Kubernetes Networking Demystified: A Brief Guide

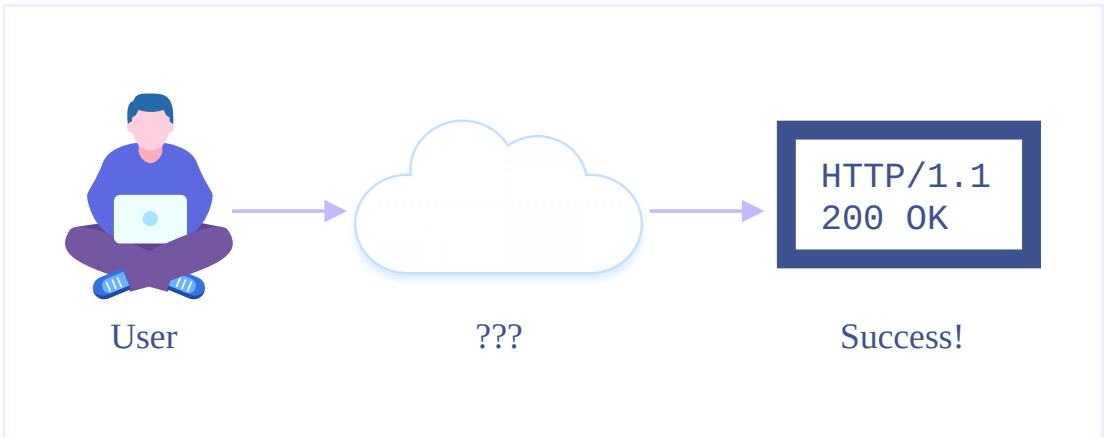
Karen Bruner (/authors/kbruner/) Jan 09, 2020

Kubernetes cluster networking can be more than a bit confusing, even for engineers with hands-on experience working with virtual networks and request routing. In this post, we will present an introduction into the complexities of Kubernetes networking by following the journey of an HTTP request to a service running on a basic Kubernetes cluster.

We will use a standard Google Kubernetes Engine (GKE) cluster with two Linux nodes for our examples, with notes on where the details might differ on other platforms.

A Request's Journey

Take the typical example of a person browsing the web. They click on a link, something happens, and a page loads.



FEATURED POSTS


Enhancing Kubernetes Security with Open Policy Agent (OPA) - Part 1 (/post/2020/04/enhancing-kubernetes-security-with-open-policy-agent-opa-part-1/)
Apr 29, 2020


AWS EKS Monitoring Best Practices for Stability and Security (/post/2020/04/aws-eks-monitoring-best-practices-for-stability-and-security/)
Apr 14, 2020

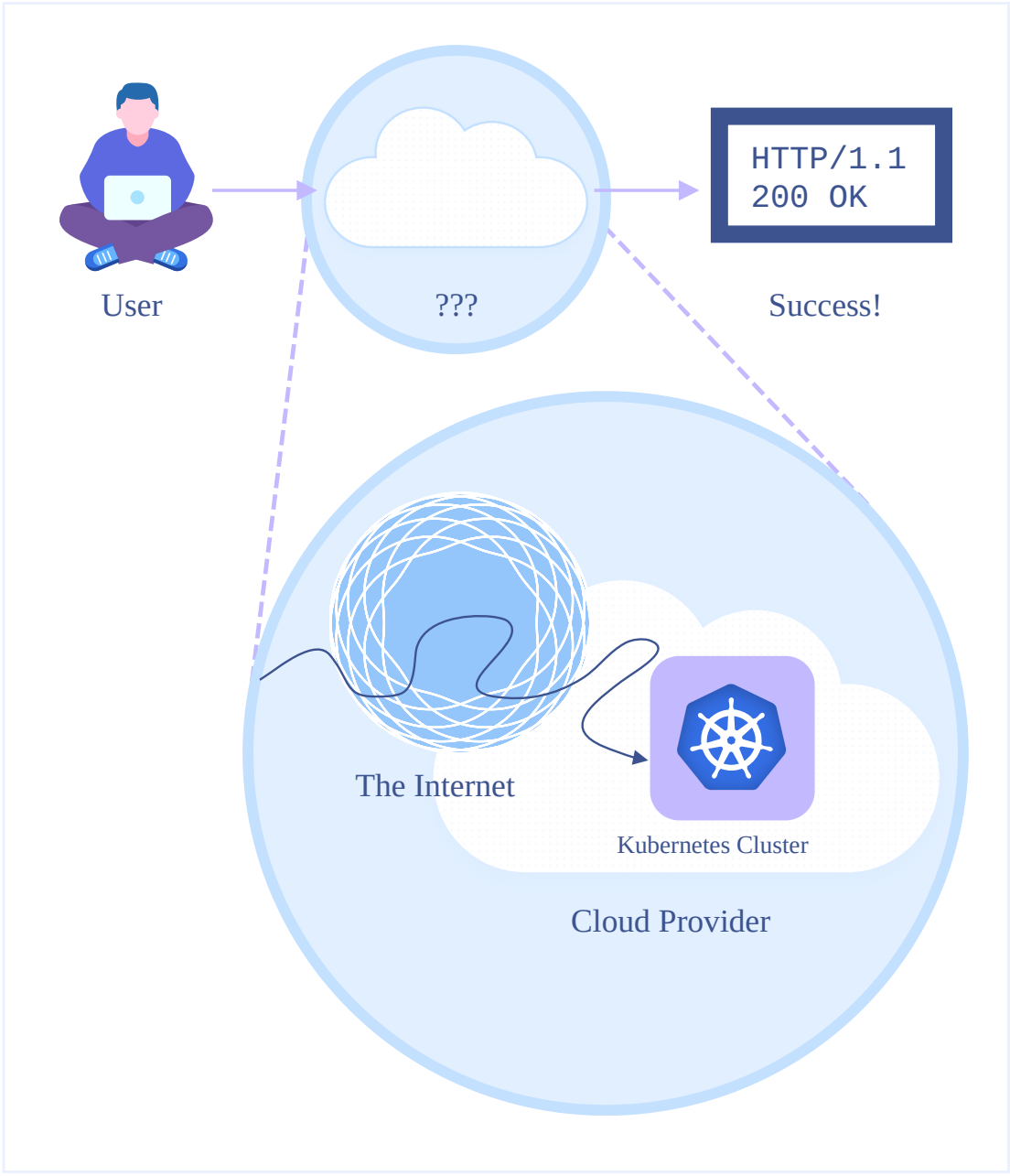

Container Image Security: Beyond Vulnerability Scanning (/post/2020/04/container-image-security-beyond-vulnerability-scanning/)
Apr 08, 2020


EKS Runtime Security Best Practices for AWS

We need to fill in those question marks a bit.

(/)

ext diagram, the request gets sent through the Internet to a very large cloud provider, then to a
ubernetes cluster hosted in the cloud provider's infrastructure.



Workloads (/post/2020/04/eks-runtime-security-best-practices-for-aws-workloads/)
Apr 06, 2020


EKS vs GKE vs AKS - April 2020 Updates (/post/2020/03/eks-vs-gke-vs-aks-april-2020-updates/)
Mar 31, 2020


EKS Networking Best Practices for Security and Operation (/post/2020/03/eks-networking-best-practices/)
Mar 30, 2020


Securing EKS Cluster Add-ons: Dashboard, Fargate, EC2 Components, and More (/post/2020/03/securing-eks-cluster-add-ons-dashboard-fargate-ec2-and-more/)
Mar 24, 2020


What's New in Kubernetes 1.18? New Features and Updates (/post/2020/03/what-is-new-in-kubernetes-1.18/)
Mar 23, 2020


Guide to Designing EKS Clusters for Better Security (/post/2020/03/guide-to-eks-cluster-design-for-better-security/)
Mar 17, 2020


Azure Kubernetes (AKS) Security Best Practices Part 4 of 4: Cluster Maintenance (/post/2020/03/azure-kubernetes-aks-security-best-practices-part-4-of-4/)
Mar 09, 2020


Azure Kubernetes (AKS) Security Best Practices Part 3 of 4: Runtime Security (/post/2020/02/azure-kubernetes-aks-security-best-practices-part-3-of-4/)
Feb 24, 2020

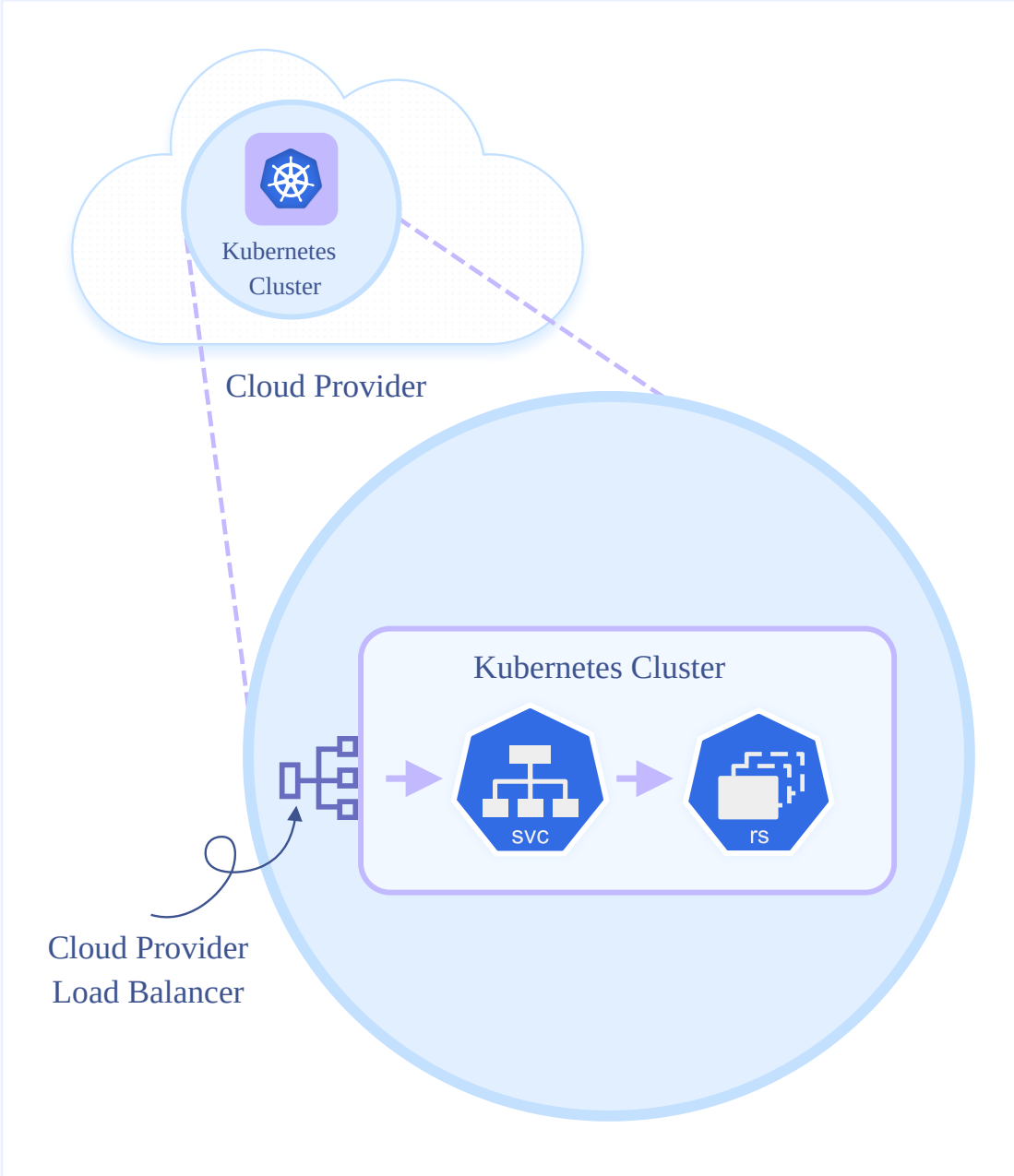

Azure Kubernetes (AKS) Security Best Practices Part 2 of 4: Networking (/post/2020/02/azure-kubernetes-aks-security-best-practices-part-2-of-4/)
Feb 11, 2020


EKS vs GKE vs AKS - Evaluating Kubernetes in the Cloud

/

(/)

Kubernetes `Service` resource, which then routes requests to pods in a Kubernetes `ReplicaSet`.



We can deploy the following YAML to create the Kubernetes `Service` (`svc`) and `ReplicaSet` (`rs`):

(/post/2020/02/eks-vs-gke-vs-aks/)
Feb 10, 2020


Azure Kubernetes (AKS) Security Best Practices Part 1 of 4: Designing Secure Clusters and Container Images (/post/2020/01/azure-kubernetes-aks-security-best-practices-part-1-of-4/)
Jan 27, 2020


Guide to Kubernetes Egress Network Policies (/post/2020/01/kubernetes-egress-network-policies/)
Jan 15, 2020


Top 5 Kubernetes Vulnerabilities of 2019 - the Year in Review (/post/2020/01/top-5-kubernetes-vulnerabilities-of-2019-the-year-in-review/)
Jan 02, 2020


What's New in Kubernetes 1.17: A Deeper Look at New Features (/post/2019/12/whats-new-in-kubernetes-1.17-a-deeper-look-at-new-features/)
Dec 09, 2019


How to Make Istio Work with Your Apps (/post/2019/11/how-to-make-istio-work-with-your-apps/)
Nov 26, 2019


Protecting Kubernetes API Against CVE-2019-11253 (Billion Laughs Attack) and Other Vulnerabilities (/post/2019/09/protecting-kubernetes-api-against-cve-2019-11253-billion-laughs-attack/)
Sep 30, 2019


12 Kubernetes configuration best practices (/post/2019/09/12-kubernetes-configuration-best-practices/)
Sep 26, 2019


Docker Container Security 101: Risks and 33 Best Practices (/post/2019/09/docker-security-101/)
Sep 13, 2019


Amazon EKS Security Best Practices

/

```
apiVersion: apps/v1
  d: ReplicaSet
metadata:
  name: hello-world
  labels:
    app: hello-world
spec:
  selector:
    matchLabels:
      app: hello-world
  replicas: 2
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
      - name: hello-world
        image: gcr.io/google-samples/node-hello:1.0
        imagePullPolicy: Always
        ports:
        - containerPort: 8080
          protocol: TCP


---
apiVersion: v1
kind: Service
metadata:
  name: hello-world
spec:
  selector:
    app: hello-world
  ports:
  - port: 80
    targetPort: 8080
    protocol: TCP
  type: LoadBalancer
  externalTrafficPolicy: Cluster
```

These manifests should result in the creation of two pods as part of the `hello-world ReplicaSet`, and a `hello-world` service resource with an external-facing load balancer, if the cloud provider and cluster network supports it. It should also create a Kubernetes `Endpoint` resource with two entries in the `host:port` notation, one for each of the pods, with the pod IP as the host value and port 8080.

(/post/2019/09/amazon-eks-security-best-practices/)
Sep 13, 2019


The Kubernetes Security Audit: 3 Key Takeaways (/post/2019/09/the-kubernetes-security-audit-3-key-takeaways/)
Sep 02, 2019


Istio Security: Running Microservices on Zero-Trust Networks (/post/2019/08/istio-security-basics-running-microservices-on-zero-trust-networks/)
Aug 01, 2019


Kubernetes Security 101: DevOps' Guide to Securing K8s (/post/2019/07/kubernetes-security-101/)
Jul 22, 2019


Gartner best practices for Kubernetes & container security (/post/2019/06/gartner-best-practices-for-securing-containers-and-kubernetes-in-production/)
Jun 25, 2019


Gartner's Top 10 Security Projects for 2019 - Container Security Makes the List (/post/2019/06/gartners-top-10-security-projects-for-2019-container-security-makes-the-list/)
Jun 24, 2019

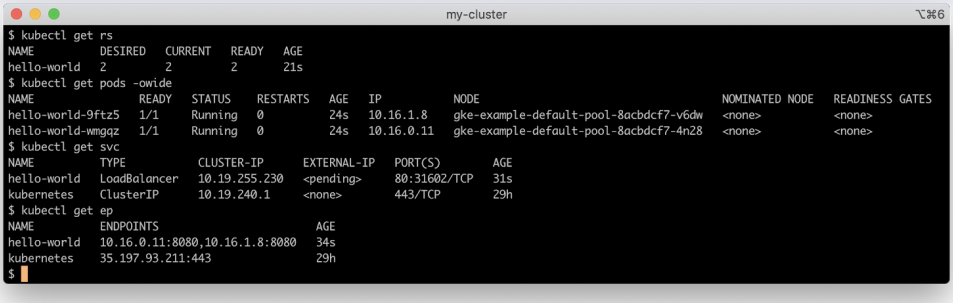

Getting started with Istio Service Mesh - What is it and what does it do? (/post/2019/06/getting-started-with-istio-service-mesh-what-is-it-and-what-does-it-do/)
Jun 20, 2019


What the Palo Alto-Twistlock Acquisition Means for Container Security Market (/post/2019/05/what-the-palo-alto-twistlock-acquisition-means-for-container-security-market/)
May 29, 2019


How to Build Production-Ready Kubernetes Clusters and Containers (/post/2019/05/how-to-build-production-ready-

On our GKE cluster, querying these resource types with `kubectl` returns the following:

(/)



For reference, our cluster has the following IP networks:

Node - 10.138.15.0/24

Cluster - 10.16.0.0/14

Service - 10.19.240.0/20

Our service has a Virtual IP address (VIP) of 10.19.240.1 in the cluster CIDR block.

We are now ready to follow the request's journey into the Kubernetes cluster, beginning at the load balancer.

The Load Balancer

While Kubernetes, both natively and through ingress controllers, offers a number of ways to expose a service, we will use the standard `Service` resource of type `LoadBalancer`. Our `hello-world` service needs a GCP network load balancer ⧉. Every GKE cluster has a cloud controller ⧉, which interfaces between the cluster and the API endpoints for GCP services needed to create cluster resources automatically, including our load balancer. (All cloud providers offer different classes of load balancers with varying options and characteristics.)

To see where the external load balancer fits in, first we need to look at the cluster from a different viewpoint.

kubernetes-clusters-and-containers/)
May 09, 2019


Guide to Kubernetes Ingress Network Policies (/post/2019/04/setting-up-kubernetes-network-policies-a-detailed-guide/)
Apr 05, 2019


New Kubernetes Security Vulnerabilities Disclosed: CVE-2019-1002101 and CVE-2019-9946 (/post/2019/03/new-kubernetes-security-vulnerabilities-discovered-cve-2019-1002101-and-cve-2019-9946/)
Mar 29, 2019


11 Kubernetes admission controller best practices for security (/post/2019/03/11-tips-to-operationalizing-kubernetes-admission-controllers-for-better-security/)
Mar 22, 2019


7 Critical Kubernetes Security Issues Resolved by Upgrading Your k8s (/post/2019/01/critical-kubernetes-security-issues-resolved-in-recent-kubernetes-versions/)
Jan 03, 2019


Detecting Docker Exploits and Vulnerabilities - Your How-to Guide (/post/2018/03/breaking-bad-detecting-real-world-container-exploits/)
Mar 08, 2018


Docker Forensics for Containers: How to Conduct Investigations (/post/2017/08/csi-container-edition-forensics-in-the-age-of-containers/)
Aug 17, 2017


Hardening Docker containers, images, and host - security toolkit

hosts-against-

urity-

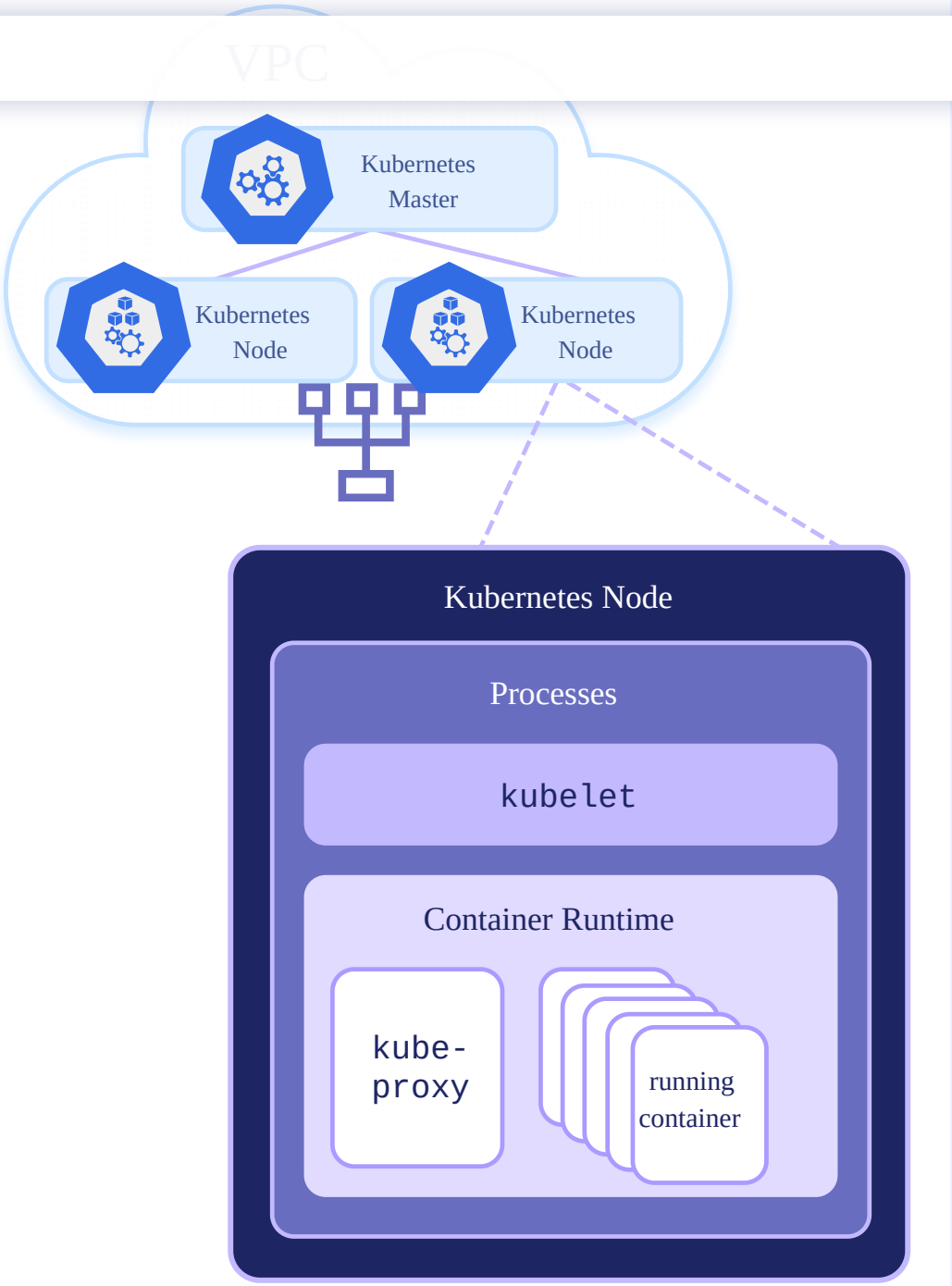
Aug 10, 2017

SUBSCRIBE TO OUR BLOG

Email Address:

SUBSCRIBE ME

/



kube-proxy

Each node has a `kube-proxy` container process. (In the Kubernetes frame of reference, that `kube-proxy` container is in a pod in the `kube-system` namespace.) `kube-proxy` manages forwarding of traffic addressed to the virtual IP addresses (VIPs) of the cluster's Kubernetes `Service` objects to the appropriate backend pods. `kube-proxy` currently supports three different operation modes:

User space: This mode gets its name because the service routing takes place in `kube-proxy` in the user process space instead of in the kernel network stack. It is not commonly used as it is slow and

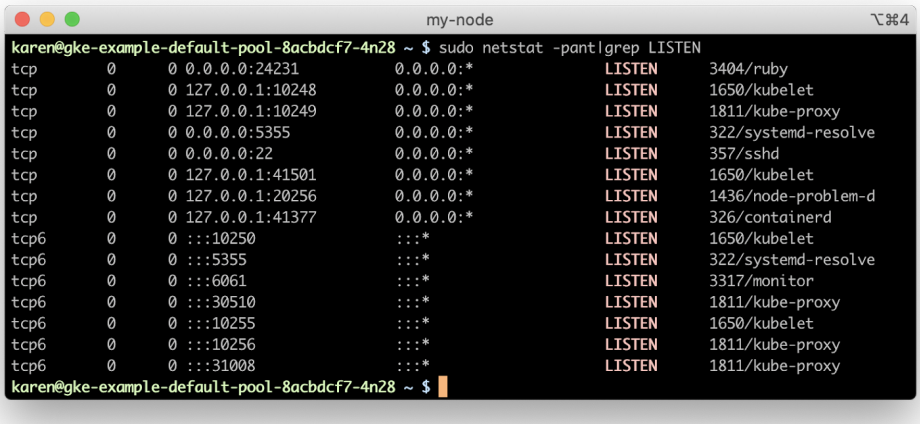
(/)

les: This mode uses Linux kernel-level Netfilter rules to configure all routing for Kubernetes Services. This mode is the default for `kube-proxy` on most platforms. When load balancing for multiple backend pods, it uses unweighted round-robin scheduling.

IPVS (IP Virtual Server): Built on the Netfilter framework, IPVS implements Layer-4 load balancing in the Linux kernel, supporting multiple load-balancing algorithms, including least connections and shortest expected delay. This `kube-proxy` mode became generally available in Kubernetes 1.11, but it requires the Linux kernel to have the IPVS modules loaded. It is also not as widely supported by various Kubernetes networking projects as the iptables mode.

`kube-proxy` in our GKE cluster runs in iptables mode, so we will look at how that mode works.

If we look at the `hello-world` service we created, we can see that it has been assigned a node port (a network port for the node's IP address) of `30510`. Dynamically-assigned ports on the node network allow multiple Kubernetes services hosted in the cluster to use the same Internet-facing port in their endpoints. If our service had been deployed to a standard Amazon Elastic Kubernetes Service (EKS) cluster, it would be served by an Elastic Load Balancer which would send incoming connections to our service's node port on nodes with a live service pod. However, Google Cloud Platform (GCP) network load balancers only forward traffic to the targets on the same port as the incoming port on the load balancer, i.e., traffic to port 80 on the load balancer will be sent to port 80 on the target backend instance. The `hello-world` pods are definitely not listening on port 80 of the node. If we run `netstat` on the node, we see that no process is listening on that port.

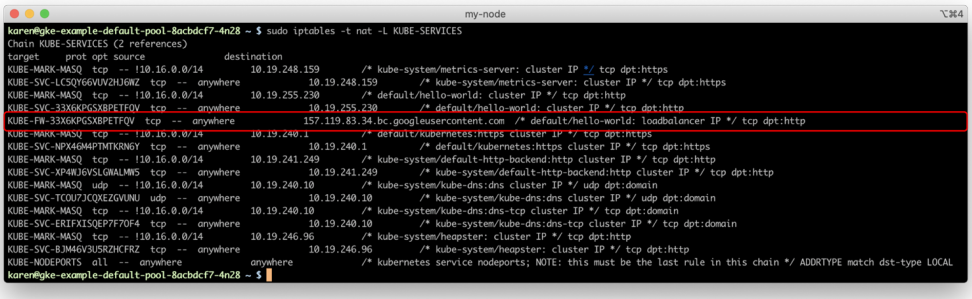


So how do requests through the load balancer make a successful connection? If `kube-proxy` were running in the user space mode, it would actually be proxying connections to backend pods. In iptables mode, though, `kube-proxy` configures Netfilter chains so the connection is routed directly to the backend container's endpoint by the node's kernel.

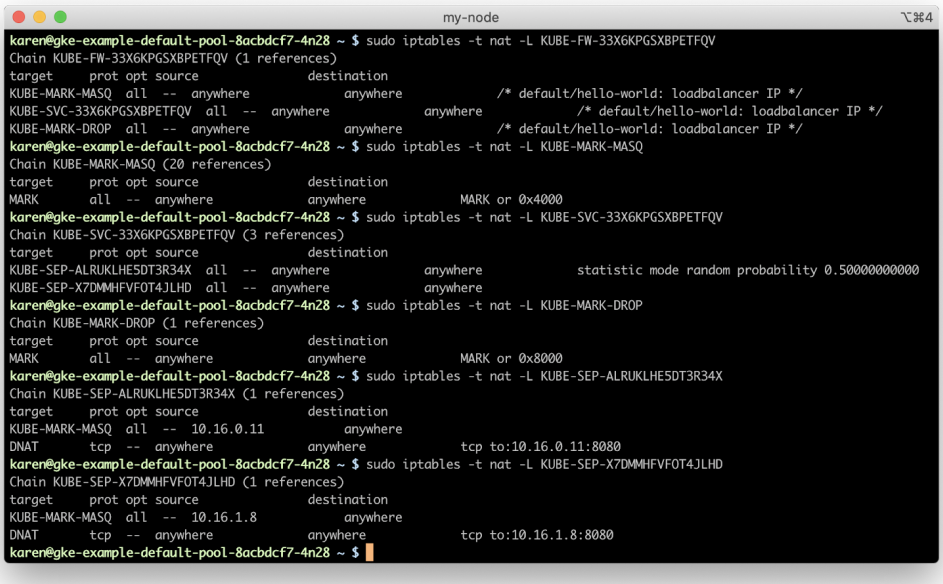
/

KE cluster, if we log in to one of the nodes and run iptables, we can see these rules.

(/)



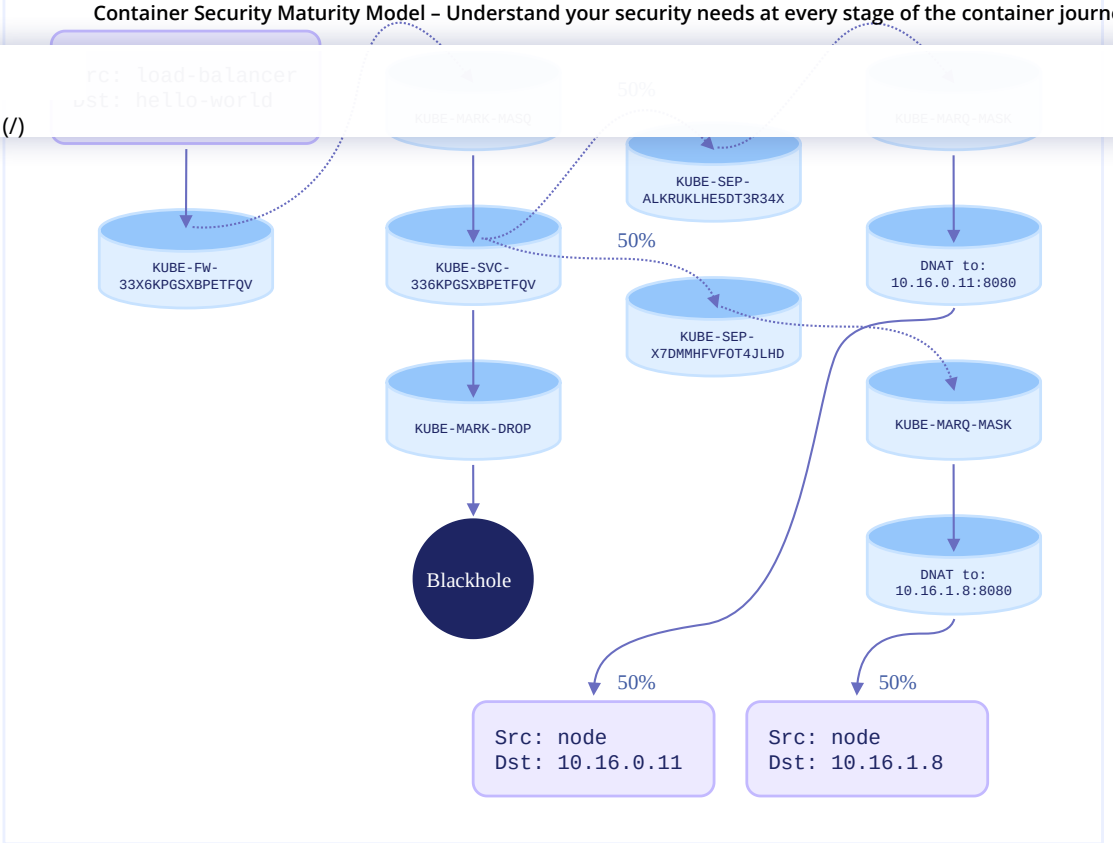
Thanks to the rule comments, we can get the name of the filter chain that matches incoming connections from the service's load balancer to our `hello-world` service and follow that chain's rules. (In the absence of a rule comment, we still could have matched the rule's source IP address to the service's load balancer.)



We can also visualize the chains and rules used in the network stack for evaluating and modifying the packet to see how the service we created in our cluster directs traffic to the replica set members.

/

(/)



The `KUBE-FW-33X6KPGSXBPETFQV` chain has three rules, each adding another chain for handling the packet.

1. `KUBE-MARK-MASQ` adds a Netfilter mark to packets destined for the `hello-world` service which originate outside the cluster's network. Packets with this mark will be altered in a `POSTROUTING` rule to use source network address translation (SNAT) with the node's IP address as their source IP address.

2. The `KUBE-SVC-33X6KPGSXBPETFQV` chain applies to all traffic bound for our hello-world service, regardless of source, and has rules for each of the service endpoints (the two pods, in this case). Which endpoint chain to use gets determined in a purely random fashion.
   1. `KUBE-SEP-ALRUKLHE5DT3R34X`
      1. `KUBE-MARK-MASQ` again adds a Netfilter mark to the packet for SNAT, if needed
      2. The `DNAT` rule sets up a destination NAT using the 10.16.0.11:8080 endpoint as the destination.
   2. `KUBE-SEP-X7DMMHFVFOT4JLHD`
      1. `KUBE-MARK-MASQ` again adds a Netfilter mark to the packet for SNAT, if needed
      2. The `DNAT` rule sets up a destination NAT using the 10.16.1.8:8080 endpoint as the destination.

3. `KUBE-MARK-DROP` adds a Netfilter mark to packets which do not have destination NAT enabled by this point. These packets will be discarded in the `KUBE-FIREWALL` chain.

/

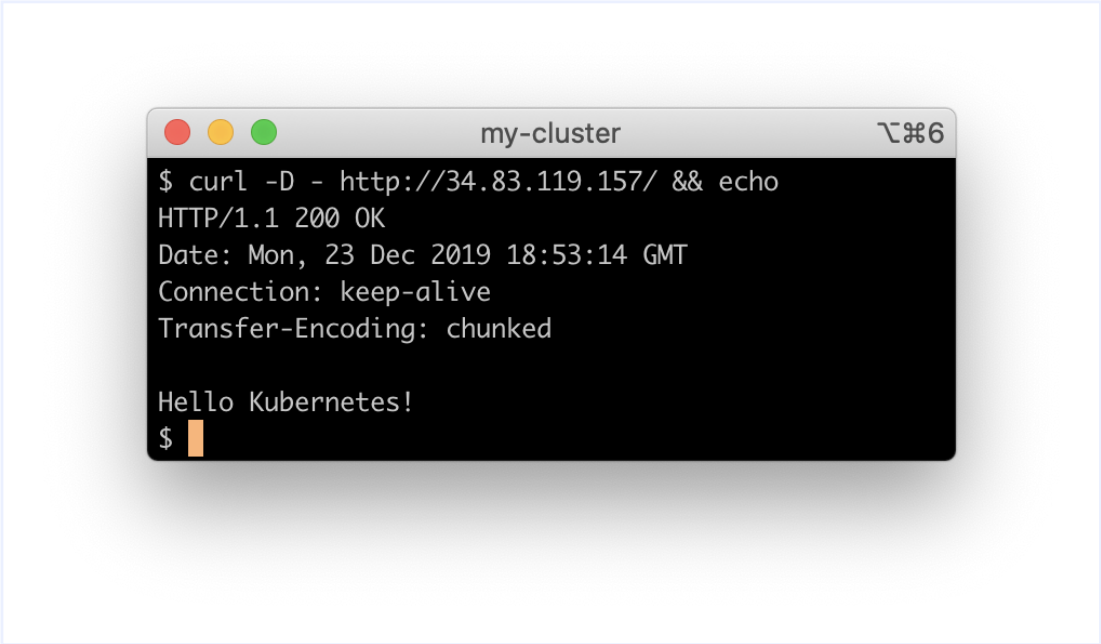
Note that, even though our cluster has two nodes, each with a `hello-world` pod, this routing method has no preference for routing to the pod on the node that receives the request from the cloud load balancer. If we change the `externalTrafficPolicy` in the service spec to `Local`, however, that would change. Not only would the request go to a pod on the node receiving the request, if one exists, but it also means a node without a service pod will refuse the connection. Therefore the `Local` policy generally needs to be used with Kubernetes daemon sets, which schedule a pod on each node in the cluster. While specifying local delivery would obviously reduce the mean network latency for requests, it can lead to uneven load across the service's pods.

The Pod Network

This post will not dive into details on pod networking, but in our GKE cluster, the pod network has its own CIDR block separate from the node's network. The Kubernetes network model ⧉ requires all pods in the cluster to be able to address each other directly, regardless of their host node. GKE clusters use the `kubenet` CNI, which creates network bridge interfaces to the pod network on each node, giving each node its own dedicated CIDR block of pod IP addresses to simplify allocation and routing. The Google Compute Engine (GCE) network can route this pod network traffic between VMs.

The Request

And this is how we get our `HTTP 200` response code.



Routing Variables

This post mentioned some of the ways different options offered by various Kubernetes platforms can change routing. Here is a non-comprehensive list:

(/)

Container Network Interface (CNI) plugins: Each cloud provider defaults to a CNI implementation compatible with their VM network model. This post used a GKE cluster with default settings as an example. Examples in Amazon EKS would have looked very different, because the AWS VPC CNI places pods directly on the nodes' VPC network.

Kubernetes Network Policy: One of the most popular CNI plugins implementing network policies, Calico, creates a virtual network interface on the nodes for each pod and uses Netfilter rules to enforce its firewall rules.

The `kube-proxy` IPVS routing mode moves the service routing and NATing out of the Netfilter rules, for the most part, although it does still make use of Netfilter.

External load balancers or other sources which can send traffic directly to the service's node port would match a different chain (`KUBE-NODEPORTS`) in iptables.

Kubernetes ingress controllers ⧉ may change edge service routing in a number of ways.

Service meshes like Istio may bypass `kube-proxy` and make direct connections for internal routing between service pods.

Securing Services

No universal method for adding firewall restrictions to cloud load balancers that are created by Kubernetes Service resources exists. Some cloud providers honor the `loadBalancerSourceRanges` ⧉ field in the Service spec, which allows you to provide a whitelist of IP CIDR blocks allowed to connect to the load balancer. If a cloud provider does not honor this field, it will be silently ignored, so take care to verify the network configuration of your external load balancers. For providers that do not support the `loadBalancerSourceRanges` field, you should assume your service endpoints on the load balancers will be open to the world unless you take action at the cloud provider level to lock down the load balancers and the cloud networks on which they run. The default firewall settings for cloud provider load balancer offerings vary wildly and depend on many factors. Some cloud providers may also support annotations to the `Service` object to configure load balancer security.

Note that we did not install the Calico CNI by enabling Kubernetes Network Policy support in our GKE cluster, because Calico creates a large number of additional iptables rules, adding extra steps when visually tracing virtual routing to a pod. However, we strongly recommend using a CNI that implements the `NetworkPolicy` API in production clusters and creating policies that restrict pod traffic (https://www.stackrox.com/post/2019/04/setting-up-kubernetes-network-policies-a-detailed-guide/).

Pods created with the `HostNetwork` attribute enabled will share the node's network space. While some valid use cases exist for doing so, generally most pods do not need to be on the host network, and particularly for pods running with root privileges, it could allow a compromised container to sniff network traffic. If you need to expose a container port on the node's network and using a

(/)

Kubernetes Service node port does not meet your needs, you have the option of specifying a Load Port for the container in the PodSpec.

Pods using the host network should not run with the `NET_ADMIN` capability, which would allow them to read and modify the node's firewall rules.

The Kubernetes network requires a large number of moving pieces. It is quite complicated, but having a basic understanding of what is taking place in your cluster will help you more effectively monitor, secure, and protect it.

Sources and Further Reading

https://kubernetes.io/docs/concepts/services-networking/service/ ⧉

https://kubernetes.io/docs/concepts/cluster-administration/networking/ ⧉

https://twitter.com/thockin/status/1191766983735296000 ⧉

https://kubernetes.io/blog/2018/07/09/ipvs-based-in-cluster-load-balancing-deep-dive/ ⧉

https://netfilter.org/documentation/HOWTO/NAT-HOWTO-6.html ⧉

Categories:

Kubernetes (/Categories/Kubernetes)

Tags:

Kubernetes Networking (/Tags/Kubernetes-Networking)

(/)

Configuration Management (/use-cases/configuration-management/)

Threat Detection (/use-cases/threat-detection/)

Incident Response (/use-cases/incident-response/)

Red Hat OpenShift (/solutions/red-hat-openshift-security/)

Pivotal Container Service (PKS) (/solutions/pks-security/)

Federal Agencies (/solutions/federal-agencies/)

Kubernetes Configuration Best Practices (https://www.stackrox.com/post/2019/12-kubernetes-configuration-best-practices/)

Intro to Istio Service Mesh (https://www.stackrox.com/post/2019/06/getting-started-with-istio-service-mesh-what-is-it-and-what-does-it-do/)

Istio Security Basics (https://www.stackrox.com/post/2019/08/istio-security-basics-running-microservices-on-zero-trust-networks/)

Kubernetes Adoption and Security Trends (https://www.stackrox.com/kubernetes-adoption-and-security-trends-and-market-share-for-containers/)

Admission Controllers (https://www.stackrox.com/post/2019/03/11-tips-to-operationalizing-kubernetes-admission-controllers-for-better-security/)

Docker Container Forensics (https://www.stackrox.com/post/2017/08/csi-container-edition-forensics-in-the-age-of-containers/)

Detecting Docker Exploits and Vulnerabilities (https://www.stackrox.com/post/2017/08/csi-container-edition-forensics-in-the-age-of-containers/)

Hardening Docker Containers and Images (https://www.stackrox.com/post/2017/08/hardening-docker-containers-and-hosts-against-vulnerabilities-a-security-toolkit/)