KubeCon CloudNativeCon Europe 2022

The Road to IPv6 Support in kOps

Ciprian Hacman, DevOps Engineer, polypoly

Ole Markus, Head Architect, Sportradar

John Gardiner Myers, Principal Engineer, Proofpoint

Justin Santa Barbara, Software Engineer, Google

Introduction

- kOps is a cloud-aware Kubernetes installer

- Provisions the underlying cloud infrastructure
- Installs all the core components of Kubernetes

- Since kOps owns the lifecycle of both the cloud resources and Kubernetes, it can handle all aspects of creating native IPv6 Kubernetes clusters

- Focus so far has been on the AWS and GCE cloud providers

Why IPv6

- Larger address space allows for a simple, flat network architecture
- Easier to avoid IP conflicts between VPCs, accounts, companies
- Native routing for Pods allows for direct targeting behind load balancers without going through "kube-proxy" or "nodeport"
- Simpler CNI architecture

Single Stack

- We decided early on we would only support single-stack addressing for pod networking and dual-stack at the edge
- Supporting dual-stack is quite complex and negates the benefits of IPv6
- Not many would be able to migrate their single-stack IPv4-clusters to dual-stack anyway

Unique Local Addresses

- We started out working on provisioning clusters with addresses from the Unique Local Address range
- This does not unlock the real benefits of IPv6, as NAT is still required, but this way we can assess the IPv6 readiness of the Kubernetes components
- Requires overlay networking and tunnels between nodes

Ready, but not quite

- IPv6 must be configured in a lot of places: control plane, CNI agents, CNI configuration
- Immediately hit a challenge with AWS cloud provider, which was not able to assign IPv6 IPs to Node objects
- No way to set priority on the Node addressing. Confuses pods that talk to nodes on NodePort
- Testing was possible now. Periodic tests were set up and most related bugs

Pivot to global addresses

- AWS released a number of IPv6 features last fall
- Replaced LUA support for global addresses.
- Required a few changes in our network provisioning:
  - NAT64/DNS64, route tables, launch templates
  - Disable masqing and overlays
- Assigning IPv6 prefix to instances during configuration.
- Assign node object's podCIDR

Pivot to global addresses


```
NAME                                                                IP
aws-cloud-controller-manager-g4698                                  172.20.63.64
cert-manager-5d46c5dc5b-7dpkk                                       2a05:d014:cd:4a01:1517::1b3f
cert-manager-cainjector-f47bf9bb5-dz448                             2a05:d014:cd:4a01:1517::6815
cert-manager-webhook-6dd6f5dcb8-5pxqt                               2a05:d014:cd:4a01:1517::ad83
cilium-64glr                                                        172.20.49.163
cilium-c7gxn                                                        172.20.87.60
cilium-ltqdt                                                        172.20.63.64
cilium-operator-74bbb8fc4b-vd2b8                                    172.20.63.64
cilium-v6tzg                                                        172.20.101.230
coredns-5dc785954d-d6cjx                                            2a05:d014:cd:4a01:7325::5cef
coredns-5dc785954d-hhwhd                                            2a05:d014:cd:4a01:7325::22b0
coredns-autoscaler-84d4cfd89c-nnlk8                                 2a05:d014:cd:4a01:7325::a317
dns-controller-57d57cdfbb-wzxh9                                     172.20.63.64
ebs-csi-controller-66fb77f96d-cswgw                                 2a05:d014:cd:4a01:1517::9873
ebs-csi-node-2qllj                                                  172.20.49.163
ebs-csi-node-fxdl4                                                  172.20.87.60
ebs-csi-node-sttfx                                                  172.20.101.230
ebs-csi-node-wswrw                                                  172.20.63.64
etcd-manager-events-ip-172-20-63-64.eu-central-1.compute.internal   172.20.63.64
etcd-manager-main-ip-172-20-63-64.eu-central-1.compute.internal     172.20.63.64
kops-controller-2fxkq                                               172.20.63.64
```

Pivot to global addresses

- We had a working cluster for about 5 minutes!
- On Ubuntu, a bug in systemd-networkd removed the instance primary IP.
- Other distros had various other issues with IPv6 prefix support as well, such as accepting router announcements.
- Eventually fixed the issues and now supports Debian 10 and Ubuntu 22.04

Custom node IPAM controller

- Custom node IPAM controller that is aware of the IPv6 prefix AWS assigns the node.
- Adds the assigned prefix to Node object's podCIDR
- Allows CNIs to be agnostic of the cloud specifics.
- Controller currently runs as part of kops-controller, but could theoretically be stand-alone or run as part of CCM

Public Topology

- Nodes are dual-stack
- Enables DNS64 in CoreDNS/NodeLocal DNS
  - Added the DNS64 plugin to kubernetes/dns
- Creates a NAT Gateway in each zone for NAT64

Private Topology

- Utility Subnets are dual-stack
  - Contains NAT Gateways and public Load Balancers
- Private Subnets are IPv6-only
  - Contains worker nodes
  - Enable DNS64 in the AWS DNS server for Subnet
  - Route NAT64 to NAT Gateway in zone's utility subnet
  - Route rest of IPv6 to Egress-only Internet Gateway
- "DualStack" Subnets
  - Contains control plane nodes
  - Because kOps uses instance targets for the API loadbalancer and NLBs only support IPv4 for instance targets
  - Like Public subnets, but no public IPv4s and use Egress-only Internet Gateway

Subnet assignments

- The Kubernetes Service network is fd00:5e4f:ce::/108
- NodeLocal DNS is fd00:90de:d95::1

Remaining work

- AWS Load Balancer Controller
  - Probably due to insufficient IAM permissions
  - Need to disable Cloud Controller Manager's Service controller
  - Need better defaults for IPv6 clusters
- External-dns
  - Support for AAAA records is work in progress

Status quo

- Cluster is working.
- Makes heavy use of NAT64/DNS64 to interact with popular registries, cloud APIs etc.
- Regularly runs the Kubernetes e2e tests on various IPv6 configurations.
- Many of the kOps addons don’t work. External-dns still working on AAAA records support.
- Services of type LoadBalancer do not support IPv6 in the AWS cloud controller

Status of GCE support

- Straightforward to add GCE equivalents to AWS work
- Some minor differences
  - e.g. IPv6 ranges auto-assigned on GCE

- Sequence:
  - VMs get a /96 at boot
  - IPAM controller writes Node's PodCIDR
  - CNI assigns pods IPs from Node's PodCIDR
  - IPv6 doesn't require cloud-specific routing

- *Almost* works!

```
NAME                READY   STATUS    IP                          NODE
guestbook-2xkdd     1/1     Running   2600:1900:4120:6e77::b      nodes-us-west2-a-dr7m
guestbook-kkfms     1/1     Running   2600:1900:4120:6e77:0:2:0:3 nodes-us-west2-a-0q31
guestbook-kmtj9     1/1     Running   2600:1900:4120:6e77:0:2:0:2 nodes-us-west2-a-0q31
redis-master-8mp55  1/1     Running   2600:1900:4120:6e77:0:2:0:4 nodes-us-west2-a-0q31
redis-slave-bm477   1/1     Running   2600:1900:4120:6e77::c      nodes-us-west2-a-dr7m
redis-slave-vnrs2   1/1     Running   2600:1900:4120:6e77:0:2:0:5 nodes-us-west2-a-0q31
```

CNI Evolution for IPv6?

- CNI requirements for IPv6 are much simpler
  - No overlay network
  - No routing logic
  - Plenty of IPs for IPAM - no IP reuse problems

- Does require *new* configuration
  - LoadBalancer services cannot do IPv4 -> IPv6 NAT
  - New roles e.g. who removes network-unavailable taint

- All surmountable, but *new* problems