Kubernetes Networking on AWS

Mike Stefaniak – Sr. Product Manager, Amazon EKS
Agenda

• Brief Kubernetes networking overview
• Amazon VPC CNI plugin architecture/features
• Demo (Security groups for pods)
• Roadmap
Networking in Kubernetes
Kubernetes Networking Model

1. All pods can communicate with all other Pods without using network address translation (NAT).
2. All nodes can communicate with all pods without NAT.
3. The IP that a pod sees itself as is the same IP that others see it as.

Kubernetes uses CNI (Container Networking Interface) as an interface between network providers and Kubernetes pod networking.

https://sookocheff.com/post/kubernetes/understanding-kubernetes-networking-model/
# Overlay networks

## Pros

- Overlay based CNI plugins can run in cloud or on-premises
- Helps with IPv4 scarcity/fragmentation
- Pod density not tied to instance type

## Cons

- Hard to debug
- No direct to endpoint communications
- Unable to rely on VPC firewalls
- Scaling challenges in large clusters
- Packet encapsulation requires node resources
- Increased network performance overhead
Native VPC networking performance

Pods have the same VPC address inside the pod as on the VPC

Simple, secure, scalable networking

Open source and on GitHub
Amazon VPC CNI Plugin

Secondary IPs:
10.0.0.1
10.0.0.2

Secondary IPs:
10.0.0.20
10.0.0.22

Instance 1

Instance 2

VPC Subnet – 10.0.0.0/24

VPC

ec2.associateaddress()
Support for advanced networking architectures

VPC - Multiple IP ranges

Subnet 1 – 10.0.0.0/24
Primary ENI
Pod
Outbound Traffic SNAT

Subnet 2 – 100.64.0.0/16
Pod – 100.64.0.200
Pod secondary ENI

On-premises 10.1.0.0/16
Corporate data center
Customer gateway
VPN or DX

Pod
worker node

https://docs.aws.amazon.com/eks/latest/userguide/cni-custom-network.html
Coming soon: Security groups for pods

Use Cases

• Maintain security in multi-tenant clusters by running applications with different network security requirements on shared compute resources.
• Control network access from pods to AWS services outside your cluster.
• Keep existing security group rules and compliance programs when migrating applications from EC2 instances to Amazon EKS, without needing to re-implement rules as Kubernetes network policies.
Simplified architecture

Without pod security groups

Worker Node VPC

Amazon RDS

Amazon ElastiCache

Availability Zone

Node group A

Node group B

kubectl taint nodes nodename
dedicated=groupA:NoSchedule

kubectl taint nodes nodename
dedicated=groupB:NoSchedule

kubectl label nodes nodename
dedicated=groupA

kubectl label nodes nodename
dedicated=groupB

application A manifest
containers:
- name: applicationA
  image: applicationA
tolerations:
  - key: "dedicated"
    operator: "Equal"
    value: "groupA"
    effect: "NoSchedule"
  nodeSelector:
    dedicated: groupA

application B manifest
containers:
- name: applicationB
  image: applicationB
tolerations:
  - key: "dedicated"
    operator: "Equal"
    value: "groupB"
    effect: "NoSchedule"
  nodeSelector:
    dedicated: groupB

With pod security groups

Worker Node VPC

Amazon RDS

Amazon ElastiCache

Availability Zone

Node group A

application A manifest
containers:
- name: applicationA
  image: applicationA

application B manifest
containers:
- name: applicationB
  image: applicationB

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
Show me how it works!
Networking Roadmap

- Simplified CNI custom networking #867
- NLB IP targeting mode (Fargate) #981
- ALB ingress grouping #847
- Increased pod density #138
- Migrate L-IPAM daemon to VPC Resource Controller #866
- IPv6 support #835

https://github.com/aws/containers-roadmap/projects/1
Where to learn more

Amazon EKS Documentation: Networking

Amazon VPC CNI Plugin

VPC CNI Plugin Proposal

Blog: De-mystifying cluster networking for Amazon EKS

Blog: Routable VPC IPv4 address conservation
Thank you!

Leave any questions/suggestions in the chat or visit our virtual booth during KubeCon EU 2020!

Twitter/GitHub: @mikestef9