Pod IP Exhaustion GKE and EKS

Olga Mirensky

Platform Engineer - ANZ Plus

Agenda

Explore k8s IPAM (IP Address Management) by understanding Pod IP exhaustion mitigation strategies.

GKE vs EKS: highly adopted managed k8s that implement networking model in very different ways.

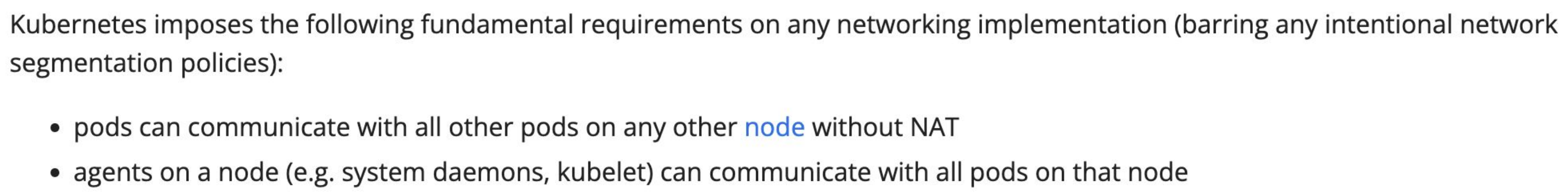
Out of Scope

Node IPs, Services IPs, IPv6, Cluster rebuild

Kubernetes Network Model and IPAM

The Kubernetes Network Model

Every Pod in a cluster gets its own unique cluster-wide IP address



Kubernetes imposes the following fundamental requirements on any networking implementation (barring any intentional network segmentation policies):

- pods can communicate with all other pods on any other node without NAT
- agents on a node (e.g. system daemons, kubelet) can communicate with all pods on that node

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

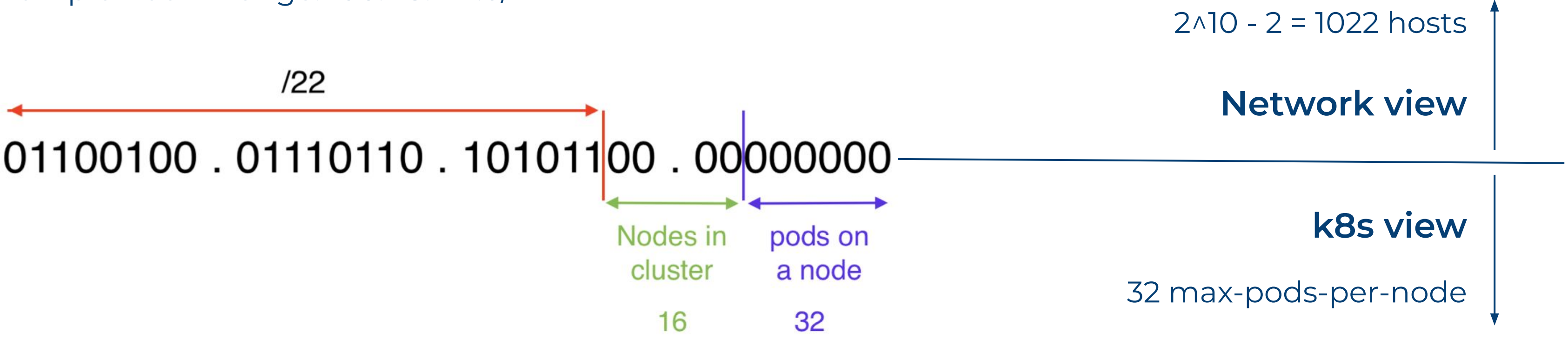
The implementation of this model is left out to CNIs - Container Network

Interface, e.g. Cilium, Calico, AWS VPC CNI and more.

Anatomy of Pod IPs

- Applicable when `kube-controller-manager` is configured with `--allocate-node-cidrs`
- Pod range CIDR is allocated when node is created
- At least x2 as many IPs reserved as max number of pods per node (6 bits for 32 pods)
- Wasted Pod IPs for hostNetwork pods (ouch, all these daemonsets!)

Example Pod IP range: 100.118.172.0/22



2^10 - 2 = 1022 hosts

Network view

/22

01100100 . 01110110 . 10101100 . 00000000

Nodes in cluster

16

pods on a node

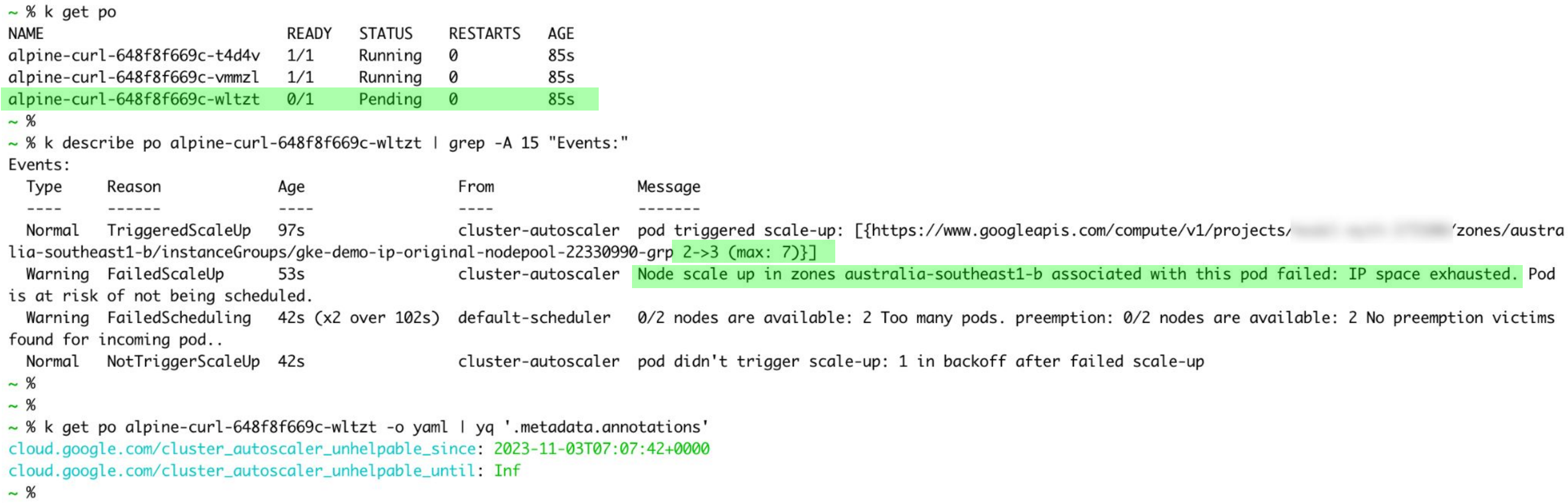
32

k8s view

32 max-pods-per-node

Pod IP Exhaustion (k8s IPAM)

When a new pod triggers a scale-up and there is no more free blocks left in Pod IP space, the result is a failure to provision a node



```
~ % k get po
NAME                            READY   STATUS    RESTARTS   AGE
alpine-curl-648f8f669c-t4d4v    1/1     Running   0          85s
alpine-curl-648f8f669c-vmmzl    1/1     Running   0          85s
alpine-curl-648f8f669c-wltzt    0/1     Pending   0          85s
~ %
~ % k describe po alpine-curl-648f8f669c-wltzt | grep -A 15 "Events:"
Events:
  Type     Reason             Age                 From                Message
  ----     ------             ----                ----                -------
  Normal   TriggeredScaleUp   97s                 cluster-autoscaler  pod triggered scale-up: [{https://www.googleapis.com/compute/v1/projects/[redacted]/zones/austra
lia-southeast1-b/instanceGroups/gke-demo-ip-original-nodepool-22330990-grp 2->3 (max: 7)}]
  Warning  FailedScaleUp      53s                 cluster-autoscaler  Node scale up in zones australia-southeast1-b associated with this pod failed: IP space exhausted. Pod
is at risk of not being scheduled.
  Warning  FailedScheduling   42s (x2 over 102s)  default-scheduler   0/2 nodes are available: 2 Too many pods. preemption: 0/2 nodes are available: 2 No preemption victims
found for incoming pod..
  Normal   NotTriggerScaleUp  42s                 cluster-autoscaler  pod didn't trigger scale-up: 1 in backoff after failed scale-up
~ %
~ %
~ % k get po alpine-curl-648f8f669c-wltzt -o yaml | yq '.metadata.annotations'
cloud.google.com/cluster_autoscaler_unhelpable_since: 2023-11-03T07:07:42+0000
cloud.google.com/cluster_autoscaler_unhelpable_until: Inf
~ %
```

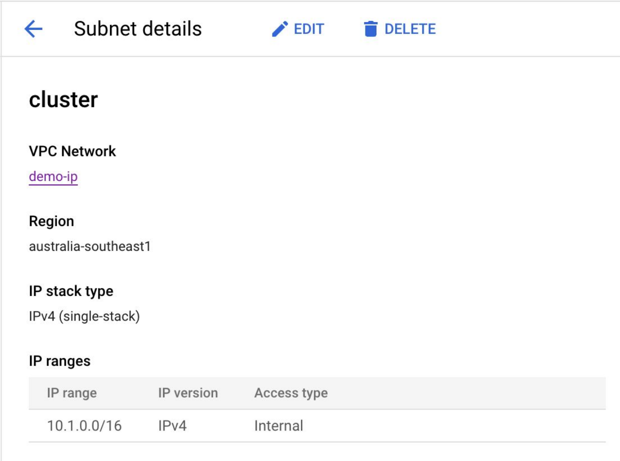
Mitigation

Review current state

- Max-pods-per-node
- Machine type
- Daemonset overhead
- Mix of machine types or nodepools with different max-per-node

Vendor Specific Mitigations - GKE

GKE Setup



cluster

VPC Network
demo-ip

Region
australia-southeast1

IP stack type
IPv4 (single-stack)

IP ranges

IP range	IP version	Access type
10.1.0.0/16	IPv4	Internal

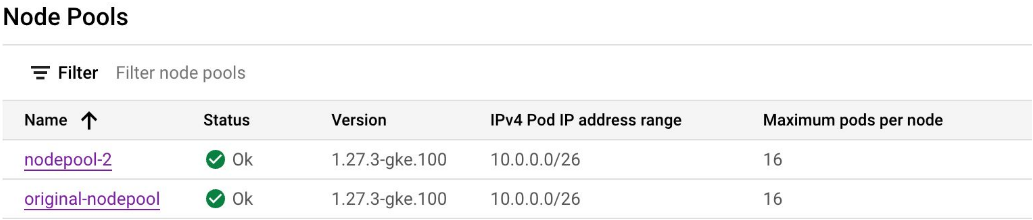
IP Range

- IPs allocated to VMs (nodes)

Secondary IPv4 Ranges

- Alias IPs - can be used for services running on a VM
- 0 - 30 ranges per subnet
- In k8s - Pod IPs and ClusterIP services.

All ranges, both primary and secondary, must be unique across all subnets in the VPC network and in any attached networks



Node Pools

Name ↑	Status	Version	IPv4 Pod IP address range	Maximum pods per node
nodepool-2	✓ Ok	1.27.3-gke.100	10.0.0.0/26	16
original-nodepool	✓ Ok	1.27.3-gke.100	10.0.0.0/26	16

GKE: Discontiguous Multi-Pod CIDR

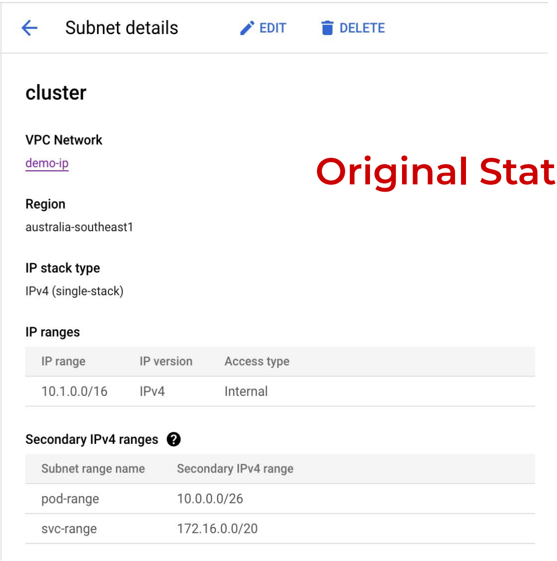
- Create and assign additional secondary ranges to the **cluster** (new in v1.26)
- Create a **node pool** with a new secondary Pod IP address range (GKE manages subnet)

→ Create a **node pool** using an existing secondary Pod IP address (you manage subnet)

Each nodepool always has one and only one subnetwork secondary range associated with it.

Cluster option is similar to nodepool except that secondary range is assigned by GKE.

Step 1 - Add new Secondary Range to the subnet



Original State

cluster

VPC Network
demo-ip

Region
australia-southeast1

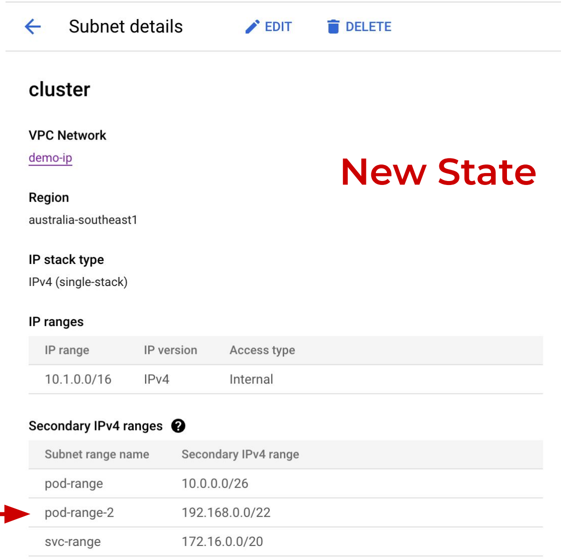
IP stack type
IPv4 (single-stack)

IP ranges

IP range	IP version	Access type
10.1.0.0/16	IPv4	Internal

Secondary IPv4 ranges

Subnet range name	Secondary IPv4 range
pod-range	10.0.0.0/26
svc-range	172.16.0.0/20



New State

cluster

VPC Network
demo-ip

Region
australia-southeast1

IP stack type
IPv4 (single-stack)

IP ranges

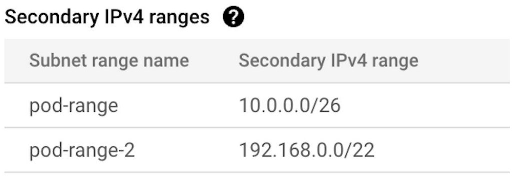
IP range	IP version	Access type
10.1.0.0/16	IPv4	Internal

Secondary IPv4 ranges

Subnet range name	Secondary IPv4 range
pod-range	10.0.0.0/26
pod-range-2	192.168.0.0/22
svc-range	172.16.0.0/20

Step 2 - Add new nodepool

```
$ gcloud container node-pools create new-nodepool \
    --cluster=demo-ip                             \
    --node-locations=$az                          \
    --location-policy=BALANCED                    \
    --enable-autoscaling                          \
    --total-max-nodes=10                          \
    --max-pods-per-node=32                        \
    --pod-ipv4-range=pod-range-2
```

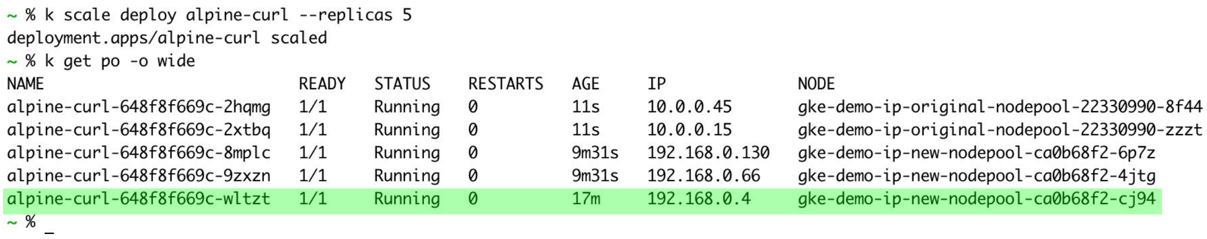


Secondary IPv4 ranges ?

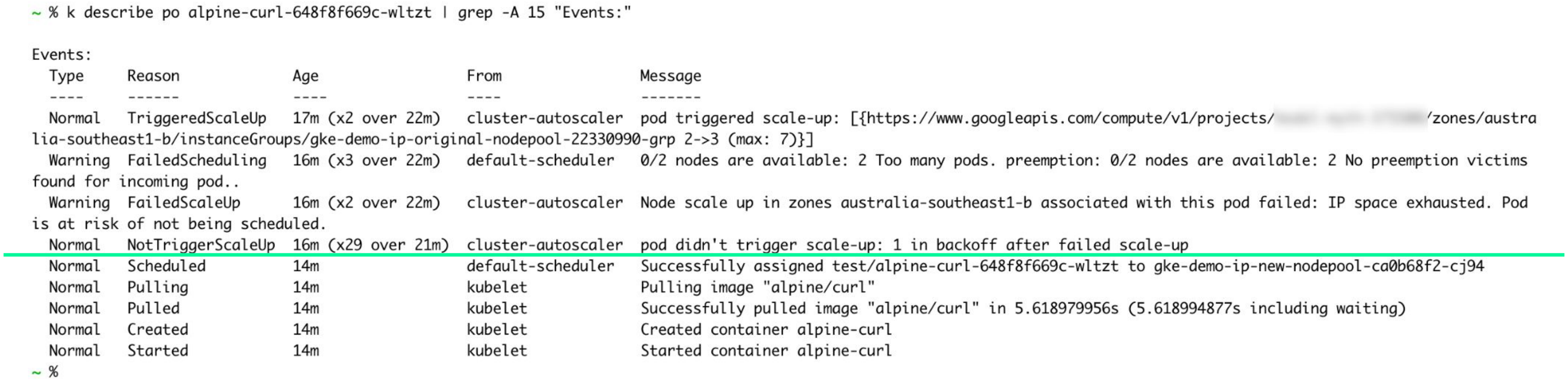
Subnet range name	Secondary IPv4 range
pod-range	10.0.0.0/26
pod-range-2	192.168.0.0/22

```
$ k get node -o yaml | yq '.items[]|{"name":
.metadata.name, "podCIDRs": .spec.podCIDRs}'

name: gke-demo-ip-new-nodepool-ca0b68f2-4jtg
podCIDRs:
  - 192.168.0.64/26
name: gke-demo-ip-new-nodepool-ca0b68f2-6p7z
podCIDRs:
  - 192.168.0.128/26
name: gke-demo-ip-new-nodepool-ca0b68f2-cj94
podCIDRs:
  - 192.168.0.0/26
name: gke-demo-ip-original-nodepool-22330990-8f44
podCIDRs:
  - 10.0.0.32/27
name: gke-demo-ip-original-nodepool-22330990-zzzt
podCIDRs:
  - 10.0.0.0/27
```

Happy End



```
~ % k scale deploy alpine-curl --replicas 5
deployment.apps/alpine-curl scaled
~ % k get po -o wide
NAME                           READY   STATUS    RESTARTS   AGE     IP              NODE
alpine-curl-648f8f669c-2hqmg   1/1     Running   0          11s     10.0.0.45       gke-demo-ip-original-nodepool-22330990-8f44
alpine-curl-648f8f669c-2xtbq   1/1     Running   0          11s     10.0.0.15       gke-demo-ip-original-nodepool-22330990-zzzt
alpine-curl-648f8f669c-8mplc   1/1     Running   0          9m31s   192.168.0.130   gke-demo-ip-new-nodepool-ca0b68f2-6p7z
alpine-curl-648f8f669c-9zxzn   1/1     Running   0          9m31s   192.168.0.66    gke-demo-ip-new-nodepool-ca0b68f2-4jtg
alpine-curl-648f8f669c-wltzt   1/1     Running   0          17m     192.168.0.4     gke-demo-ip-new-nodepool-ca0b68f2-cj94
~ % _
```



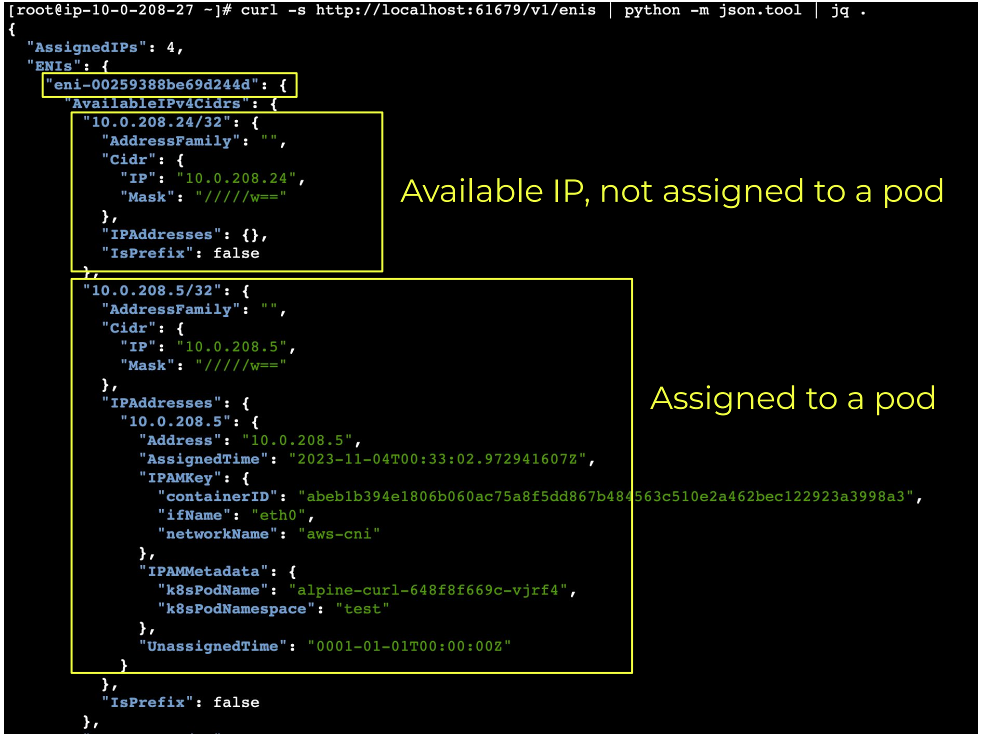
```
~ % k describe po alpine-curl-648f8f669c-wltzt | grep -A 15 "Events:"

Events:
  Type     Reason              Age                 From                Message
  ----     ------              ----                ----                -------
  Normal   TriggeredScaleUp    17m (x2 over 22m)   cluster-autoscaler  pod triggered scale-up: [{https://www.googleapis.com/compute/v1/projects/[redacted]/zones/austra
lia-southeast1-b/instanceGroups/gke-demo-ip-original-nodepool-22330990-grp 2->3 (max: 7)}]
  Warning  FailedScheduling    16m (x3 over 22m)   default-scheduler   0/2 nodes are available: 2 Too many pods. preemption: 0/2 nodes are available: 2 No preemption victims
found for incoming pod..
  Warning  FailedScaleUp       16m (x2 over 22m)   cluster-autoscaler  Node scale up in zones australia-southeast1-b associated with this pod failed: IP space exhausted. Pod
is at risk of not being scheduled.
  Normal   NotTriggerScaleUp   16m (x29 over 21m)  cluster-autoscaler  pod didn't trigger scale-up: 1 in backoff after failed scale-up
  Normal   Scheduled           14m                 default-scheduler   Successfully assigned test/alpine-curl-648f8f669c-wltzt to gke-demo-ip-new-nodepool-ca0b68f2-cj94
  Normal   Pulling             14m                 kubelet             Pulling image "alpine/curl"
  Normal   Pulled              14m                 kubelet             Successfully pulled image "alpine/curl" in 5.618979956s (5.618994877s including waiting)
  Normal   Created             14m                 kubelet             Created container alpine-curl
  Normal   Started             14m                 kubelet             Started container alpine-curl
~ %
```

Vendor Specific Mitigations - EKS

EKS IPAM

- AWS VPC CNI consists of two components:
    - CNI Daemonset `aws-node`
    - `ipamd` daemon running on the host
- No Pods CIDR allocation at node creation
- ENIs are attached to the node as needed
- Pod IPs are allocated to each ENI as needed
- `WARM_ENI_TARGET`, `WARM_IP_TARGET` - ENI/IP allocation headroom

Secondary IP



```
[root@ip-10-0-208-27 ~]# curl -s http://localhost:61679/v1/enis | python -m json.tool | jq .
{
  "AssignedIPs": 4,
  "ENIs": {
    "eni-00259388be69d244d": {
      "AvailableIPv4Cidrs": {
        "10.0.208.24/32": {
          "AddressFamily": "",
          "Cidr": {
            "IP": "10.0.208.24",
            "Mask": "/////w=="
          },
          "IPAddresses": {},
          "IsPrefix": false
        },
        "10.0.208.5/32": {
          "AddressFamily": "",
          "Cidr": {
            "IP": "10.0.208.5",
            "Mask": "/////w=="
          },
          "IPAddresses": {
            "10.0.208.5": {
              "Address": "10.0.208.5",
              "AssignedTime": "2023-11-04T00:33:02.972941607Z",
              "IPAMKey": {
                "containerID": "abeb1b394e1806b060ac75a8f5dd867b484563c510e2a462bec122923a3998a3",
                "ifName": "eth0",
                "networkName": "aws-cni"
              },
              "IPAMMetadata": {
                "k8sPodName": "alpine-curl-648f8f669c-vjrf4",
                "k8sPodNamespace": "test"
              },
              "UnassignedTime": "0001-01-01T00:00:00Z"
            }
          },
          "IsPrefix": false
        },
```

Available IP, not assigned to a pod

Assigned to a pod

ipamd provided debug tools on the host VM (amazon-vpc-cni-k8s repo)

docs/troubleshooting.md

EKS: Pod IP exhaustion

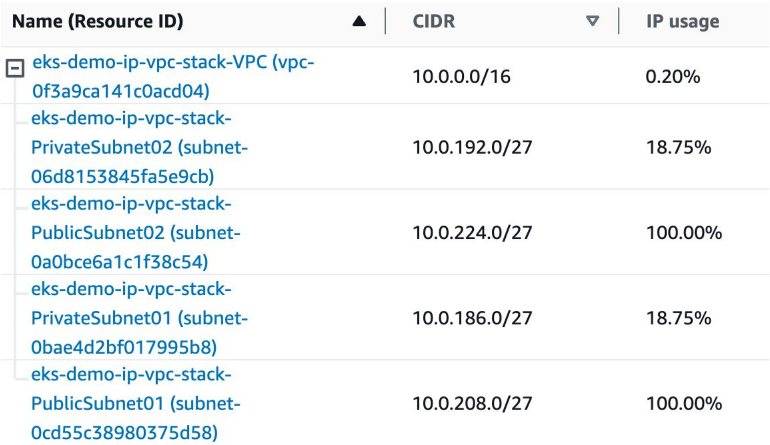
Manifests at pod startup time:

```
NAME                                 READY    STATUS               RESTARTS    AGE
mydeploy-6bdbfc484d-2frv4            0/1      ContainerCreating    0           9m52s

-
Warning  FailedCreatePodSandBox  7m26s       kubelet    Failed to create pod sandbox: rpc error: code =
Unknown desc = failed to setup network for sandbox <id>: plugin type="aws-cni" name="aws-cni" failed
(add): add cmd: failed to assign an IP address to container
```

Demo EKS Network Setup

- EKS requires at least 2 AZs
- The demo cluster uses 2 public and 2 private subnets in a VPC with `10.0.0.0/16` CIDR.

```
$ aws ec2 describe-subnets
--filters Name=vpc-id,Values=$vpc_id
--query 'Subnets[*].{SubnetId: SubnetId,AvailabilityZone:
AvailabilityZone,CidrBlock: CidrBlock}'
--output table
-------------------------------------------------------------------
|                         DescribeSubnets                         |
+------------------+-----------------+----------------------------+
| AvailabilityZone |    CidrBlock    |          SubnetId          |
+------------------+-----------------+----------------------------+
|  ap-southeast-2a |  10.0.208.0/27  |  subnet-0cd55c38980375d58  |
|  ap-southeast-2a |  10.0.186.0/27  |  subnet-0bae4d2bf017995b8  |
|  ap-southeast-2b |  10.0.192.0/27  |  subnet-06d8153845fa5e9cb  |
|  ap-southeast-2b |  10.0.224.0/27  |  subnet-0a0bce6a1c1f38c54  |
+------------------+-----------------+----------------------------+
```



Name (Resource ID) ▲	CIDR ▽	IP usage
⊟ eks-demo-ip-vpc-stack-VPC (vpc-0f3a9ca141c0acd04)	10.0.0.0/16	0.20%
eks-demo-ip-vpc-stack-PrivateSubnet02 (subnet-06d8153845fa5e9cb)	10.0.192.0/27	18.75%
eks-demo-ip-vpc-stack-PublicSubnet02 (subnet-0a0bce6a1c1f38c54)	10.0.224.0/27	100.00%
eks-demo-ip-vpc-stack-PrivateSubnet01 (subnet-0bae4d2bf017995b8)	10.0.186.0/27	18.75%
eks-demo-ip-vpc-stack-PublicSubnet01 (subnet-0cd55c38980375d58)	10.0.208.0/27	100.00%

EKS Custom Networking for Pods

1. Associate new CIDR range to the VPC

```
$ aws ec2 associate-vpc-cidr-block --vpc-id $vpc_id --cidr-block 100.64.0.0/20
```

2. Create subnets in each AZ with CIDRs from the new range

```
$ aws ec2 create-subnet --vpc-id $vpc_id --availability-zone $az_1 --cidr-block 100.64.1.0/24
$ aws ec2 create-subnet --vpc-id $vpc_id --availability-zone $az_2 --cidr-block 100.64.2.0/24
```

3. Configure AWS VPC CNI to Custom Networking

```
$ kubectl set env daemonset aws-node AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG=true
```

4. Provide subnets data via ENIConfig CRD
5. Create Node Group

EKS Custom Networking for Pods (cont)

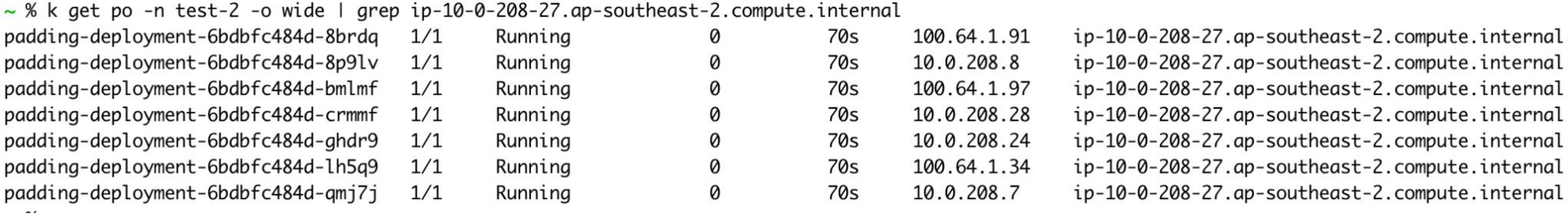
ENI Config - 1 for each AZ

```
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_1
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_1
```

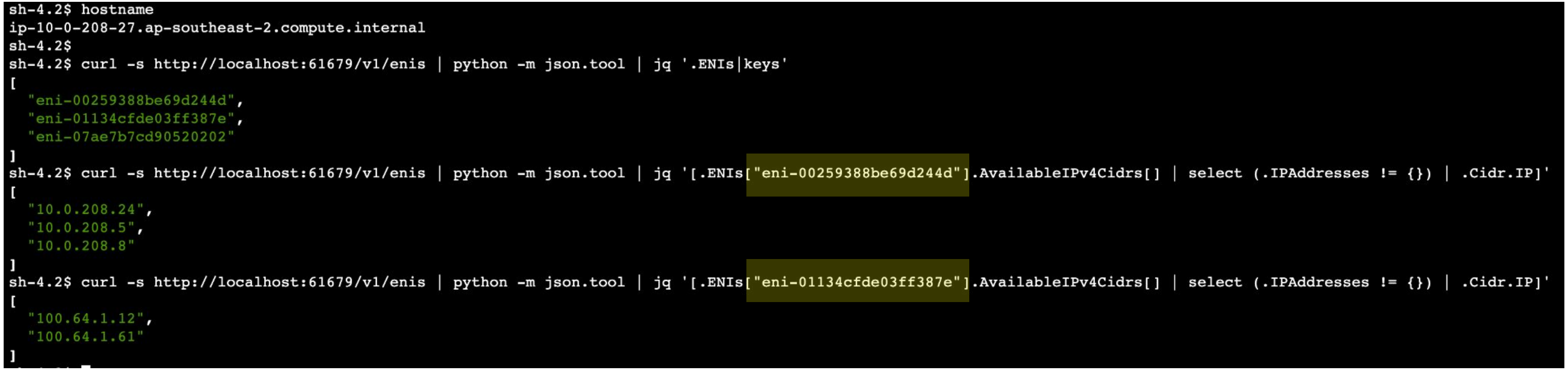
New NodeGroup (eksctl config)

```
managedNodeGroups:
  - name: managed-ng-new
    availabilityZones: [$az_1, $az_2]
    …
```

Any Pod IP on Any Node



```
~ % k get po -n test-2 -o wide | grep ip-10-0-208-27.ap-southeast-2.compute.internal
padding-deployment-6bdbfc484d-8brdq   1/1     Running            0          70s     100.64.1.91    ip-10-0-208-27.ap-southeast-2.compute.internal
padding-deployment-6bdbfc484d-8p9lv   1/1     Running            0          70s     10.0.208.8     ip-10-0-208-27.ap-southeast-2.compute.internal
padding-deployment-6bdbfc484d-bmlmf   1/1     Running            0          70s     100.64.1.97    ip-10-0-208-27.ap-southeast-2.compute.internal
padding-deployment-6bdbfc484d-crmmf   1/1     Running            0          70s     10.0.208.28    ip-10-0-208-27.ap-southeast-2.compute.internal
padding-deployment-6bdbfc484d-ghdr9   1/1     Running            0          70s     10.0.208.24    ip-10-0-208-27.ap-southeast-2.compute.internal
padding-deployment-6bdbfc484d-lh5q9   1/1     Running            0          70s     100.64.1.34    ip-10-0-208-27.ap-southeast-2.compute.internal
padding-deployment-6bdbfc484d-qmj7j   1/1     Running            0          70s     10.0.208.7     ip-10-0-208-27.ap-southeast-2.compute.internal
```



```
sh-4.2$ hostname
ip-10-0-208-27.ap-southeast-2.compute.internal
sh-4.2$
sh-4.2$ curl -s http://localhost:61679/v1/enis | python -m json.tool | jq '.ENIs|keys'
[
  "eni-00259388be69d244d",
  "eni-01134cfde03ff387e",
  "eni-07ae7b7cd90520202"
]
sh-4.2$ curl -s http://localhost:61679/v1/enis | python -m json.tool | jq '[.ENIs["eni-00259388be69d244d"].AvailableIPv4Cidrs[] | select (.IPAddresses != {}) | .Cidr.IP]'
[
  "10.0.208.24",
  "10.0.208.5",
  "10.0.208.8"
]
sh-4.2$ curl -s http://localhost:61679/v1/enis | python -m json.tool | jq '[.ENIs["eni-01134cfde03ff387e"].AvailableIPv4Cidrs[] | select (.IPAddresses != {}) | .Cidr.IP]'
[
  "100.64.1.12",
  "100.64.1.61"
]
```

Questions ?

Demo source code and references

https://github.com/olga-mir/k8s/tree/main/demo/2023-gke-eks-pod-ip-exhaustion