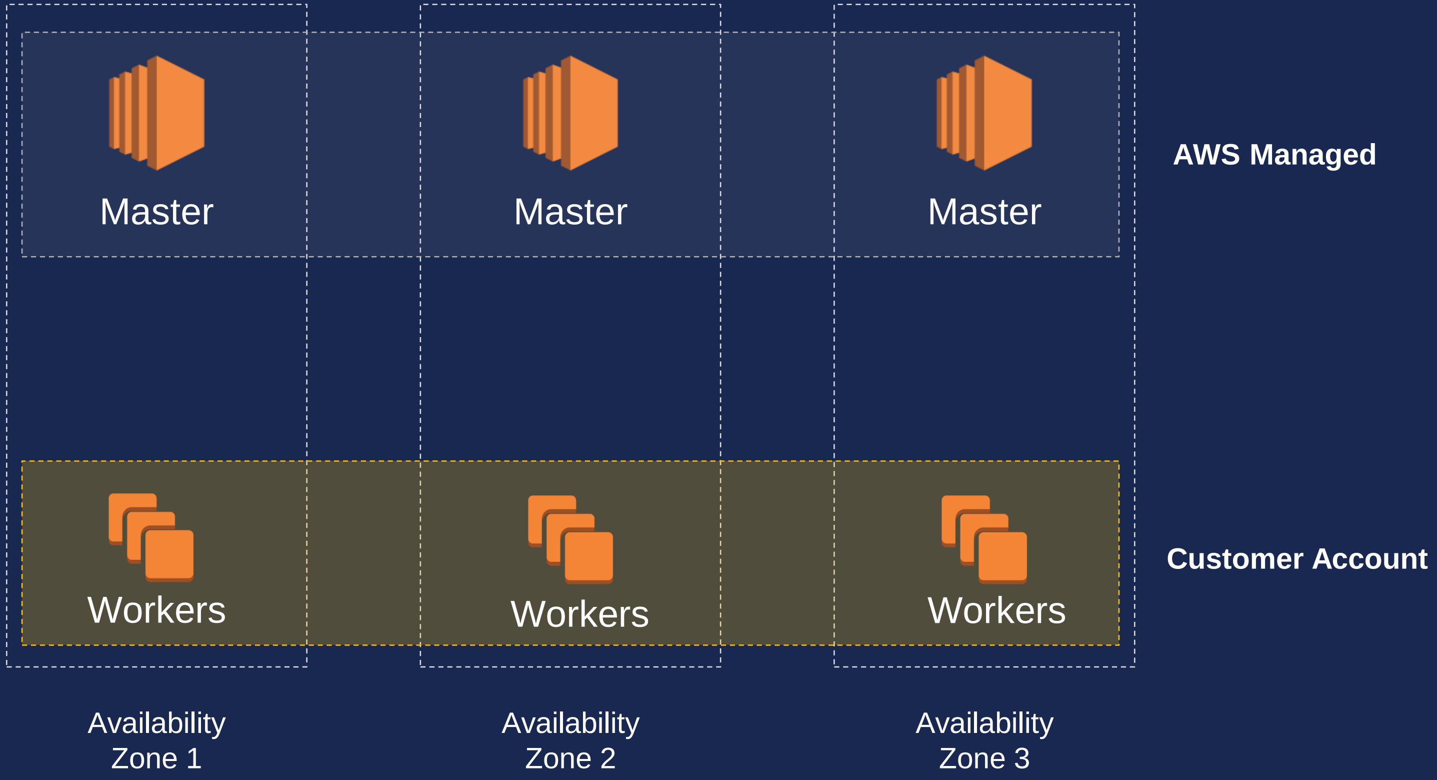
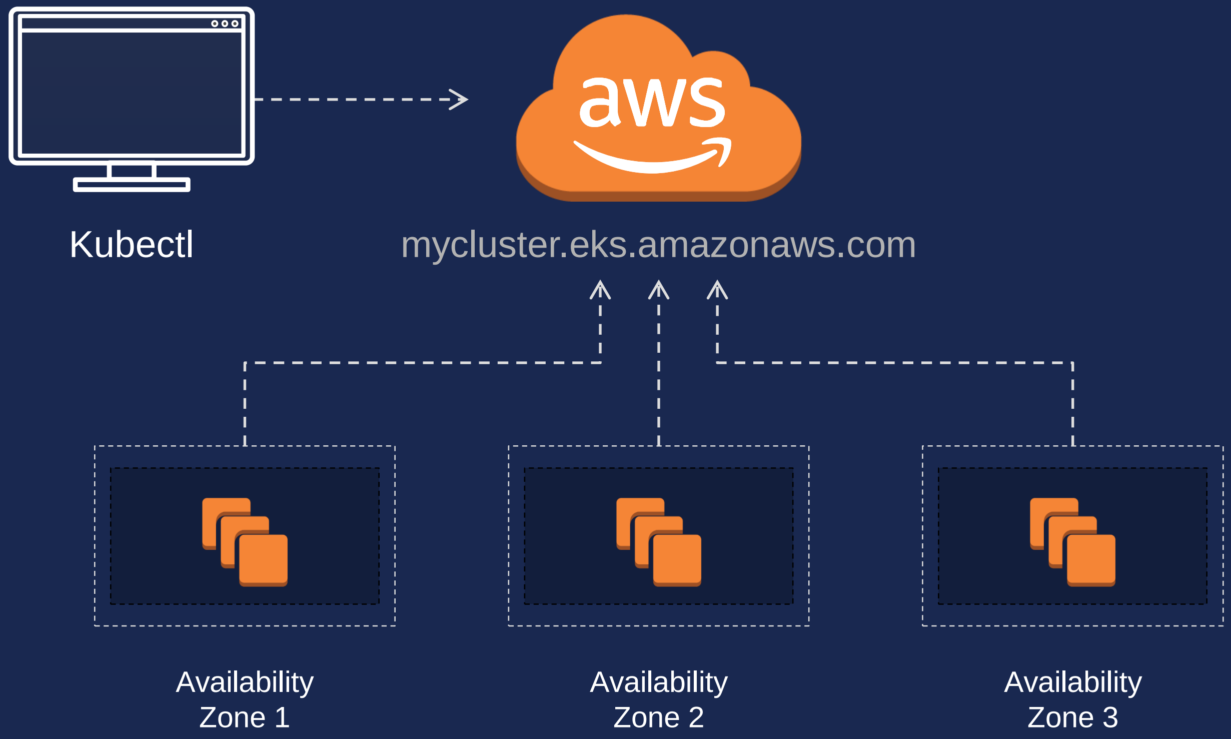
aws SUMMIT

Effective Cloud Native Design with Amazon EKS

Pahud Hsieh(謝洪恩)

Solutions Architect, Amazon Web Services

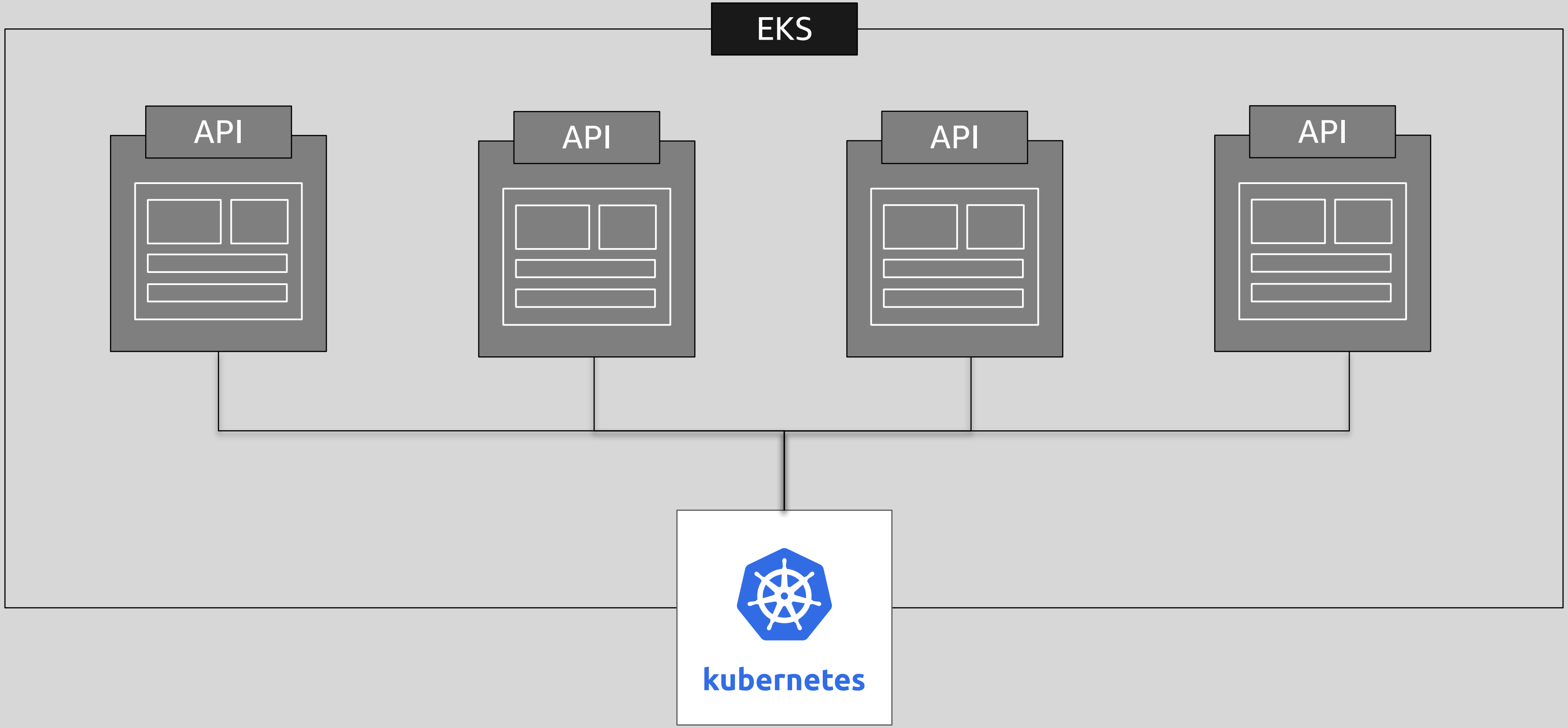

Master
Master
Master
AWS Managed
Workers
Workers
Workers
Customer Account
Availability Zone 1
Availability Zone 2
Availability Zone 3


Kubectl
mycluster.eks.amazonaws.com
Availability Zone 1
Availability Zone 2
Availability Zone 3

EKS is Kubernetes Certified

certified

kubernetes


EKS
API
API
API
API
kubernetes

```
aws eks create-cluster –cluster-name summit2018 –desired-master-version 1.10
–role-arn arn:aws:iam::account-id:role/role-name
```

```
aws eks describe-cluster –cluster-name summit2018
```

Cluster Metadata

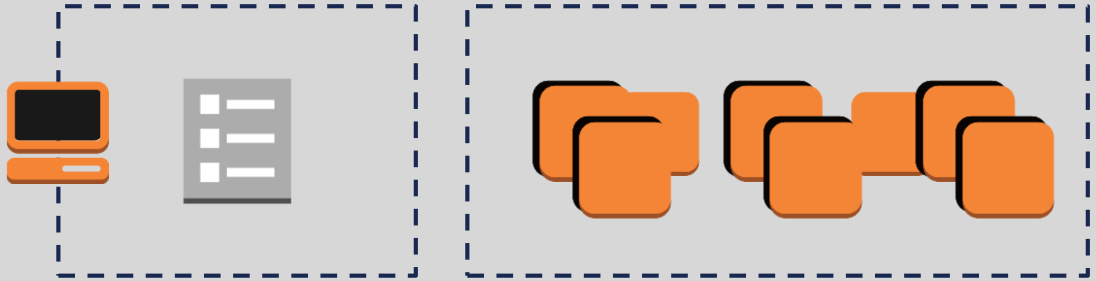
```
HTTP/1.1 200 Content-type:
application/json

{ "cluster":
    {
    "clusterName": "string",
    "createdAt": number,
    "currentMasterVersion": "string",
    "desiredMasterVersion": "string",
    "masterEndpoint": "string",
    "roleArn": "string",
    "status": "string",
    "statusMessage": "string"
    }
}
```

```
aws eks list-clusters
aws eks delete-cluster –cluster-name
summit2018
```

EKS Architecture

EKS Master Autoscaling



CNI(Container Network Interface)

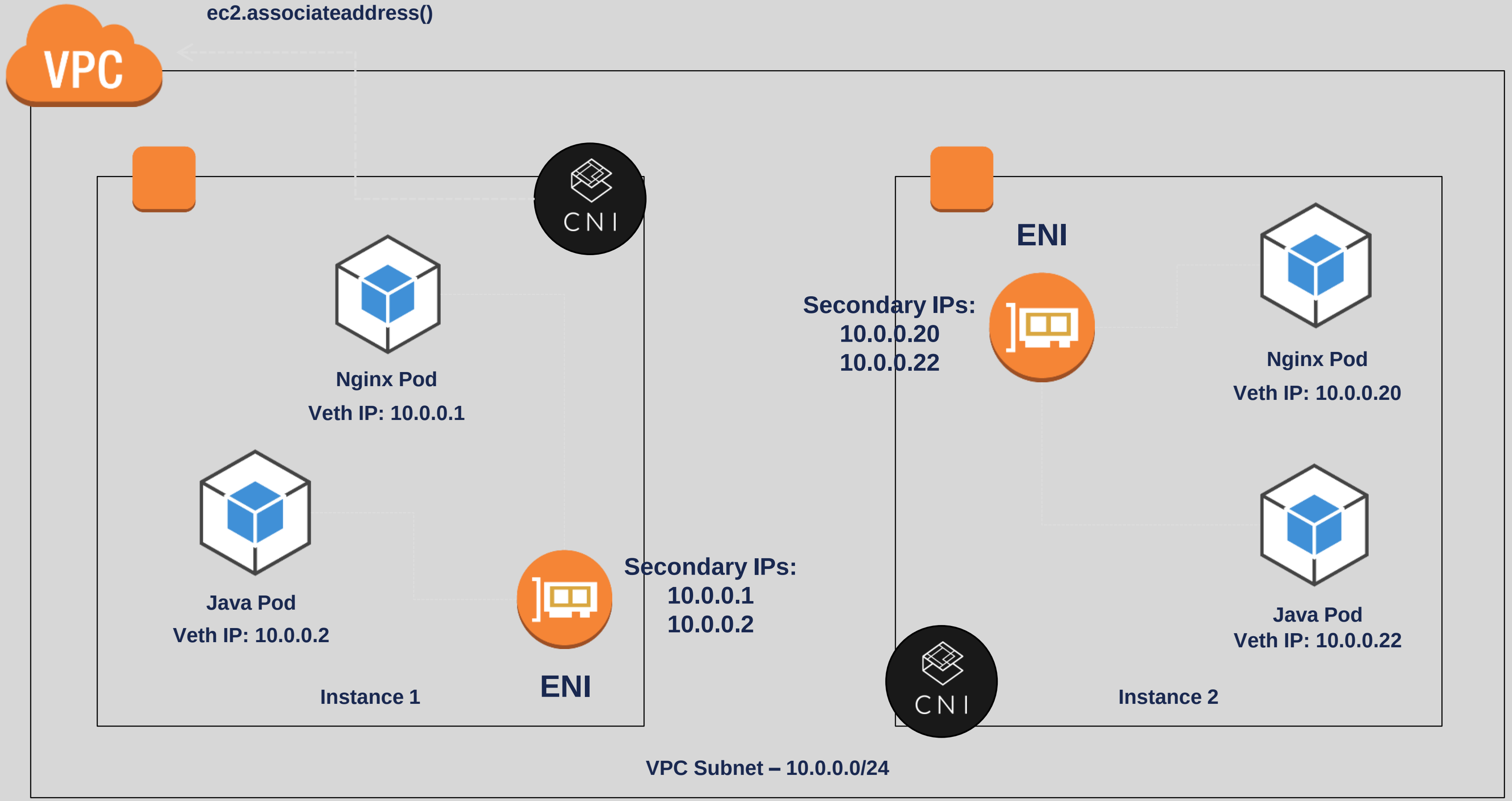
Native VPC networking with CNI plugin

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

Simple, secure networking

Open source and on Github

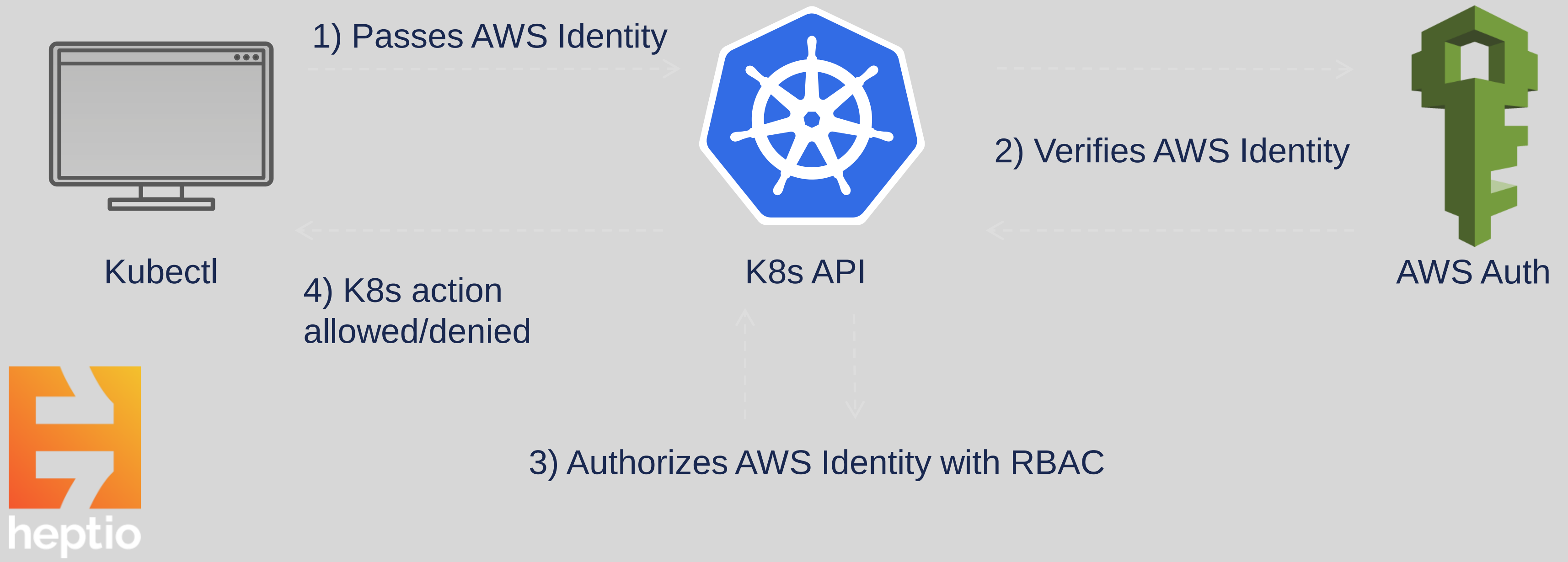
https://github.com/aws/amazon-vpc-cni-k8s


ec2.associateaddress()
VPC
CNI
Nginx Pod
Veth IP: 10.0.0.1
Java Pod
Veth IP: 10.0.0.2
Secondary IPs:
10.0.0.1
10.0.0.2
ENI
Instance 1
ENI
Secondary IPs:
10.0.0.20
10.0.0.22
Nginx Pod
Veth IP: 10.0.0.20
Java Pod
Veth IP: 10.0.0.22
CNI
Instance 2
VPC Subnet – 10.0.0.0/24

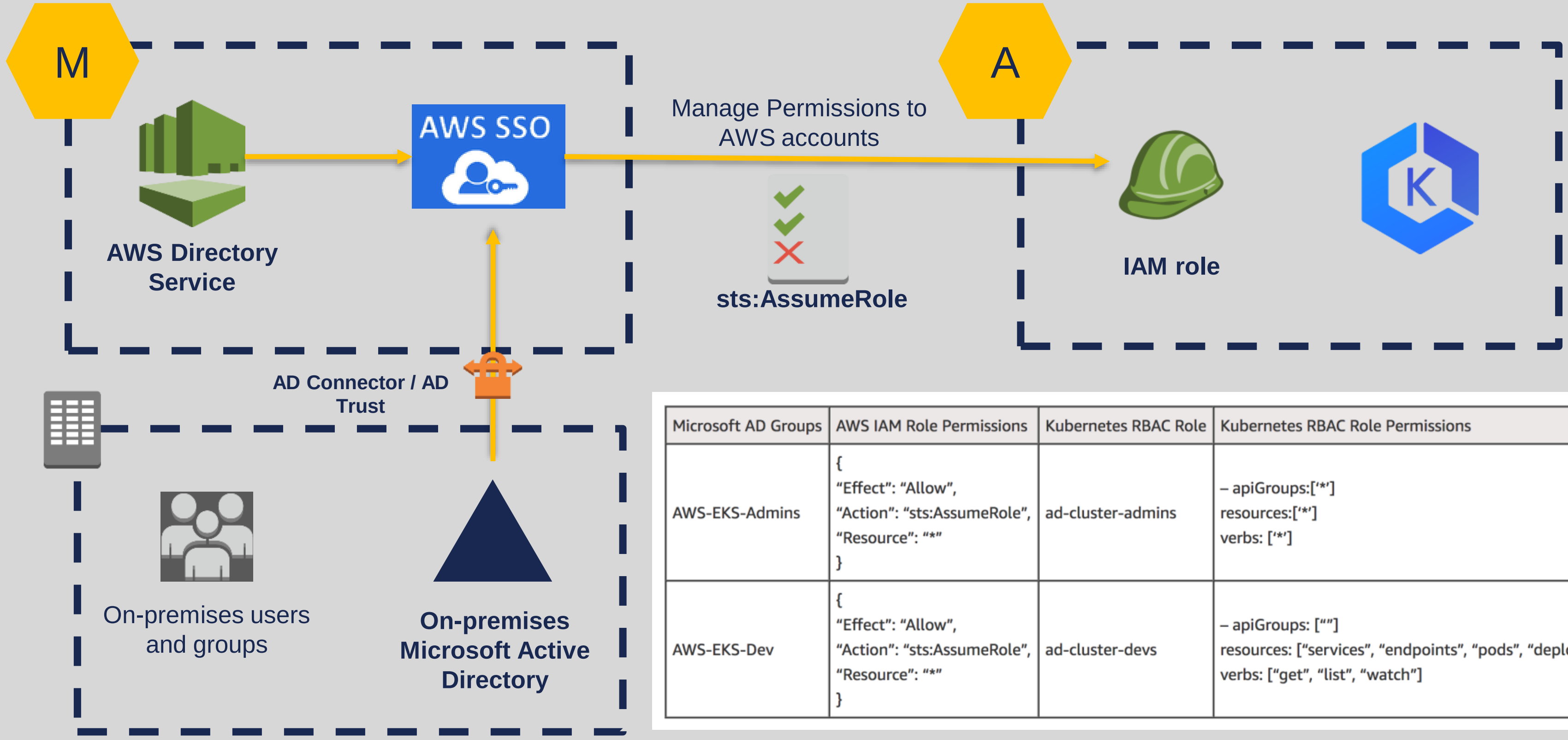
IAM Integration

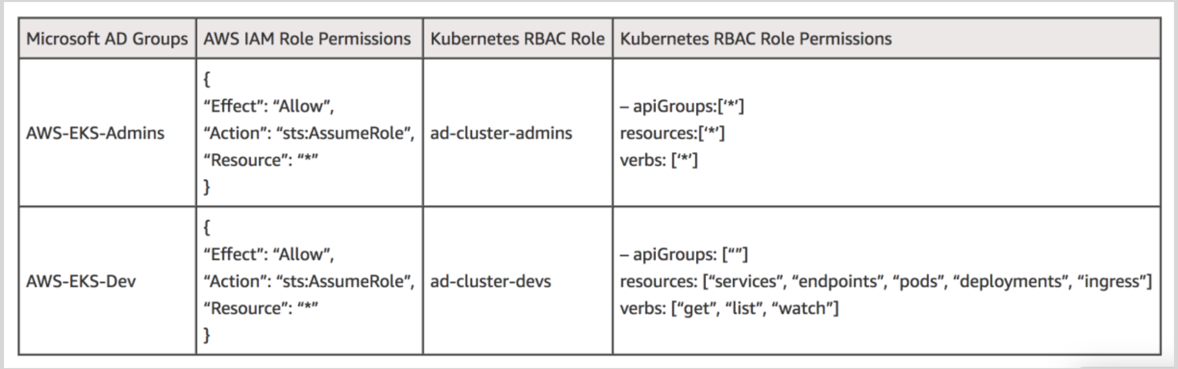
IAM authentication with Kubernetes

IAM + Kubectl


1) Passes AWS Identity
2) Verifies AWS Identity
3) Authorizes AWS Identity with RBAC
4) K8s action allowed/denied
Kubectl
K8s API
AWS Auth
heptio

SAML 2.0 – Integrate with AD and SSO


M
AWS Directory Service
AWS SSO
Manage Permissions to AWS accounts
sts:AssumeRole
A
IAM role
K
AD Connector / AD Trust
On-premises users and groups
On-premises Microsoft Active Directory



Microsoft AD Groups	AWS IAM Role Permissions	Kubernetes RBAC Role	Kubernetes RBAC Role Permissions
AWS-EKS-Admins	{ “Effect”: “Allow”, “Action”: “sts:AssumeRole”, “Resource”: “*” }	ad-cluster-admins	– apiGroups:[‘*’] resources:[‘*’] verbs: [‘*’]
AWS-EKS-Dev	{ “Effect”: “Allow”, “Action”: “sts:AssumeRole”, “Resource”: “*” }	ad-cluster-devs	– apiGroups: [“”] resources: [“services”, “endpoints”, “pods”, “deployments”, “ingress”] verbs: [“get”, “list”, “watch”]

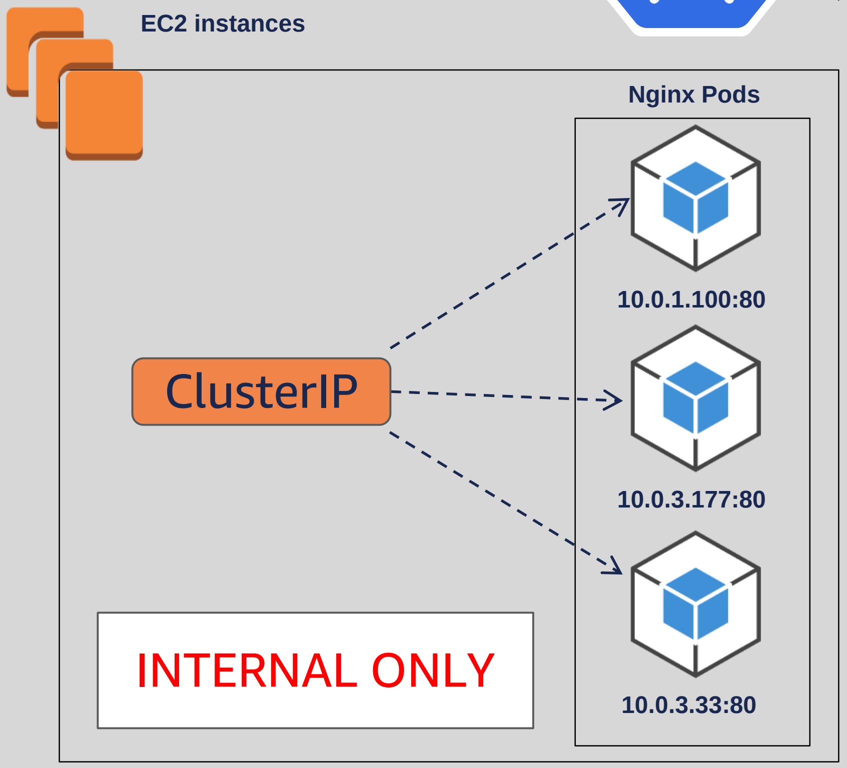
https://aws.amazon.com/tw/blogs/opensource/integrating-ldap-ad-users-kubernetes-rbac-aws-iam-authenticator-project

Service Types and Ingress

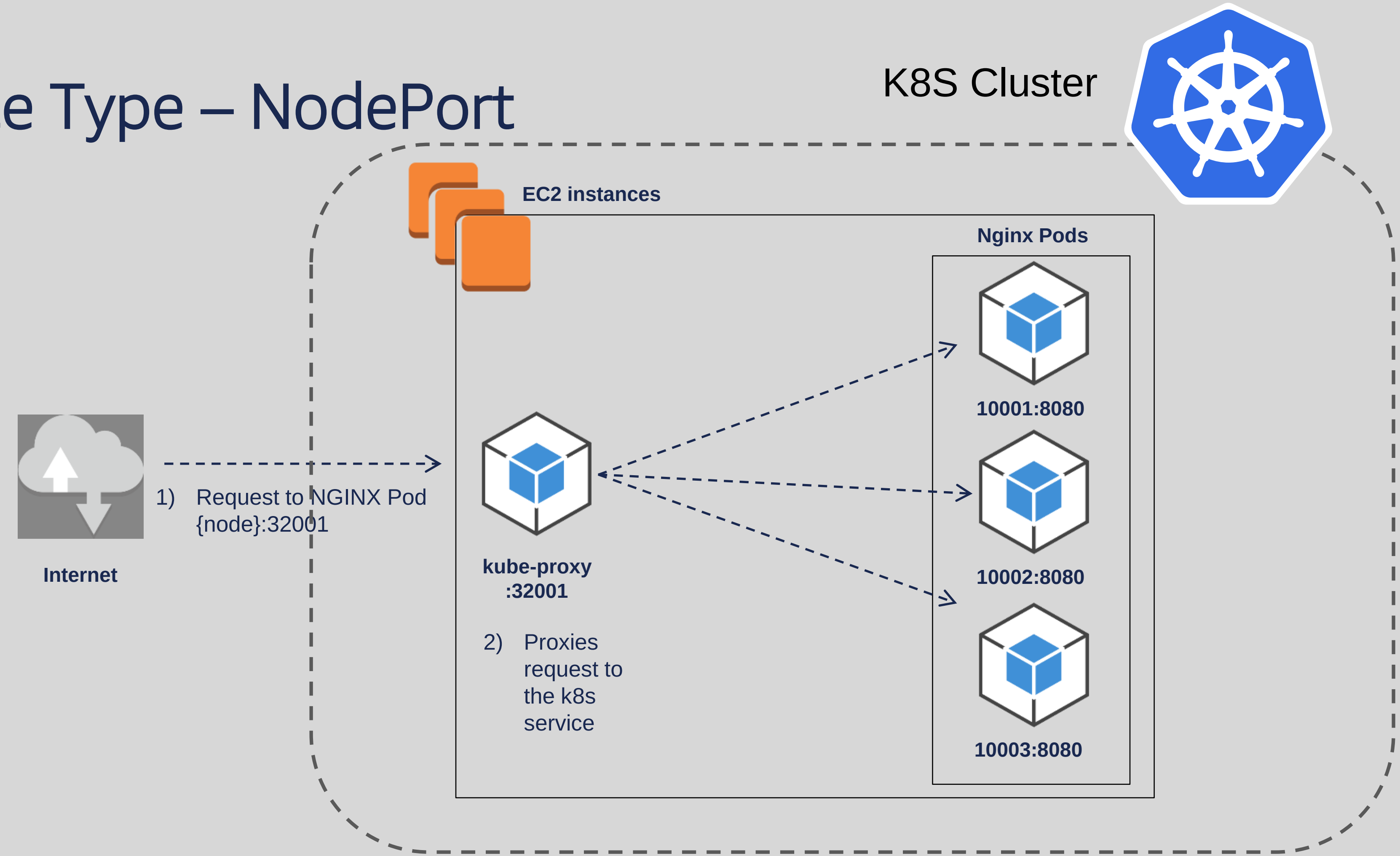
Service Type – ClusterIP(virtual)

K8S Cluster

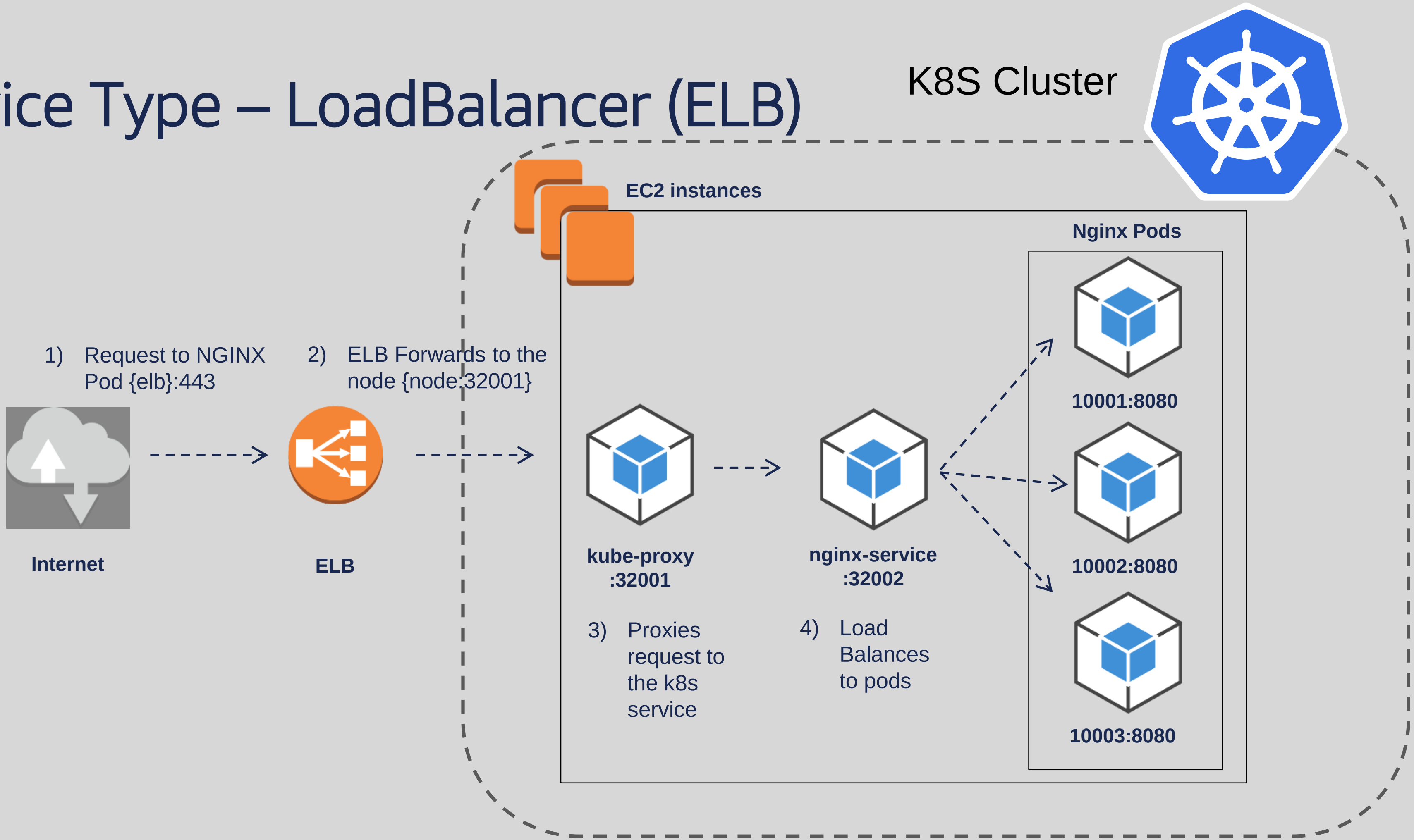
```
-A KUBE-SEP-JNHR7XFBS7L5NBRR -p tcp -m comment --comment
"default/nginx-service:web" -m tcp -j DNAT --to-destination 10.0.3.33:80
-A KUBE-SEP-MJGBAZNA2WGVIMWN -s 10.0.3.177/32 -m comment --
comment "default/nginx-service:web" -j KUBE-MARK-MASQ
-A KUBE-SEP-MJGBAZNA2WGVIMWN -p tcp -m comment --comment
"default/nginx-service:web" -m tcp -j DNAT --to-destination 10.0.3.177:80
-A KUBE-SEP-XZ3DUOZYSFB3ILKR -s 10.0.1.100/32 -m comment --
comment "default/nginx-service:web" -j KUBE-MARK-MASQ
-A KUBE-SEP-XZ3DUOZYSFB3ILKR -p tcp -m comment --comment
"default/nginx-service:web" -m tcp -j DNAT --to-destination 10.0.1.100:80
-A KUBE-SERVICES -d 172.20.169.0/32 -p tcp -m comment --comment
"default/nginx-service:web cluster IP" -m tcp --dport 80 -j KUBE-SVC-
MCOVNBHDEGIKKKLL
-A KUBE-SVC-MCOVNBHDEGIKKKLL -m comment --comment "default/nginx-
service:web" -m statistic --mode random --probability 0.33332999982 -j KUBE-
SEP-XZ3DUOZYSFB3ILKR
-A KUBE-SVC-MCOVNBHDEGIKKKLL -m comment --comment "default/nginx-
service:web" -m statistic --mode random --probability 0.50000000000 -j KUBE-
SEP-MJGBAZNA2WGVIMWN
-A KUBE-SVC-MCOVNBHDEGIKKKLL -m comment --comment "default/nginx-
service:web" -j KUBE-SEP-JNHR7XFBS7L5NBRR
```



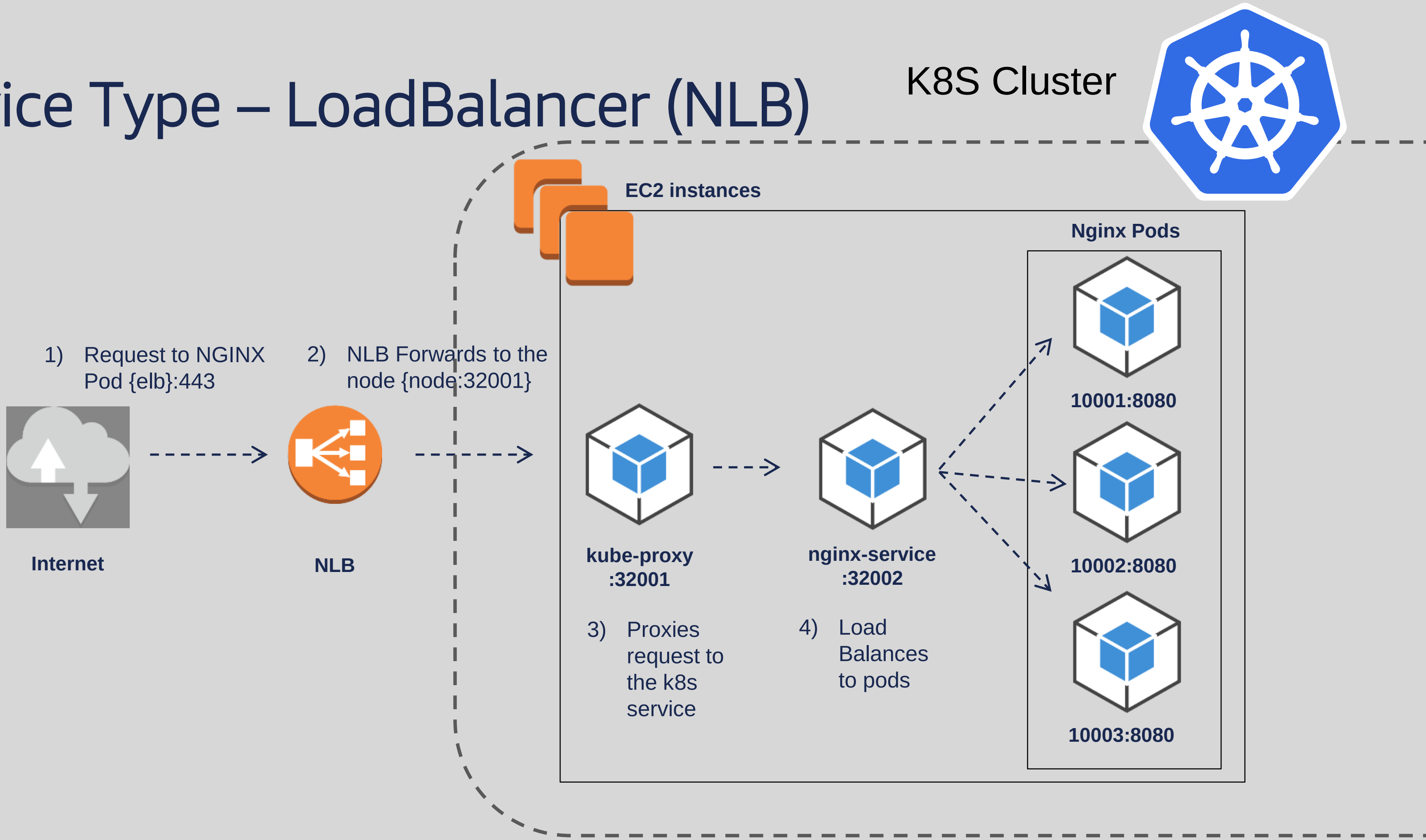
Service Type – NodePort



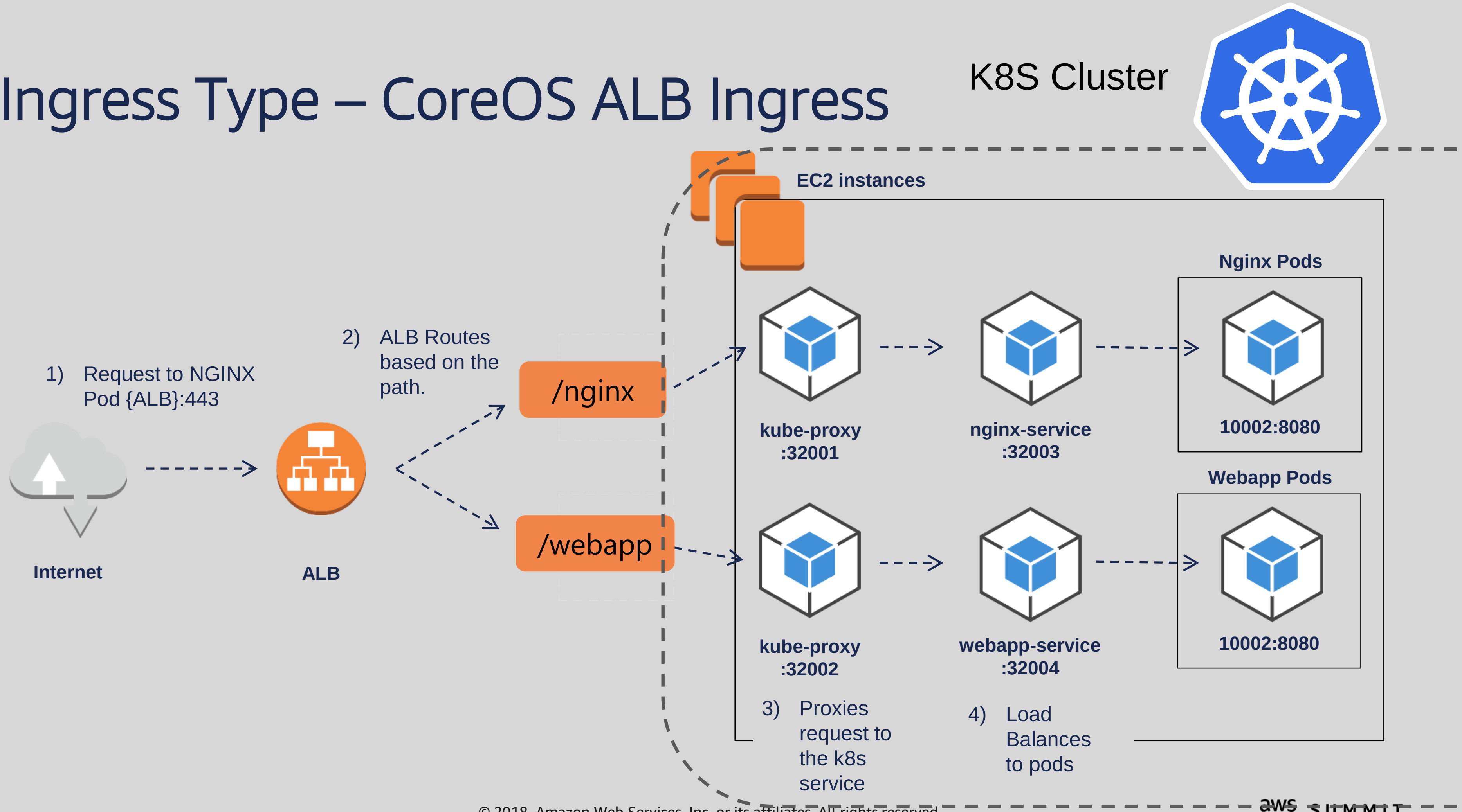
Service Type – LoadBalancer (ELB)



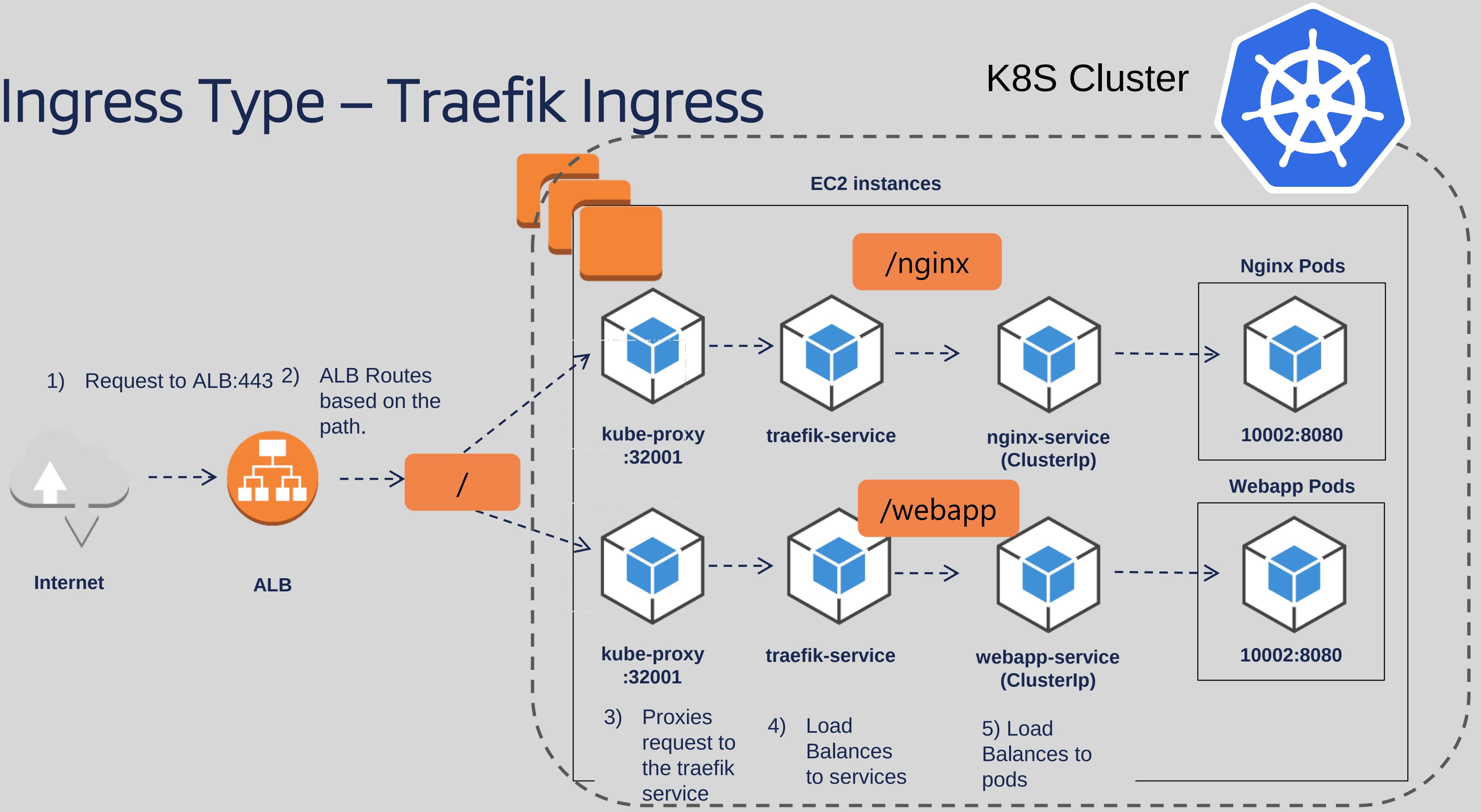
Service Type – LoadBalancer (NLB)



Ingress Type – CoreOS ALB Ingress



Ingress Type – Traefik Ingress



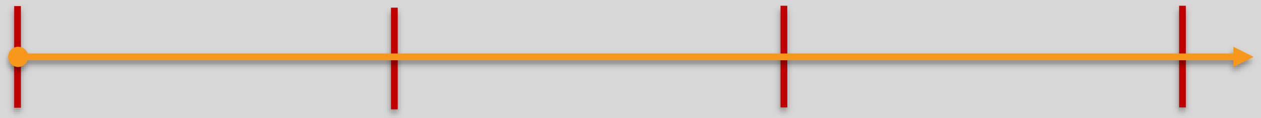
https://docs.traefik.io/user-guide/kubernetes/

Storage

Storage

- Persistent Volume
- Persistent Volume Claims
- StatefulSets
- Storage classes

Lifecycle of the storage volume



Provisioning

- Static
- Dynamic*

Binding

- Control loop watches for PVC requests and satisfies if PV is available.
- For Dynamic, PVC will provision PV
- PVC to PV binding is one-to-one mapping

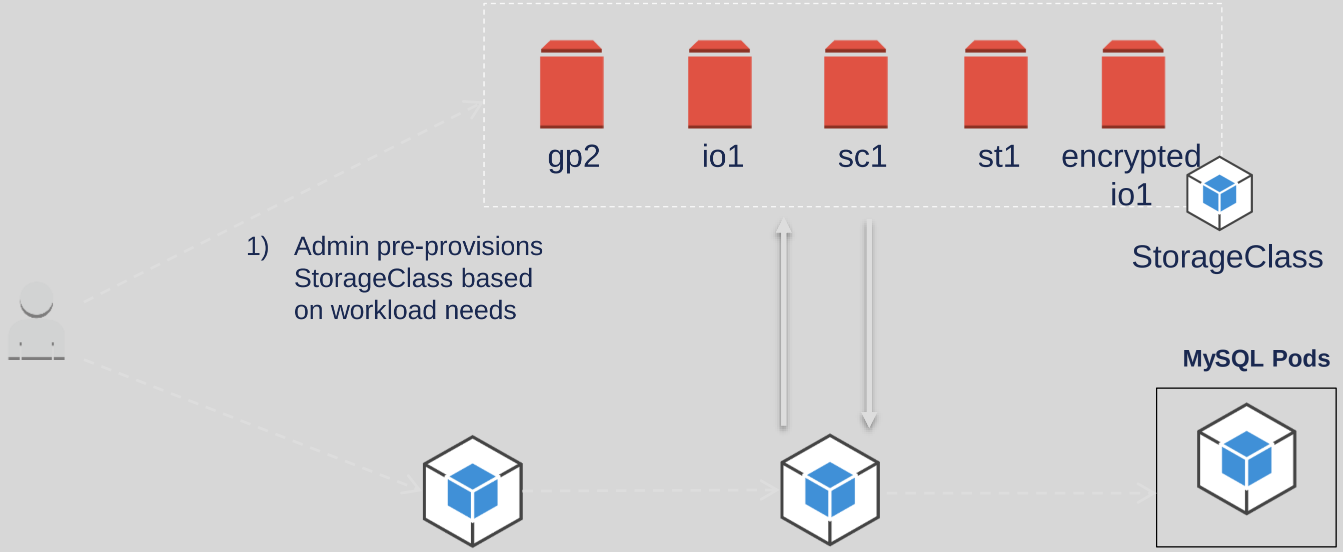
Using

- Cluster mounts volume based on PVC

Reclaiming

- Retain (default)
- Recycle
- Delete

If we need specific volume type?



1) Admin pre-provisions StorageClass based on workload needs

2) End user requests for specific volume types (For ex, encrypted io1 volume)

3) Control loop watches PVC request and allocates volume if PV exists

4) End user creates stateful workload

Scheduling

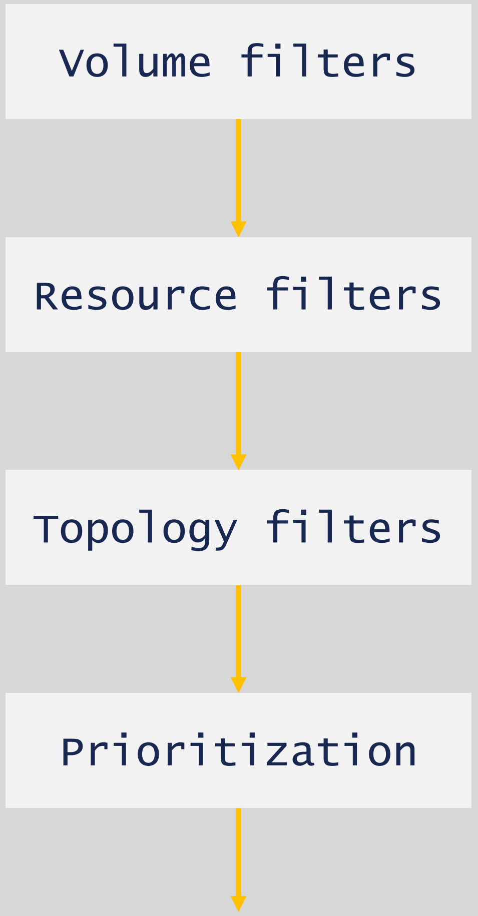
Scheduling Control

Resource requirements

Constraints

• Taints	Node-level
• Tolerations	Pod-level

Affinity/Anti-Affinity



Taints and Tolerations

```
# Taint node
$ kubectl taint nodes ip-10-0-32-12.us-west-2.compute.internal \
    skynet=false:NoSchedule

# Tolerations
kind: Pod
spec:
  tolerations:
  - key: skynet
    operator: Equal
    value: “false”
    effect: NoSchedule

[...]
```

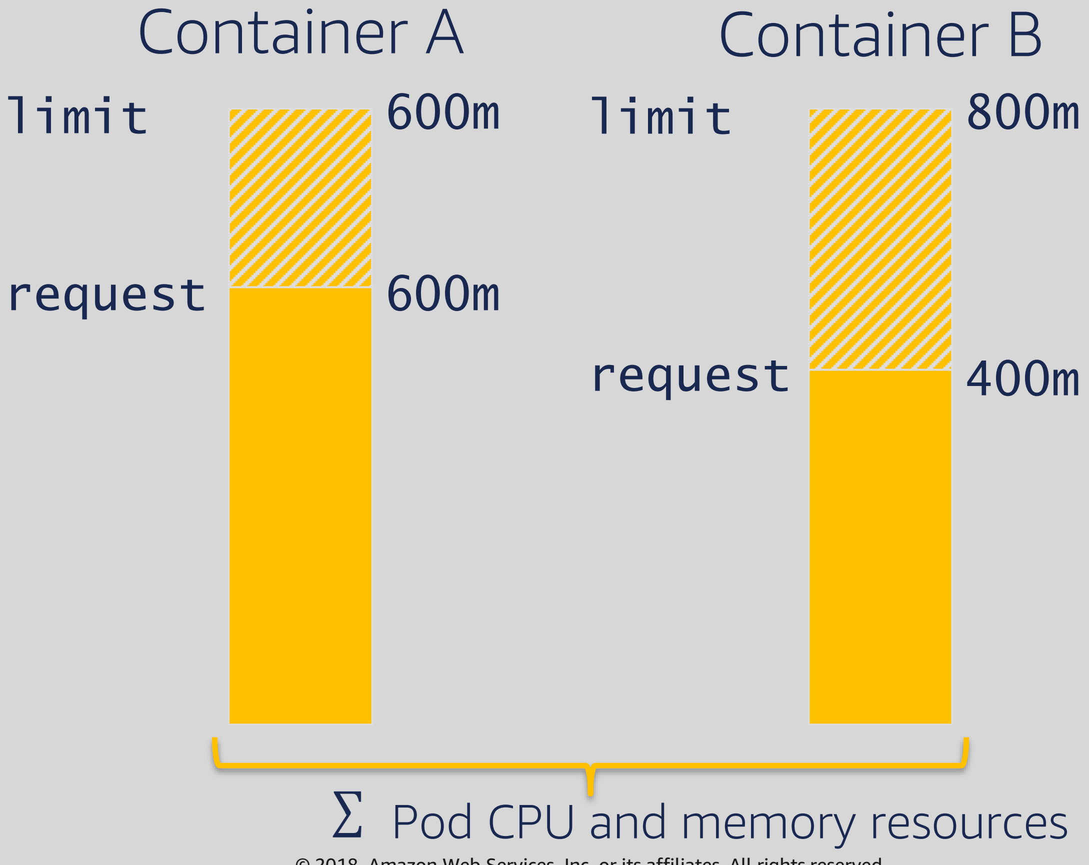
$\}$ Match taint to schedule onto tainted node

Affinity / Anti-Affinity

- Control scheduling onto nodes
  - Combine with Taints & Tolerations
- Distribute Pods across cluster

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
          - key: "beta.kubernetes.io/instance-type"
            operator: In
            values: [“r4.large",“r4.xlarge"]
```

Restrict Resource Usage



aws SUMMIT

Resource Quota

Applied per Namespace

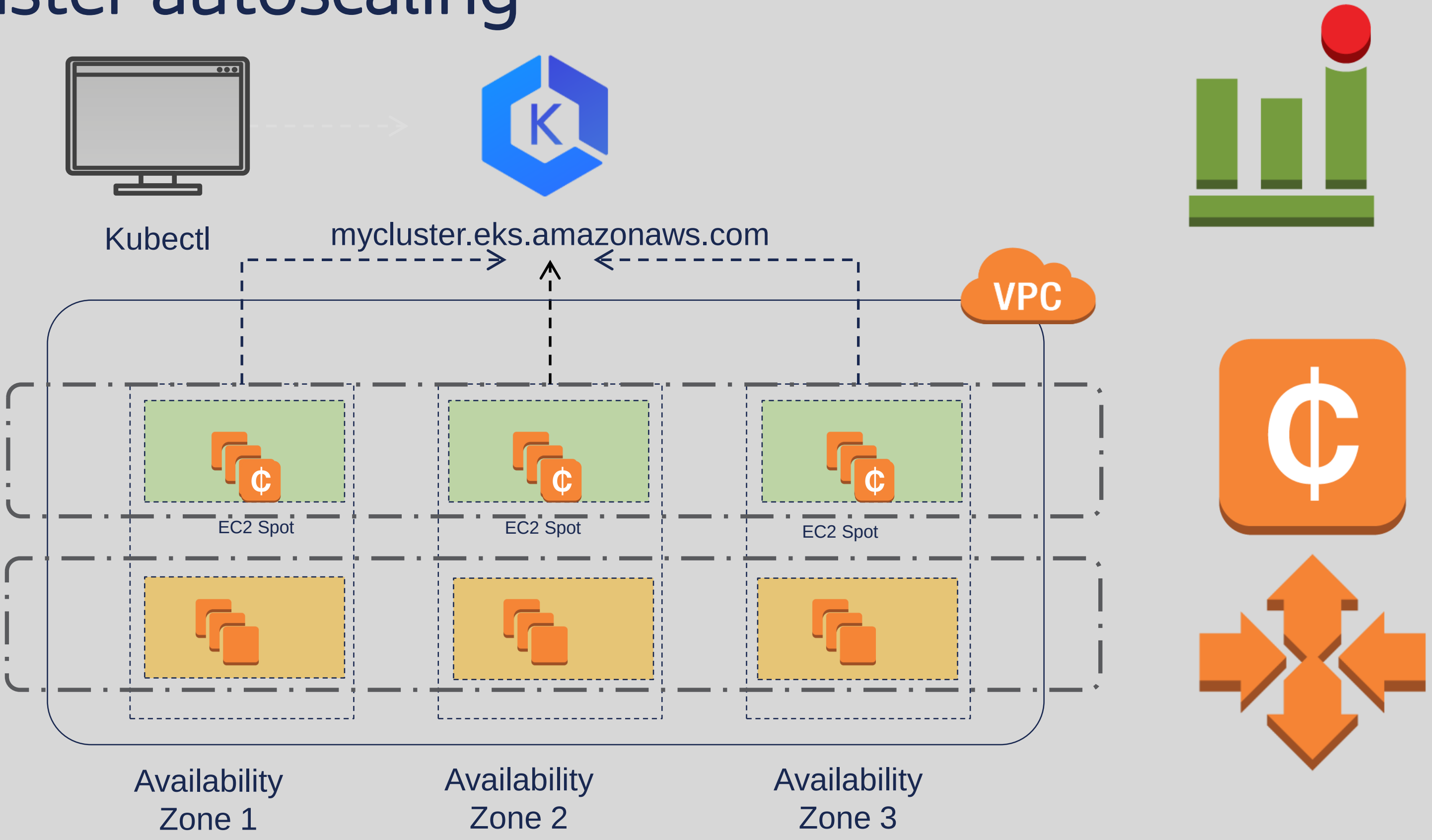
```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: production
spec:
  hard:
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
```

Pod Resource Request

```
apiVersion: v1
kind: Pod
metadata:
  name: production
spec:
  containers:
  - name: nginx-pod
    image: nginx
    resources:
      limits:
        memory: "800Mi"
        cpu: "800m" # 0.8 vCPU
      requests:
        memory: "600Mi"
        cpu: "400m“ # 0.4 vCPU
```

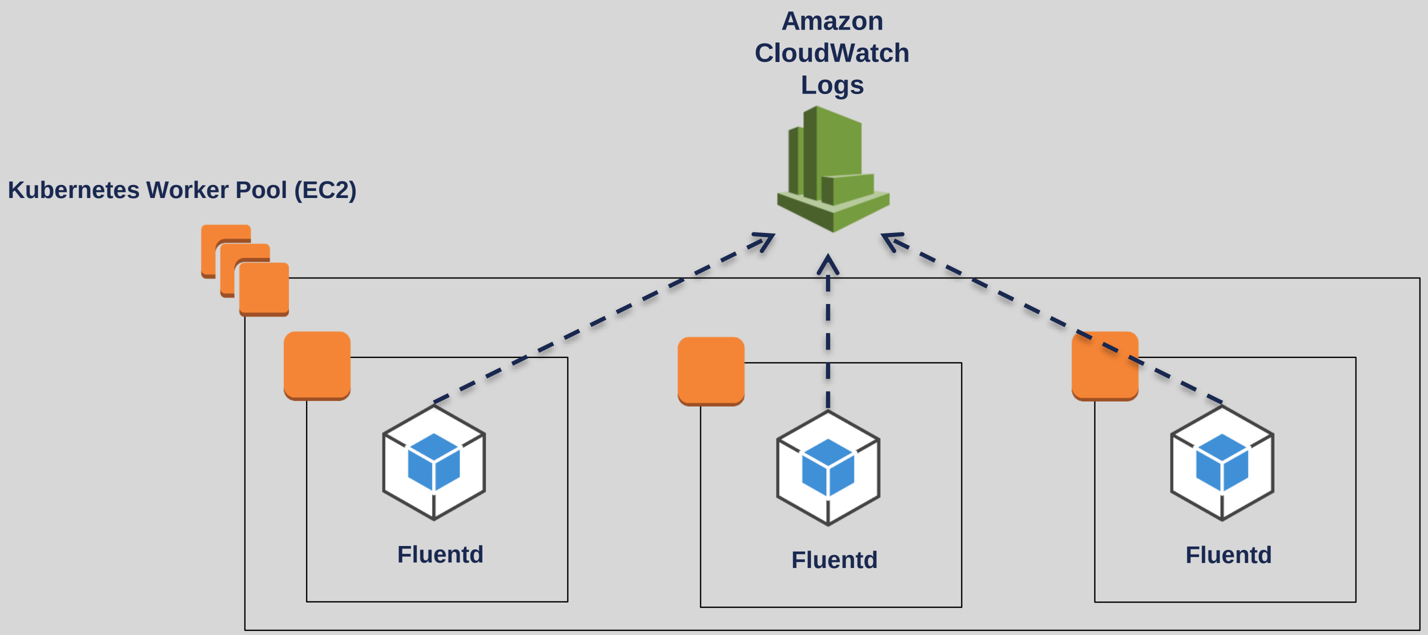
ResourceQuota defined both, so Pod must define both

Cluster autoscaling



Logging

Log aggregation in Cloudwatch Logs via Fluentd



Fluentd Daemonset
Ensures a pod with a Fluentd container on each node in the worker pool with the host's /var/lib/docker/containers mounted so that it can package and ship container logs to CWLogs.

Provisioning

CloudFormation

Terraform

- Popular cloud provisioning provider
- Amazon EKS support on 0-Day
- Can provision multiple node groups
- Can provision spot fleet

https://www.terraform.io/docs/providers/aws/guides/eks-getting-started.html

Vishwakarma by AMIS

https://github.com/getamis/vishwakarma

```
$ git clone https://github.com/getamis/vishwakarma.git
$ cd examples/eks_worker
$ terraform init
$ terraform plan
$ terraform apply //create cluster, autoscaling group and spot fleet

data.ignition_systemd_unit.locksmithd: Refreshing state...
data.template_file.aws_auth_cm: Refreshing state...
data.template_file.max_user_watches: Refreshing state... ... Apply complete!
Resources: 74 added, 0 changed, 0 destroyed.
```

aws SUMMIT

GO BUILD

aws SUMMIT

Thank You!