FEATURE BRIEF

Data Center Solutions
Kubernetes with Container Bare Metal

Multiple Network Interfaces

Multiple network interface connections in Kubernetes pods are important when enabling delivery of cloud and communication service provider services

Kubernetes is the leading open source container orchestration engine. Standard Kubernetes pods allow for connection to one network interface. But these pods can host containers supporting service-provisioning applications or virtual network functions (VNFs) that are typically requiring connectivity to multiple network interfaces. The following are a few examples of applications that typically require connectivity to multiple network interfaces:

- Storage/legacy applications: Multiple connections are needed to allow the service provisioned within the pod to access a legacy application or storage drive and, at the same time, support pod communications.
- Split data plane/control plane applications: This use case is most applicable to VNF-based services because the VNF must connect to both the data plane and the control plane (and possibly require a separate management connection). This is very important for applications that need an extra high-performance connection such as IPTV or media streaming.
- Virtual private network (VPN)/router applications: Multiple network interfaces are essential for VPN and router use cases where security capabilities need to be extended into the pod. In these cases, the VPN/router links are encrypted, while others are unencrypted for normal pod communications.
- Multi-tenant networks: If pod resources are shared between multiple tenants, multiple interfaces are needed to enable dedicated network connections for each of those customers.

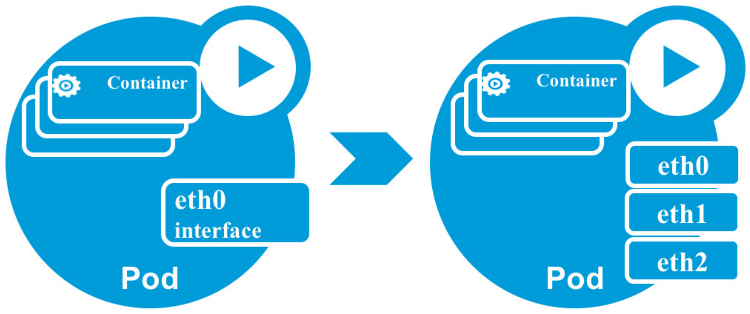

Figure 1: Standard Kubernetes pod (left) showing a single network interface shared among the containers within the pod; and (right) a Kubernetes pod with multiple network interfaces, the capability enabled with Multus.

Multiple network interfaces are available through the Multus CNI plugin

Kubernetes natively supports only one network interface and proposals to support multiple network interfaces are being discussed currently in the community. Today, however, it's possible to implement multiple network interfaces using Multus, a Kubernetes CNI plugin that enables the creation of additional pod network interfaces. Figure 1 shows containers in a pod sharing one interface and then a pod with multiple network interface with Multus CNI.

With Multus, other CNI plugins can create network connections

Multus is a CNI proxy and arbiter of other CNI plugins. Multus invokes other CNI plugins for network interface creation. When Multus is used, a master plugin (flannel, Calico, weave) is identified to manage the primary network interface (eth0) for the pod and it is returned to Kubernetes.Other CNI minion plugins (SR-IOV, vHost CNI, etc.) can create additional pod interfaces (net0, net1, etc.) during their normal instantiation process.

Figure 2 shows the network control flow with Multus. The Kubernetes Kubelet is the primary agent that runs on each node in the Kubernetes cluster. Its main functions are to register the node with the Kubernetes control plane and life cycle management of any pods that are subsequently scheduled to that node.

Kubelet is responsible for establishing the network interfaces for each pod; it does this by invoking its configured CNI plugin. When Multus is invoked, it recovers pod annotations related to Multus, in turn, then it uses these annotations to recover a Kubernetes custom resource definition (CRD), which is an object that informs which plugins to invoke and the configuration needing to be passed to them. The order of plugin invocation is important as is the identity of the master plugin.

In the right side of the figure 2, we see the benefit in a virtual firewall (vFW) use case. By using the SR-IOV CNI plugin in DPDK mode, the vFW can get full-speed line rate packet interfaces to the networks on which it is expected to perform its function. Additionally, there exists the management and control eth0 interface, which is available for control of the vFW itself and also possibly other functions, such as logging whose job may be to scrape the vFW logs and export via the management network interface to a centralized logging service.

Conclusion

Support for multiple network interfaces is becoming more important as Kubernetes platforms are used in Communication Service Providers and cloud networks. While a native solution is now being discussed, Multus provides this multiple interface support today with an easy CNI plugin that works with all other Kubernetes networking plugins. For more information on what Intel is doing with containers, go to https://networkbuilders.intel.com/network-technologies/intel-container-experience-kits.

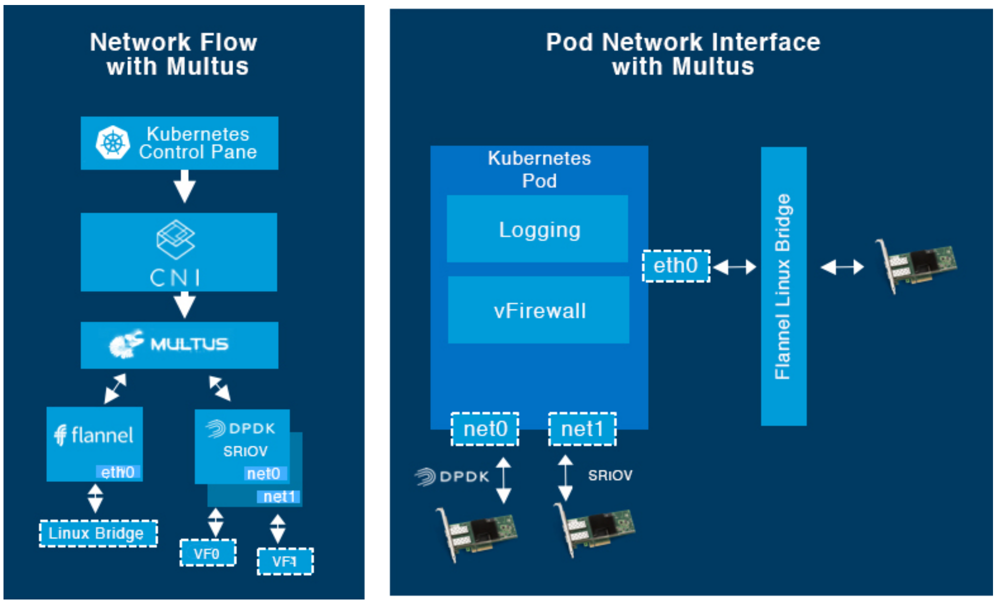

Figure 2: Network control flow with Multus.