Scaling to Large Clusters in EKS

Operational experience and challenges

Agenda

- Introduction
- Scaling Challenges
  1. Autoscaling
  2. Custom Network Interface (CNI)
  3. DNS
  4. API server
  5. Cost Optimization
- Vision
- Conclusion

Introduction

About Us

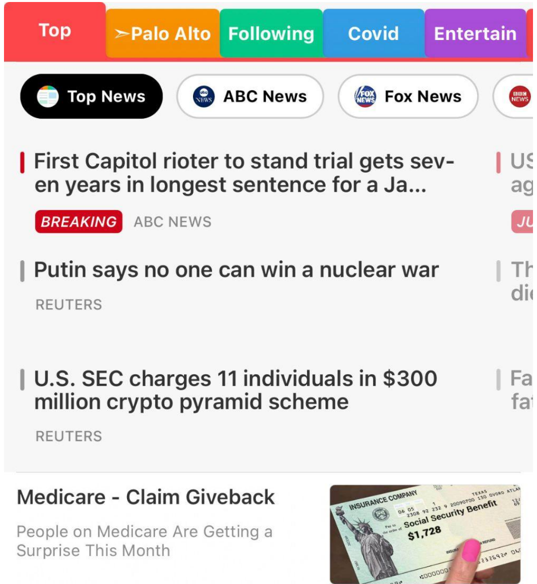
Marc Brandenburg
Group Head
Platform Backend

Luke (Feixiang Li)
Senior Software Engineer
Platform Infrastructure

About SmartNews

Delivering the world's quality information to the people who need it.

Global user base with focus on Japan and the United States



First Capitol rioter to stand trial gets seven years in longest sentence for a Ja...

BREAKING ABC NEWS

Putin says no one can win a nuclear war

REUTERS

U.S. SEC charges 11 individuals in $300 million crypto pyramid scheme

REUTERS

US ag

JU

Th di

Fa fa

Medicare - Claim Giveback

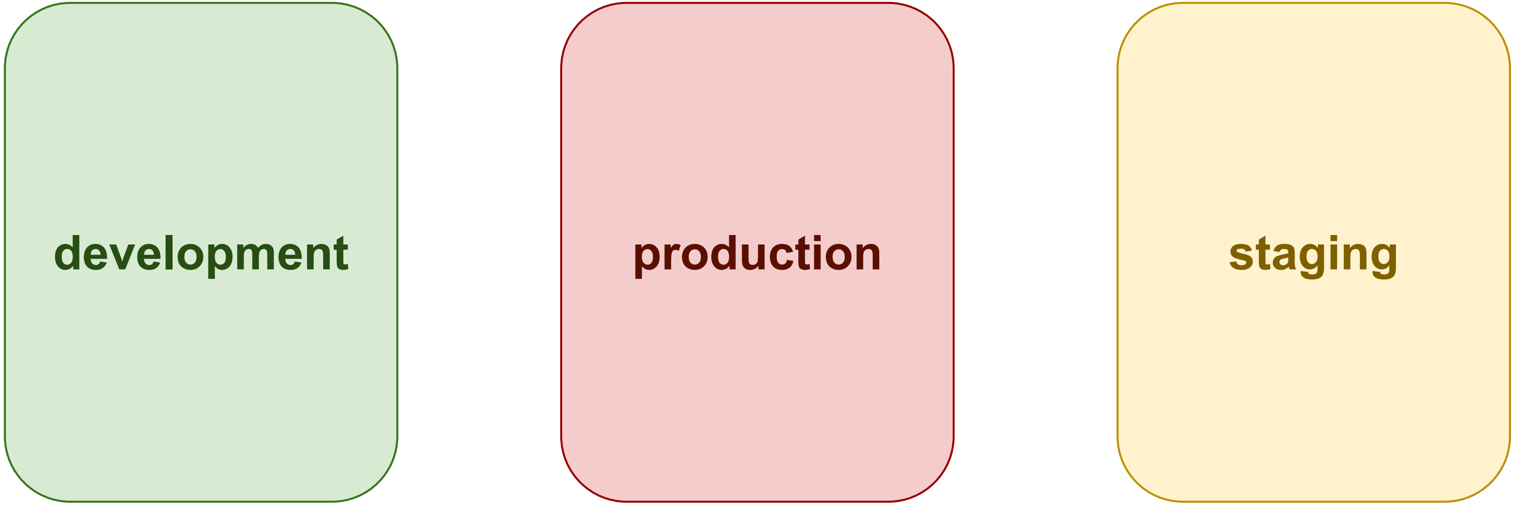
People on Medicare Are Getting a Surprise This Month

EKS Usage

- Started using EKS for smaller services from **July 2019**
- Most new services after around **Mid 2020** run on EKS
- From **February 2021** started to deploy stateful services
- Scaled up processing workloads around **End of 2021**

Cluster Topology

Environment



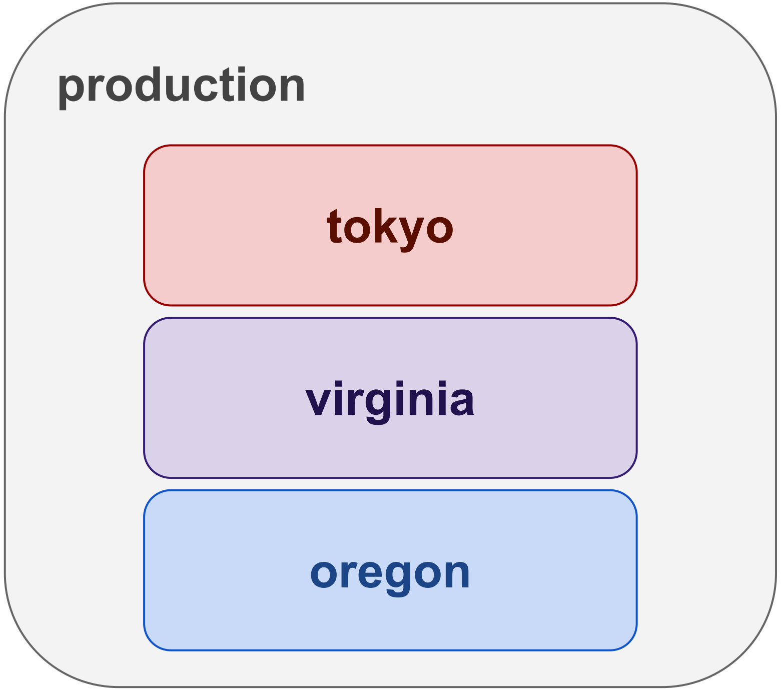
development

production

staging

Cluster Topology

Region



production

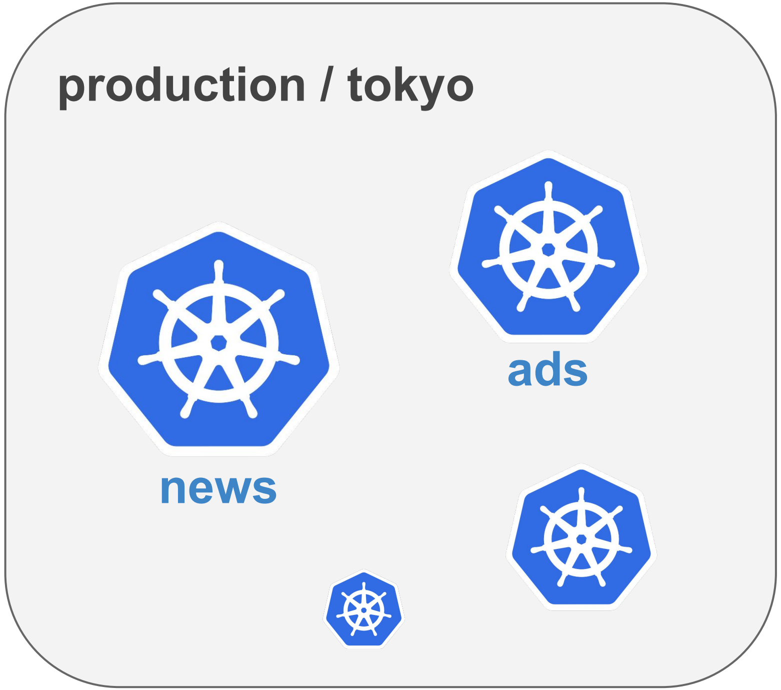
tokyo

virginia

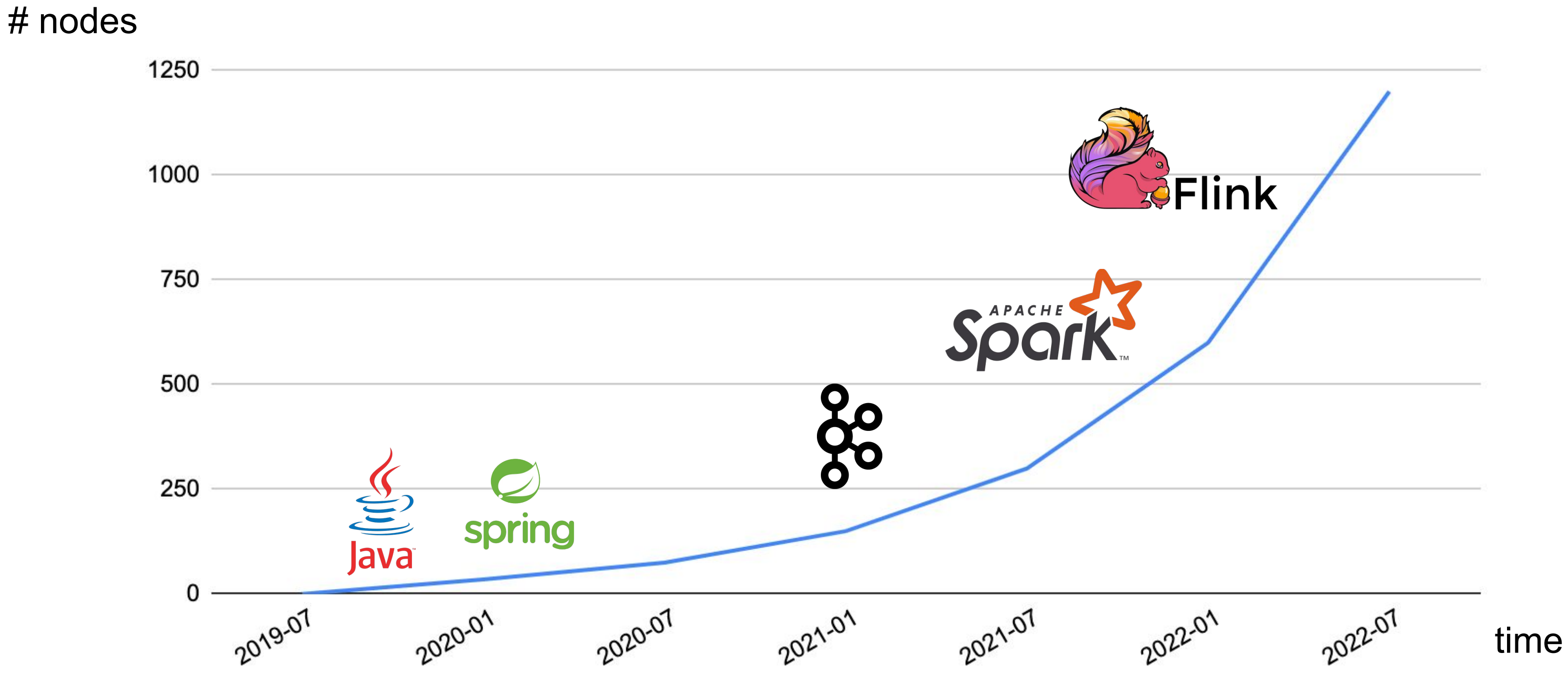
oregon

Cluster Topology

Cluster



Cluster Scale



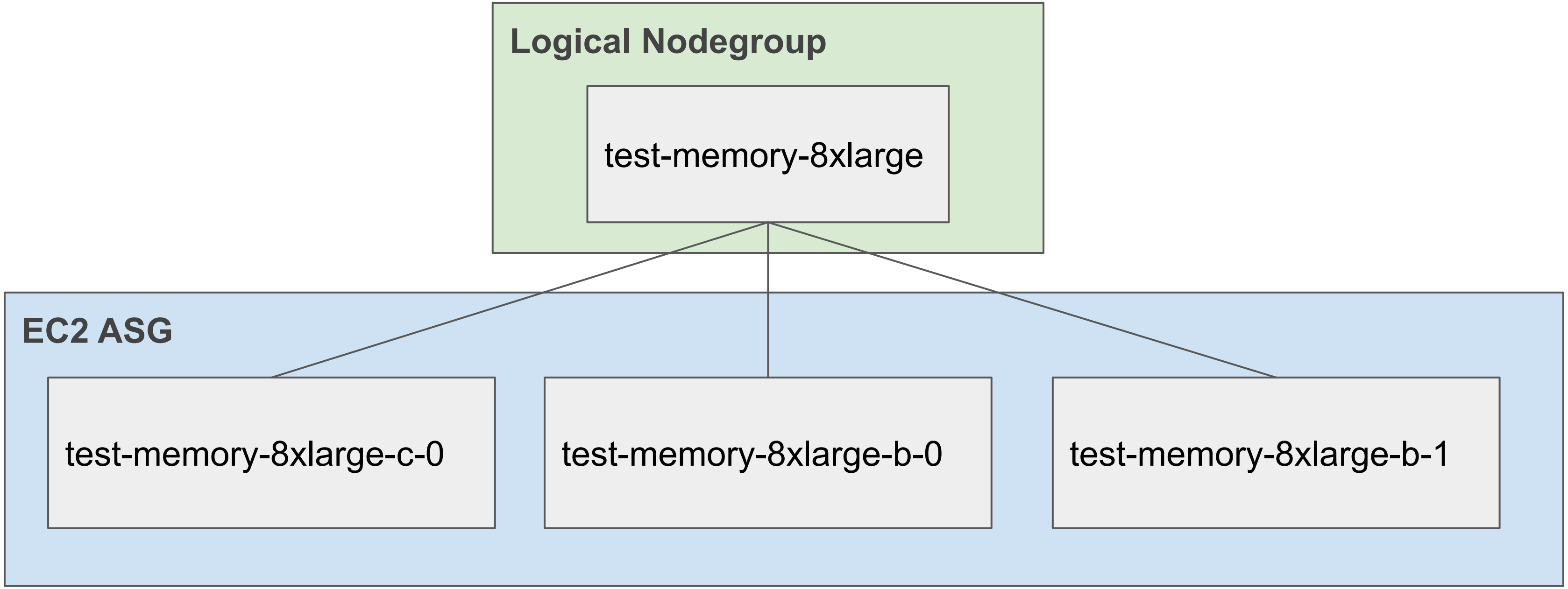
Scaling Challenges

#1 Autoscaling

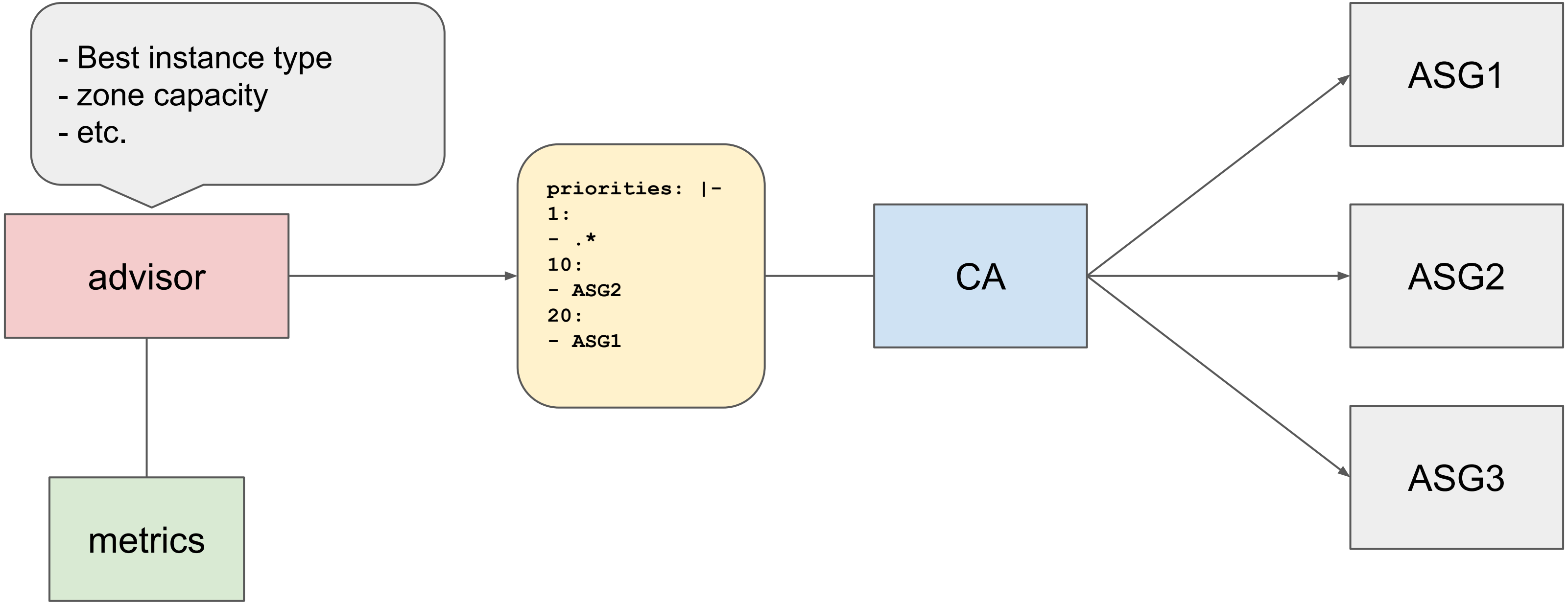
#1 Autoscaling

- Challenges
  - Large number of nodes
  - The load changes significantly during a day
- Solutions
  - Pod Autoscaling: Keda
    - Cron scaler
    - Cloudwatch scaler
  - Node Autoscaling: Cluster Autoscaler

Node Autoscaling



Dynamic Priority



```
priorities: |-
1:
- .*
10:
- ASG2
20:
- ASG1
```

#2 Custom Network Interface

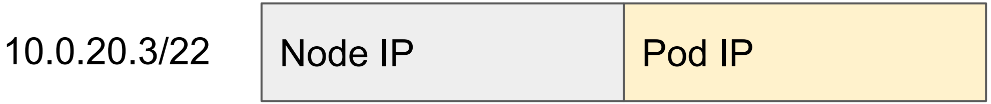
#2 AWS-CNI

- Challenges
  - Pod failed to start due to IP assignment failure
  - *InsufficientFreeAddressesInSubnet: The specified subnet does not have enough free addresses to satisfy the request.*
  - *RequestLimitExceeded: Request limit exceeded*
- Solutions
  - Understand its source code
  - Use custom network: *AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG*
  - Use prefix delegation: *ENABLE_PREFIX_DELEGATION*

Insufficient IP

Before

- aws-cni tries to get IP from the same subnet of the node
- cluster autoscaler doesn’t know that, it will try to add more nodes if there are available IP when scaling up



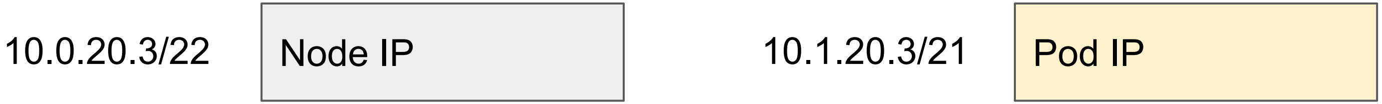
10.0.20.3/22

Node IP

Pod IP

Now

- separated node and pod subnets



10.0.20.3/22

Node IP

10.1.20.3/21

Pod IP

Prefix Delegation

aws-cni applies for new IP when warm target isn’t satisfied and releases when there are too much

If pods are frequently scheduled and deleted on the same node, aws-cni would make many requests to AWS API to get IP.

By using prefix, it applies for one subnet(/28) instead of one IP

```
func (c *IPAMContext) datastorePrefixTargetState() (short int,
enabled bool) {
        …
        freePrefixesInStore := c.dataStore.GetFreePrefixes()
        toAllocate := max(c.warmPrefixTarget-freePrefixesInStore,
0)      log.Debugf("Prefix target is %d, short of %d prefixes, free
%d prefixes", c.warmPrefixTarget, toAllocate, freePrefixesInStore)
        return toAllocate, true
 }
```

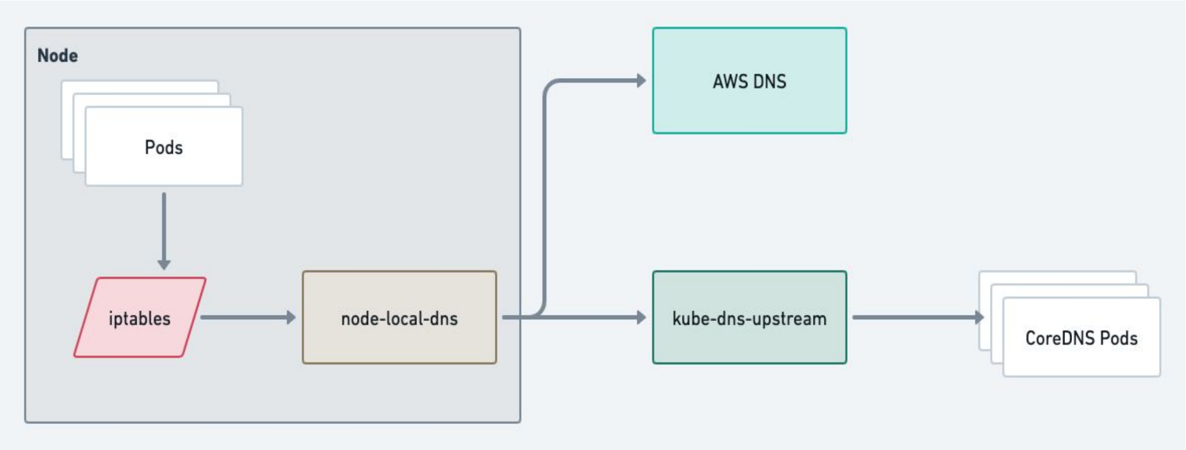
#3 DNS

#3 DNS

- Challenges
  - High latency
  - AWS API limit
- Solutions
  - NodeLocal DNS cache
  - Increase TTL of external domains

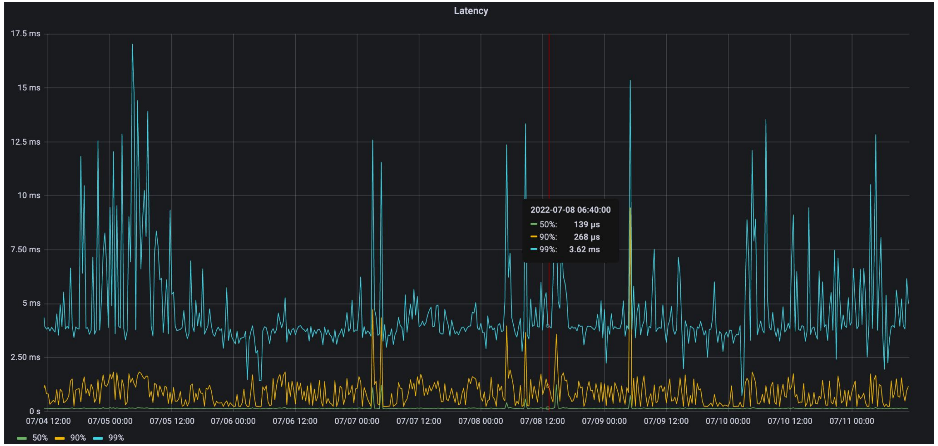
Local DNS cache

```
cluster.local:53 {
        errors
        cache {
                success 9984 60
                denial 9984 5
        }
        reload
        loop
        bind 169.254.20.10 10.100.0.10
        forward . __PILLAR__CLUSTER__DNS__ {
                force_tcp
        }
        prometheus :9253
        health 169.254.20.10:8080
        }
.:53 {
        errors
        cache 300
        reload
        loop
        bind 169.254.20.10 10.100.0.10
        forward . /etc/resolv.conf
        prometheus :9253
}
```

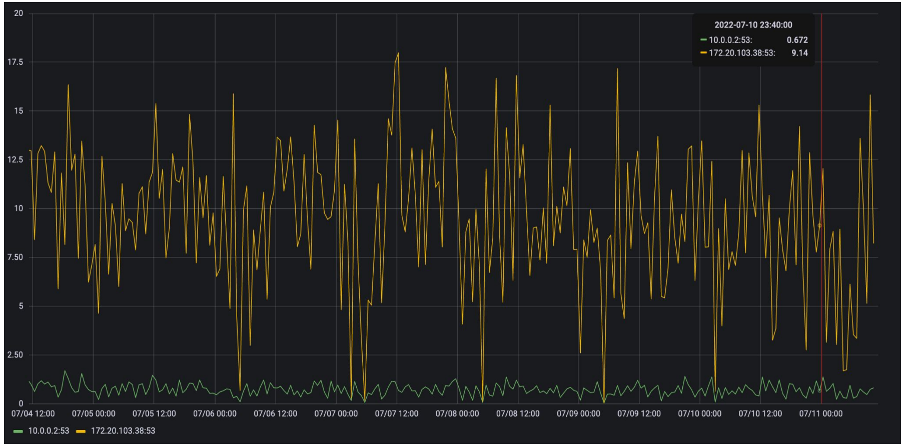


DNS Improvement

Latency



Requests

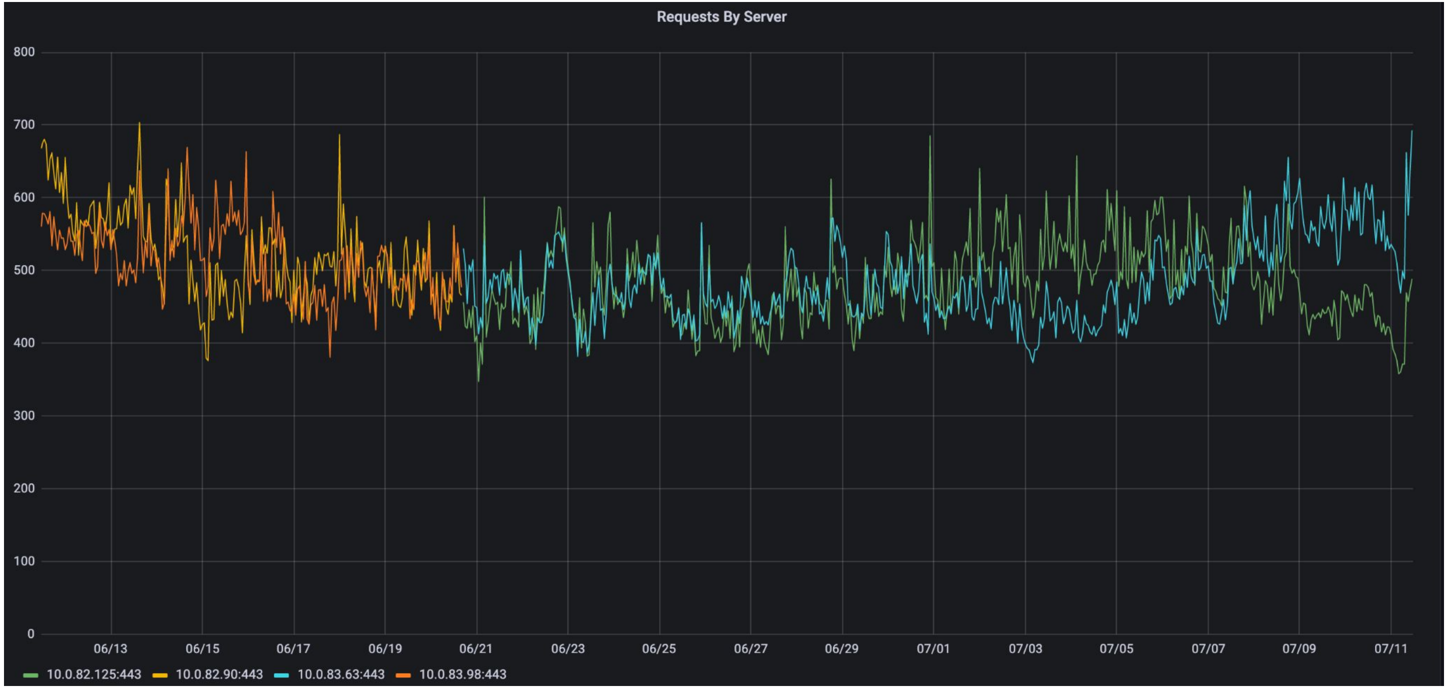


#4 API server

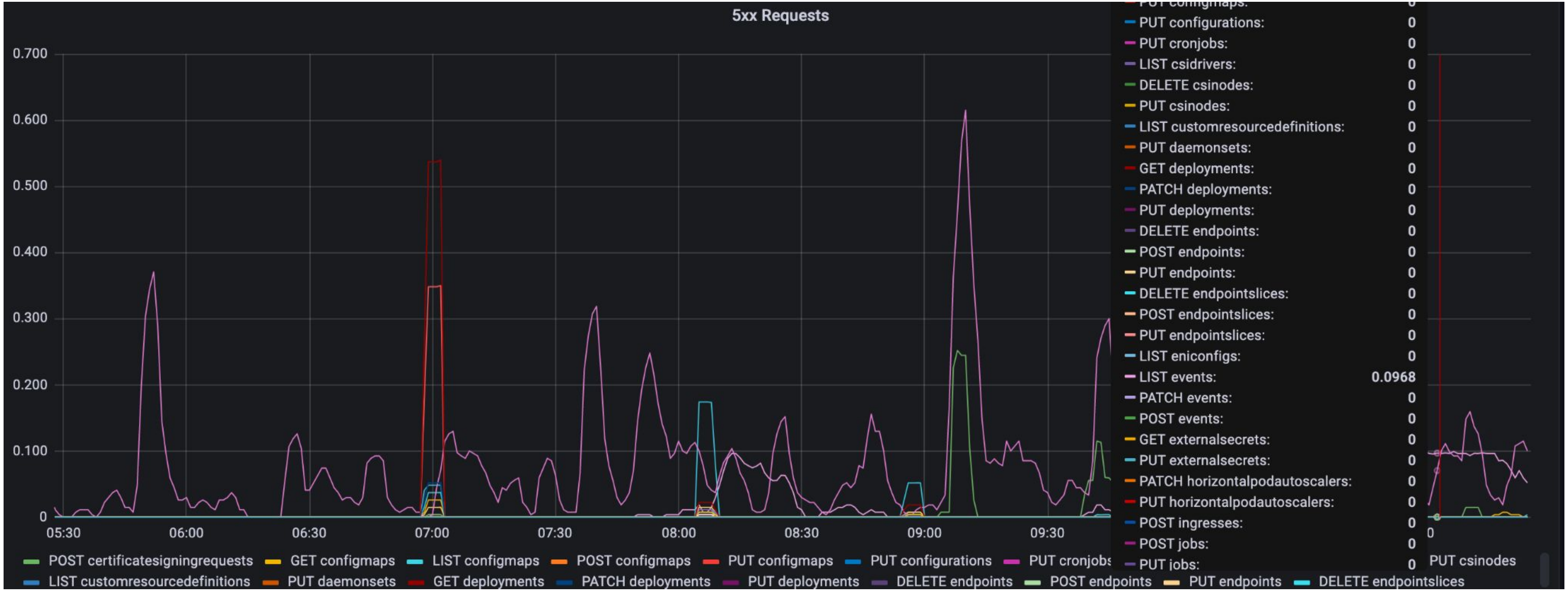
#4 API server

- Challenges
  - Outages
  - Unable to access EKS API
  - High latency
- Solutions
  - Setup monitoring
  - (WIP) Setup proxy server

EKS Monitoring- QPS

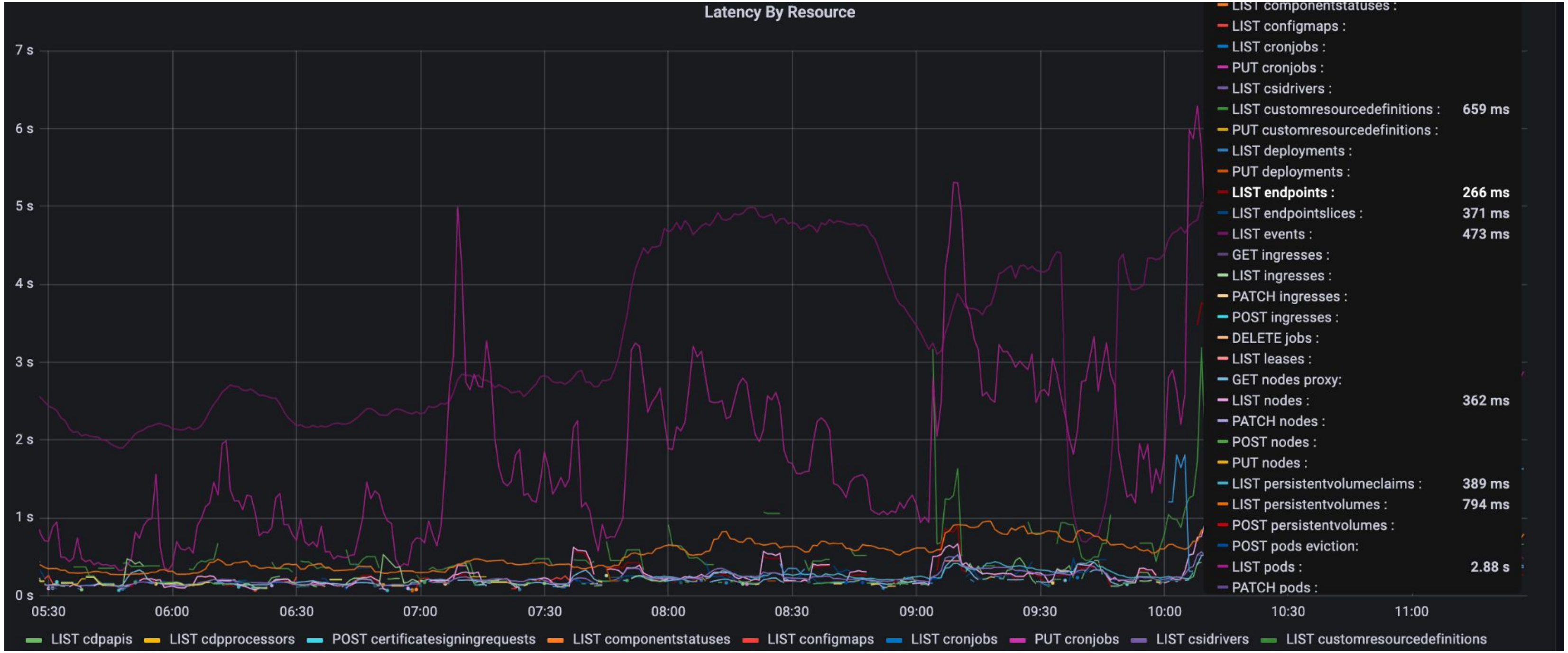


EKS Monitoring- 5xx



PUT configmaps:	0
PUT configurations:	0
PUT cronjobs:	0
LIST csidrivers:	0
DELETE csinodes:	0
PUT csinodes:	0
LIST customresourcedefinitions:	0
PUT daemonsets:	0
GET deployments:	0
PATCH deployments:	0
PUT deployments:	0
DELETE endpoints:	0
POST endpoints:	0
PUT endpoints:	0
DELETE endpointslices:	0
POST endpointslices:	0
PUT endpointslices:	0
LIST eniconfigs:	0
LIST events:	0.0968
PATCH events:	0
POST events:	0
GET externalsecrets:	0
PUT externalsecrets:	0
PATCH horizontalpodautoscalers:	0
PUT horizontalpodautoscalers:	0
POST ingresses:	0
POST jobs:	0
PUT jobs:	0

EKS Monitoring– latency



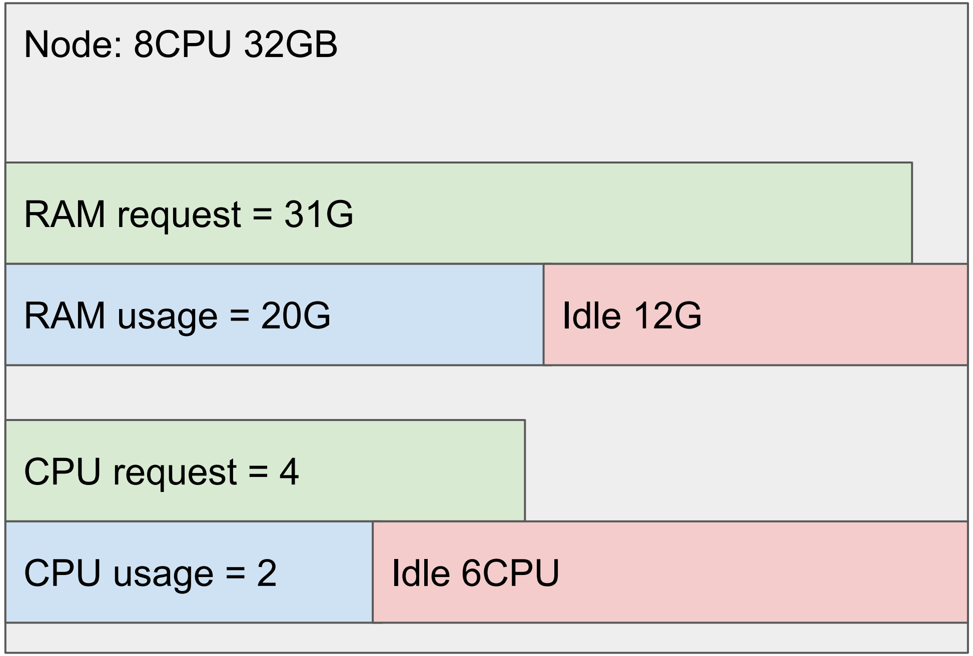
LIST componentstatuses :	
LIST configmaps :	
LIST cronjobs :	
PUT cronjobs :	
LIST csidrivers :	
LIST customresourcedefinitions :	659 ms
PUT customresourcedefinitions :	
LIST deployments :	
PUT deployments :	
LIST endpoints :	**266 ms**
LIST endpointslices :	371 ms
LIST events :	473 ms
GET ingresses :	
LIST ingresses :	
PATCH ingresses :	
POST ingresses :	
DELETE jobs :	
LIST leases :	
GET nodes proxy:	
LIST nodes :	362 ms
PATCH nodes :	
POST nodes :	
PUT nodes :	
LIST persistentvolumeclaims :	389 ms
LIST persistentvolumes :	794 ms
POST persistentvolumes :	
POST pods eviction:	
LIST pods :	2.88 s
PATCH pods :	

#5 Cost Optimization

#5 Cost Optimization

- Challenges
  - Cost increased significantly in last 6 month
- Solutions
  - Resource quota
  - Spot adoption
  - Resource utilization

Resource utilization



Node: 8CPU 32GB

RAM request = 31G

RAM usage = 20G

Idle 12G

CPU request = 4

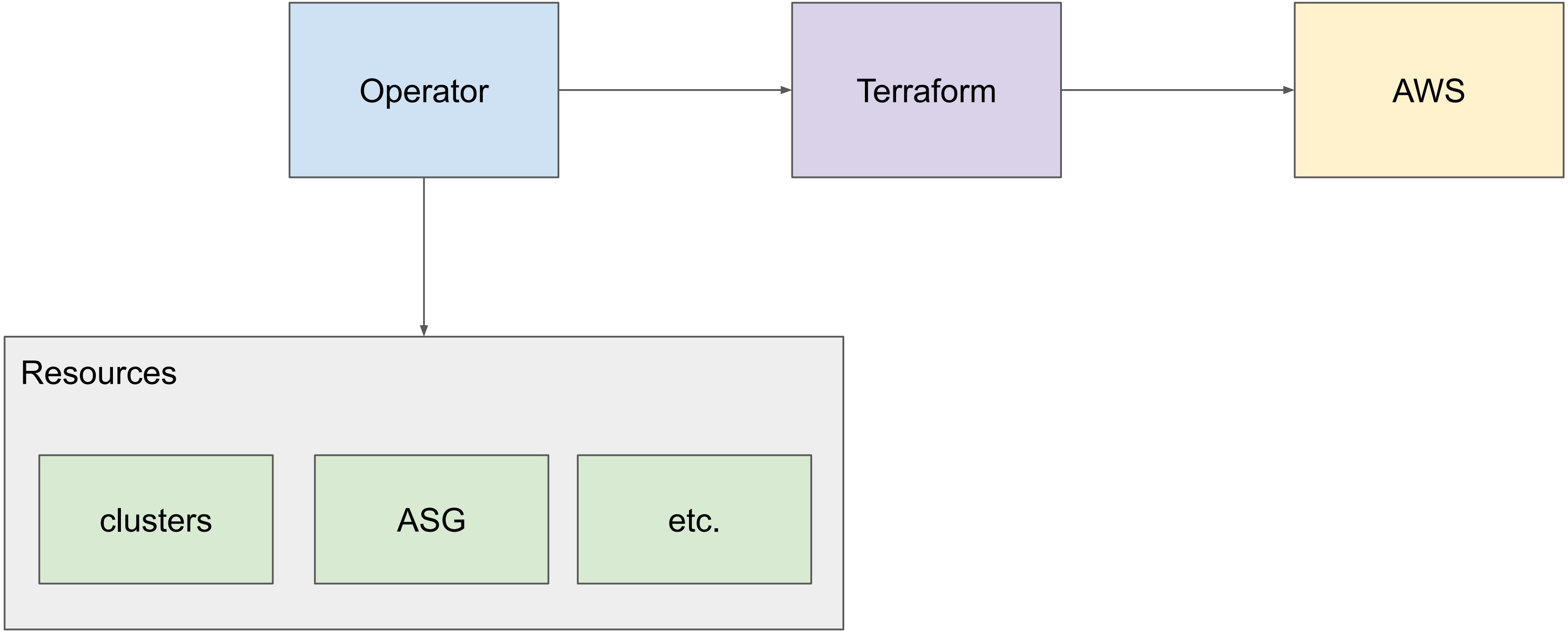
CPU usage = 2

Idle 6CPU

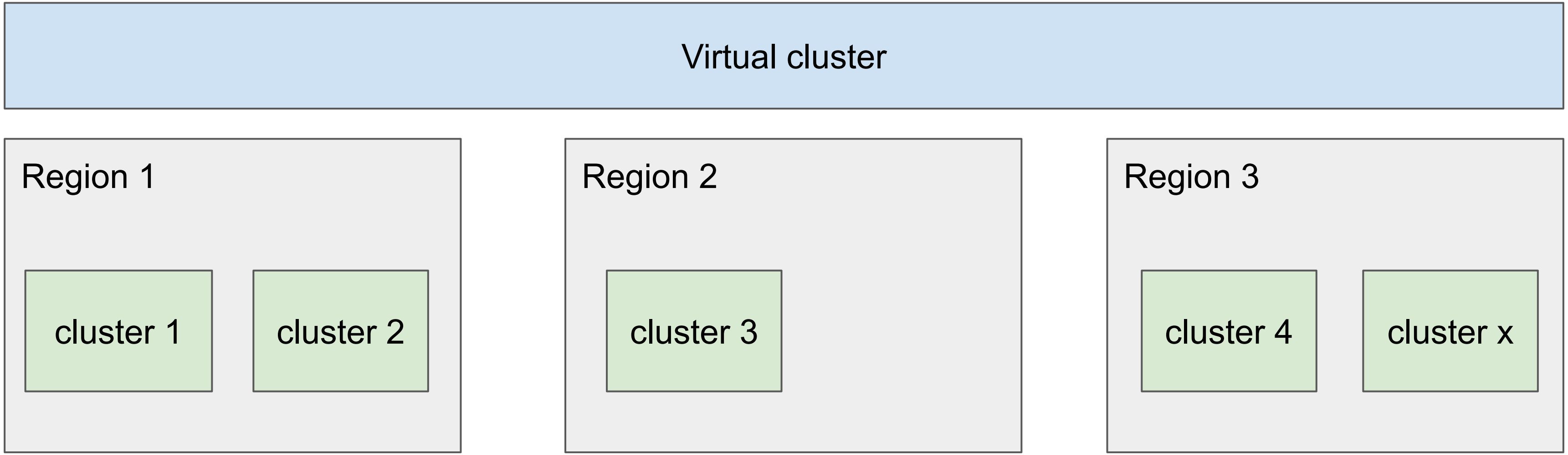
- Use Kubecost to identify low efficiency workloads
- CA dynamic priority

Vision

Cluster Operator



Virtual Cluster



Conclusion

Conclusion

- Prepare for a larger scale when providing new features
    - Especially for operators that work with the entire cluster
- Have a deep understanding of crucial components
    - Even on EKS you will run components (CNI, CA, etc) yourself
- Be cost sensitive all the time
    - Hard to manage node utilization and cross-AZ traffic