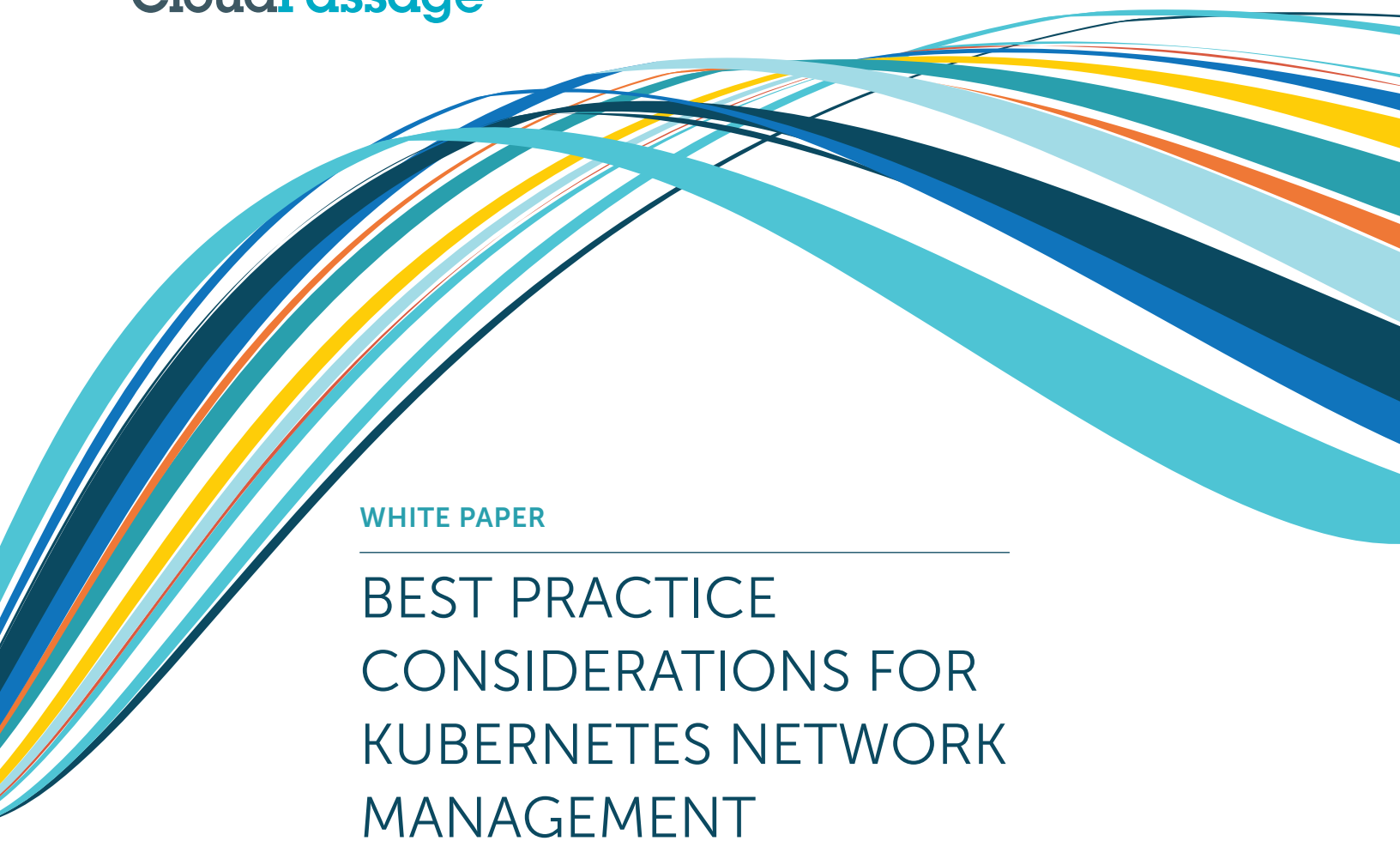
CloudPassage



WHITE PAPER

BEST PRACTICE CONSIDERATIONS FOR KUBERNETES NETWORK MANAGEMENT

Table of contents

This paper provides guidelines and recommendations for selecting and planning the implementation of Kubernetes-managed overlay networks. This paper examines the implementation of the Flannel, a popular overlay network, within the context of the design considerations set forth in this document. The intended audience for this guide are individuals involved in the architecture or implementation of Kubernetes. This guide focuses on Kubernetes deployed within the context of a private data center.

The number of tools available for IP address management, network traffic flow management, and segmentation in a containerized application is enormous. New tools for managing overlay networks emerge frequently, and tracking evolving feature sets can be a prohibitively time-consuming task. This paper provides guidance for tool selection and considerations for architecture and implementation. While this document does not cover every possible network configuration for Kubernetes clusters, following these guidelines will enable more graceful scaling from an operational standpoint, and higher-resolution observables for better-quality security visibility.

Networking in Kubernetes

The basic unit of a Kubernetes deployment is called a pod. A pod is a collection of resources that includes storage, a virtual network interface, and at least one container. Nodes are the hosts that run Kubernetes pods. Kubernetes assigns a unique IP address to each pod, and each pod in a Kubernetes cluster can reach every other pod – as well as every Kubernetes application node – via a simple IP routing. Favoring simplicity and the avoidance of unnecessary complication, Network Address Translation (NAT) is not used in container-to-container, node-to-container, or container-to-node communication. This design allows a great deal of flexibility and simplicity.

IP address management

The rules for IP address management haven't changed with the introduction of overlay networks. Maintaining unique IP addresses for all networked workloads in an enterprise is essential for the correct and unambiguous operation of network security tools – from network flow message generators to traffic inspection appliances. Being able to positively correlate an IP address and a timestamp to a single workload is absolutely essential for avoiding unnecessary work in digital forensics and incident response activities.

Exhaustion of RFC1918 address space within an enterprise should bring the enterprise to the threshold of IPv6 adoption, rather than the creation of isolated network enclaves and re-use of already-allocated subnets. Protecting enclaves with overlapping IP space can require the duplicate implementation of detection tools and extra work for automating the contextualization of input from the duplicated toolsets. Holding to good IP address allocation practices allows the business to avoid the unnecessary expense associated with licensing duplicated tools and the human labor overhead of managing otherwise unnecessary systems.

Planning network addressing and topology with respect to route convergence in the data center is essential. The time and network device compute resources required for a network to automatically converge in the event of a link or router failure will be impacted by the efficiency of the overall design of the network. The fewer routes that need to be converged in the event of a layer-3 device failure, the better – so design your network with convergence in mind and tend toward summarized (or summarizable) routes whenever possible.

This paper provides guidelines and recommendations for designing Kubernetes-managed overlay networks. The intended audience for this guide are individuals involved in the architecture or implementation of Kubernetes. This guide focuses on Kubernetes deployed within the context of a private data center.

Overlay networks

Overlay networks share some high-level characteristics with more traditional technologies. For instance, implementing VPLS over MPLS allows the extension of a layer-2 network over multiple sites. While this can provide convenience in certain circumstances, it can also add confusion if an application workload expects the same latency from all members of a subnet – because members of the same subnet could, in fact, be located at completely separate physical sites. Similar challenges can exist with VXLAN, a popular standard for implementing distributed overlay networks. This should be taken into consideration when designing the overlay networks that serve an application. Other layer-2 considerations apply here as well, like the potential for broadcast storms. In addition to the issues that can be inherent in certain network design patterns, the back-end configuration for overlay networks can have performance trade-offs. Research your overlay network's back-end configuration options (including potential integrations with network hardware SDN management systems) to ensure that you're picking the best back-end for your network design requirements.

Consistency and reliability

Ensuring consistent performance and reliability for a Kubernetes cluster requires that close attention be paid to the underlying enterprise network. The overlay network cannot reconverge faster than the underlying enterprise network, so design your cluster network with this in mind. Revisiting application design and pod scheduling characteristics across nodes may be necessary during this phase of the design process to ensure the survivability of the application in the event of an enterprise network service outage or degradation.

Segmentation and policy enforcement

Network policy enforcement within a Kubernetes application must work closely with pod scheduling and the overlay network management components, to ensure that accessibility is effectively managed through scaling operations. This may seem like an aspect to cluster configuration that should just work. However, as the number of nodes in the application grows, even monitoring tools can cause problems if the polling interval is too aggressive (*https://blog.openai.com/scaling-kubernetes-to-2500-nodes/#etcd*). If your configuration

state is unavailable, your segmentation and policy enforcement tools will certainly suffer. Understand the limitations of your cluster configuration and have a plan for crossing a difficult application size threshold long before the need presents itself.

Focus on Flannel

Performance and scale

Two of Flannel's most common options for back-end configuration are VXLAN and UDP. Always choose VXLAN over UDP for the Flannel back-end when implementing in a private data center. UDP should only be used for troubleshooting, as the latency and throughput characteristics for the UDP back-end are insufficient for production use.

As mentioned in a prior section, scaling characteristics depend greatly on the performance of etcd, so ensuring that etcd has the resources to scale is absolutely critical. Remember to allocate Flannel's subnet as a part of the greater enterprise IP addressing plan in order to ensure the unambiguous operation of network management and monitoring tools.

Consistency and reliability

Flannel typically stores its configuration information via the Kubernetes API, which is backed by etcd. Again, etcd performance is absolutely critical to making your Kubernetes and Flannel cluster run smoothly. Ensure that only the workloads and processes that should be interacting with etcd are able to reach the etcd service ports.

Segmentation options for Flannel

There should never be more than one management tool for network traffic policy enforcement within a single Kubernetes cluster. Multiple visibility tools may be required, but Kubernetes must be able to manage its own network flow and segmentation. Network traffic policy enforcement within the underlying enterprise network should still be employed to protect against unauthorized access to the Kubernetes cluster. At the time of this writing, a strong emerging solution for combining overlay network management and traffic policy enforcement is Canal. Canal is a deployment pattern for running Calico alongside Flannel, which allows the simplified deployment of both Flannel and Calico and allows Kubernetes to manage its own overlay network as well as policy enforcement within the Kubernetes cluster.

CloudPassage Halo for security visibility:

CloudPassage® Halo® can be used for visibility into network traffic flow between Kubernetes nodes, as well as between a Kubernetes cluster and the broader enterprise network, and the Internet at large. Because Halo installs into the workload, it is well-suited to be an up-to-date source of truth for dynamic workload attributes, like IP addresses, which provide the high-quality information required for automated network management and continuous compliance in an application environment designed for rapid elasticity.

Summary

The same rules of good application design for distributed, containerized applications apply to Kubernetes-managed applications. Kubernetes can make application survivability and elasticity easier to achieve but implementing monolithic design patterns can make these desirable aspects unnecessarily difficult to achieve. Plan to scale and know where your weaknesses are, especially from a scalability perspective. Ensure that your Kubernetes and supporting application infrastructure is resilient in the event of underlying service or network degradation. From an IP address management perspective, create the Kubernetes cluster as a part of the enterprise IP address space – a true participant in the business technology ecosystem. Start with high-quality application performance and security visibility tooling from the start. If these points are well-understood and codified at the onset of the project, (before production implementation begins) it will be easier to load-test ahead of necessary scaling events and protect the application environment over time.

ABOUT CLOUDPASSAGE

Founded in 2010, CloudPassage® was the first company to obtain a U.S. patent for universal cloud infrastructure security and has been a leading innovator in cloud security automation and compliance monitoring for high-performance application development and deployment environments. CloudPassage Halo® is an award-winning workload security automation platform that provides universal visibility and continuous protection for servers in any combination of data centers, private/public clouds and containers. The Halo platform is delivered as a service, so it deploys in minutes and scales effortlessly. Fully integrated with popular infrastructure automation and orchestration tools such as Puppet and Chef, as well as leading CI/CD tools such as Jenkins, Halo secures the enterprise where it's most vulnerable—application development and workload deployment. Today, CloudPassage Halo secures the critical infrastructure of many of the leading global finance, insurance, media, ecommerce, high-tech service providers, transportation and hospitality companies.

CloudPassage

www.cloudpassage.com | 800.215.7404