Amazon EKS

云上托管的K8S集群

杨历，AWS解决方案架构师

议程

- **AWS 容器技术介绍**
- Amazon EKS(云上托管的K8S集群)介绍
- Amazon EKS对K8S网络的创新
- K8S Service 与AWS的集成
- Amazon EKS与AWS安全解决方案整合
- Amazon EKS日志及监控
- Demo演示

AWS容器生态系统

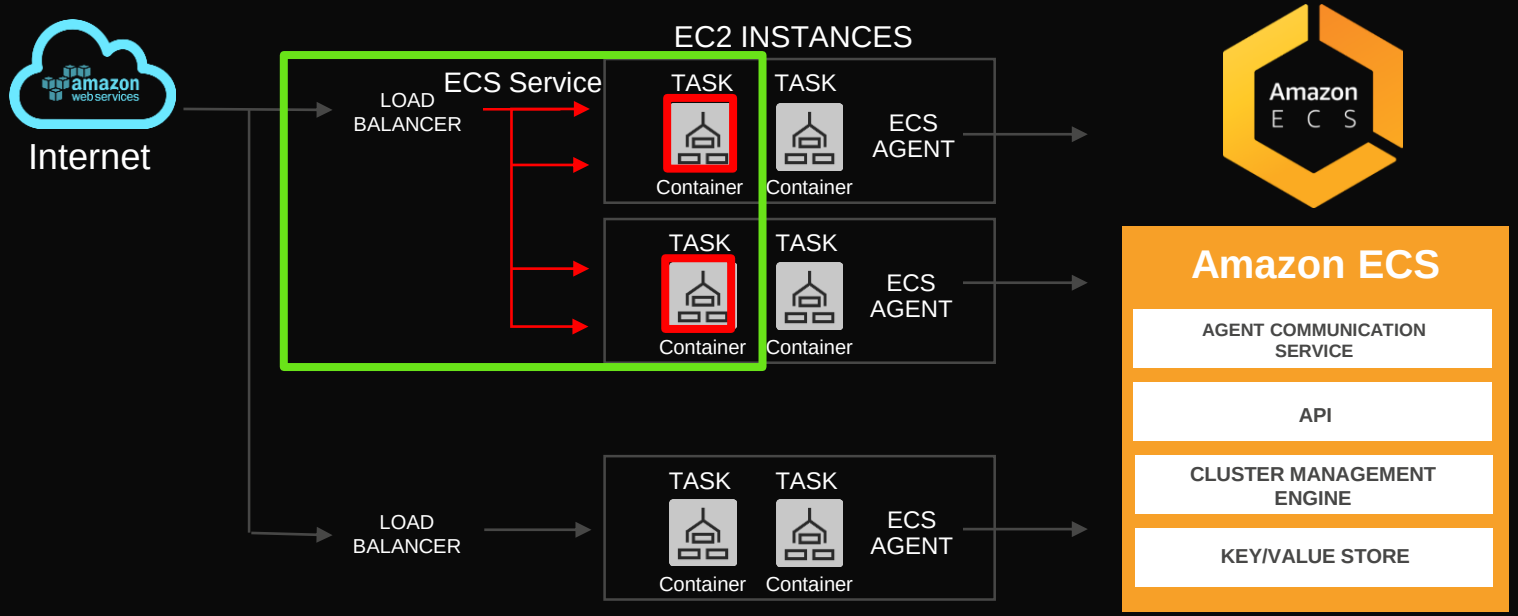
Amazon ECS

Amazon EKS

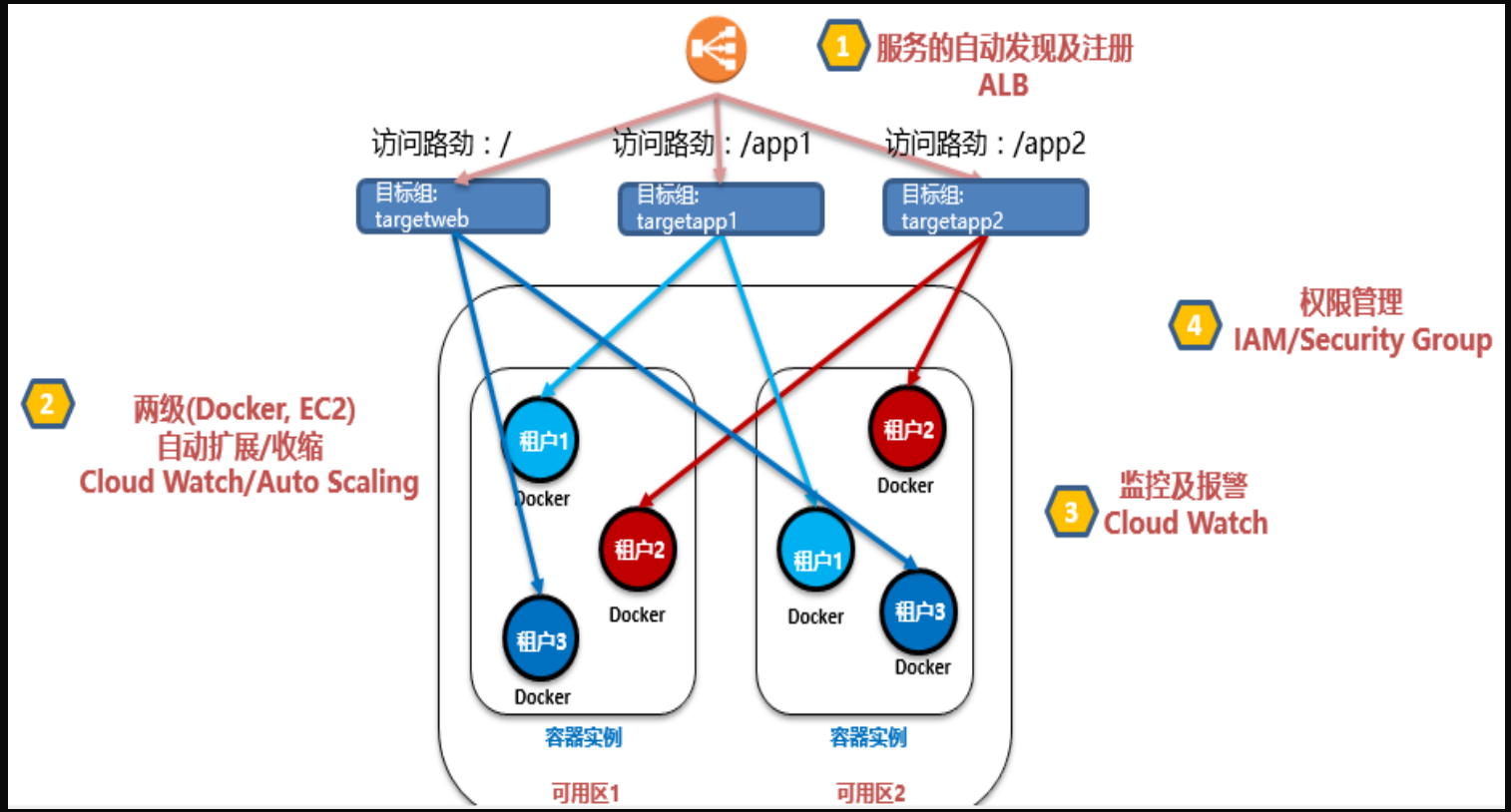
Fargate

Amazon ECR

Amazon ECS—Task & Service



AWS ECS 适用场景

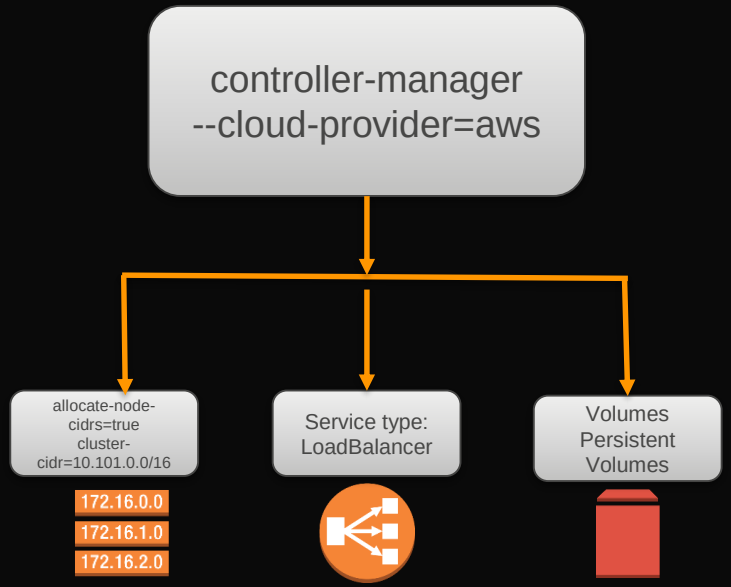

1 服务的自动发现及注册
ALB
访问路劲：/
访问路劲：/app1
访问路劲：/app2
目标组: targetweb
目标组: targetapp1
目标组: targetapp2
4 权限管理
IAM/Security Group
2 两级(Docker, EC2)
自动扩展/收缩
Cloud Watch/Auto Scaling
3 监控及报警
Cloud Watch
租户1
Docker
租户2
Docker
租户3
Docker
租户2
Docker
租户1
Docker
租户3
Docker
容器实例
容器实例
可用区1
可用区2

57%

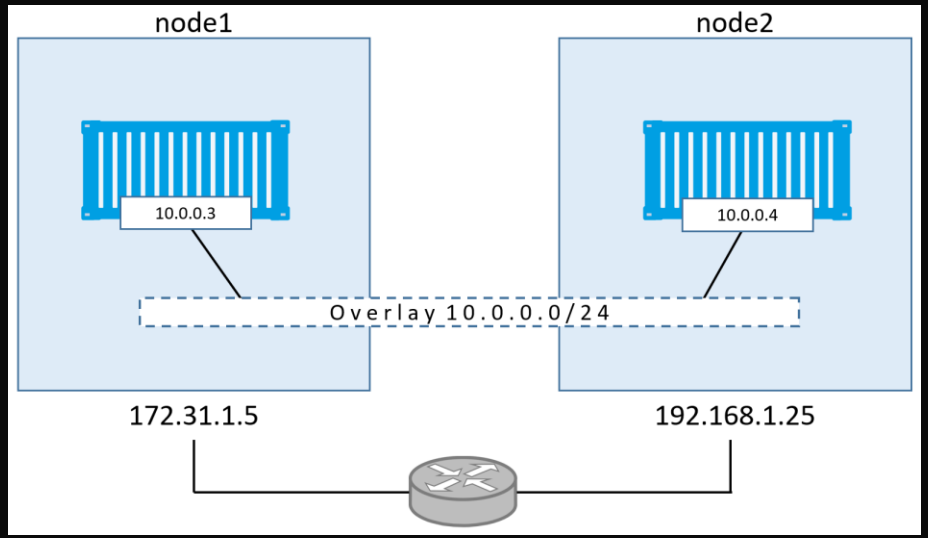
Kubernetes工作负载运行在AWS上

—CNCF survey

KOPS (在AWS上构建K8S集群的工具)



云特性

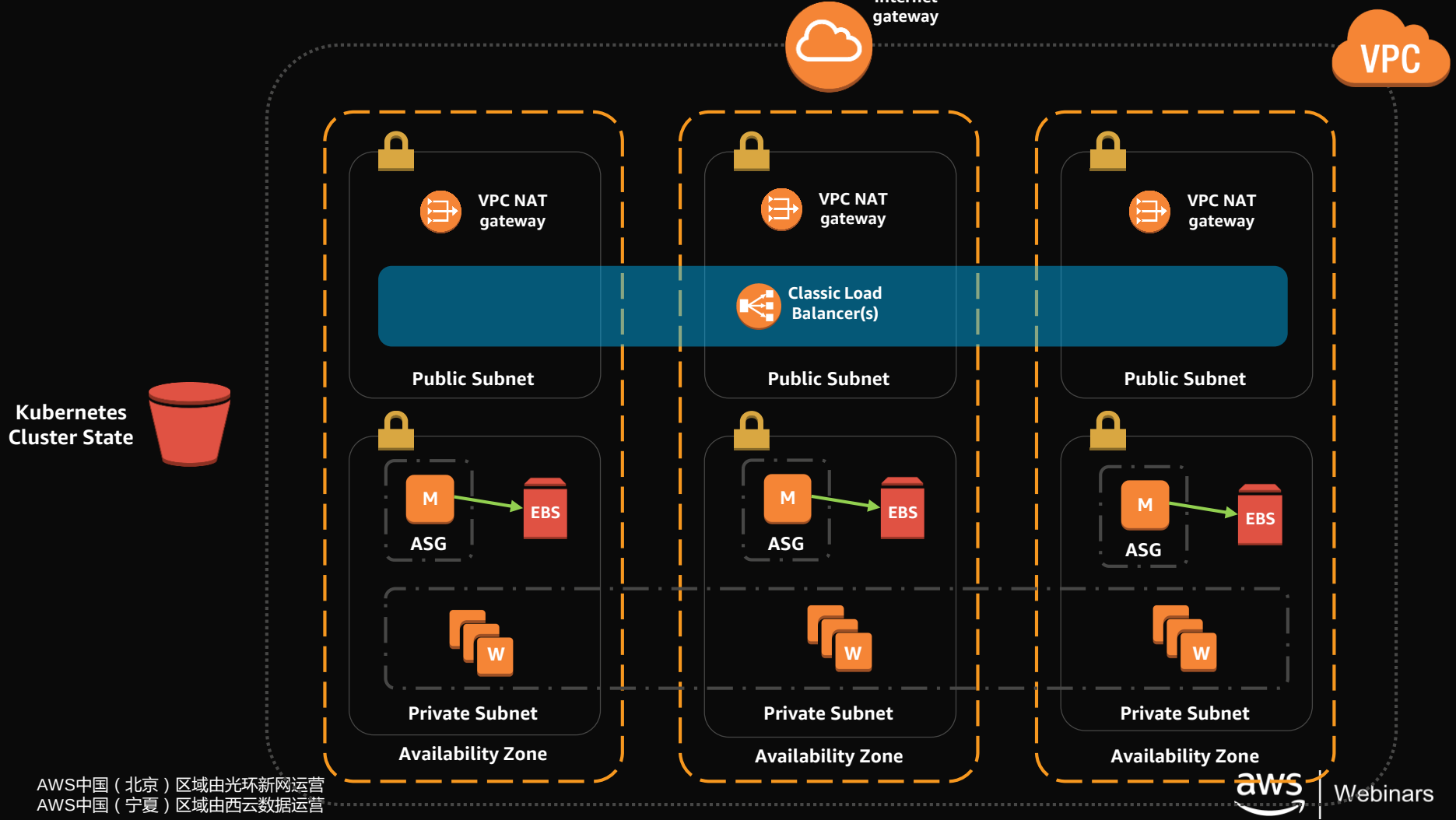


CNI 插件

部署文档：https://kubernetes.io/docs/getting-started-guides/aws/

1. Install Binaries & Tools: kops, AWS CLI tools, kubectl
2. Set IAM User to “kops”
3. Allow “kops” user Full access to EC2, Route53, S3, IAM, VPC
4. Configure AWS client to new IAM user “kops”
5. Configure DNS (or) Deploy a gossip-based cluster:
   - We hosted the subdomain “dnishi.k8sdemolabs.com” in Route53
6. Create a S3 bucket to save cluster config: “dnishi-kops-store”
7. Set the “kops environmental variables”
8. Create cluster: ”kops create cluster” and “kops validate cluster”

```
kops create cluster \
    --api-loadbalancer-type=public \
    --vpc vpc-7d4ef914 \
    --network-cidr 10.2.0.0/16 \
    --master-zones cn-northwest-1a,cn-northwest-1b,cn-northwest-1c --master-count 3 \
    --zones cn-northwest-1a,cn-northwest-1b,cn-northwest-1c --node-count 3 \
    --node-size t2.medium \
    --master-size t2.medium \
    --topology private \
    --networking amazon-vpc-routed-eni \
    --cloud-labels "Team=Dev,Owner=Martin Yang" \
    --image ami-1b7f6879 \
kopsdemo.k8s.local
```



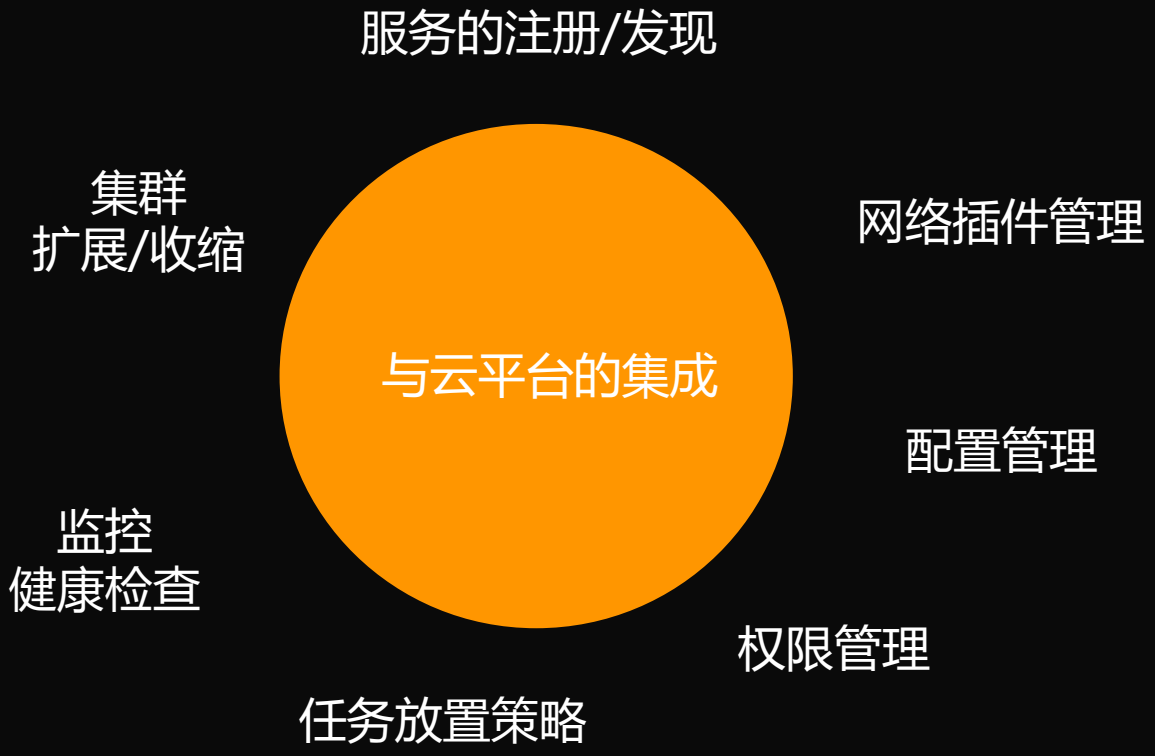
AWS中国（北京）区域由光环新网运营
AWS中国（宁夏）区域由西云数据运营

aws | Webinars

议程

- AWS 容器技术介绍
- **Amazon EKS(云上托管的K8S集群)**
- EKS对K8S网络的创新
- K8S Service 与AWS的集成
- EKS与AWS安全解决方案整合
- EKS日志及监控
- Demo演示

容器编排的痛点



Amazon EKS

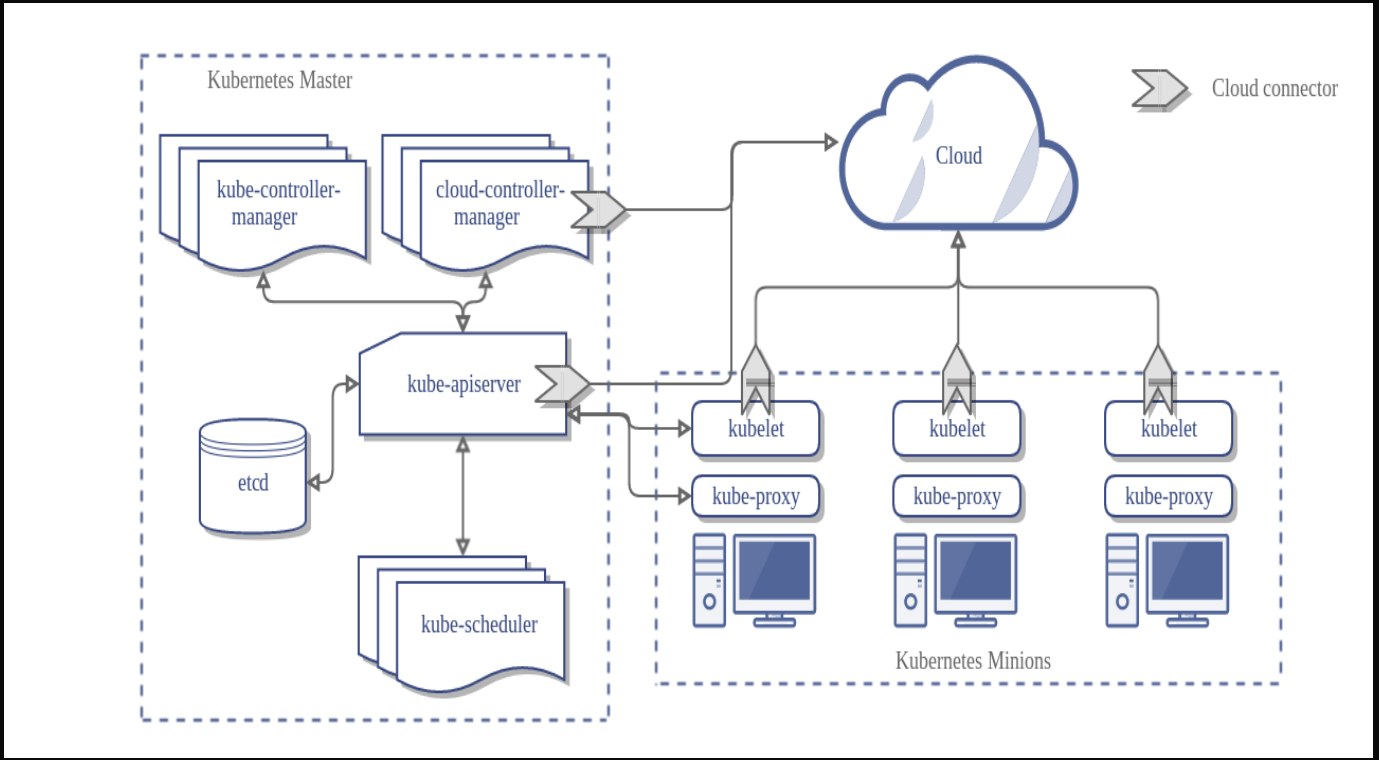
和开源 Kubernetes 一致体验

Upstream 保持和上游同步

支持企业生产级别的容器应用

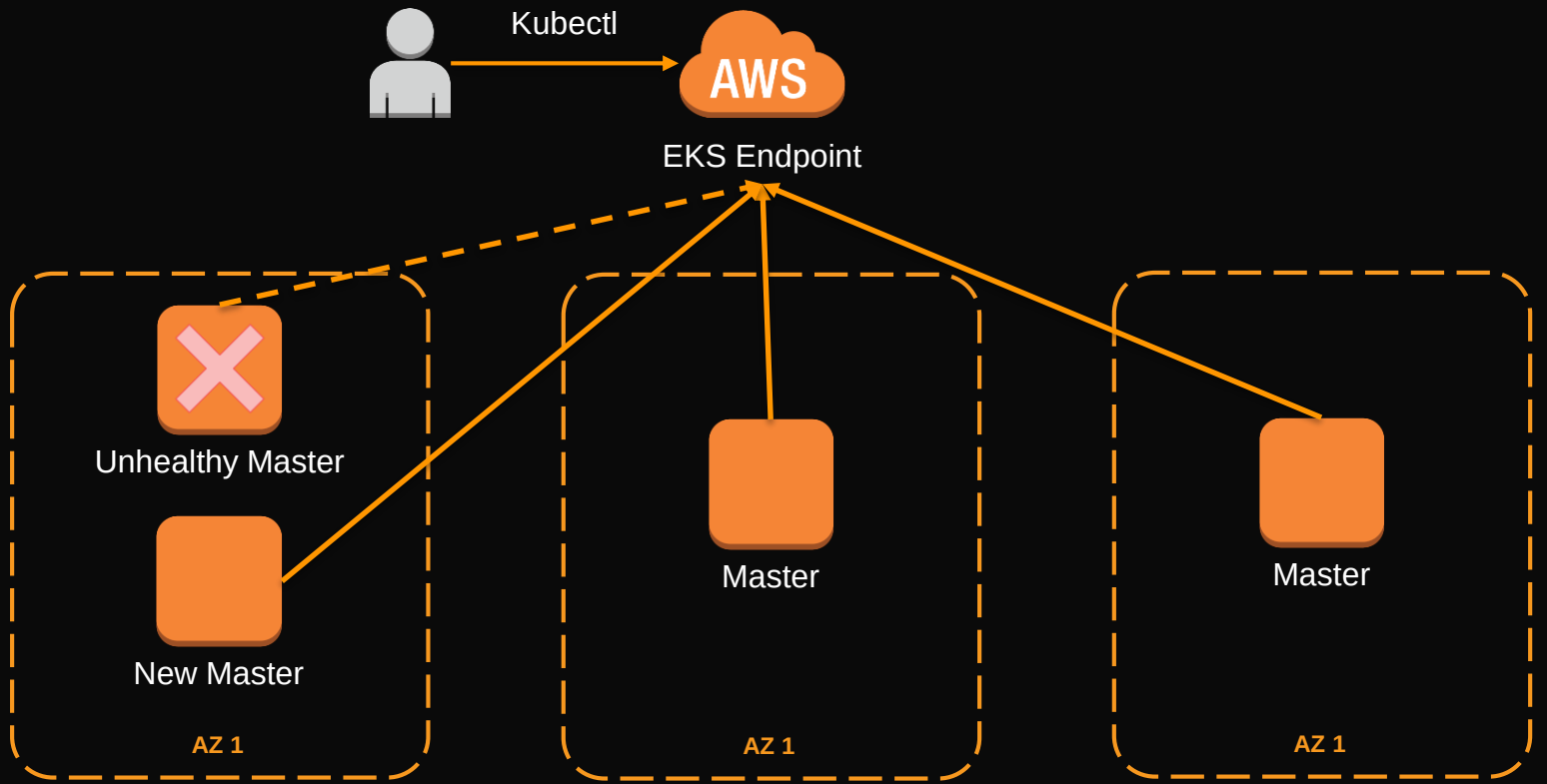
按需和 AWS 服务无缝集成

自动升级打补丁

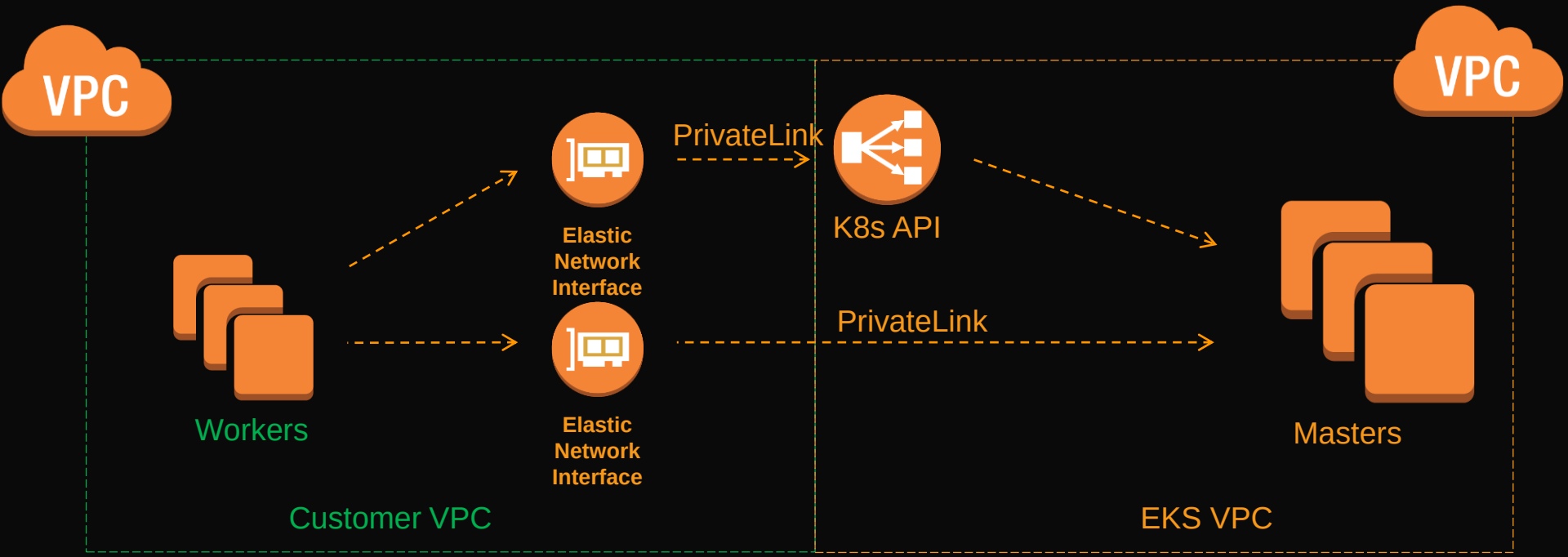


AWS托管的高可用性方案

3个可用区，3个主节点

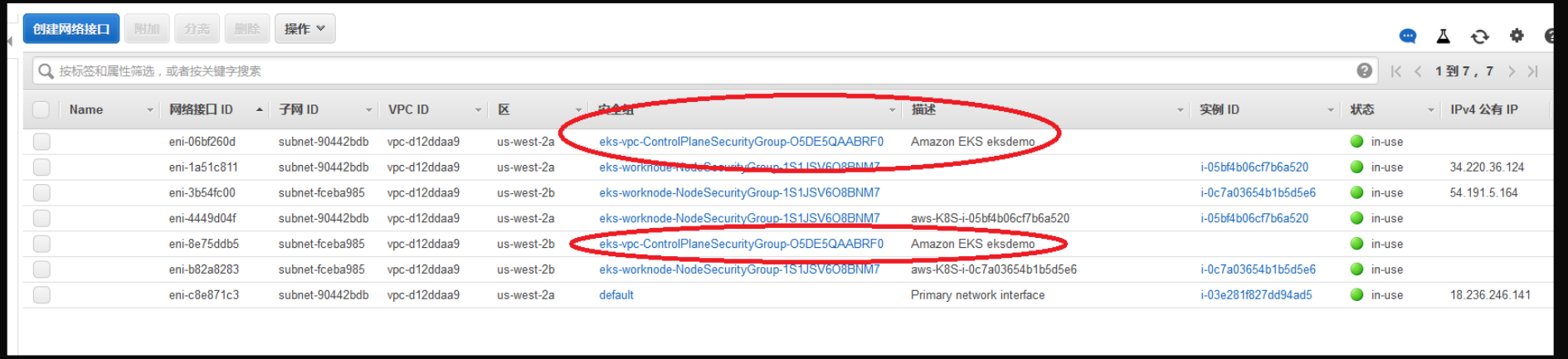


主节点与工作节点的网络连接 PrivateLink



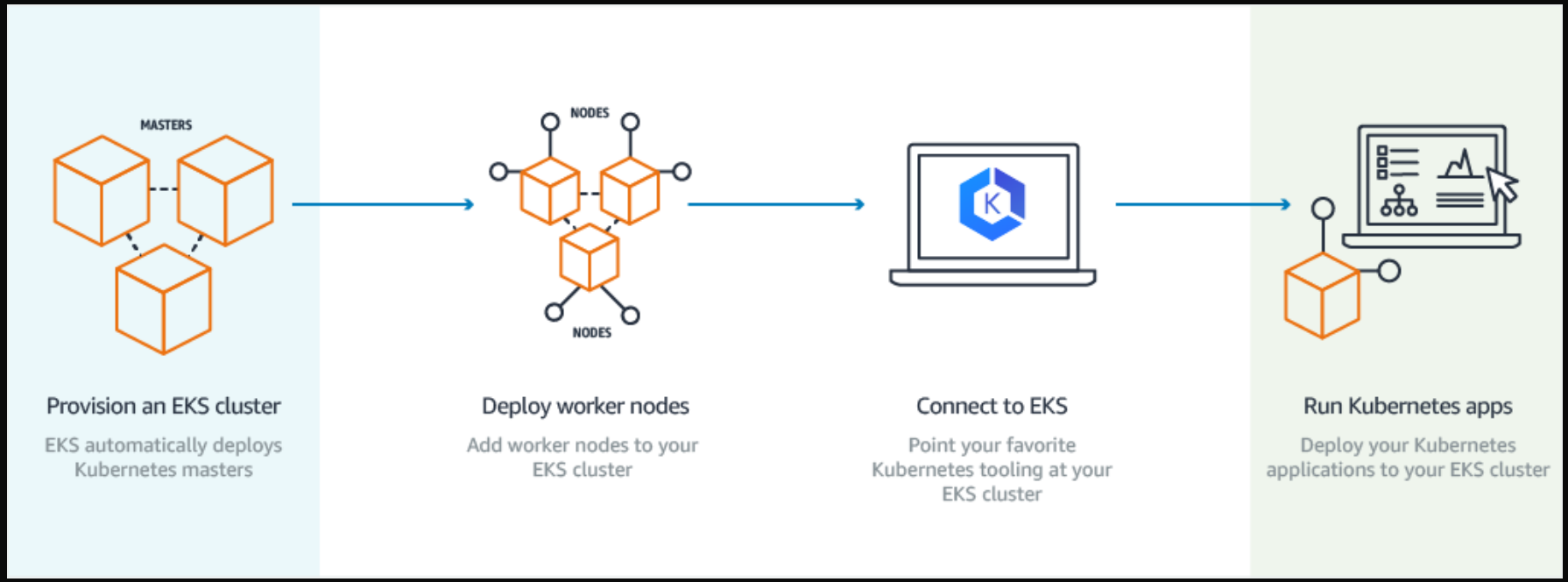
AWS中国（北京）区域由光环新网运营
AWS中国（宁夏）区域由西云数据运营

主节点与工作节点的网络连接 PrivateLink



在EKS中，主节点和API服务器通过PrivateLink向工作节点公开。在工作节点的VPC中以ENI方式出现，在工作节点和主节点之间提供高速网络，而无需穿越公共互联网。

Amazon EKS 的工作原理



Amazon EKS总结

- EKS是CNCF基金会认证的原生Kubernetes
- 管理新版本升级

- 主节点由AWS托管，托管的3个主节点分布在三个可用区实现高可用性

- 您的工作节点在您自己的VPC中, VPC可以是新创建或现有的VPC
- 工作节点可以运行在私有子网, 工作节点由用户自己管理，可以使用定制的AMI(AWS提供工具)，GPU实例等等.

- 通过PrivateLink来实现主节点与工作节点的高速，稳定网络连接
- AWS提供CloudFormation脚本来自动创建VPC及工作节点

Amazon EKS总结

- EKS 会自动备份，监控etcd
- Kube API server, controller, scheduler 日志会发送到CloudWatch
- 通过CloudTrail 来监控EKS API调用
- 未来会支持无服务器化Fargate

怎样迁移到Amazon EKS

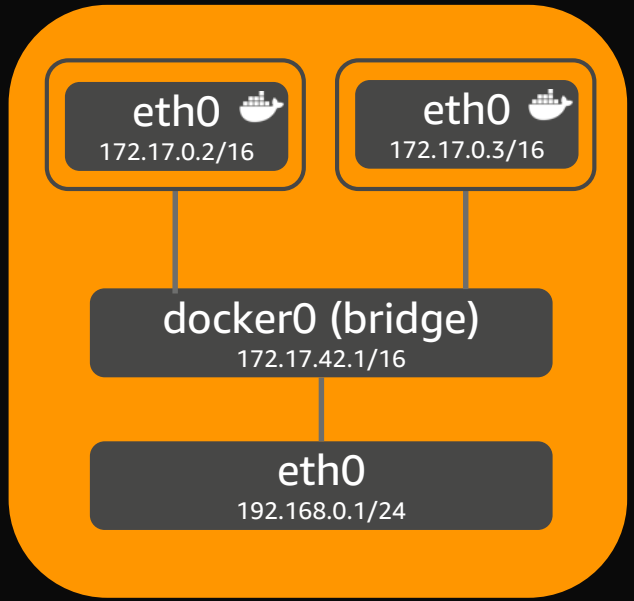
修改kubectl的配置文件指向 EKS
重新发布应用程序

不能通过备份/恢复etcd的方式来迁移

议程

- AWS 容器技术介绍
- Amazon EKS(云上托管的K8S集群)
- **EKS对K8S网络的创新**
- K8S Service 与AWS的集成
- Amazon EKS与AWS安全解决方案整合
- Amazon EKS日志及监控
- Demo演示

Docker: 网络模式 - bridge (默认)



$ docker run –p 8080:8080 maddox/fast-http

Running a container with ports mapped sets up a NAT with iptables

192.168.0.1:8080 -> 172.17.0.2:8080

Docker:网络模式 - host

$ docker run –p 8080:8080 maddox/fast-http
--network host

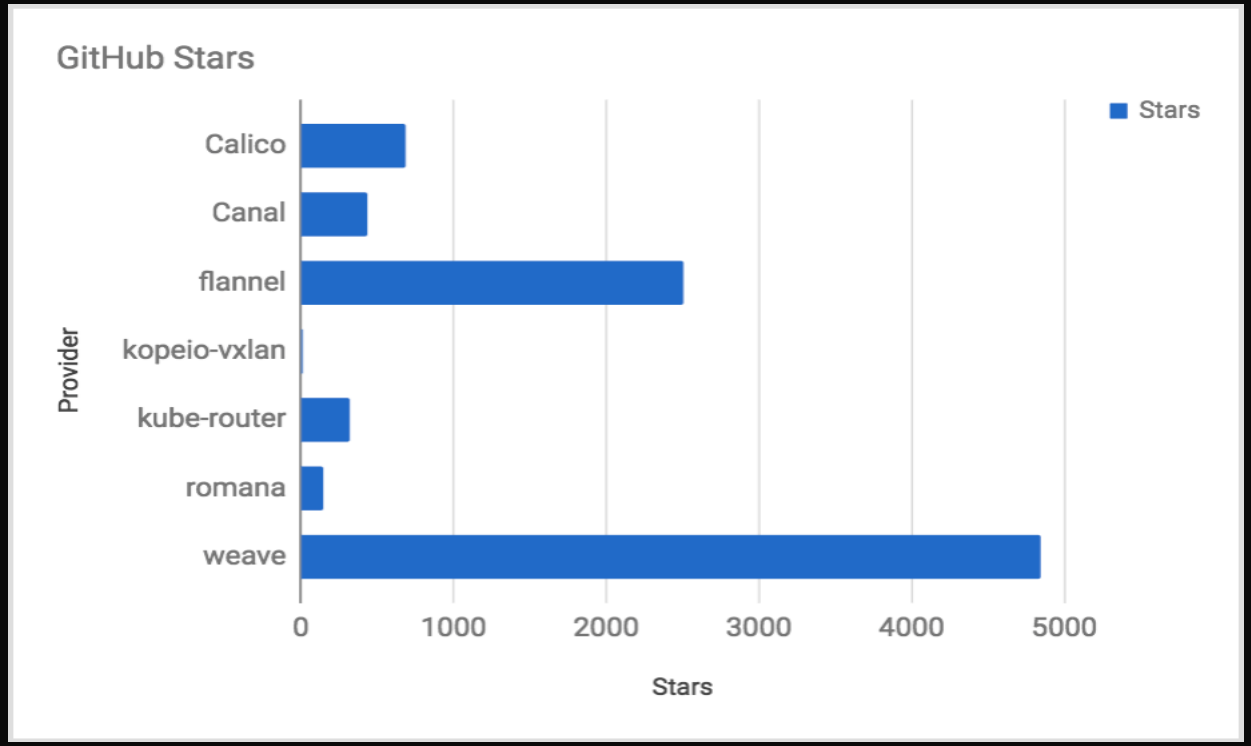
☺ **No performance overhead**

☹ **Only one port 8080 per host**

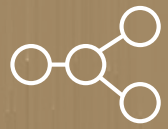
Kubernetes网络

- 每个pod都有一个IP地址
- 容器看到的IP与其他人看到的IP相同
- Kubernetes通过插件的模式来实现网络解决方案(CNI)

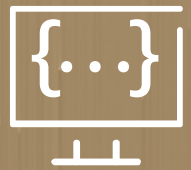
主流的开源网络插件


GitHub Stars
Stars
Provider
Calico
Canal
flannel
kopeio-vxlan
kube-router
romana
weave
0
1000
2000
3000
4000
5000
Stars

amazon-vpc-cni-k8s 网络插件



与VPC网络集成的插件



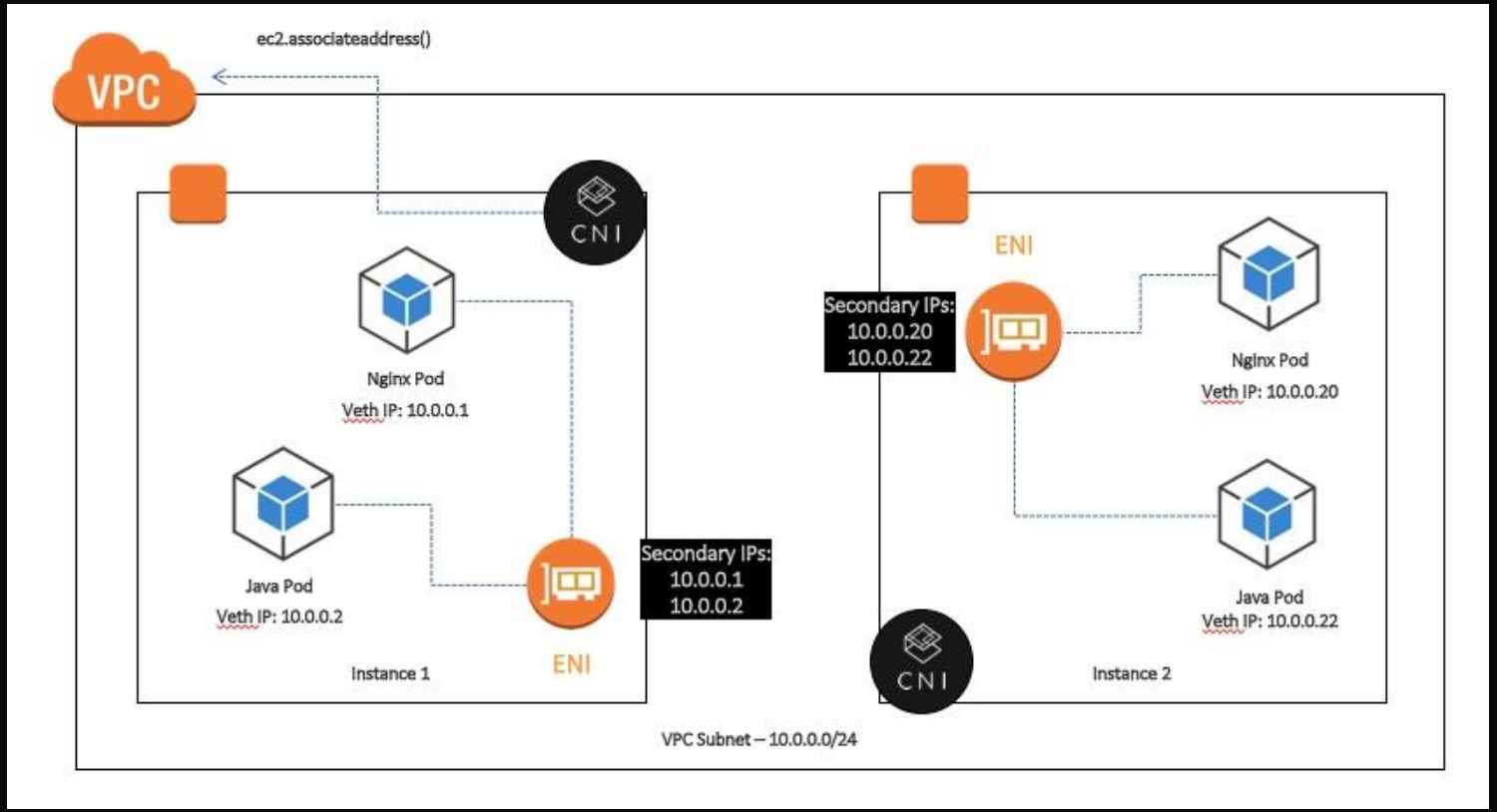
Pods 具有物理的VPC网络地址

简单，安全的网络

开源，在Github上

In Kops, use `--networking amazon-vpc-routed-eni`

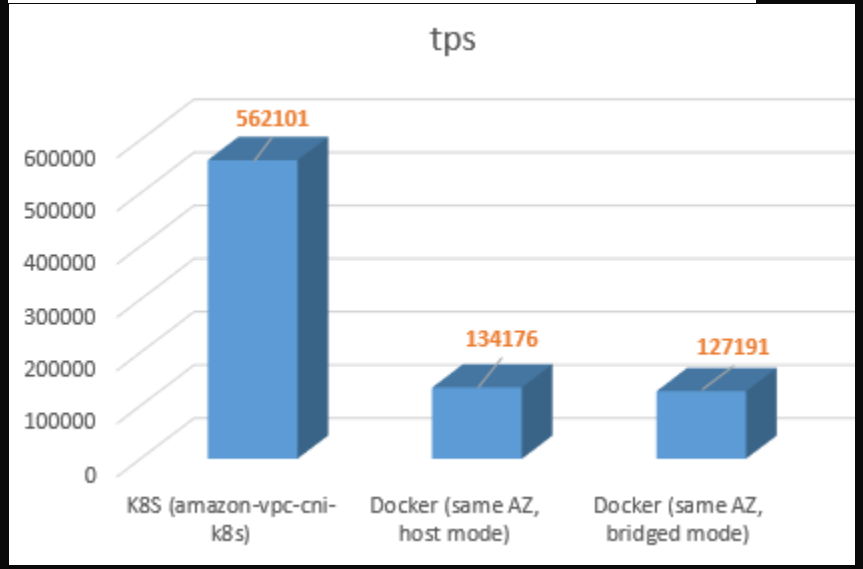
amazon-vpc-cni-k8s插件原理



amazon-vpc-cni-k8s插件原理

实例类型	最大网络接口数	每个接口的 IPv4 地址数	每个接口的 IPv6 地址数
c1.medium	2	6	不支持 IPv6
c1.xlarge	4	15	不支持 IPv6
c3.large	3	10	10
c3.xlarge	4	15	15
c3.2xlarge	4	15	15
c3.4xlarge	8	30	30
c3.8xlarge	8	30	30
c4.large	3	10	10
c4.xlarge	4	15	15
c4.2xlarge	4	15	15
c4.4xlarge	8	30	30
c4.8xlarge	8	30	30
c5.large	3	10	10

私有 DNS	ip-192-168-160-210.ec2.internal
私有 IP	192.168.160.210, 192.168.161.39, 192.168.143.151
辅助私有 IP	192.168.143.220, 192.168.131.140, 192.168.149.77, 192.168.145.15, 192.168.190.193, 192.168.157.188, 192.168.189.173, 192.168.136.174, 192.168.183.14, 192.168.147.241, 192.168.130.73, 192.168.158.10, 192.168.178.13, 192.168.181.161, 192.168.177.164



议程

- AWS 容器技术介绍
- Amazon EKS(云上托管的K8S集群)
- Amazon EKS对K8S网络的创新
- **K8S Service 与AWS的集成**
- Amazon EKS与AWS安全解决方案整合
- Amazon EKS日志及监控
- Demo演示

Kubernetes Services

将一组容器（pod）部署到Kubernetes时

ClusterIP virtual IP, accessible from all nodes

LoadBalancer automatically creates a public ELB (using IAM role)

NodePort bind service to the same port on every host

Services: ClusterIP

$ kubectl run nginx --image=nginx --replicas 3 --port=80 $ kubectl expose deployment nginx

$ kubectl get services

```
NAME        TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
nginx       ClusterIP   100.67.104.10   <none>           80/TCP       17s
```

Now all hosts can connect to 100.67.104.10 (or via DNS as nginx)

Services: ClusterIP

Cluster IP doesn't actually exist on any host, or anywhere.

Each node runs a 'kube-proxy' container.

kube-proxy creates iptables rules on each host to redirect ClusterIP to pod(s) IPs:

```
$ kubectl get services
NAME         TYPE         CLUSTER-IP      EXTERNAL-IP       PORT(S)       AGE
nginx        ClusterIP    100.67.104.10   <none>            80/TCP        1m

$ iptables-save | grep nginx
-A KUBE-SEP-26VNBZMXBHB7VWNZ -s 100.123.148.131/32 -m comment --comment "default/nginx:" -j KUBE-MARK-MASQ
-A KUBE-SEP-26VNBZMXBHB7VWNZ -p tcp -m comment --comment "default/nginx:" -m tcp -j DNAT --to-destination 100.123.148.131:80
-A KUBE-SEP-BHBFJ25VE6FZ3NVP -s 100.122.216.130/32 -m comment --comment "default/nginx:" -j KUBE-MARK-MASQ
-A KUBE-SEP-BHBFJ25VE6FZ3NVP -p tcp -m comment --comment "default/nginx:" -m tcp -j DNAT --to-destination 100.122.216.130:80
-A KUBE-SEP-TVBSTANGUXMLRRM2 -s 100.123.46.130/32 -m comment --comment "default/nginx:" -j KUBE-MARK-MASQ
-A KUBE-SEP-TVBSTANGUXMLRRM2 -p tcp -m comment --comment "default/nginx:" -m tcp -j DNAT --to-destination 100.123.46.130:80

-A KUBE-SERVICES ! -s 100.96.0.0/11 -d 100.67.104.10/32 -p tcp -m comment --comment "default/nginx: cluster IP" -m tcp --dport 80 -j KUBE-MARK-MASQ
-A KUBE-SERVICES -d 100.67.104.10/32 -p tcp -m comment --comment "default/nginx: cluster IP" -m tcp --dport 80 -j KUBE-SVC-4N57TFCL4MD7ZTDA

-A KUBE-SVC-4N57TFCL4MD7ZTDA -m comment --comment "default/nginx:" -m statistic --mode random --probability 0.33332999982 -j KUBE-SEP-BHBFJ25VE6FZ3NVP
-A KUBE-SVC-4N57TFCL4MD7ZTDA -m comment --comment "default/nginx:" -m statistic --mode random --probability 0.50000000000 -j KUBE-SEP-26VNBZMXBHB7VWNZ
-A KUBE-SVC-4N57TFCL4MD7ZTDA -m comment --comment "default/nginx:" -j KUBE-SEP-TVBSTANGUXMLRRM2
```

pod1
pod2
pod3

Iptables的弱点

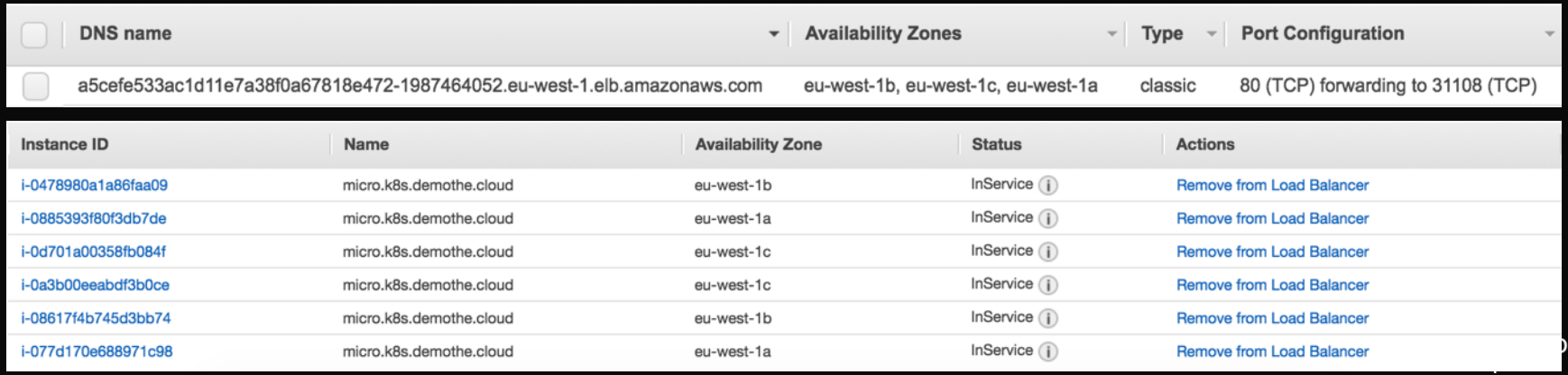
- Iptables在有大量规则时, 效率低.
- 更新期间锁定
    - 5,000 services == 40,000 rules == ~11分
    - 20,000 services == 160,000 rules == ~5 小时
- 较早的服务规则出现在iptables链的头部= faster
- 较新的服务规则稍后会出现在iptables链后部= slower

新版本（1.11）会用IPVS取代iptables
这将160,000规则的更新时间缩短到2毫秒

Services: LoadBalancer

```
$ kubectl run nginx --image=nginx --replicas 3 --port=80
$ kubectl expose deployment nginx --type=LoadBalancer
$ kubectl get services -o=wide
```

```
NAME    TYPE          CLUSTER-IP     EXTERNAL-IP                                                                PORT(S)
nginx   LoadBalancer  100.70.217.164 a5cefe533ac1d11e7a38f0a67818e472-1987464052.eu-west-1.elb.amazonaws.com   80:31108/TCP
```



DNS name	Availability Zones	Type	Port Configuration
a5cefe533ac1d11e7a38f0a67818e472-1987464052.eu-west-1.elb.amazonaws.com	eu-west-1b, eu-west-1c, eu-west-1a	classic	80 (TCP) forwarding to 31108 (TCP)

Instance ID	Name	Availability Zone	Status	Actions
i-0478980a1a86faa09	micro.k8s.demothe.cloud	eu-west-1b	InService	Remove from Load Balancer
i-0885393f80f3db7de	micro.k8s.demothe.cloud	eu-west-1a	InService	Remove from Load Balancer
i-0d701a00358fb084f	micro.k8s.demothe.cloud	eu-west-1c	InService	Remove from Load Balancer
i-0a3b00eeabdf3b0ce	micro.k8s.demothe.cloud	eu-west-1c	InService	Remove from Load Balancer
i-08617f4b745d3bb74	micro.k8s.demothe.cloud	eu-west-1b	InService	Remove from Load Balancer
i-077d170e688971c98	micro.k8s.demothe.cloud	eu-west-1a	InService	Remove from Load Balancer

通过annotations来配置ELB

service.beta.kubernetes.io/	aws-load-balancer-type
service.beta.kubernetes.io/	aws-load-balancer-internal
service.beta.kubernetes.io/	aws-load-balancer-proxy-protocol
service.beta.kubernetes.io/	aws-load-balancer-access-log-emit-interval
service.beta.kubernetes.io/	aws-load-balancer-access-log-enabled
service.beta.kubernetes.io/	aws-load-balancer-access-log-s3-bucket-name
service.beta.kubernetes.io/	aws-load-balancer-access-log-s3-bucket-prefix
service.beta.kubernetes.io/	aws-load-balancer-connection-draining-enabled
service.beta.kubernetes.io/	aws-load-balancer-connection-draining-timeout
service.beta.kubernetes.io/	aws-load-balancer-connection-idle-timeout
service.beta.kubernetes.io/	aws-load-balancer-cross-zone-load-balancing-enabled
service.beta.kubernetes.io/	aws-load-balancer-extra-security-groups
service.beta.kubernetes.io/	aws-load-balancer-ssl-cert
service.beta.kubernetes.io/	aws-load-balancer-ssl-ports
service.beta.kubernetes.io/	aws-load-balancer-ssl-negotiation-policy
service.beta.kubernetes.io/	aws-load-balancer-backend-protocol
service.beta.kubernetes.io/	aws-load-balancer-additional-resource-tags
service.beta.kubernetes.io/	aws-load-balancer-healthcheck-healthy-threshold
service.beta.kubernetes.io/	aws-load-balancer-healthcheck-unhealthy-threshold
service.beta.kubernetes.io/	aws-load-balancer-healthcheck-timeout
service.beta.kubernetes.io/	aws-load-balancer-healthcheck-interval

- Draining
- Logging
- SSL Certs
- Tagging
- Security groups
- Health checks

https://github.com/kubernetes/kubernetes/blob/master/pkg/cloudprovider/providers/aws/aws.go

Currently aws only supports elb classic and nlb in EKS. aws-alb-ingress-controller plugin that enables AWS ALB for Kubernetes
https://github.com/kubernetes-sigs/aws-alb-ingress-controller)

Network Load Balancer (layer 4)

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: "nlb"
spec:
  type: LoadBalancer
  externalTrafficPolicy: Local
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
```

议程

- AWS 容器技术介绍
- Amazon EKS(云上托管的K8S集群)
- Amazon EKS对K8S网络的创新
- K8S Service 与AWS的集成
- **Amazon EKS与AWS安全解决方案整合**
- Amazon EKS日志及监控
- Demo演示

安全

IAM

VPC

PrivateLink

Heptio kubernete RBAC support
Calico network policy

TIGERA

PROJECT CALICO

amazon-vpc-cni-k8s插件不支持NetworkPolicy。 EKS使用Calico实现NetworkPolicy
虽然Calico本身就是CNI，但我们只会使用它的NetworkPolicy功能

Kubernetes网络策略强制执行网络安全规则

Calico是网络策略API的领导者

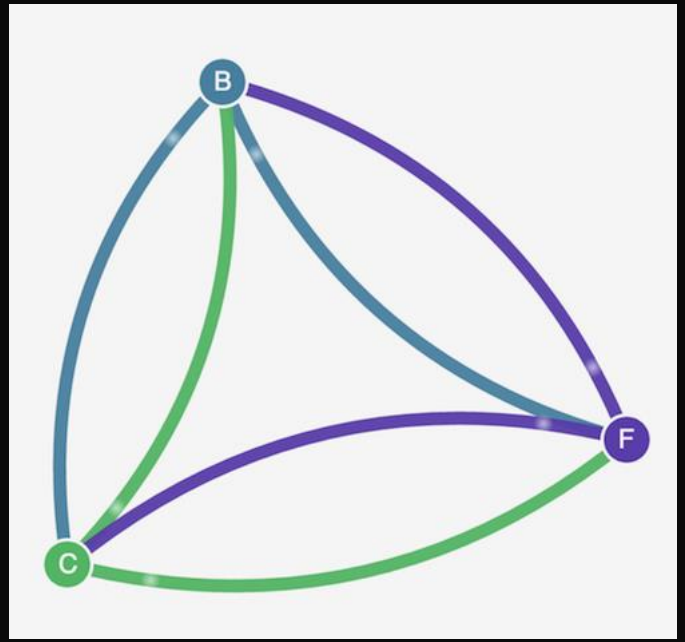
开源，活动的开发者（> 100个贡献者）

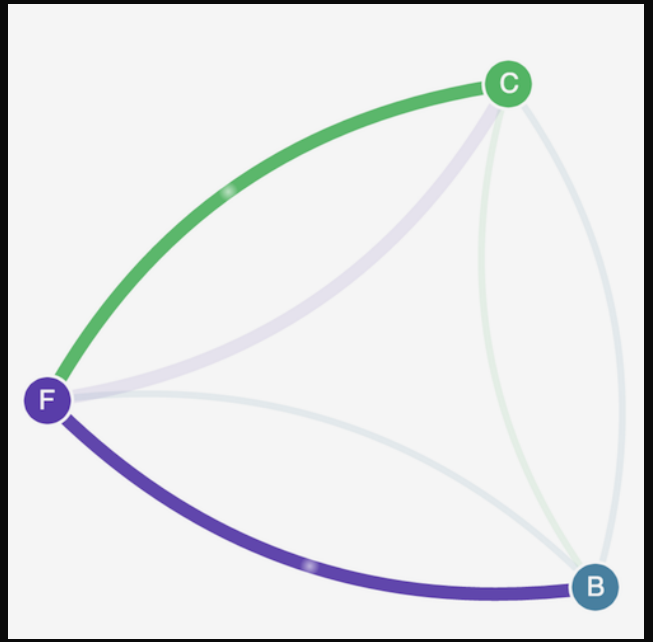
Tigera提供商业支持

AWS中国（北京）区域由光环新网运营
AWS中国（宁夏）区域由西云数据运营

aws | Webinars

Calico网络安全策略





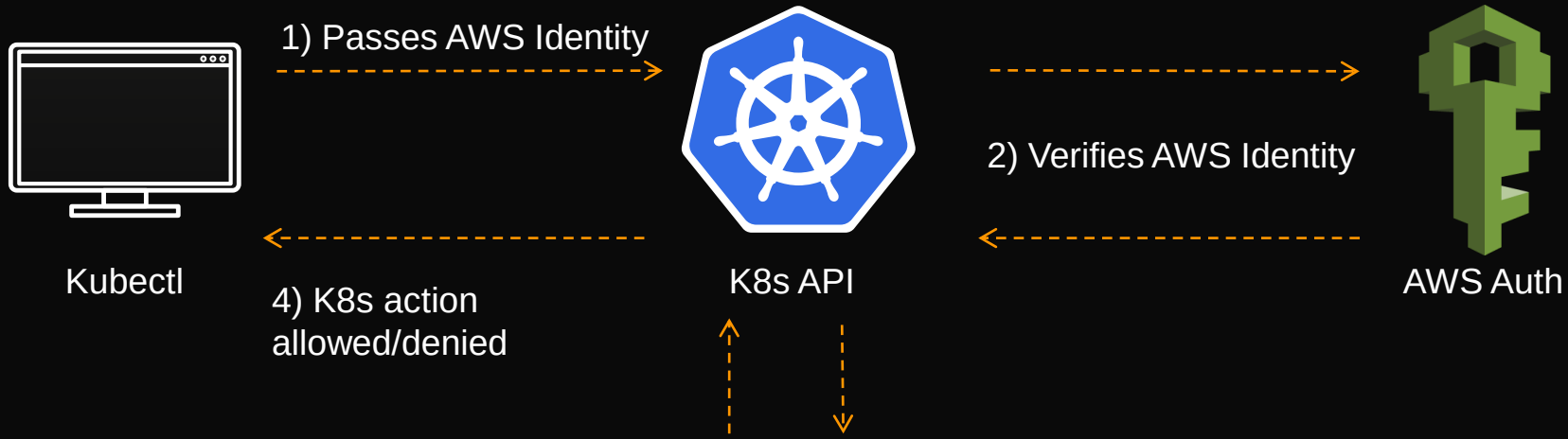
AWS中国（北京）区域由光环新网运营
AWS中国（宁夏）区域由西云数据运营

通过heptio来整合Kubernetes RBAC与IAM

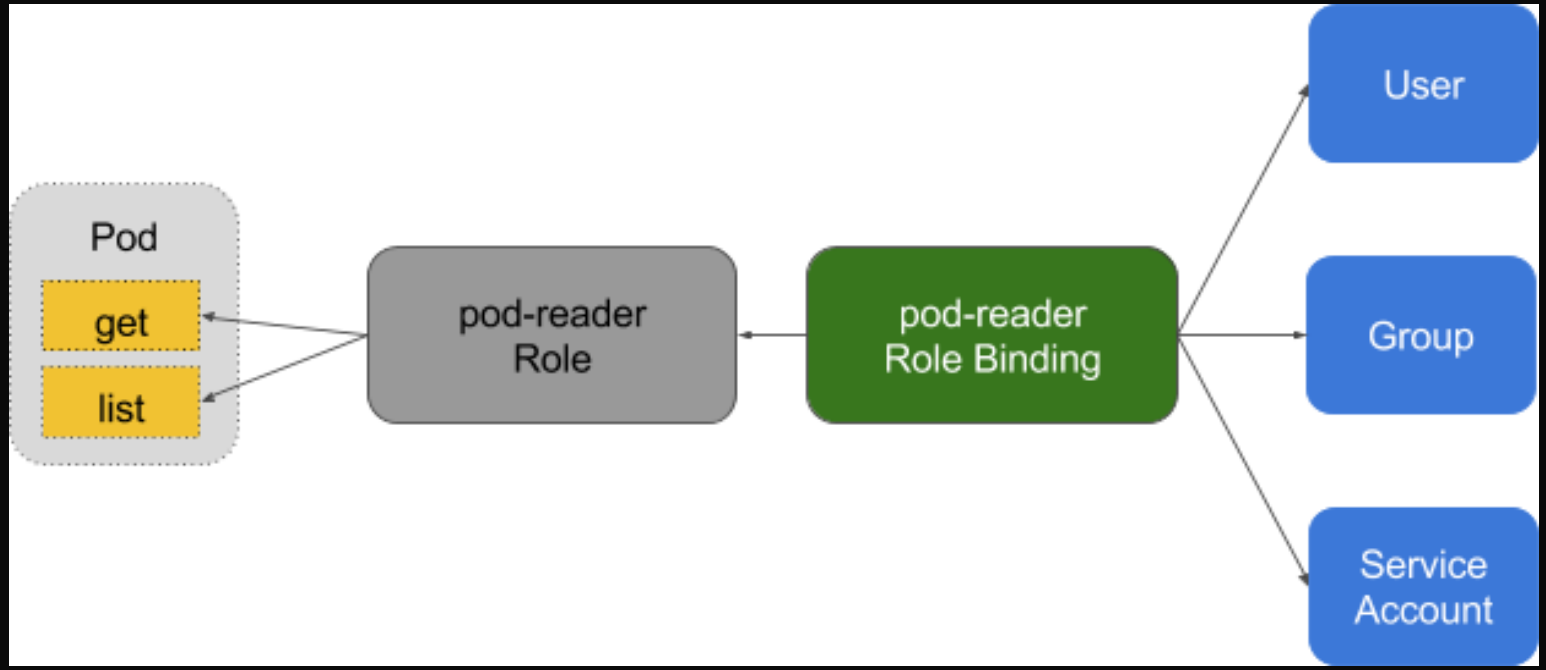
https://github.com/heptiolabs/kubernetes-aws-authenticator

一种开源解决方案，可以将AWS IAM身份验证与Kubernetes集成

Kubectl使用IAM进行身份验证



RBAC


Pod
get
list
pod-reader
Role
pod-reader
Role Binding
User
Group
Service
Account

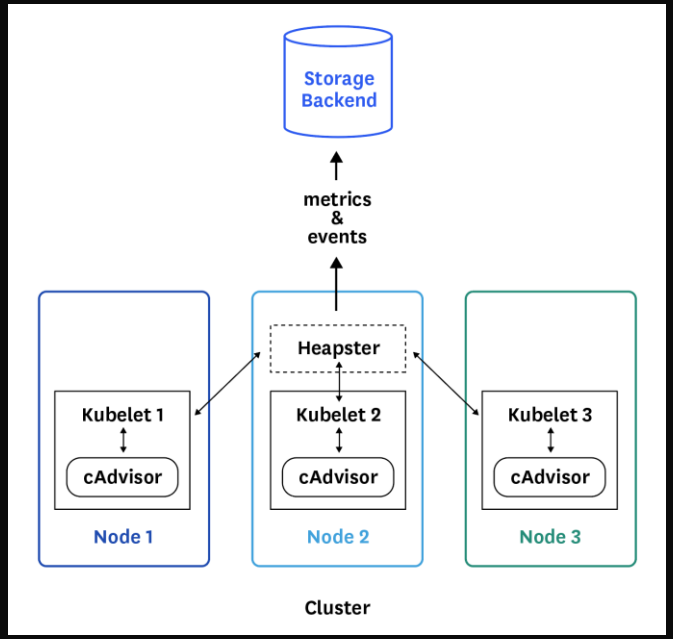
Kubernetes 角色

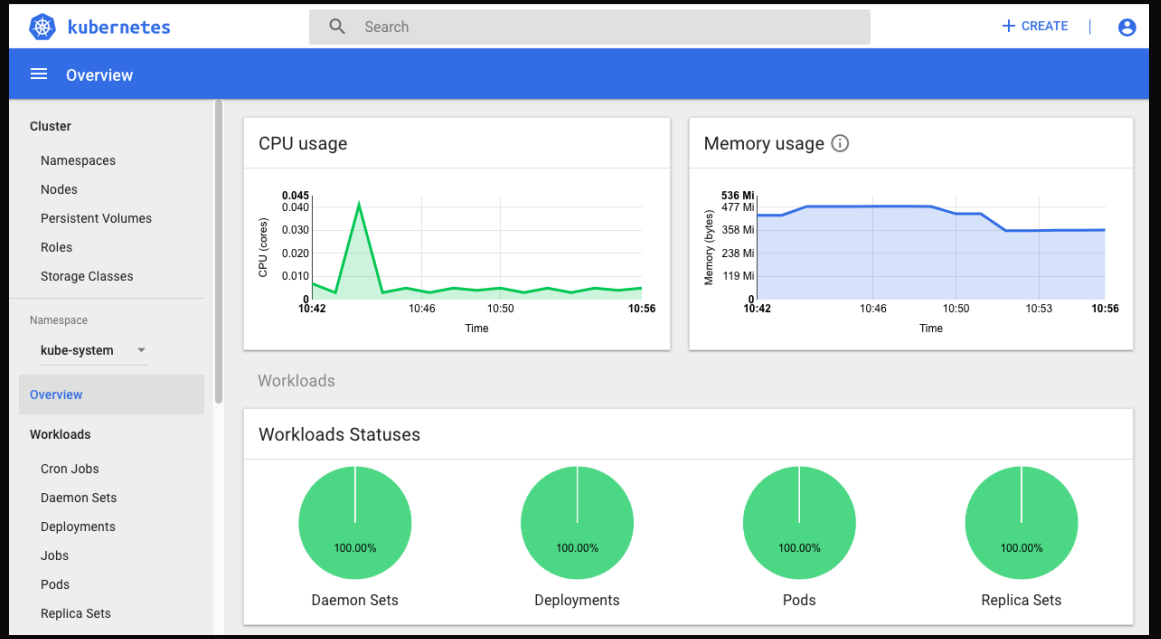
- 用户（在我们的例子中，IAM用户）绑定到Kubernetes角色
- 通常只有一个Admin角色
- 具有有限权限的其他角色（使用RBAC分配）允许用户访问群集中的特定命名空间或其他组件

议程

- AWS 容器技术介绍
- Amazon EKS(云上托管的K8S集群)
- Amazon EKS对K8S网络的创新
- K8S Service 与AWS的集成
- Amazon EKS与AWS安全解决方案整合
- **Amazon EKS日志及监控**
- Demo演示

监控






```
clusterrolebinding.rbac.authorization.k8s.io "heapster" created
[ec2-user@ip-192-168-94-119 ~]$ kubectl get svc  --namespace kube-system
NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)         AGE
heapster               ClusterIP   10.100.76.130   <none>        80/TCP          55m
kube-dns               ClusterIP   10.100.0.10     <none>        53/UDP,53/TCP   11h
kubernetes-dashboard   ClusterIP   10.100.76.225   <none>        443/TCP         58m
monitoring-influxdb    ClusterIP   10.100.190.49   <none>        8086/TCP        54m
```

AWS中国（北京）区域由光环新网运营
AWS中国（宁夏）区域由西云数据运营

日志

CloudWatch Logs

Kube API server, controller, scheduler logs in CloudWatch

/var/log/kube-apiserver.log - API Server, responsible for serving the API
/var/log/kube-scheduler.log - Scheduler, responsible for making scheduling decisions
/var/log/kube-controller-manager.log - Controller that manages replication controllers

CloudTrail

EKS API calls logged to CloudTrail

议程

- AWS 容器技术介绍
- Amazon EKS(云上托管的K8S集群)
- Amazon EKS对K8S网络的创新
- Amazon EKS与AWS安全解决方案整合
- Amazon EKS日志及监控
- **Demo演示**