- Automate day-two operations for application-specific tasks such as deployment, self healing, scaling, and upgrades, thus providing a highly available fault tolerant infrastructure.
- Provide evergreen extensibility to run new application and technologies without modifying the cluster manager or any of the existing applications built on top of it.
- Elastically scale the application and the underlying infrastructure from a handful of nodes to tens of thousands.

Mesos, unlike other cluster managers, has a two-level scheduler architecture. This allows Mesos to allocate individual workloads to their own application scheduler. This modular approach enables different types of services to be deployed on the same architecture.

Apache Mesos at its core is a distributed systems kernel and a cluster manager. It can manage many different types of workloads (frameworks) including container orchestrators, Java applications, analytics, streaming, and machine learning. Marathon is an example of a container orchestrator that is made to run on top of a Mesos managed cluster. Marathon has the ability to orchestrate both applications and other frameworks. Marathon allows for services to be ran with different container runtimes: Docker, Mesos, and Universal Container Runtime (UCR). UCR allows for the provisioning of Docker and AppC container images with the Mesos containerizer. UCR is more stable on Mesos architecture and allows for GPU and CNI support as well as a host of other container management features.

Comparison

Since DC/OS allows for multiple frameworks on the same compute space you have the ability to run services where they make the best sense. Kubernetes is great for the container orchestration of 12-factor applications; however, it lacks features for some stateful apps, distributed databases, machine learning and GPU support. These services are better served to be handled by Mesos/Marathon, and DC/OS allows for that choice.

04

Docker Enterprise

Kubernetes Integration

Following the example of Mesosphere, and responding to both market reality and to enterprise customers who have made it clear they don't want to be locked into a proprietary product, Docker embraced pragmatism and adopted integrated support for Kubernetes 1.8. Multiple vendors, including Red Hat, CoreOS, Canonical, and VMware, have commercially supported Kubernetes distributions in the market that have been directly competing with Docker Swarm.
Docker Enterprise and Kubernetes are managed with the same dashboard, so administrators do not need to manage different silos. The whole stack is supported and certified end-to-end for Windows and all major Linux distributions so you are never locked into a single operating system.

Docker's implementation of Kubernetes looks to hide much of the complexity of Kubernetes behind the scenes. Docker took the approach of providing a design that allows for running both Kubernetes and Swarm within the same cluster. When deploying Swarm, the installer provides an option for installing Kubernetes. If selected, Kubernetes inherits the redundancy design of the Swarm install. It also integrates two different approaches to subsystems such as networking. Ecosystems designed for either orchestrator should still work independently. Applications built using a Docker Compose file run natively against Swarm-provided Kubernetes infrastructure. While Swarm and Kubernetes can run on a single host, each orchestrator remains unaware of the other. Each orchestrator assumes complete use of a single host. For this reason, Docker does not recommend running Kubernetes and Swarm workloads on the same host.

By integrating Docker Enterprise with Kubernetes, Docker expands the argu-ment of existing not just as a development platform, but a production-level ecosystem that competes with PaaS offerings. This integration brings many key orchestration capabilities, such as service discovery, load balancing, and horizontal scaling. Most importantly, it gives organizations a flexible platform to run their workloads on-premise or in the public cloud without making any changes to the application layer.

Native Solution

There are no immediate plans to discontinue Swarm despite the significant gap in popularity between it and Kubernetes.

Docker Enterprise Swarm features include:

- Use the Docker Engine CLI to create a swarm of Docker Engines where you can deploy application services.
- Deploy both kinds of nodes, managers, and workers using the Docker Engine. This means you can build an entire swarm from a single disk image.
- Docker Engine uses a declarative approach to define the desired state of the various services in the application stack.
- For each service, declare the number of tasks you want to run. When you scale up or down, the swarm manager automatically adapts by adding or removing tasks to maintain the desired state.
- The swarm manager node constantly monitors the cluster state and reconciles any differences between the actual state and your expressed desired state.
- Specify an overlay network for your services. The swarm manager automatically assigns addresses to the containers on the overlay network when it initializes or updates the application.
- Swarm manager nodes assign each service in the swarm a unique DNS name and load balances running containers.
- Expose the ports for services to an external load balancer. Internally, the swarm specifies how to distribute service containers between nodes.
- Each node in the swarm enforces TLS mutual authentication and encryption to secure communications between itself and all other nodes.
- Apply service updates to nodes incrementally at rollout time. The swarm manager lets you control the delay between service deployments to different sets of nodes.

Docker's official position is that there are plans to continue to evolve the Swarm platform, but no details to this end are being disclosed whatsoever.

The cluster management and orchestration features embedded in the Docker Engine are built using swarmkit. Swarmkit is a separate project which implements Docker's orchestration layer and is used directly within Docker.

A swarm consists of multiple Docker hosts which run in swarm mode and act as managers (to manage membership and delegation) and workers (which run swarm services). A given Docker host can be a manager, a worker, or perform both roles. A node is an instance of the Docker engine participating in the swarm. To deploy an application to a swarm, you submit a service definition to a manager node. The manager node dispatches units of work called tasks to worker nodes. Manager nodes also perform the orchestration and cluster management functions required to maintain the desired state of the swarm. Manager nodes elect a single leader via the raft gossip protocol to conduct orchestration tasks. Worker nodes receive and execute tasks dispatched from manager nodes. The worker node notifies the manager node of the current state of its assigned tasks so that the manager can maintain the desired state of each worker.

A service is the definition of the tasks to execute on the manager or worker nodes. It is the central structure of the swarm system and the primary root of user interaction with the swarm.

The swarm manager uses ingress load balancing to expose the services you want to make available externally to the swarm. External components, such as cloud load balancers, can access the service on the PublishedPort of any node in the cluster. This applies whether or not the node is currently running the task for the service. All nodes in the swarm route ingress connections to a running task instance.

Swarm mode has an internal DNS component that automatically assigns each service in the swarm a DNS entry. The swarm manager uses internal load balancing to distribute requests among services within the cluster based upon the DNS name of the service.

Comparison

The differences between Swarm and Kubernetes can be succinctly summarized in the following tradeoff: complexity versus features. Whereas Kubernetes boasts far more robustness and power than Swarm, Swarm is simpler to setup, deploy, and manage. However, given that Docker Enterprise (and others) simplifies these aspects of Kubernetes, the comparison then boils down to many features versus fewer features.

The future of Swarm is very unclear at the moment. The most likely hypothesis is that Docker will pivot it towards some other functionality, much like is occurring for Nomad and Cattle (see Rancher section). Perhaps the most convincing argument thus far for the retirement of Swarm is from enterprise companies that claim Swarm falls far short of their security requirements.

04

Rancher

Kubernetes Integration

With the upcoming release of Rancher 2.0, Rancher will be pivoting towards Kubernetes as the featured solution for its stack. Canonical and Rancher have partnered to develop a turn-key application delivery platform built on Ubuntu, Kubernetes, and Rancher 2.0. The new Cloud Native Platform will make it easy for users to deploy, manage, and operate containers on Kubernetes through a single workflow management portal from dev and test to production environments. Cloud Native Platform will simplify enterprise usage of Kubernetes with seamless user management, access control, and cluster administration.

Rancher 2.0 will be generally available early this year. Users can stand up and manage new Kubernetes clusters using Canonical's Kubernetes distribution, or a cloud-hosted Kubernetes service such as Amazon EKS, Azure ACS, or Google GKE. Rancher 1.0 is currently powering more than ten thousand Docker clusters, over a thousand of which are running Kubernetes. Beginning with Rancher 2.0, every cluster will now be based on Kubernetes. Rancher 2.0 will accelerate adoption of Kubernetes within the enterprise.

New features in Rancher 2.0 include:

- Manage Kubernetes everywhere: Rancher 2.0 enables users to manage existing Kubernetes clusters from cloud service providers.
- Multi-cluster management: Rancher 2.0 provides centralized man-agement of user authentication, monitoring, and health checks.
- Improved user experience: The Docker CLI and Docker Compose have been brought to Kubernetes.
- Enriched application catalog: The Rancher catalog has been extended to support Docker Compose, Kubernetes templates, and Helm charts.

Setting up Kubernetes from scratch is normally a non-trivial endeavor as it requires setting up etcd, networking plugins, DNS servers, and certificate author-ities. Rancher does all of this setup. All of these features mean that Kubernetes is also to support large and diverse workloads in a way that Docker Swarm is not ready for at the moment.

All pods in a deployment will be registered with a service as they come and go. Services also abstract away multiple deployments. Thus, if you want to run rolling deployments, you will register two Kubernetes deployments with the same service, then gradually add pods to one while reducing pods from the other. You can even do blue-green deployments where you point the service at a new Kubernetes deployment in one go.

Kubernetes supports Jobs, Scheduled Jobs, and Pet Sets. Jobs create one or more pods and wait until they terminate. A job makes sure that the specified number of pods terminate successfully. Jobs are great for issuing batch style work loads to your cluster (not services) that always need to be up and tasks that need to run to completion and then be cleaned up.

Pet Sets support stateful service workloads that are normally very difficult to containerize. This includes databases and real-time connected applications. Pet Sets provide stable hostnames for each pet and are indexed. Pet Sets also provide stable storage using persistent volumes and peer-discovery. Pet Sets provide startup and tear down ordering which is essential for persistent and scalable services. Pet Sets are one of the big differentiators for Kubernetes as persistent stateful workloads are almost impossible to run at production scale on Docker without this support.

Kubernetes also provides namespaces to isolate workloads on a cluster, secrets management, and auto-scaling.

Native Solution

With the current version of Rancher 1.x, users are able to choose between Mesos, Swarm, Kubernetes, and the intrinsic Cattle container orchestration solutions. In Rancher, everything is an API resource with a process lifecycle.

Rancher is built on top of containers: Web UI, API, Server that manages Rancher Agents, Database, Machine microservice - docker-machine binary.

RancherOS is a small distribution released by the Rancher team. It is an easy way to run containers at scale in production, and includes only the services needed to run Docker. It only includes the latest version of Docker and removes any unneeded library that a normal Linux distribution might have. In RancherOS, everything is a container. The traditional init system is replaced so that Docker runs directly on the Kernel.

Cattle is a container orchestration and scheduling framework. At its inception it was designed as an extension to Docker Swarm, but since then Cattle and Swarm have diverged. Cattle is used extensively by Rancher itself to orchestrate infrastructure services as well as setting up, managing, and upgrading Swarm, Kubernetes, and Mesos clusters.

Cattle application deployments are organized into stacks which can be used for User or Infrastructure stacks. A Cattle Stack is a collection of Services and the Infrastructure stack is primarily a Docker image with its networking, scalability, storage, health checks, service discovery links, environment, and all of the other configurations.

A Cattle Service could be a load balancer or an external service. A Stack can be launched using a docker-compose.yml, rancher-compose.yml, or by starting containers. In addition to this, you can start several applications using an application catalog. More than fifty different applications are available in the catalog.

Portainer is a lightweight management UI which allows you to easily manage your Docker host or Swarm cluster. Portainer is meant to be as simple to deploy as it is to use. It consists of a single container that can run on any Docker engine (Docker for Linux and Docker for Windows are supported). Portainer allows you to manage your Docker containers, images, volumes, and networks. It is compatible with the standalone Docker engine and with Docker Swarm.

Comparison

Rancher gives you the option of running Swarm, Mesos, Cattle, or Kubernetes as the container orchestration layer of its container management stack. However, Mesos is much better suited as a component of DC/OS. Swarm is certainly better suited as part of Rancher than individually. Cattle is not as sophisticated as any of the three largest competitors, but still enjoys good integrations into Rancher. However, all of these arguments are about to be moot. With the upcoming 2.0, Rancher will only support Kubernetes, and will be pivoting Cattle away from container orchestration and towards other functionality.

04 Azure

Kubernetes Integration

Azure is a cloud computing service created by Microsoft. Microsoft launched the former version of Azure Container Service (ACS) in 2015 with support for multiple container orchestrators. This has changed as they have standardized Azure Container Service around Kubernetes and have rebranded it as AKS. AKS also has the option to run open-source orchestrators, Docker Swarm, or even DC/OS from the Azure Marketplace.

Microsoft has recently released their version of managed Kubernetes on Microsoft Azure cloud services known as Azure Container Service (AKS). AKS differs from other managed services in that you pay for the agent nodes within your clusters and not for the masters. AKS provides version upgrades, patching, cluster scaling, and self-healing to the infrastructure. Microsoft has joined the Cloud Native Computing Foundation (CNCF) which has certified AKS as an official deployment method. As a fully managed product, AKS does not expose the masters to the customer which enables you to focus on the workloads. Once the cluster is created, API endpoints are automatically exposed to enable usage of kubectl, helm, or draft via the command line interface. Through the CLI you can deploy services like on a normal K8s cluster. When making a LoadBalancer type service, AKS negotiates with the Azure networking stack to create a Layer 4 load balancer. This load balancer is assigned a public IP address through which the service is exposed.

Native Solution

Even though Microsoft has rebranded their Azure Container Service, the legacy version is still available to customers. This allows for customers to deploy Docker Swarm, DC/OS, and open-source orchestrators from marketplaces templates. With these legacy options the customer is responsible for management and maintenance.

Comparison

With legacy ACS, you are able to use any orchestrator that you would choose; however, you are responsible for the management and deployment of the cluster if it is not available on the marketplace. Even though AKS abstracts away most of the maintenance functions of K8s, some customers may find it restricting. This is due to AKS only supporting K8s versions 1.7.7, 1.7.9, 1.8.1 and 1.8.2. In addition, AKS restricts networking to the Azure provided subnet. The only way around this is to use a custom versioning or network. Then you would have to deploy an unmanaged version of K8s.

04 Conclusion

At this juncture in time, the only competitor to Kubernetes that has not given in to customer demand by integrating Kubernetes is Hashicorp's Nomad. all other competitors have integrated Kubernetes and look to be supporting it as their primary platform moving forward. Since Kubernetes has become the default for container orchestration in these integrations, the question now becomes which pairing to use.

It is worth noting that these integrations all simplify the setup, deployment, and maintenance of Kubernetes. however, if one wishes to simplify the heavy lifting of Kubernetes without adding additional layers to the stack, then Tectonic by CoreOS becomes the easy enterprise choice. It simplifies Kubernetes setup, deployment, and management without adding any extra overhead to the solution stack.

If one wishes to use software integrated with Kubernetes for more than just simplifying usage, then the choice becomes dependent upon use case. For big data and machine learning, DC/OS is well regarded as the leading option for Kubernetes pairing due to its support of stateful applications with persistent data. For a developer-focused environment that does not rely so much on security as on simplicity, Swarm is the clear winner for Kubernetes pairing. For those looking for a DC/OS-lite option, Rancher is readily available to manage your Kubernetes clusters.

Regardless of the decision, the core software is Kubernetes, and will be for some time to come.

ABOUT SHADOW-SOFT

We are mavens in open software and open standards. It's how we help our customers "Take the Power Back" from vendors and own their technology future.

We do consulting and managed services across three specialties: DevOps, Application Infrastructure, and Cloud. We help customers modernize and optimize infrastructure, deploy Agile and DevOps methodologies, provide orchestration and automation, and maintain these environments on public or private clouds.

Call 770-546-0077 or email contact@shadow-soft.com