Comparison with Docker Swarm

In this lesson, we will compare Kubernetes Ingress to Docker Swarm equivalent.

The Similarities

Both Kubernetes and Docker Swarm have Ingress, and it might sound compelling to compare them and explore the differences. While that, at first glance, might seem like the right thing to do, there is a problem. Ingress works quite differently across the two.

Swarm Ingress networking is much more similar to Kubernetes Services. Both can, and should, be used to expose ports to clients both inside and outside a cluster. If we compare the two products, we'll discover that Kubernetes Services are similar to a combination of Docker Swarm's Overlay and Ingress networking. The Overlay is used to provide communication between applications inside a cluster, and Swarm's Ingress is a flavor of Overlay network that publishes ports to the outside world.

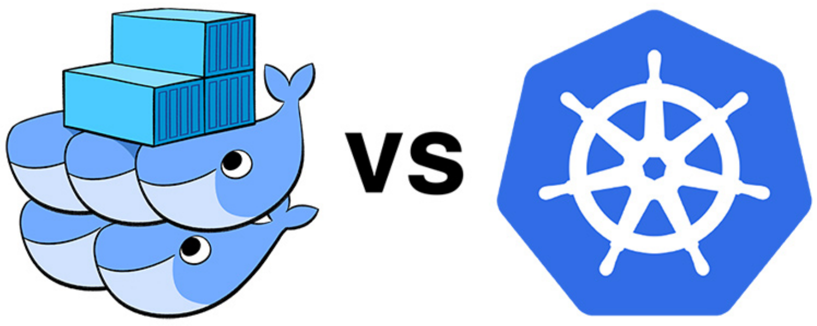
> The truth is that Swarm **does not** have an equivalent to Kubernetes Ingress Controllers. That is, *if we do not include Docker Enterprise Edition to the mix*.

The fact that a Kubernetes Ingress equivalent does not ship with Docker

Swarm does not mean that similar functionality cannot be accomplished through other means. It can. Traefik, for example, can act both as a

Kubernetes Ingress Controller, as well as a dynamic Docker Swarm proxy. It provides, more or less, the same functionality no matter which scheduler you choose. If you're looking for a Swarm specific alternative, you might choose Docker Flow Proxy.

All in all, as soon as we stop comparing Ingress on both platforms and start looking for a similar set of functionality, we can quickly conclude that both Kubernetes and Docker Swarm allow a similar set of features. We can use paths and domains to route traffic from a single set of ports (e.g., `80` and `443`) to a specific application that matches the rules. Both allow us to offload SSL certificates, and both provide solutions that make all the necessary configurations dynamically.



The Differences

If on the functional level both platforms provide a very similar set of features, can we conclude that there is no essential difference between the two schedulers when taking into account only dynamic routing and load balancing? *We would say no*. Some important differences might not be of functional nature.

The Ingress API

Let's discuss Kubernetes and Docker Swarm based on the Ingress API.

Kubernetes

Kubernetes provides a well-defined Ingress API that third-party solutions can utilize to deliver a seamless experience. Let's take a look at one example.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: devops-toolkit
spec:
  rules:
  - host: devopstoolkitseries.com
    http:
      paths:
      - backend:
          serviceName: devops-toolkit
          servicePort: 80
```

This definition can be used with many different solutions. Behind this Ingress resource could be *nginx, voyager, haproxy, or trafficserver* Ingress Controller. All of them use the same Ingress API to deduce which Services should be used by forwarding algorithms. Even Traefik, known for its incompatibility with commonly used Ingress annotations, would accept that YAML definition.

Having a well-defined API still leaves a lot of room for innovation. We can use `annotations` to provide the additional information our Ingress Controller of choice might need. Some of the same annotations are used across different solutions, while the others are specific to a Controller.

All in all, Kubernetes Ingress Controller combines a well-defined (and simple) specification that all Ingress Controllers must accept and, at the same time, it leaves ample room for innovation through custom `annotations` specified in `metadata`.

Docker Swarm

Docker Swarm does not have anything resembling an Ingress API. Functionality similar to Kubernetes Ingress Controllers can be accomplished either by using Swarm Kit or using the Docker API. The problem is that there is no defined API that third-party solutions should follow, so each is a world in itself. For example, understanding how Traefik works will not help you much when trying to switch to Docker Flow Proxy. Each is operated differently in isolation. There is no standard because Docker did not focus on making one.

Docker's approach to scheduling is based entirely on the features baked into Docker Server. There is only one way to do things. Often, that provides a very user-friendly and reliable experience. If Swarm does what you need it to do, it is an excellent choice. However, the problem occurs when you need more. In

that case, you might experience difficulties finding a solution with Docker Swarm.

> If we limit the comparison to Kubernetes Ingress Controllers and their equivalents in Docker Swarm, the former is a clear winner.

Assuming that the current strategy continues, Docker would need to add layer 7 forwarding into Docker Server if it is to get back in the game on this front. If we limit ourselves only to this set of features, Kubernetes wins through its Ingress API that opened the door, not only to internal solutions, but also to third-party Controllers.

We are still at the beginning. There are many more features worth comparing. We only scratched the surface. Stay tuned for more.

Let’s move to the next chapter and learn how to access the host’s file system using Volumes.