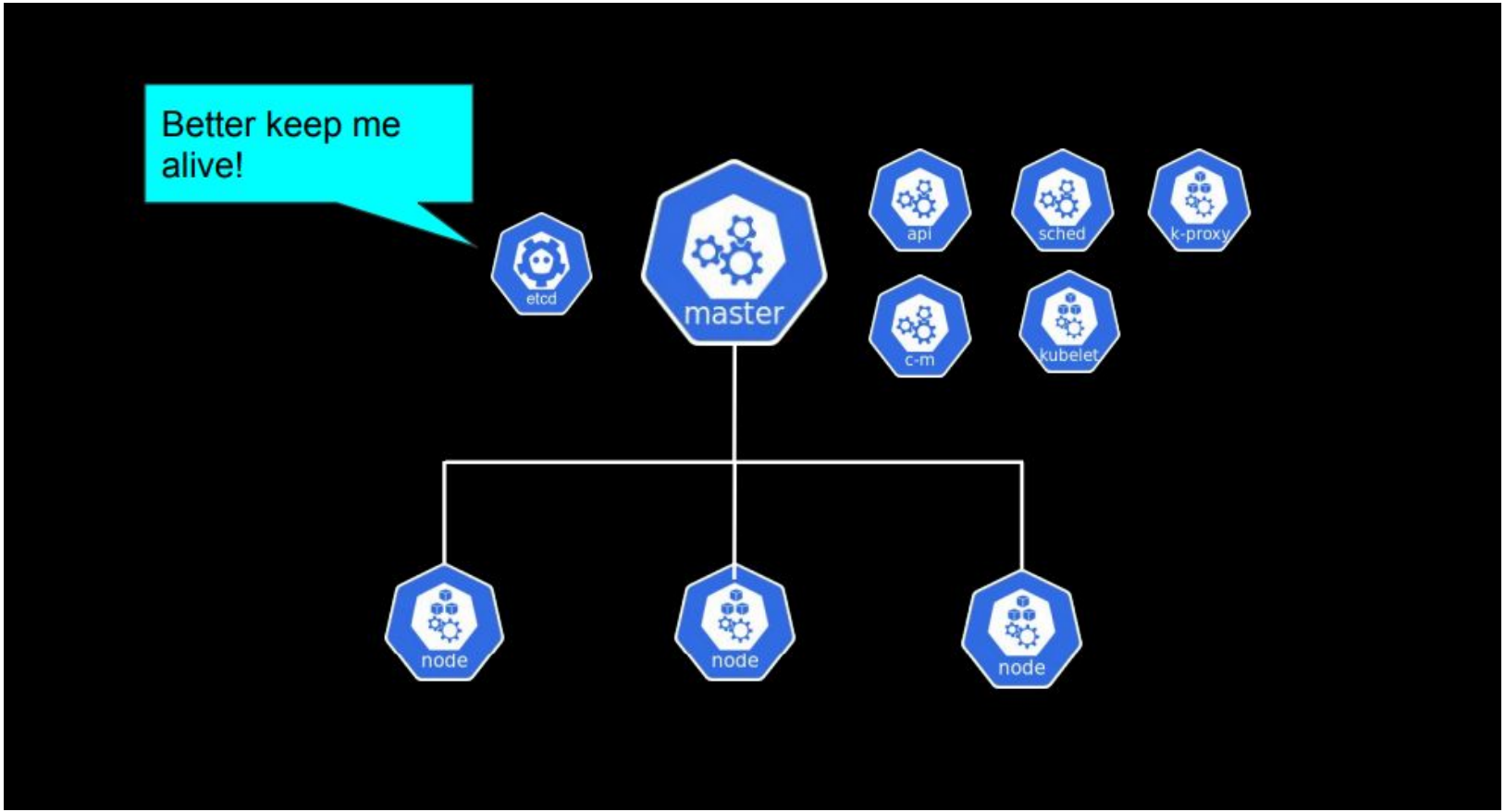
Introduction to Amazon EKS

Outline

- Kubernetes Basics
- EKS Basics
- Logging And Monitoring
- EKS Advanced Concepts
- Securing EKS
- Fargate
- Deploying EKS with DevOps
- Real World EKS Projects

What is EKS?

Kubernetes Architecture



Kubernetes Control Plane - Self Managed

Amazon EC2

- Need to make Control Plane Highly Available
  - Maintain multiple EC2 in multiple AZ
- Scale Control Plane if needed
- Keep etcd up and running
- Overhead of managing EC2s
  - AMI Rehydration
  - Security Patching
  - Replace failed EC2s
  - Orchestration for Kubernetes Version Upgrade

Kubernetes Control Plane - AWS Managed

AWS Manages Kubernetes Control Plane

- AWS maintains High Availability - Multiple EC2s in Multiple AZs
- AWS Detects and Replaces Unhealthy Control Plane Instances
- AWS Scales Control Plane
- AWS Maintain etcd
- Provides Automated Version Upgrade and Patching
- Supports Native and Upstream Kubernetes
- Integrated with AWS Ecosystem

EKS Data Plane

Amazon EC2

Self Managed Node Groups

- You maintain worker EC2s
- You orchestrate version upgrade, security patching, AMI Rehydration, keeping pods up during upgrade
- Can use custom AMI

Amazon EC2

Managed Node Groups

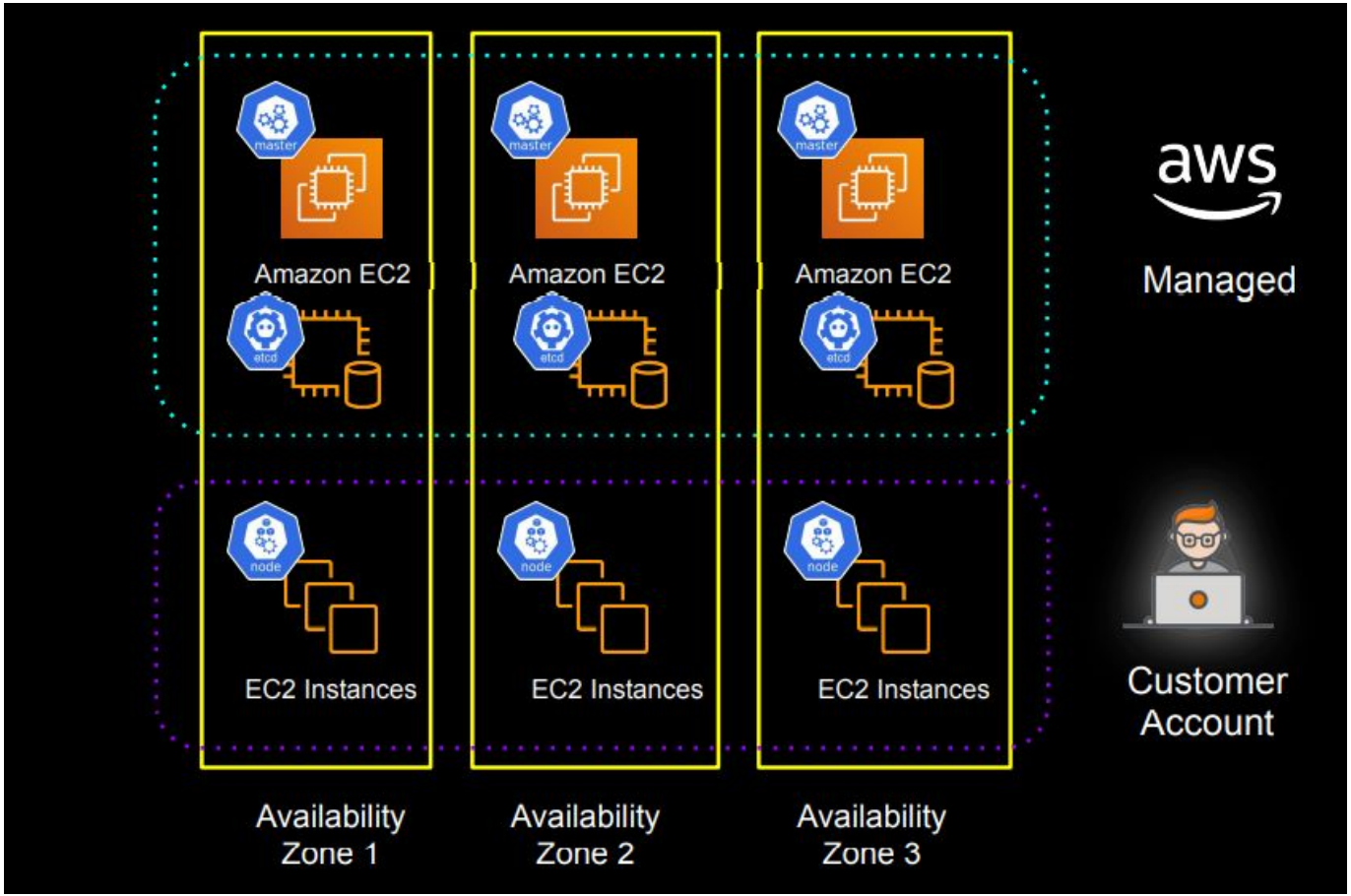
- AWS manages worker EC2s
- AWS provides AMI with security patches, version upgrade
- AWS manages pod disruption during upgrade
- Doesn't work with custom AMI

AWS Fargate

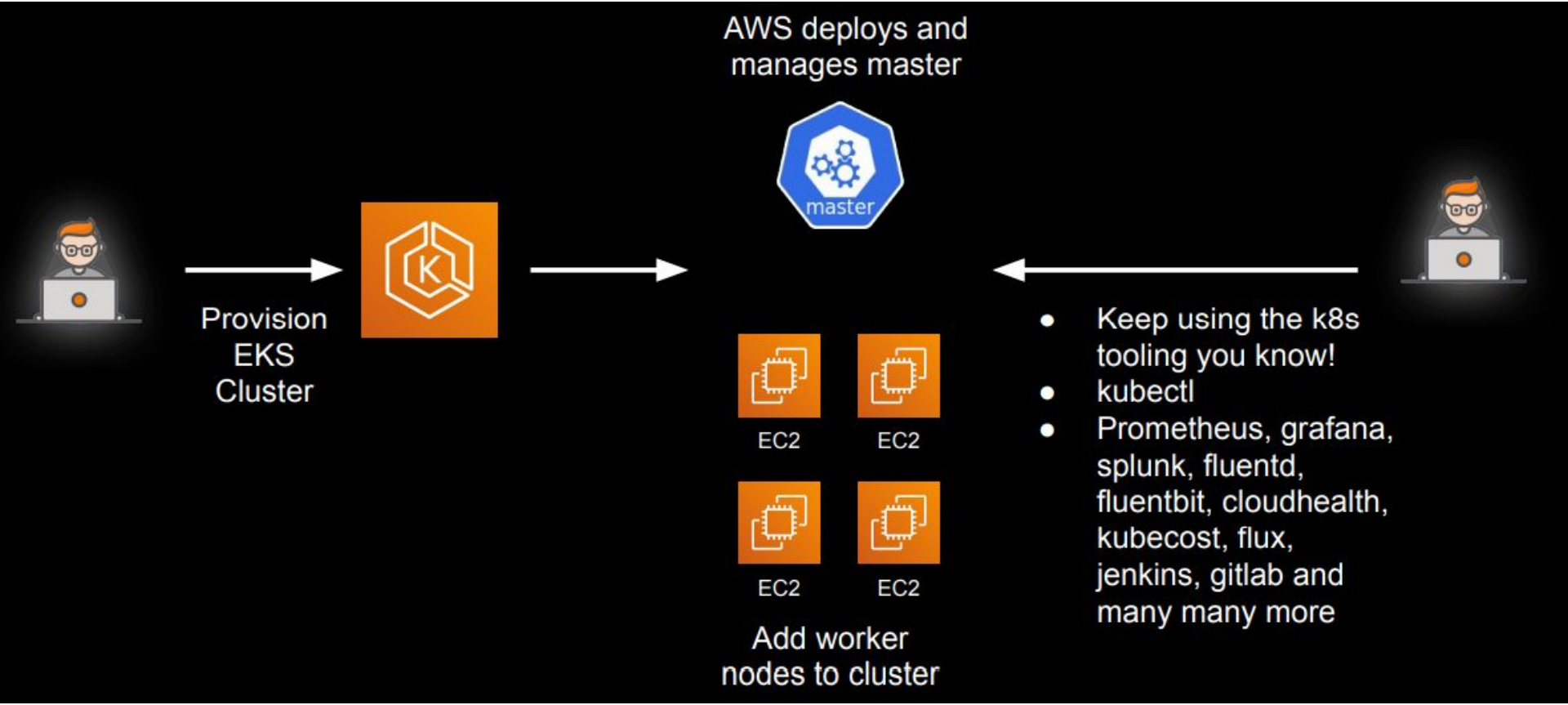
- No worker EC2 whatsoever!
- You define and deploy pods
- Container + Serverless!

EKS in AWS Ecosystem

EKS High Level Architecture



High Level Flow



Money Matters

Amazon Elastic Container Service for Kubernetes

Flat Charge 10 cents/Hour = $72/Month

EC2 EC2 EC2

Price based on EC2

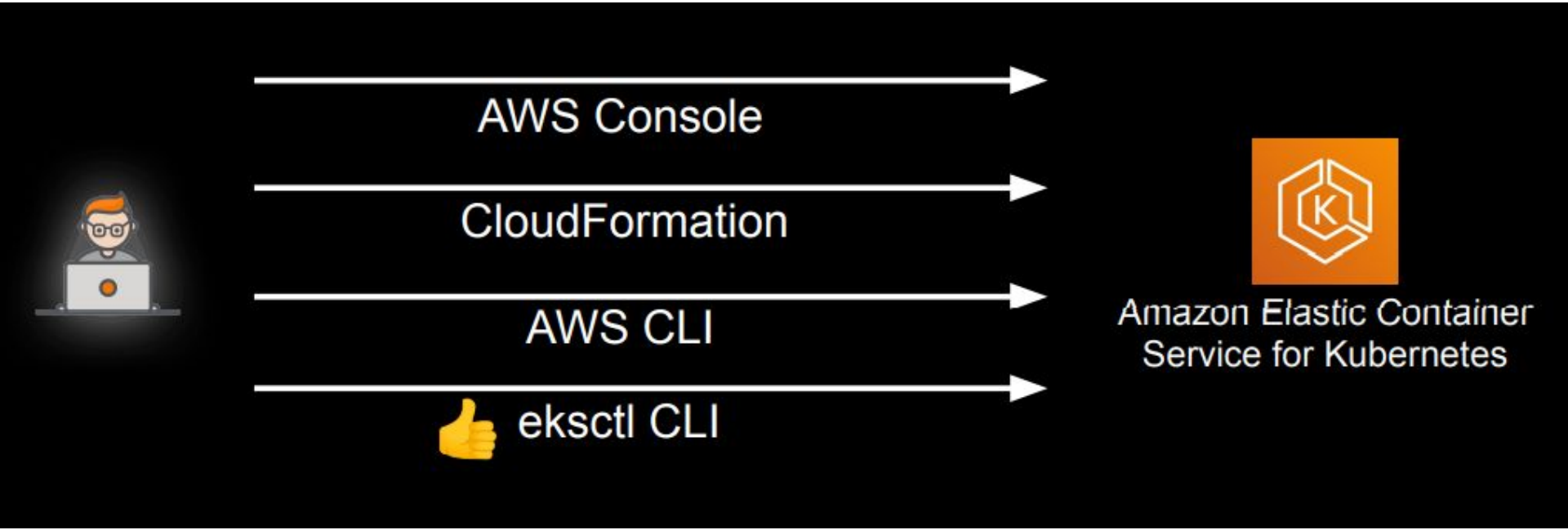
EKS with 3 m5.large in us-east-1 = EKS Control Plane + Worker Nodes (m5.large X 3)

= $72/mo + $219.24/mo

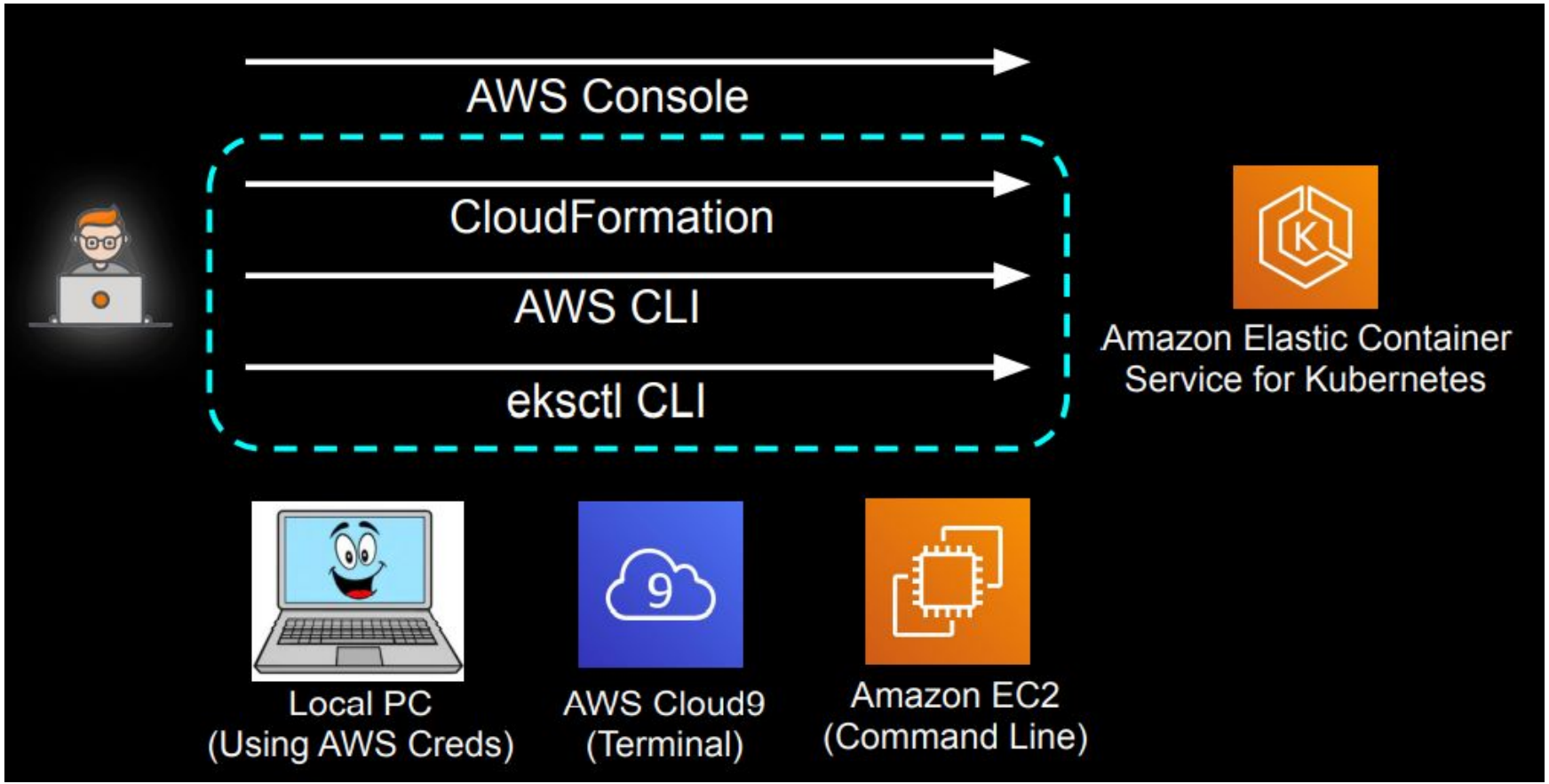
= $291.24/mo

eksctl & kubectl

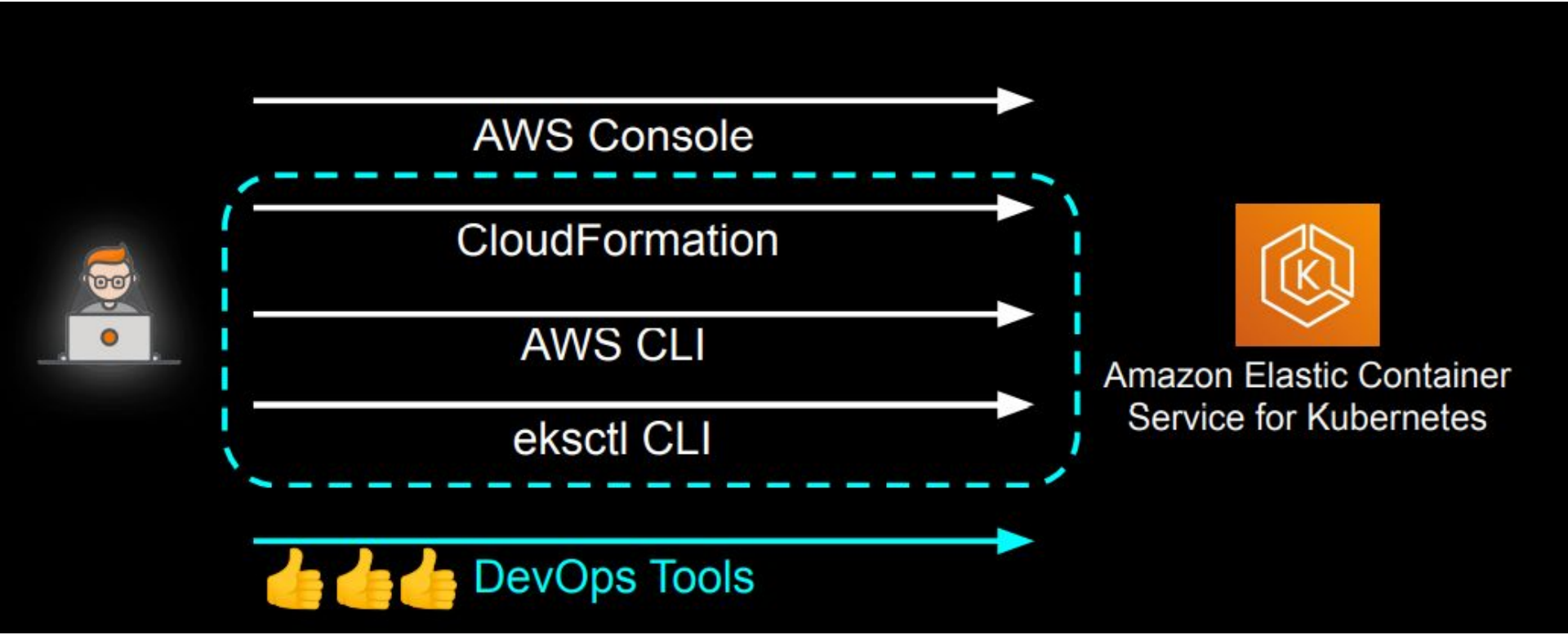
Ways To Spin Up EKS Cluster



Learning Medium



Automation


AWS Console
CloudFormation
AWS CLI
eksctl CLI
DevOps Tools
Amazon Elastic Container Service for Kubernetes

What is eksctl?

- CLI tool for creating clusters on EKS
- Easier than console, for real!
- Abstracts lots of stuff - VPC, Subnet, Sec. Group etc. using CloudFormation

eksctl create cluster

Amazon Elastic Container Service for Kubernetes

AWS Fargate (On EKS)

Available eksctl features (Only on EKS)

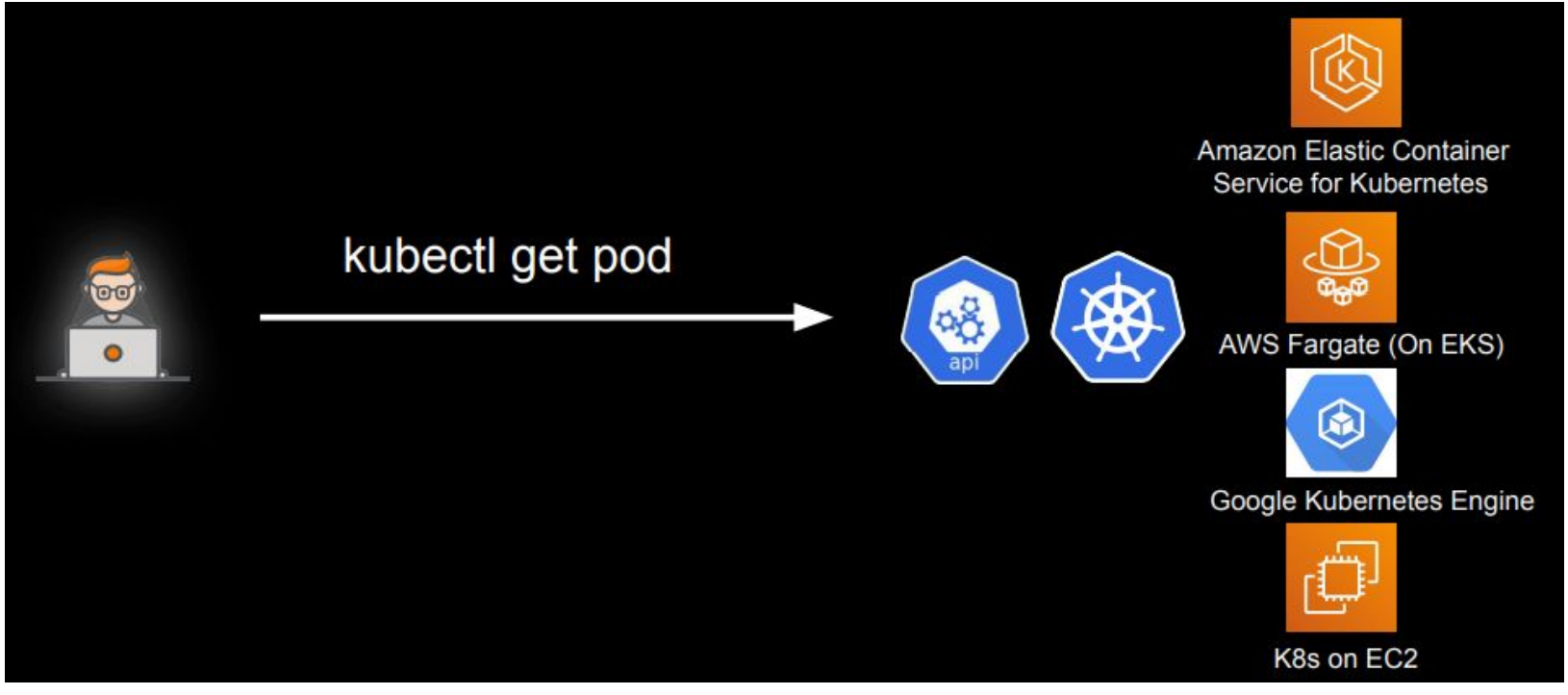
- Create, drain and delete nodegroups
- Scale a nodegroup
- Update a cluster
- Use custom AMIs
- Configure VPC Networking
- Configure access to API endpoints
- Support for GPU nodegroups
- IAM Management and Add-on Policies
- Spot instances and mixed instances
- List cluster Cloudformation stacks
- Install coredns
- Write kubeconfig file for a cluster
- Spot instances and mixed instances
- Create, get, list and delete clusters

eksctl Commands

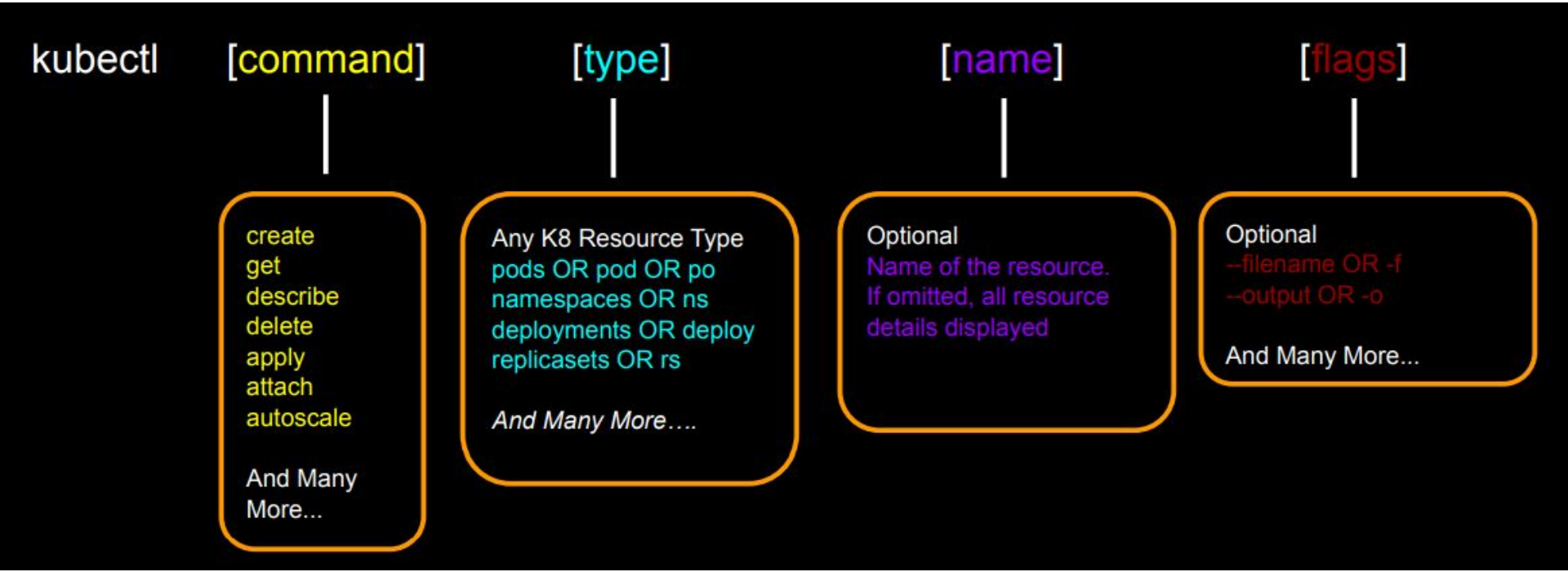
Command	Brief Description
`eksctl create cluster`	Create EKS Cluster with one nodegroup containing 2 m5.large
`eksctl create cluster --name <name> --version 1.15 --node-type t3.micro --nodes 2`	Create EKS Cluster with K8 version 1.15 with 2 t3.micro nodes
`eksctl create cluster --name <name> --version 1.15 --nodegroup-name <nodegrpname> --node-type t3.micro --nodes 2 --managed`	Create EKS cluster with managed node group
`eksctl create cluster --name <name> --fargate`	EKS Cluster with Fargate Profile

What is kubectl?

- CLI for running commands against a cluster on K8s resources
- Communicate via cluster API Server
- Works for any K8s cluster - EKS, K8s on EC2, GKE etc.



kubectl Command Syntax



All available command and resource type : https://kubernetes.io/docs/reference/kubectl/overview

kubectl Command Syntax

kubectl	[command]	[TYPE]	[NAME]	[flags]
kubectl	get	pod	pod1	
kubectl	get	pod		
kubectl get po				
kubectl	get	pod	pod2	-o yaml

Most Used kubectl Commands

Command	Brief Description
`kubectl apply -f ./manifest-file.yaml`	Create resources based on manifest. Declarative Way! Best Way!
`kubectl get nodes`	List all node info
`kubectl get services`	List all services
`kubectl get pods -o wide`	List pods with more details
`kubectl get pod my-pod -o yaml`	Get a pod's YAML
`kubectl get deployment my-dep`	List a particular deployment
`kubectl exec -it podname -- /bin/bash`	Get a shell to the running Container

DEMO TIME!

- Spin up EKS Cluster using eksctl
- Use kubectl
  - Deploy nginx using manifest file
  - Get resources info

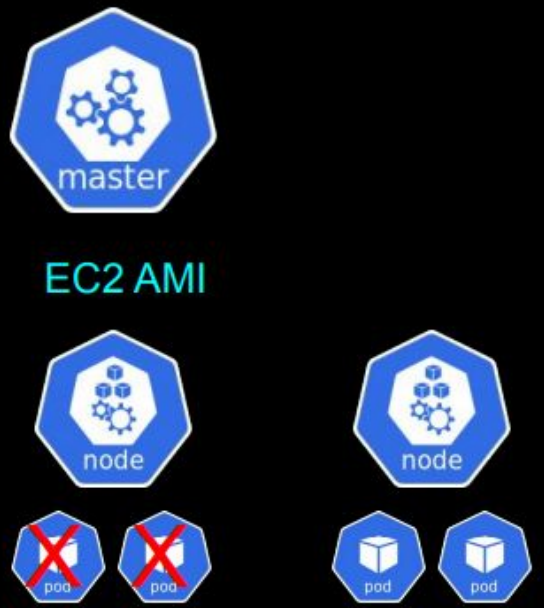
EKS Managed Nodegroups

EKS Regular Nodes and Pods

- K8 Version needs to be updated
- Deploy Security Patches
- EC2 AMI Need to Be Updated

EKS Nodes Regular Update

- K8 Version needs to be updated
- Deploy Security Patches
- EC2 AMI Need to Be Updated
  - You need to create AMIs



- Application will be DOWN!!
- Or You have to orchestrate HA

EKS Managed Nodegroups

- Create and Manage EC2 Workers for you
- Amazon releases AMIs with bug fix, security patches for EKS Worker Nodes
  - Custom AMIs not supported (As of April, 2020)
- Automated deployment of updated AMIs with security patches, CVEs
  - No app downtime
  - No overhead of user managed orchestration
  - Auto scaling group is used behind the scenes

EKS Nodes: Managed Nodegroup Update

- Worker Node K8s Version updated with one click/API call
- AWS Provides AMI with security updates
  - No patching/Rehydrate effort for you

- Application will be up and running
- AWS orchestrate HA through Auto Scaling Group
- Respects Pod Disruption Budget

EKS Logging & Monitoring

EKS Logging

EKS Control Plane Logging

EKS Worker Nodes Logging

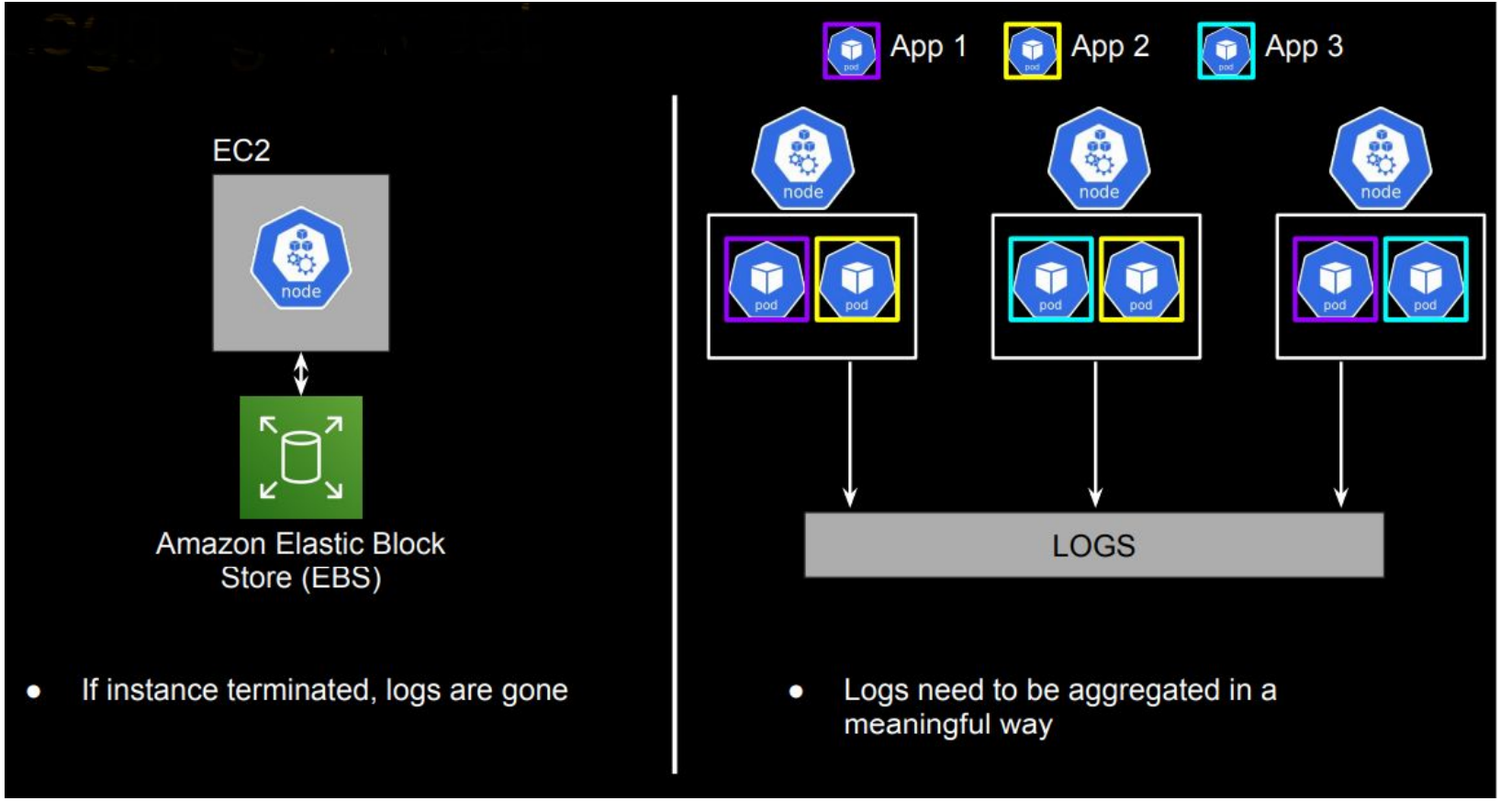
EKS Logging

- EKS Control Plane Logging
  - K8 api
  - audit
  - authenticator
  - controllerManager
  - scheduler
- EKS Worker Nodes Logging

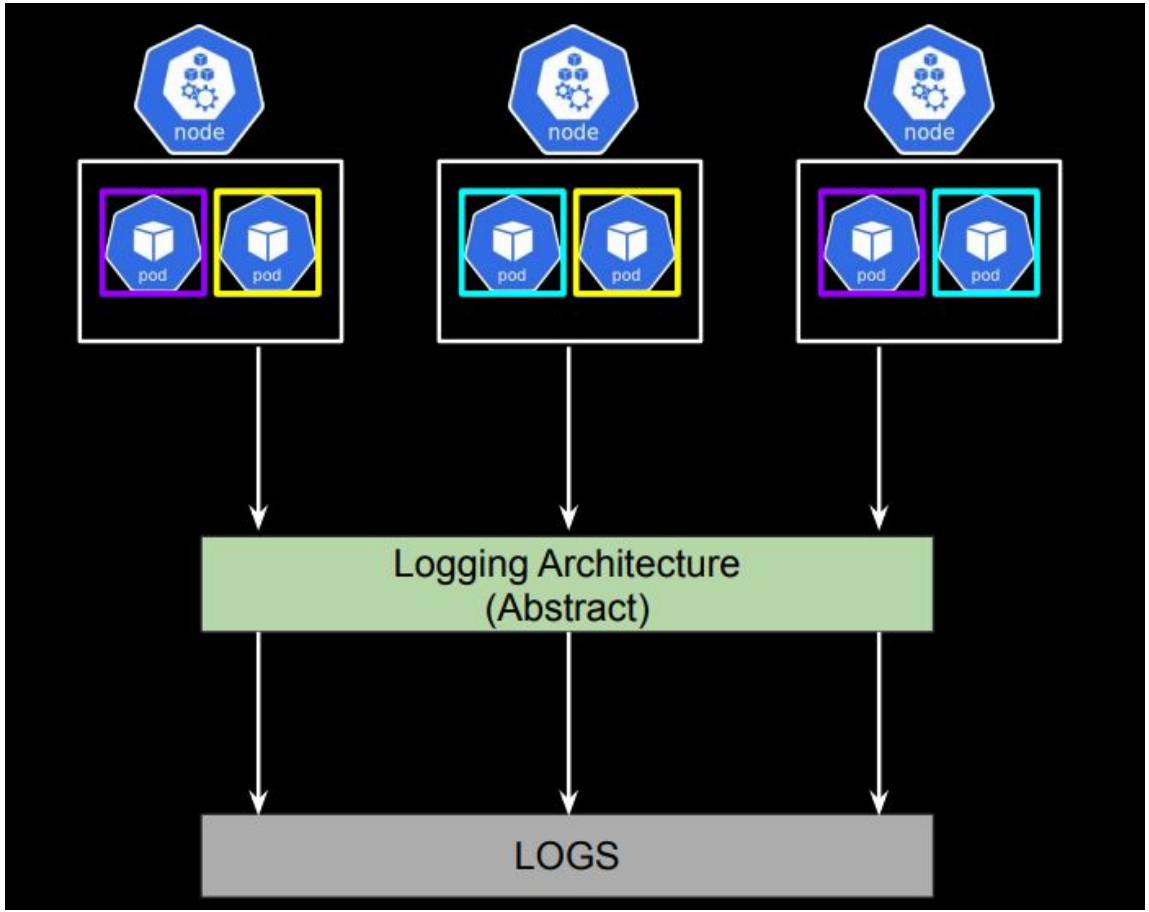
EKS Logging

- EKS Control Plane Logging
  - K8 api
  - audit
  - authenticator
  - controllerManager
  - scheduler
- EKS Worker Nodes Logging
  - System logs from kubelet, kube-proxy, or dockerd
  - Application logs from application containers

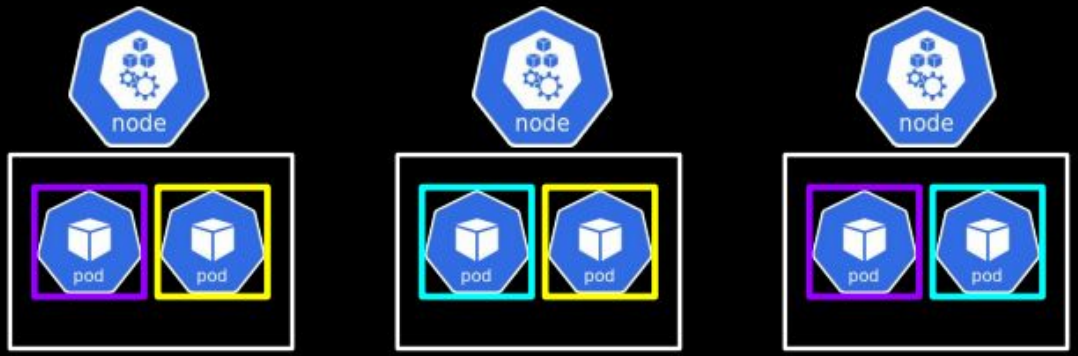
Logging Caveat



Logging Caveat

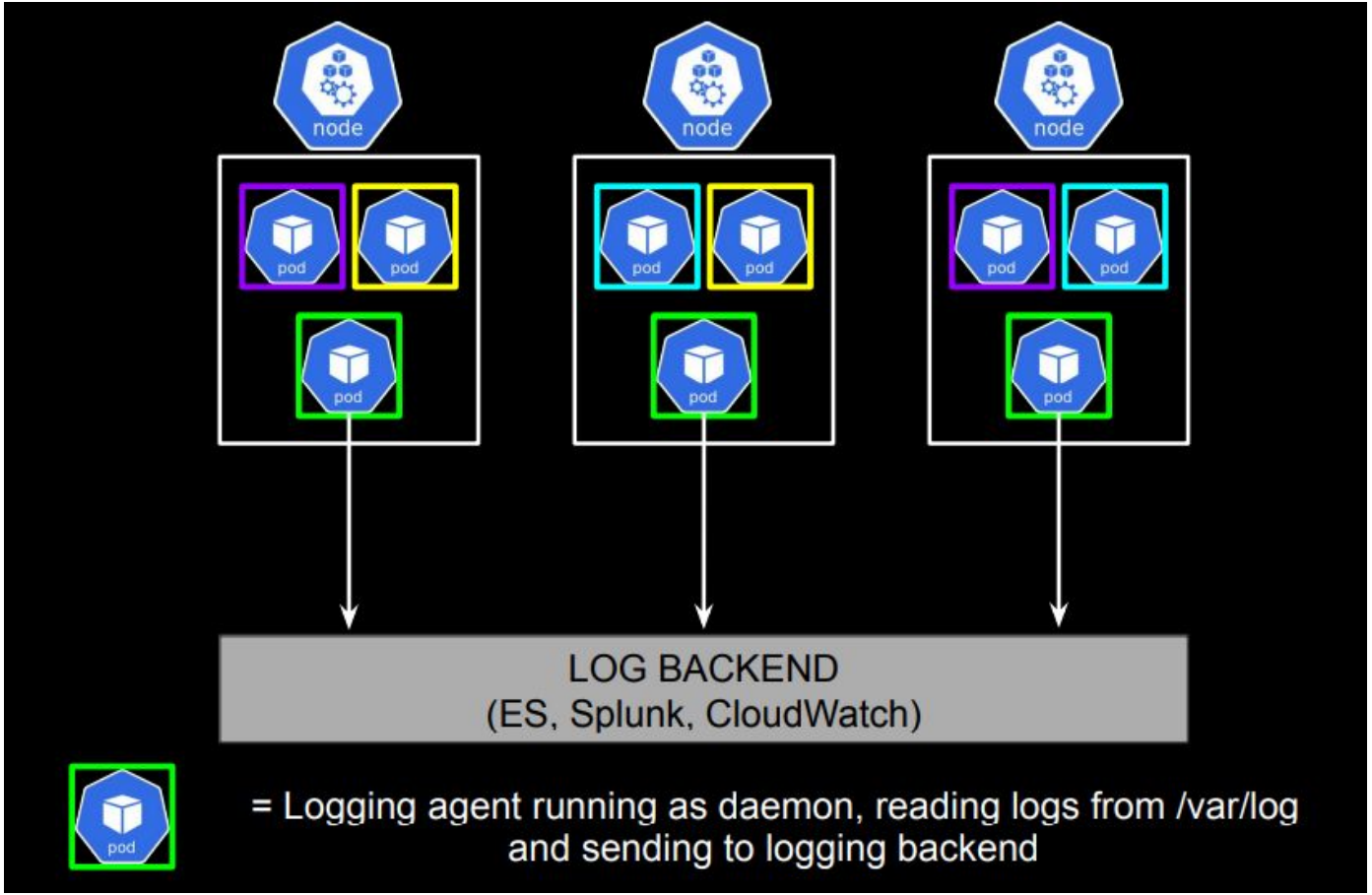


Kubernetes Worker Nodes

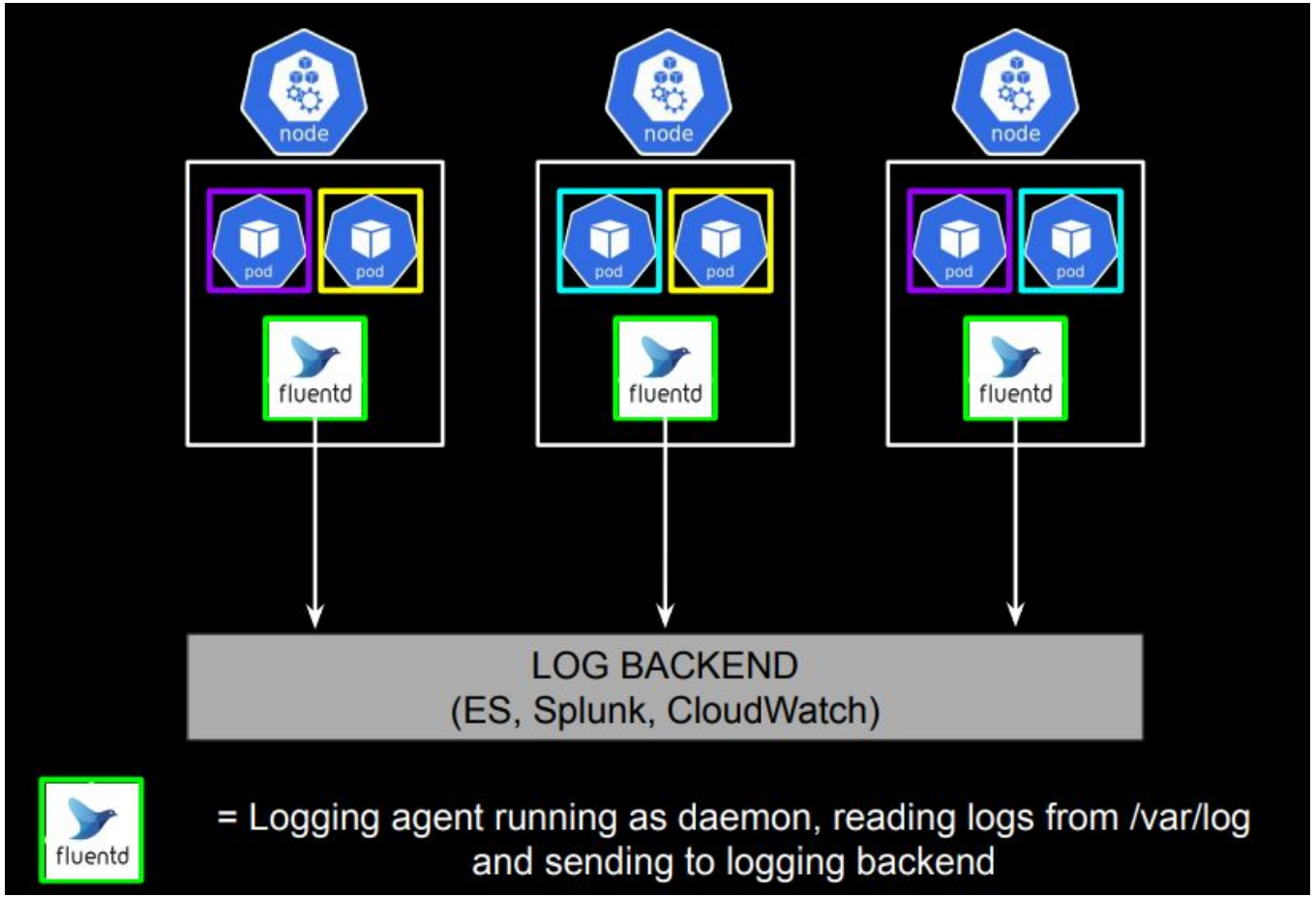


- Containerized application writes to
  - stdout and stderr
- System logs go to
  - systemd
- Container redirect logs to /var/log/containers/*.log files
- Now we know where to extract logs from!

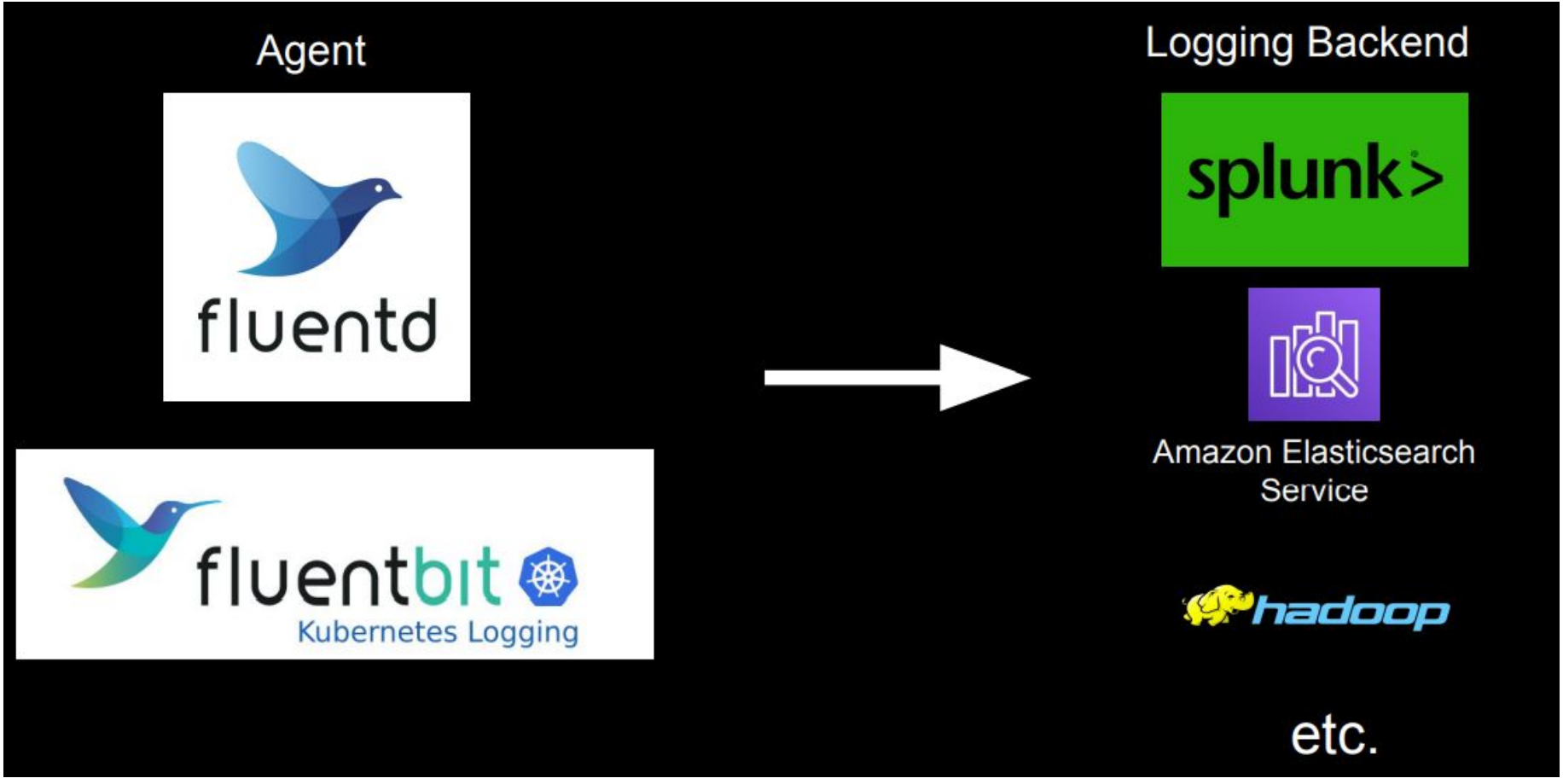
Implementation



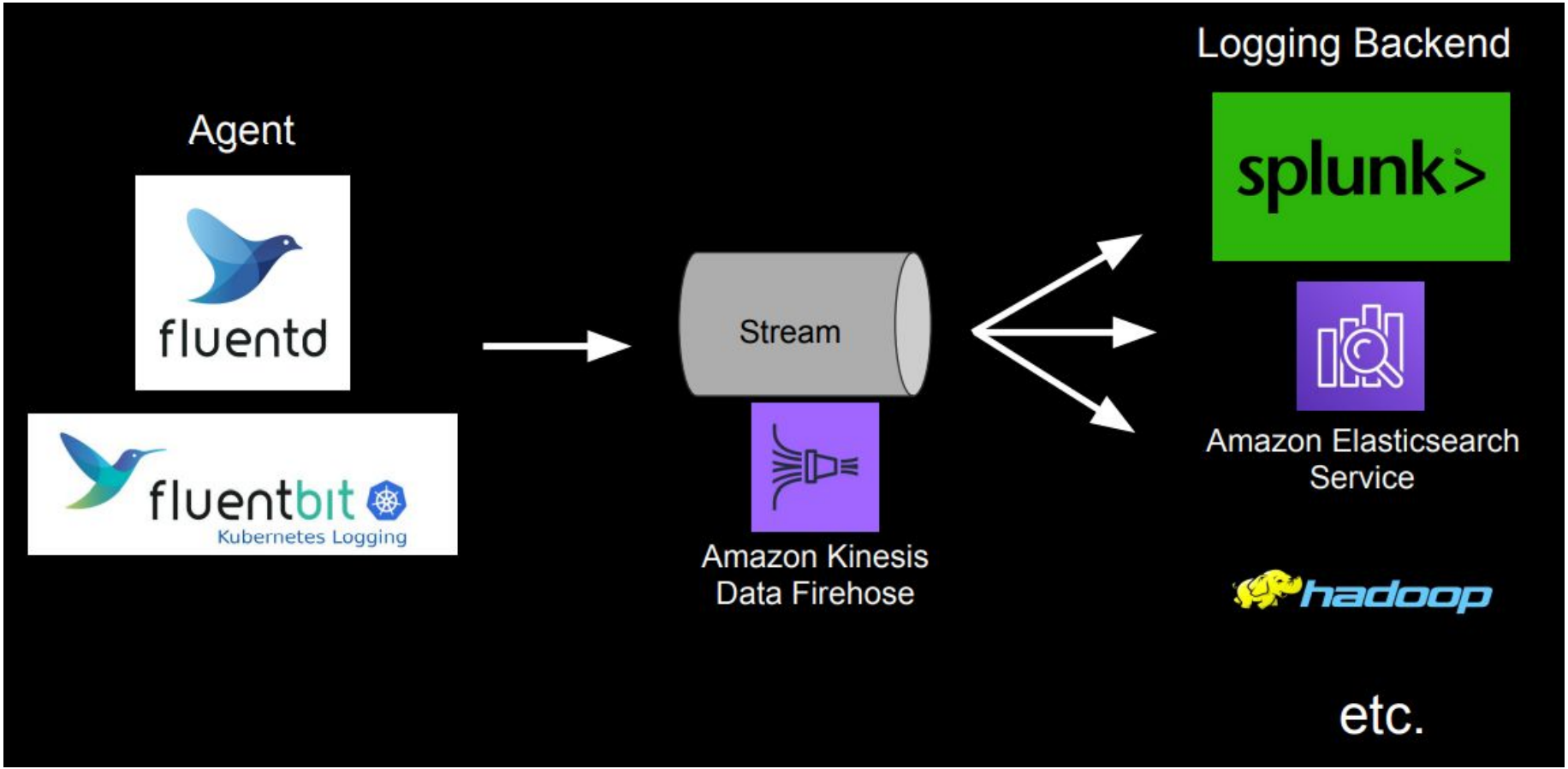
Implementation



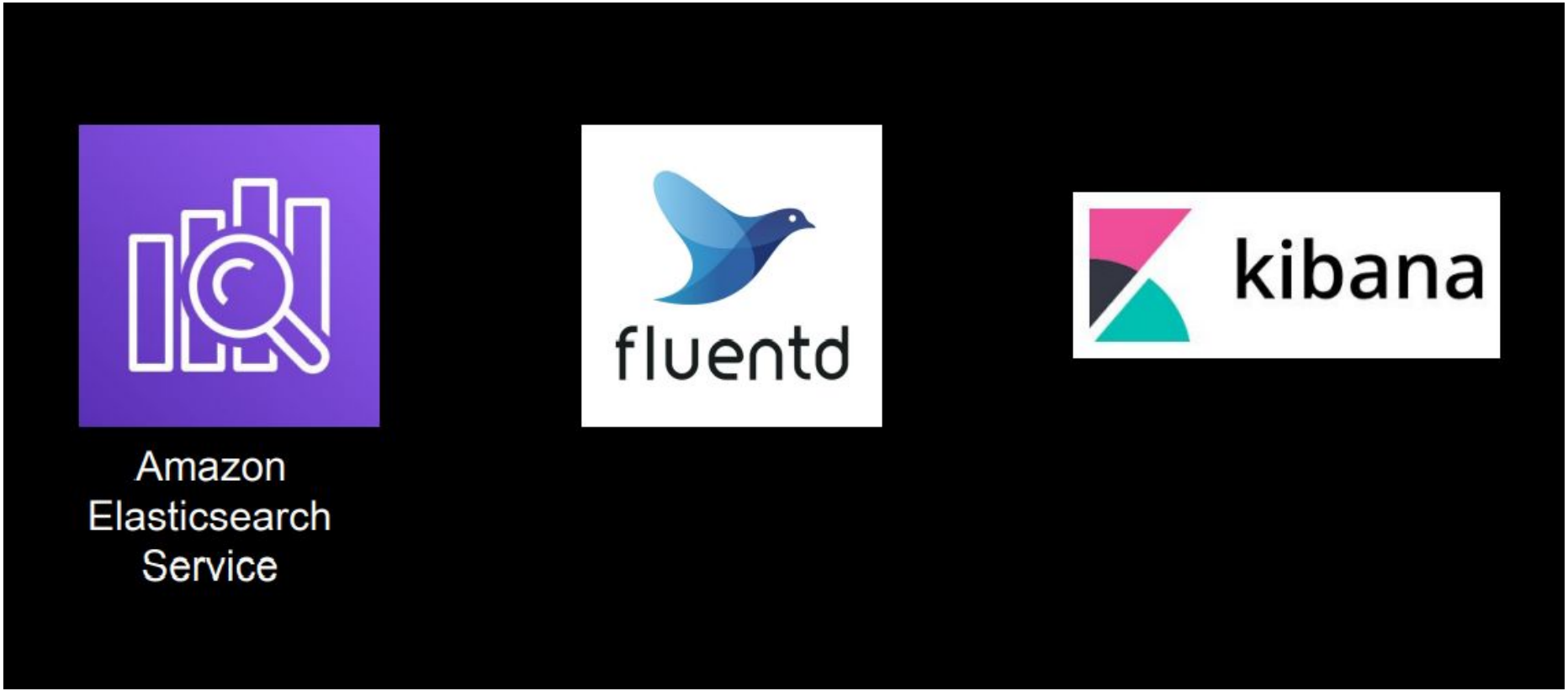
Different Options



For Streaming to Multiple Log Backends



EFK Stack



Under Pressure!

- fluentd has 100+ plugins, fluentbit has around 20 (April 2020)
- However as traffic goes up, fluentd can't keep up
  - fluentd based on Ruby and memory intensive
  - Slow propagation of logs
  - Loss of logs
  - fluentd buffer can be increased to solve this, but not dynamic

Contd on next slide..

Under Pressure!

Implementation

EKS Control Plane Logging

EKS Control Plane Logging

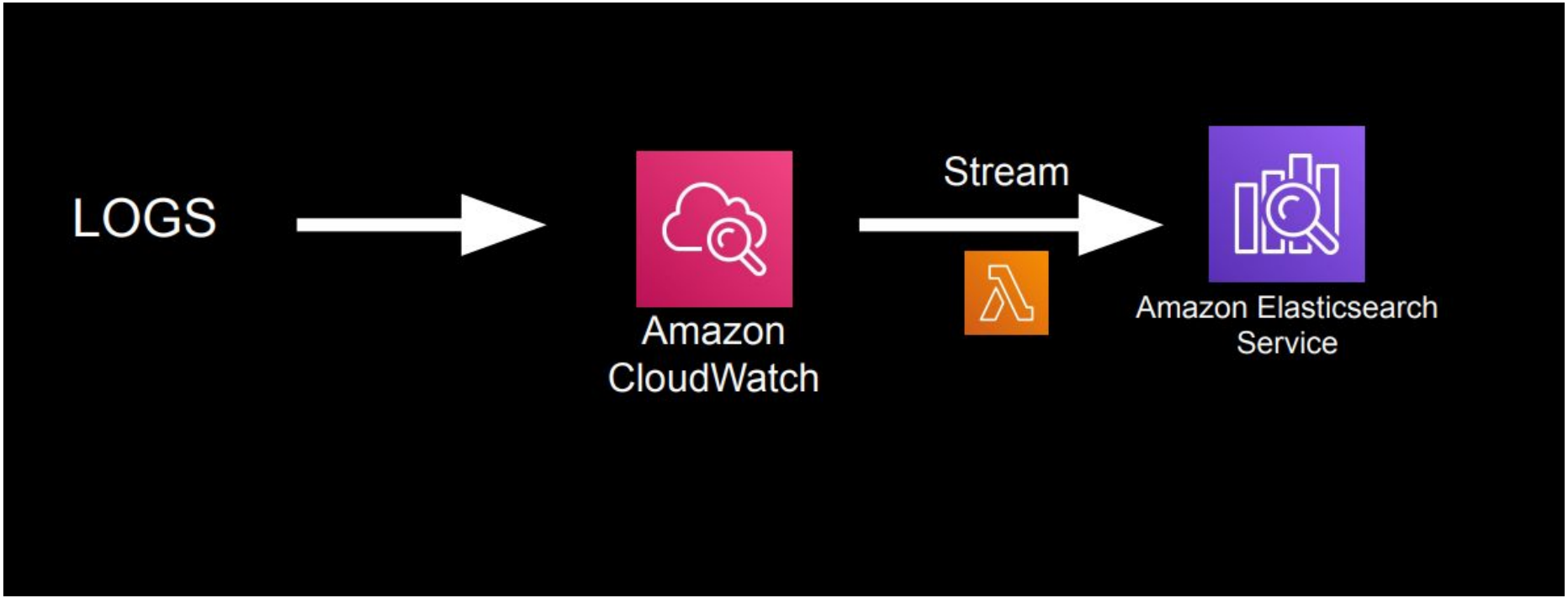
- K8s api
- audit
- authenticator
- controllerManager
- scheduler

EKS Control Plane Logging

- EKS Control Plane Logging
  - K8s api
  - audit
  - authenticator
  - controllerManager
  - scheduler

Amazon
CloudWatch

EKS Control Plane Logging



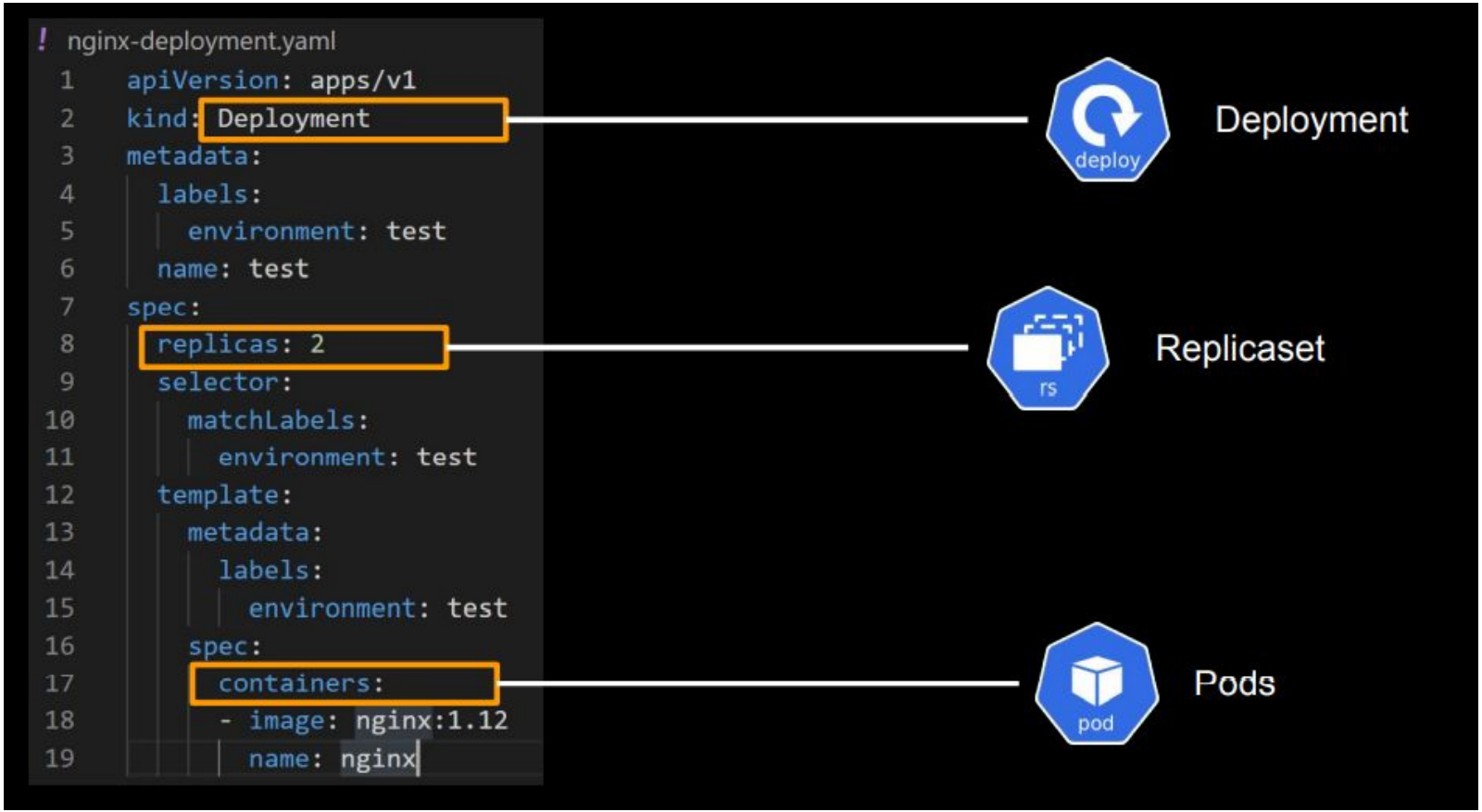
Kubernetes Dashboard

- Web-based Kubernetes user interface
- Overview of applications and resources running on cluster
- Create and modify resources!
  - Pod
  - Deployments
  - Jobs
  - etc

Kubernetes Dashboard

EKS ADVANCED CONCEPTS

Quiz: What Resources Created?

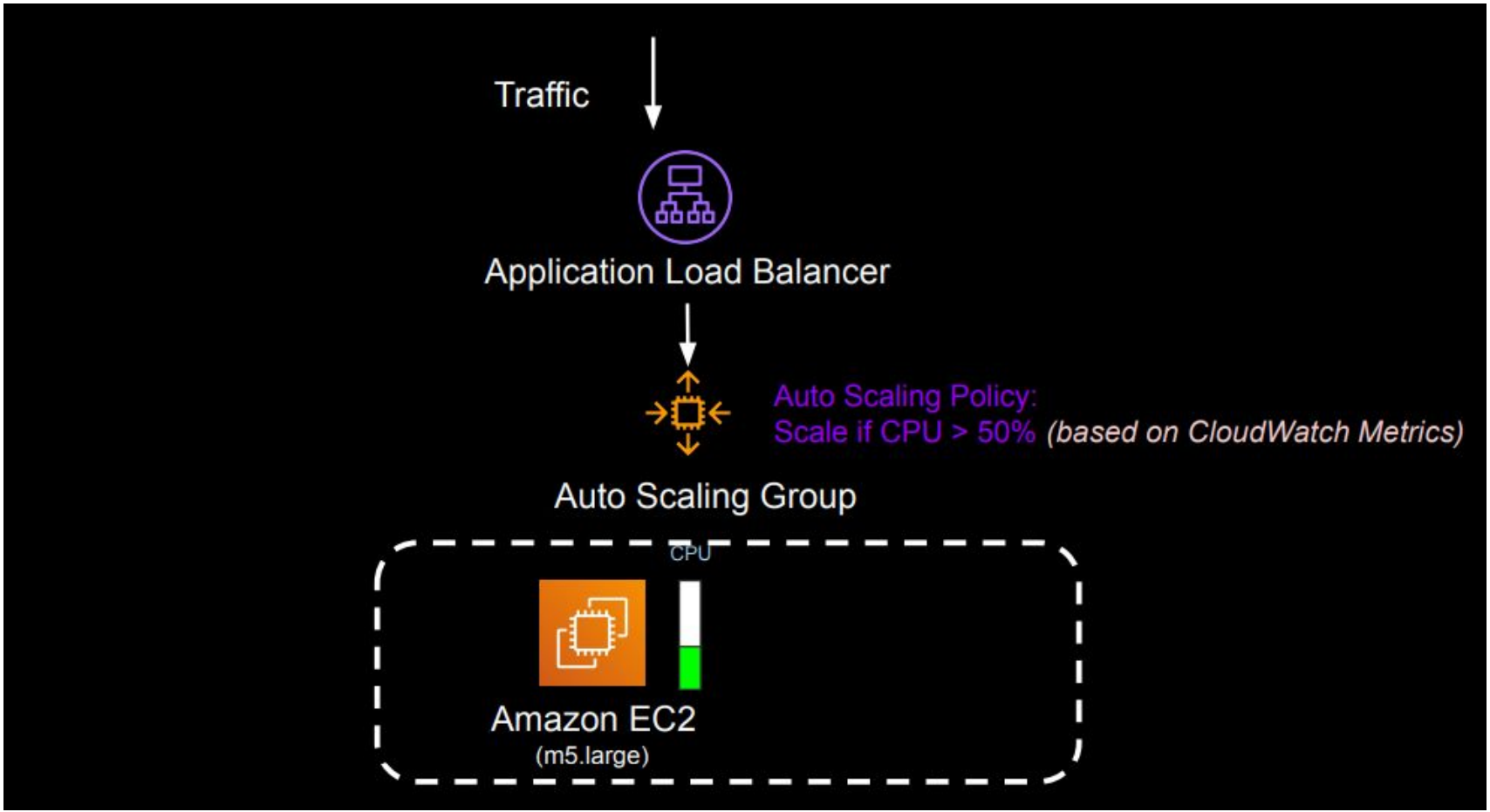


Container Scaling

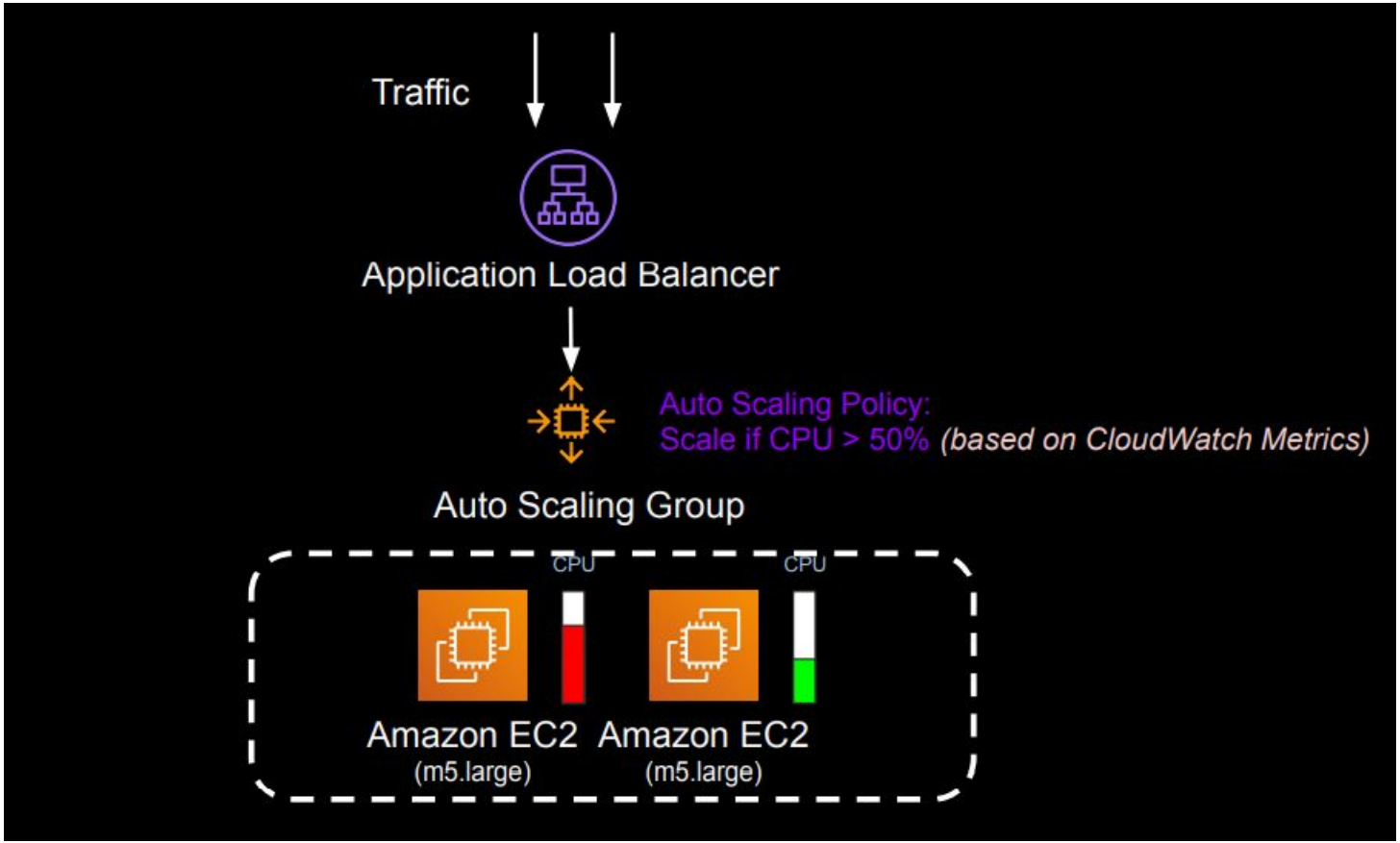
- Quick Look into EC2 Scaling
- Container Scaling
- Understand Pod Limits and Requests
- Horizontal Pod Autoscaler
- Understand Manifest File
- HPA Demo

Horizontal Pod Autoscaler

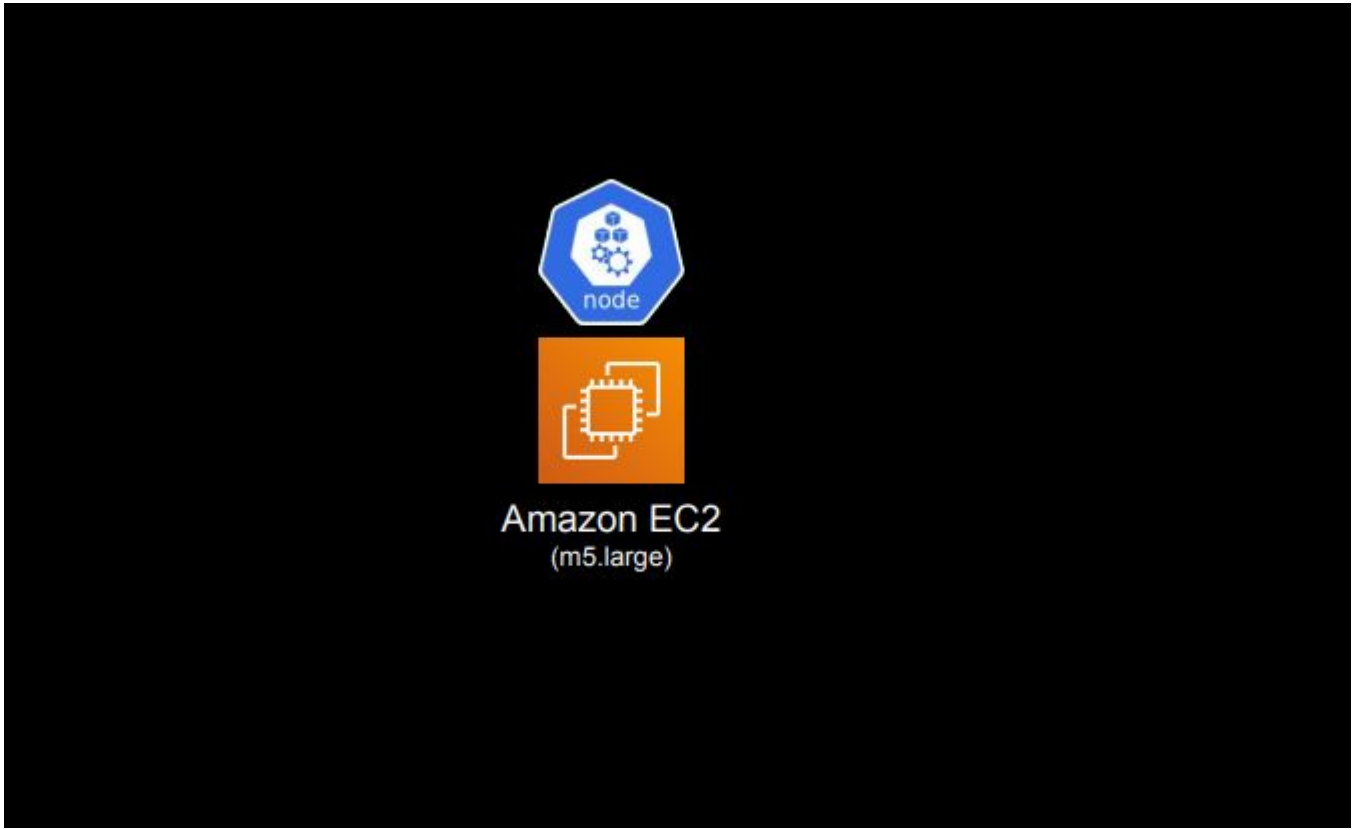
Going Back to EC2 Scaling



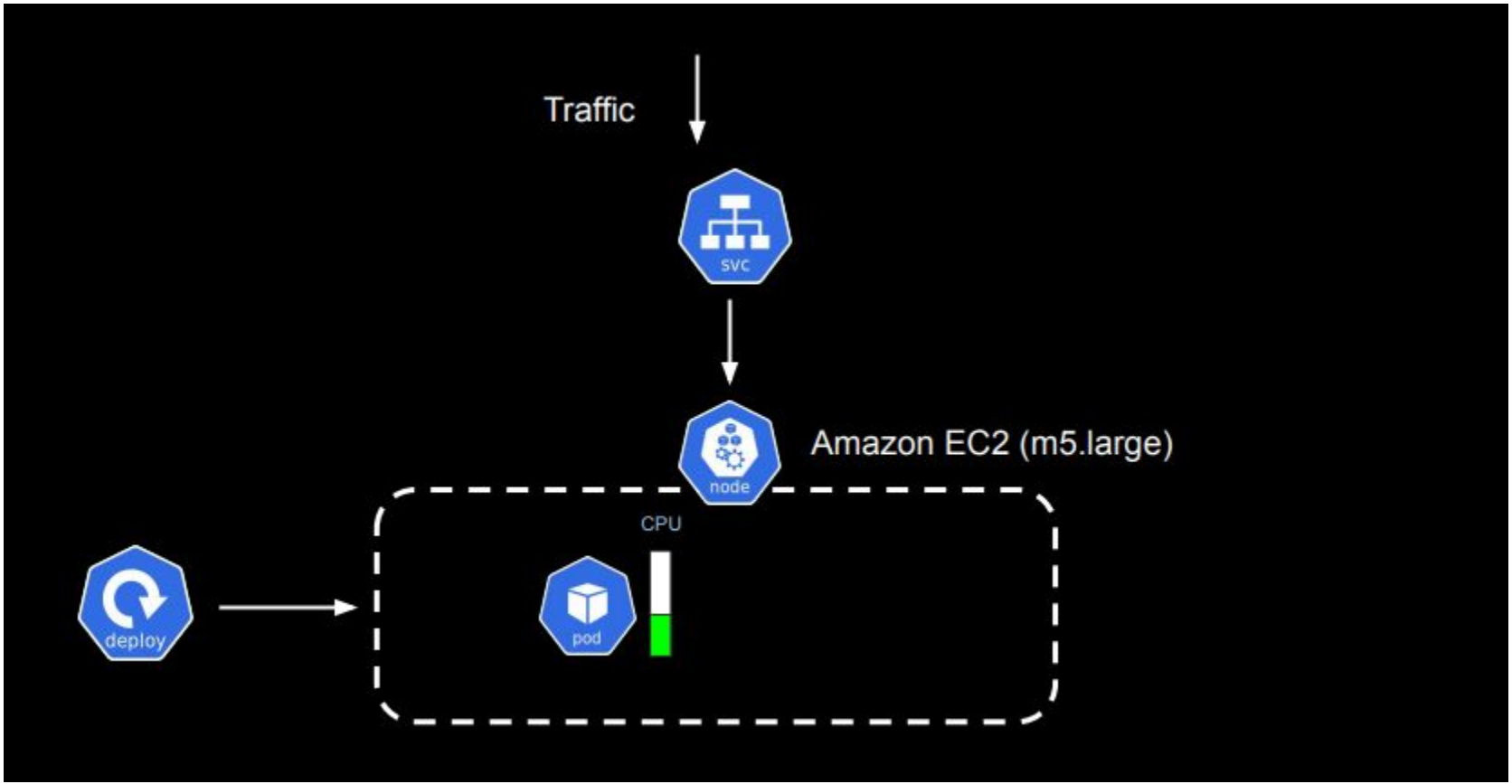
Going Back to EC2 Scaling



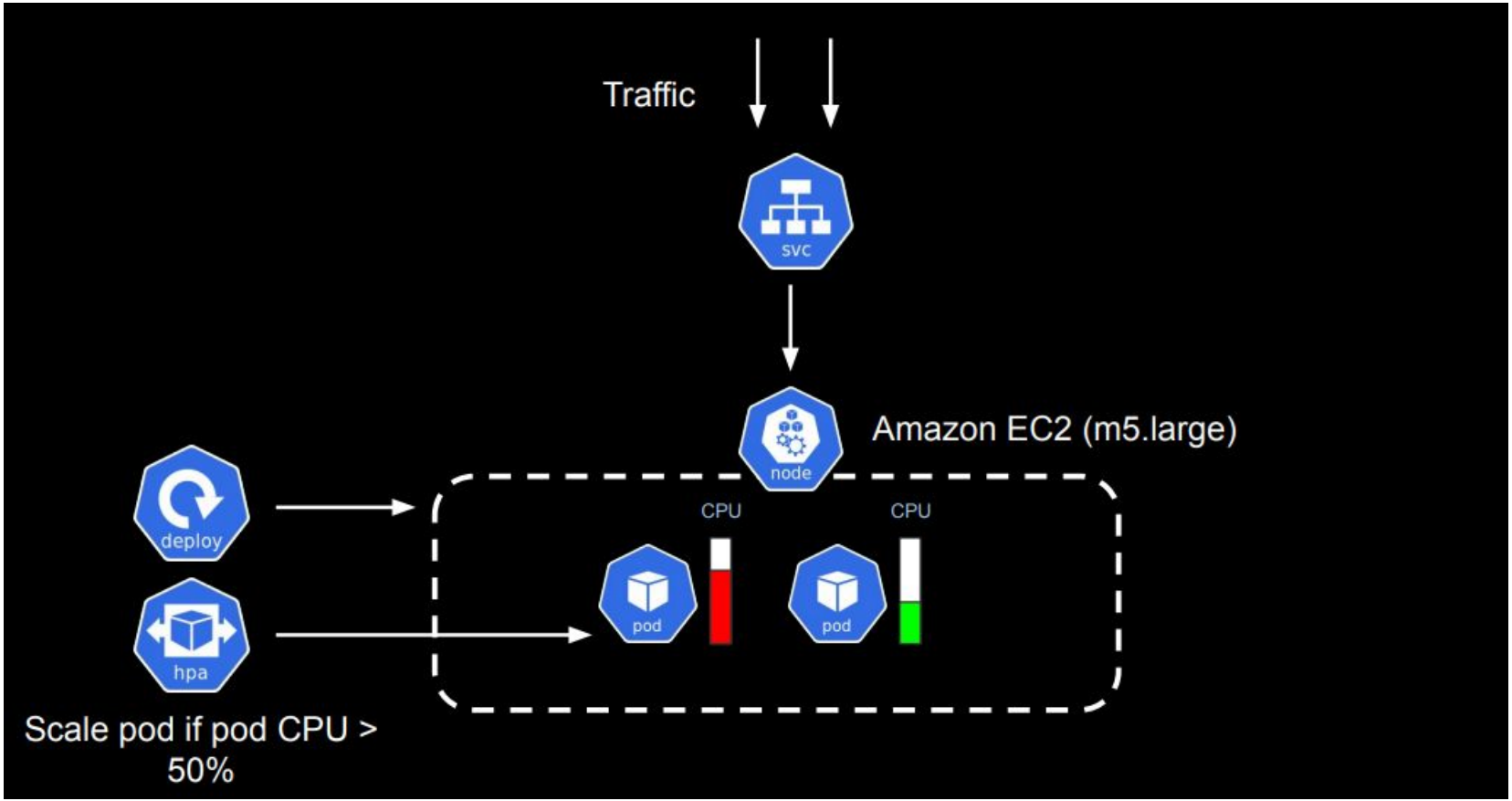
EKS Container Scaling



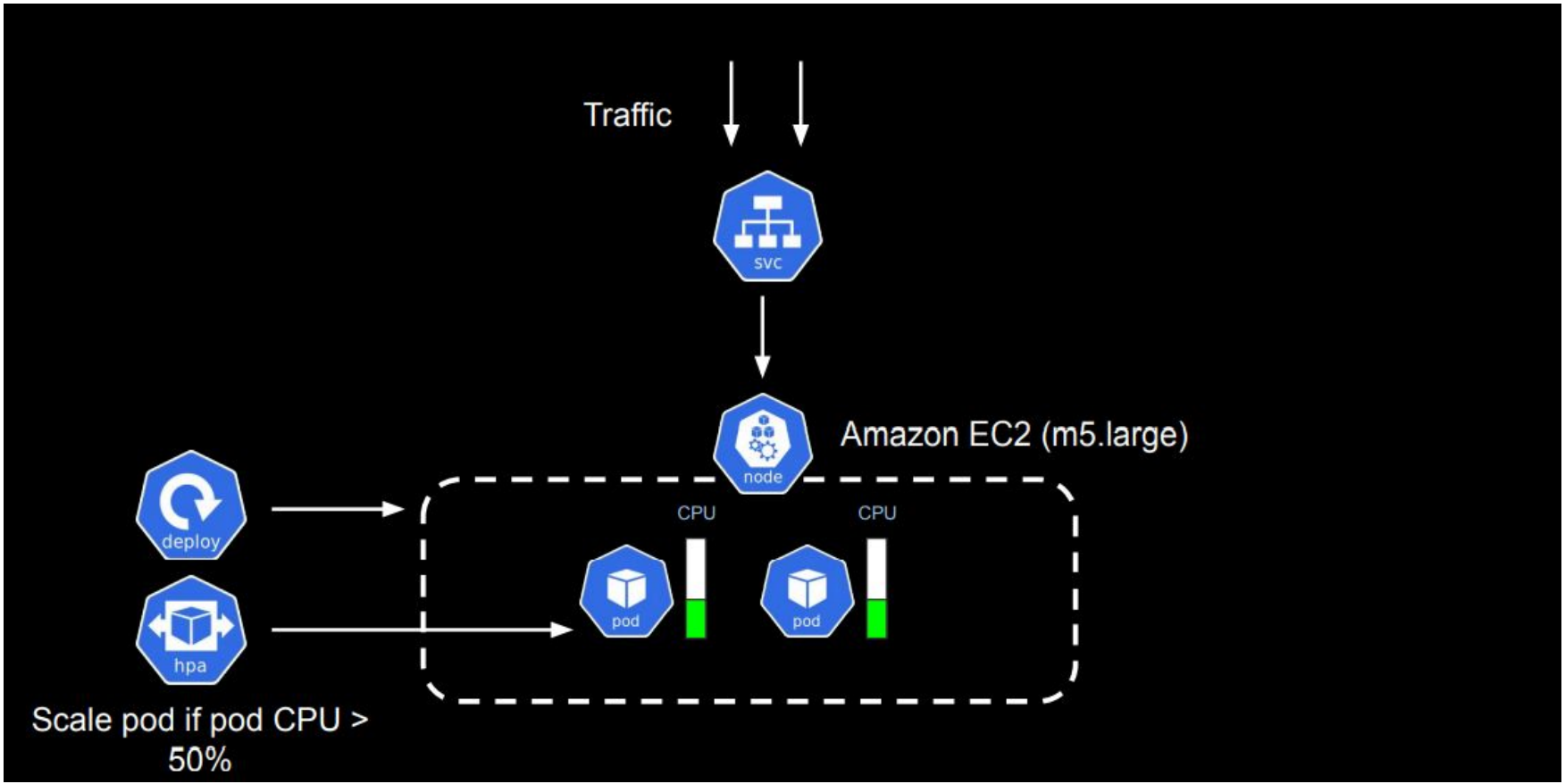
Horizontal Pod Autoscaler (HPA)



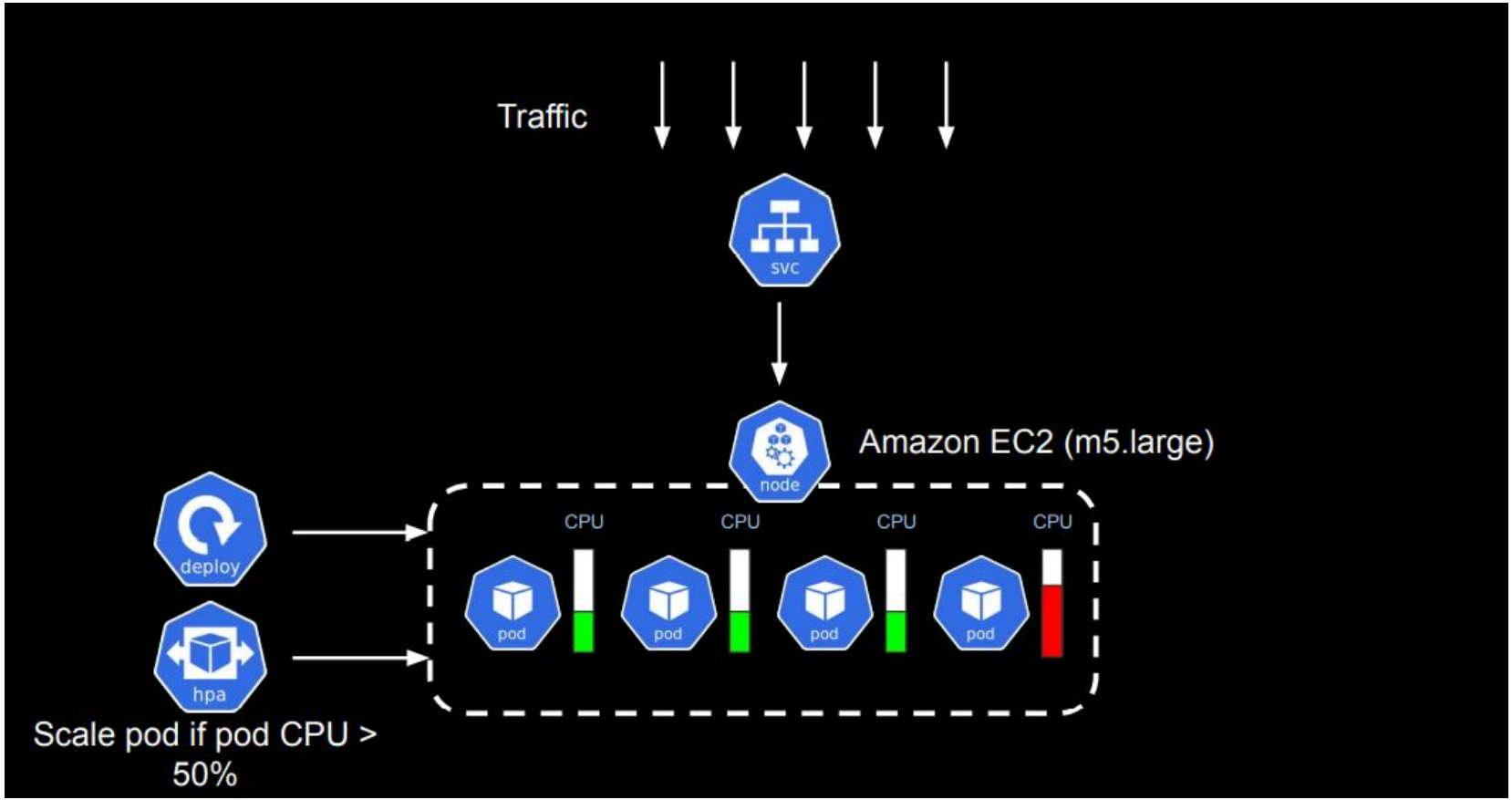
Horizontal Pod Autoscaler (HPA)



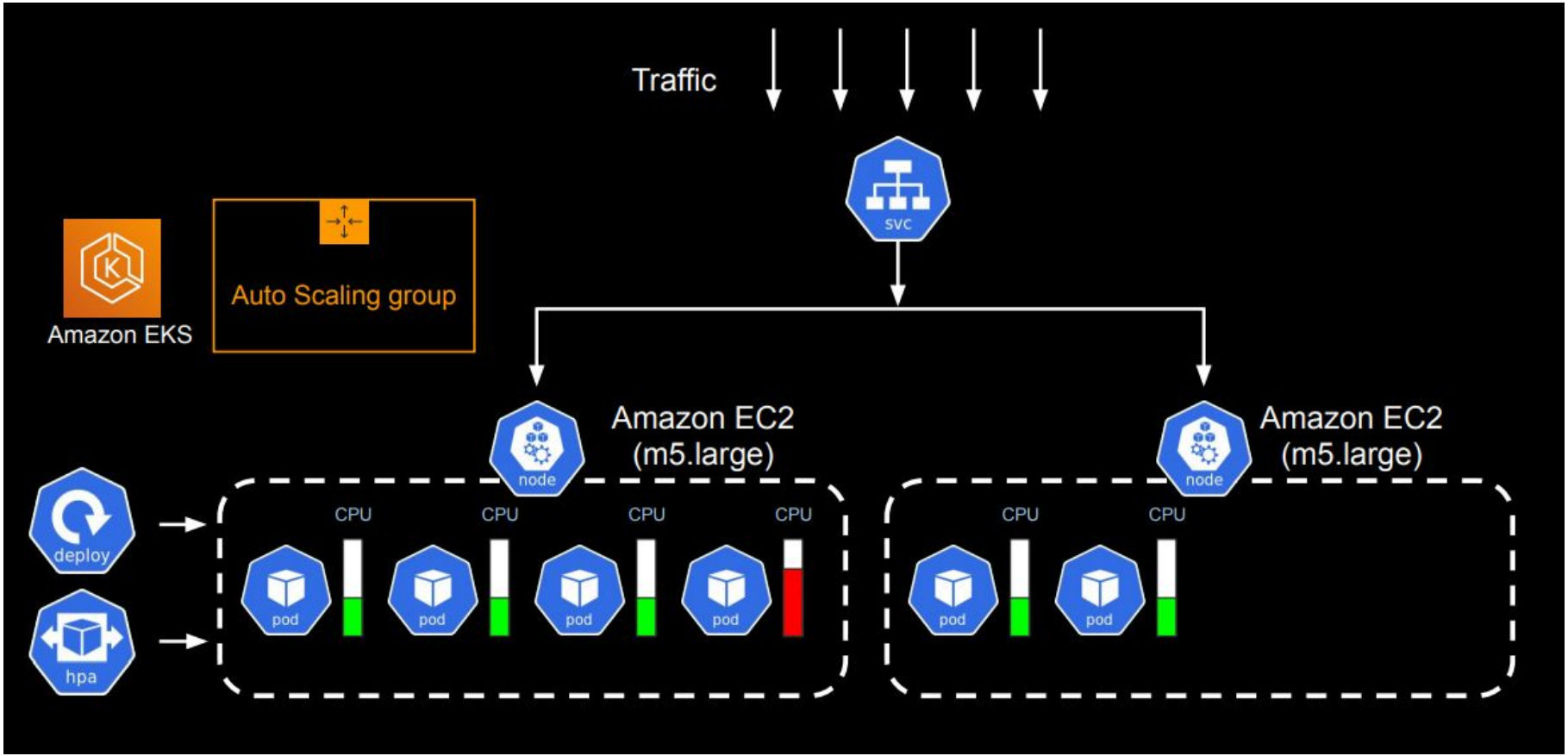
Horizontal Pod Autoscaler (HPA)



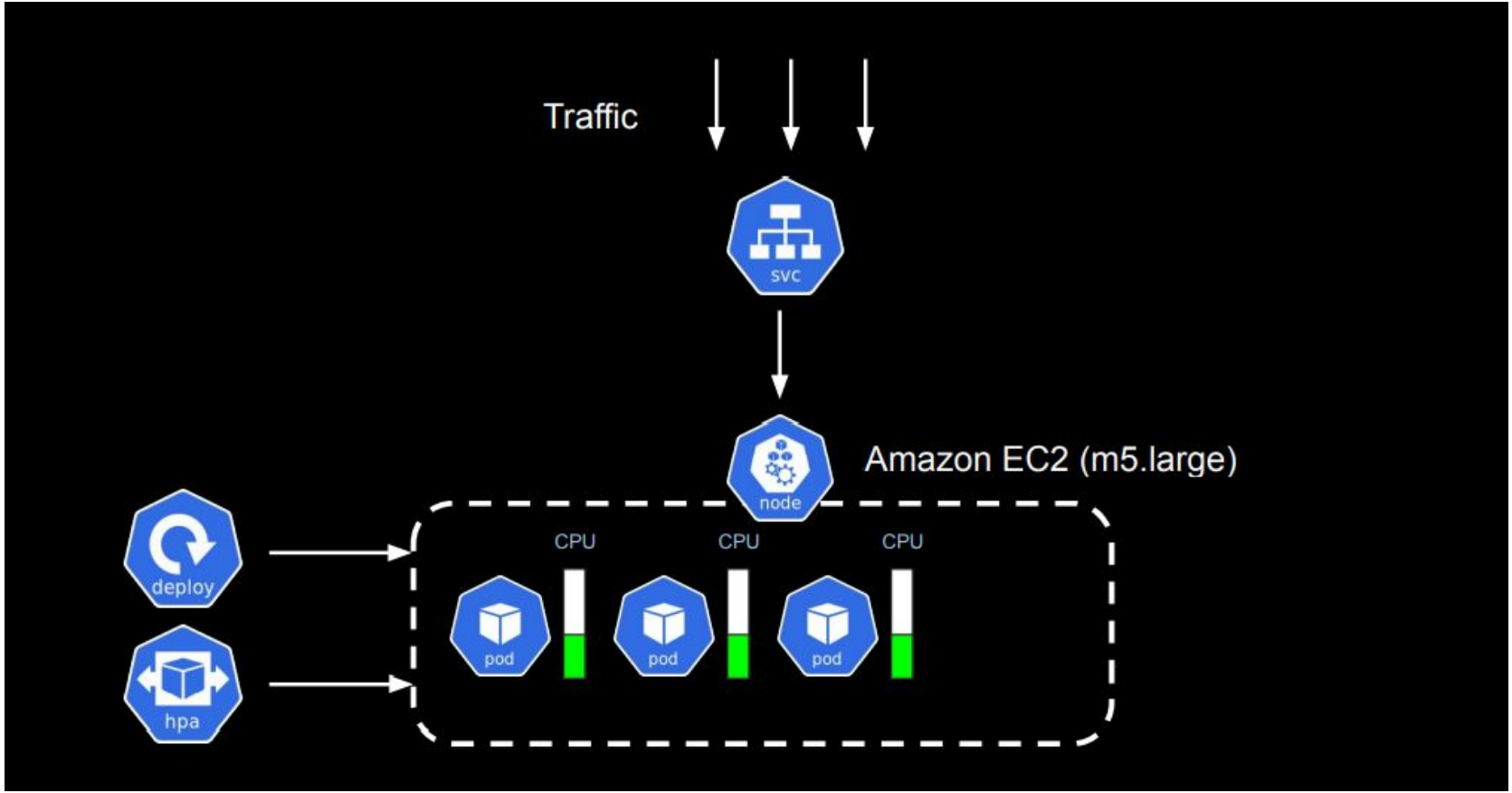
Horizontal Pod Autoscaler (HPA)



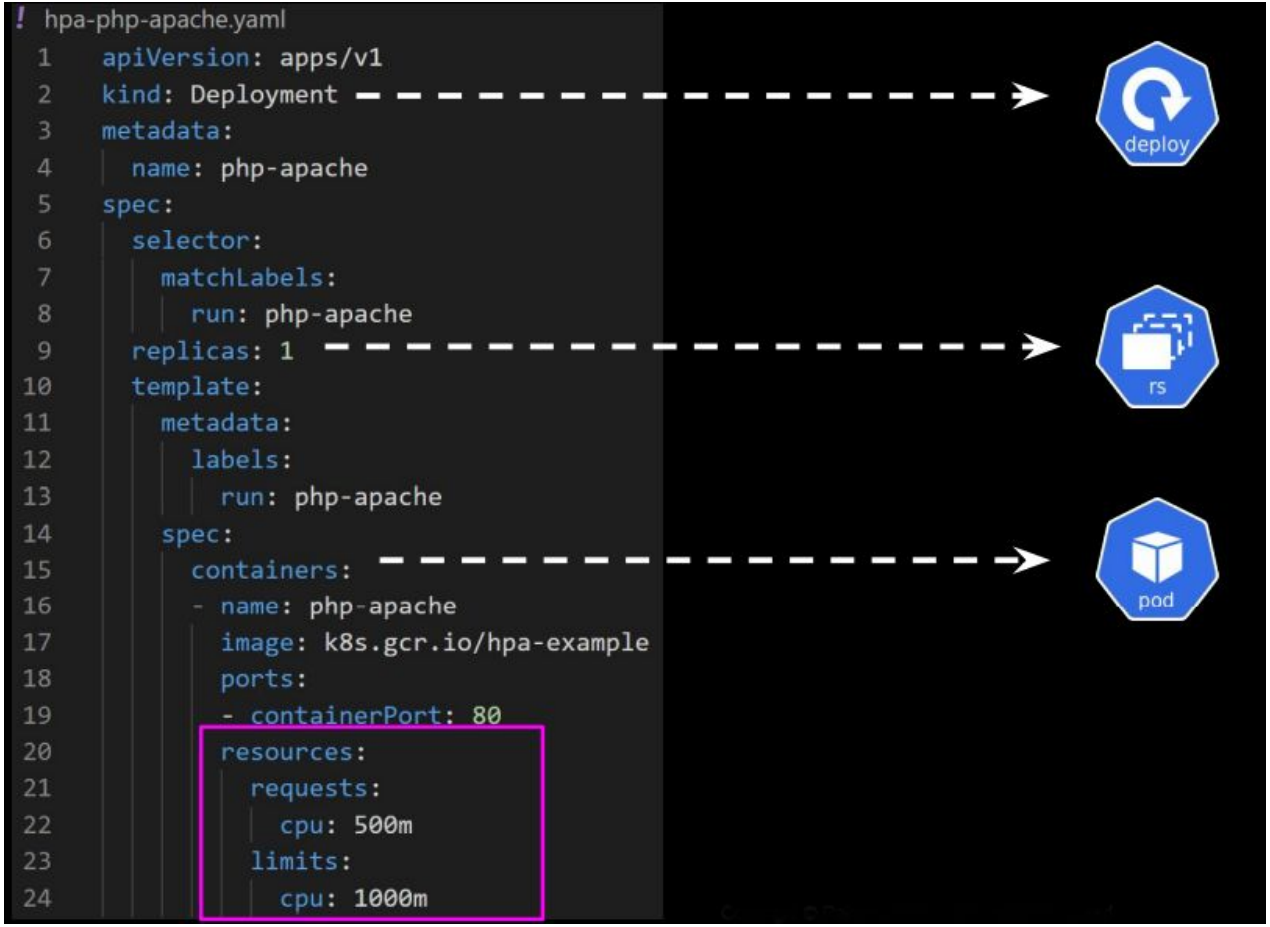
EKS Cluster Autoscaler



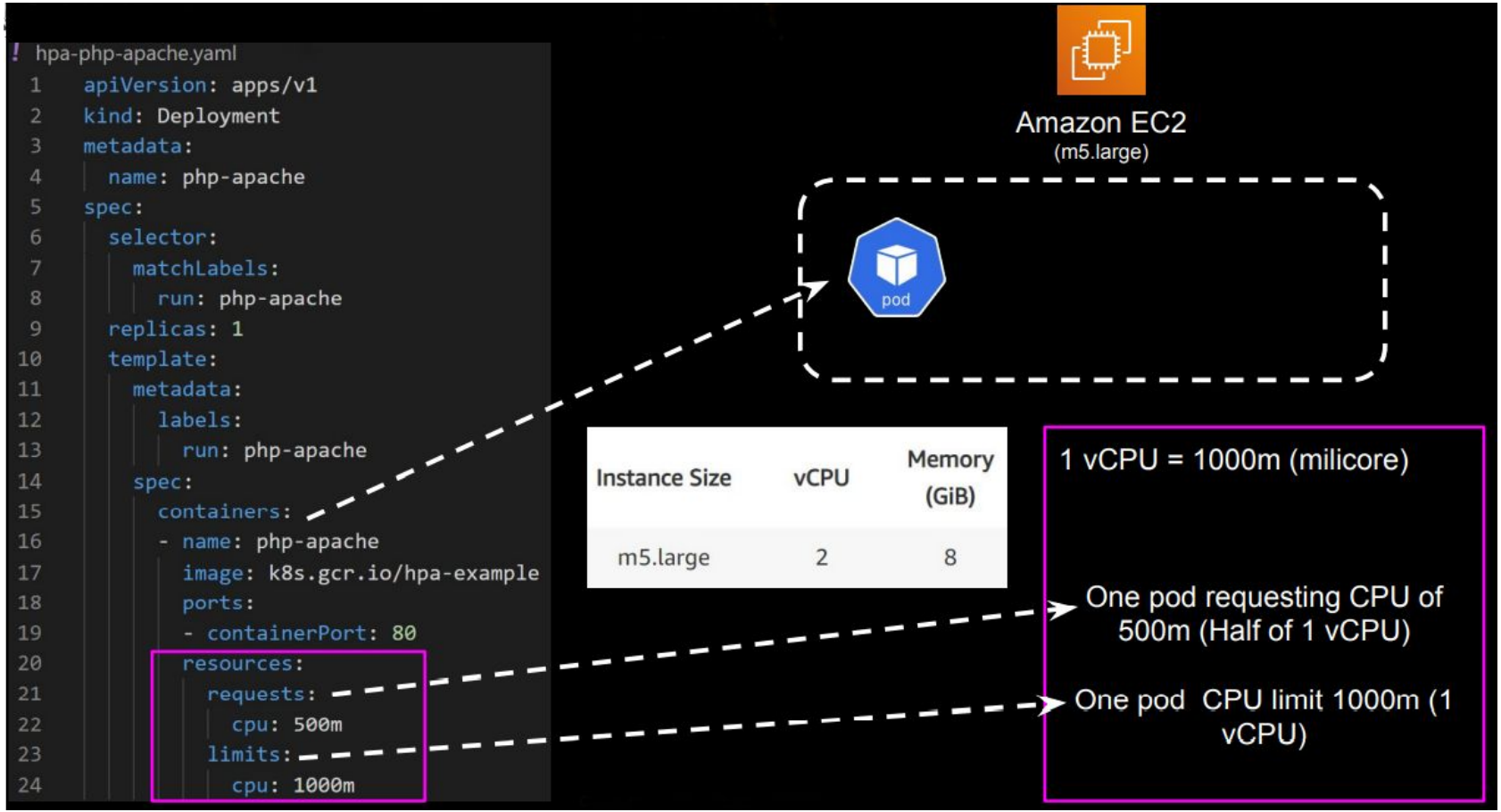
Horizontal Pod Autoscaler (HPA)



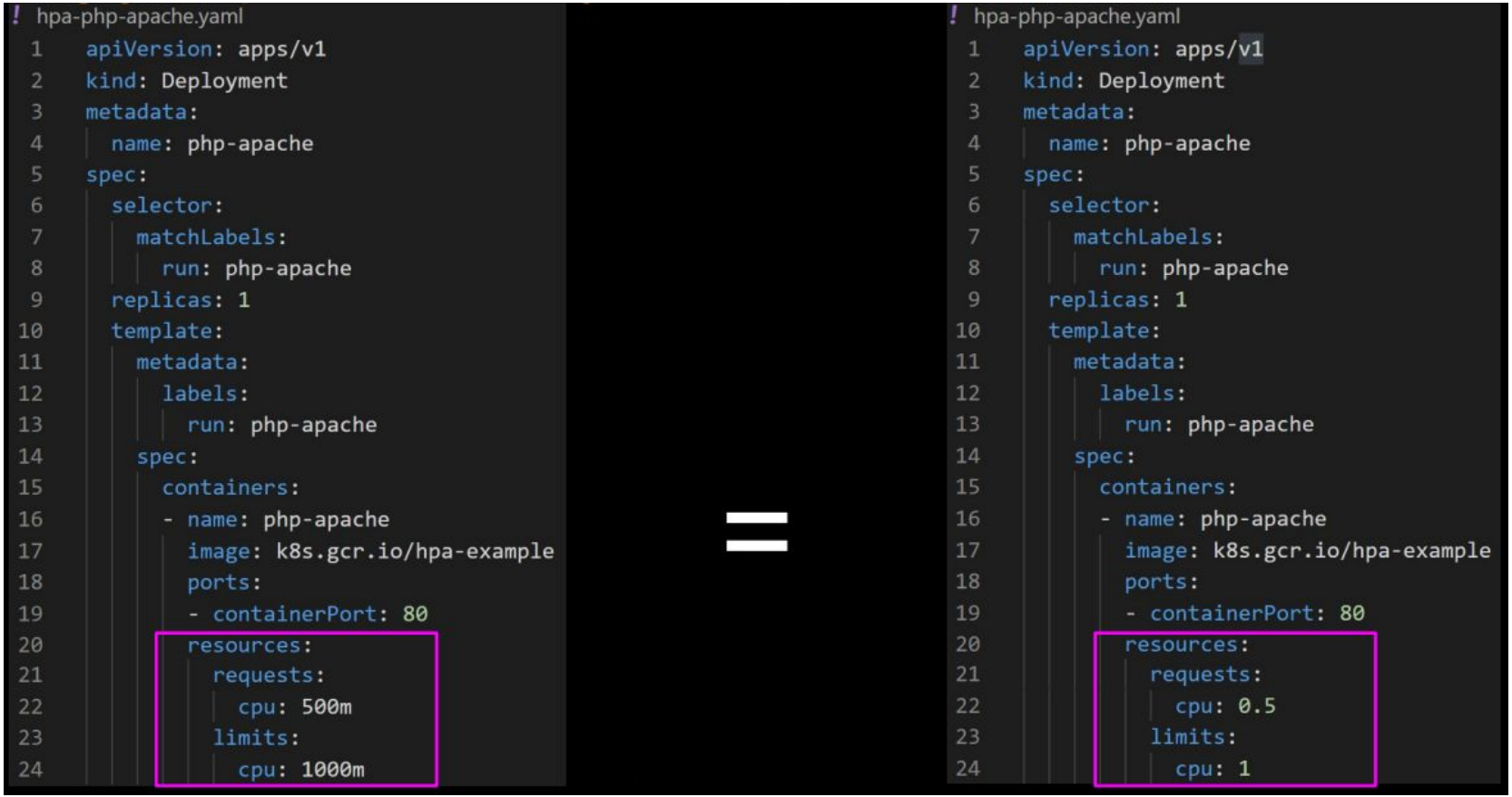
Pod Request And Limit



Pod Request And Limit



Pod Request And Limit



Pod Request And Limit

hpa-cpu-and-memory-php-apache.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: php-apache
spec:
  selector:
    matchLabels:
      run: php-apache
  replicas: 1
  template:
    metadata:
      labels:
        run: php-apache
    spec:
      containers:
      - name: php-apache
        image: k8s.gcr.io/hpa-example
        ports:
        - containerPort: 80
        resources:
          requests:
            cpu: 500m
            memory: 256Mi
          limits:
            cpu: 1000m
            memory: 512Mi
```

Amazon EC2
(m5.large)

Instance Size	vCPU	Memory (GiB)
m5.large	2	8

1 vCPU = 1000m (milicore)

One pod requesting CPU of 500m (Half of 1 vCPU) and 256 MiB of Memory

One pod CPU limit 1000m (1 vCPU) and 512 MiB of Memory

Pod Request And Limit + HPA

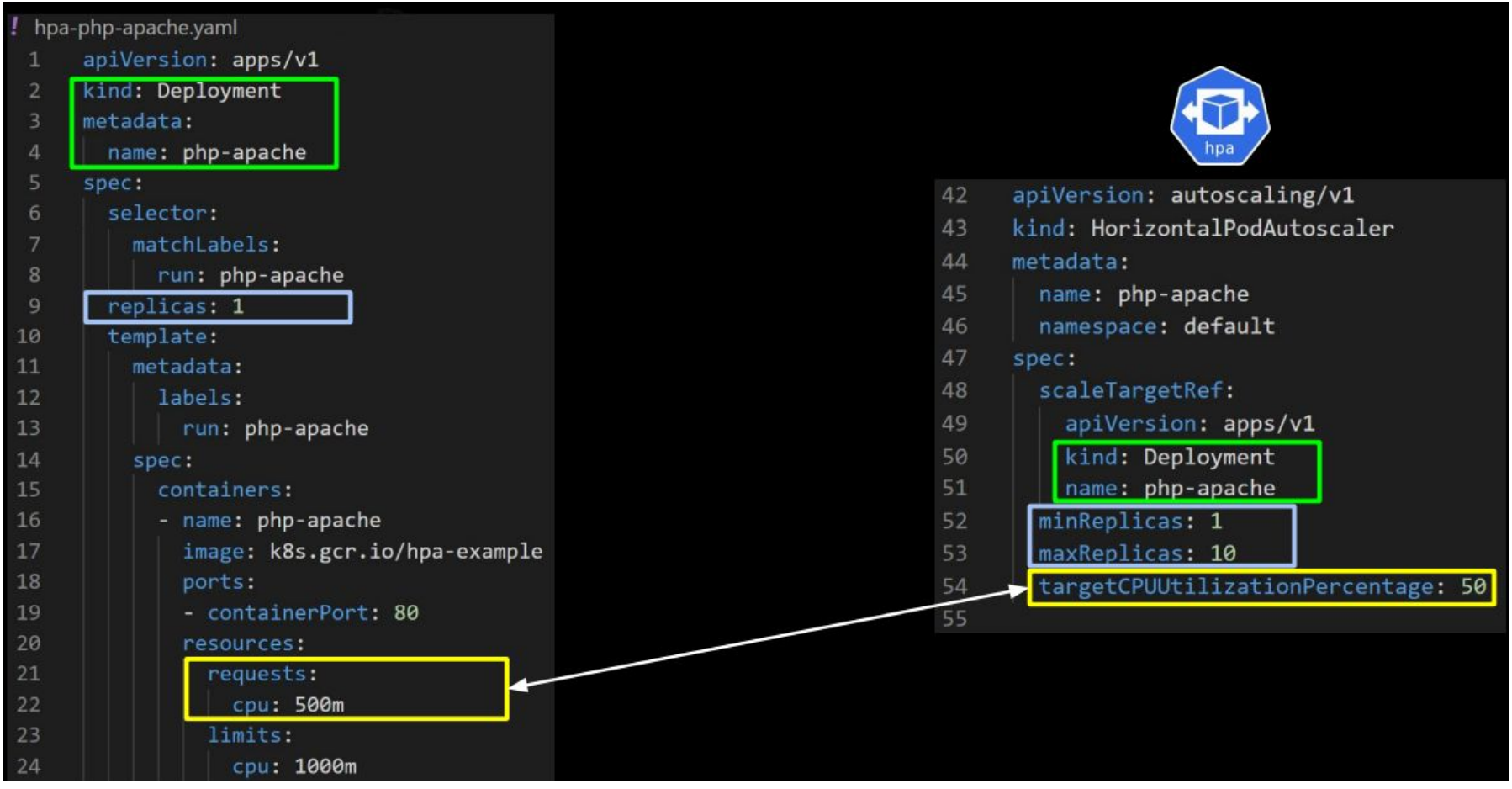
hpa-php-apache.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: php-apache
spec:
  selector:
    matchLabels:
      run: php-apache
  replicas: 1
  template:
    metadata:
      labels:
        run: php-apache
    spec:
      containers:
      - name: php-apache
        image: k8s.gcr.io/hpa-example
        ports:
        - containerPort: 80
        resources:
          requests:
            cpu: 500m
          limits:
            cpu: 1000m
```

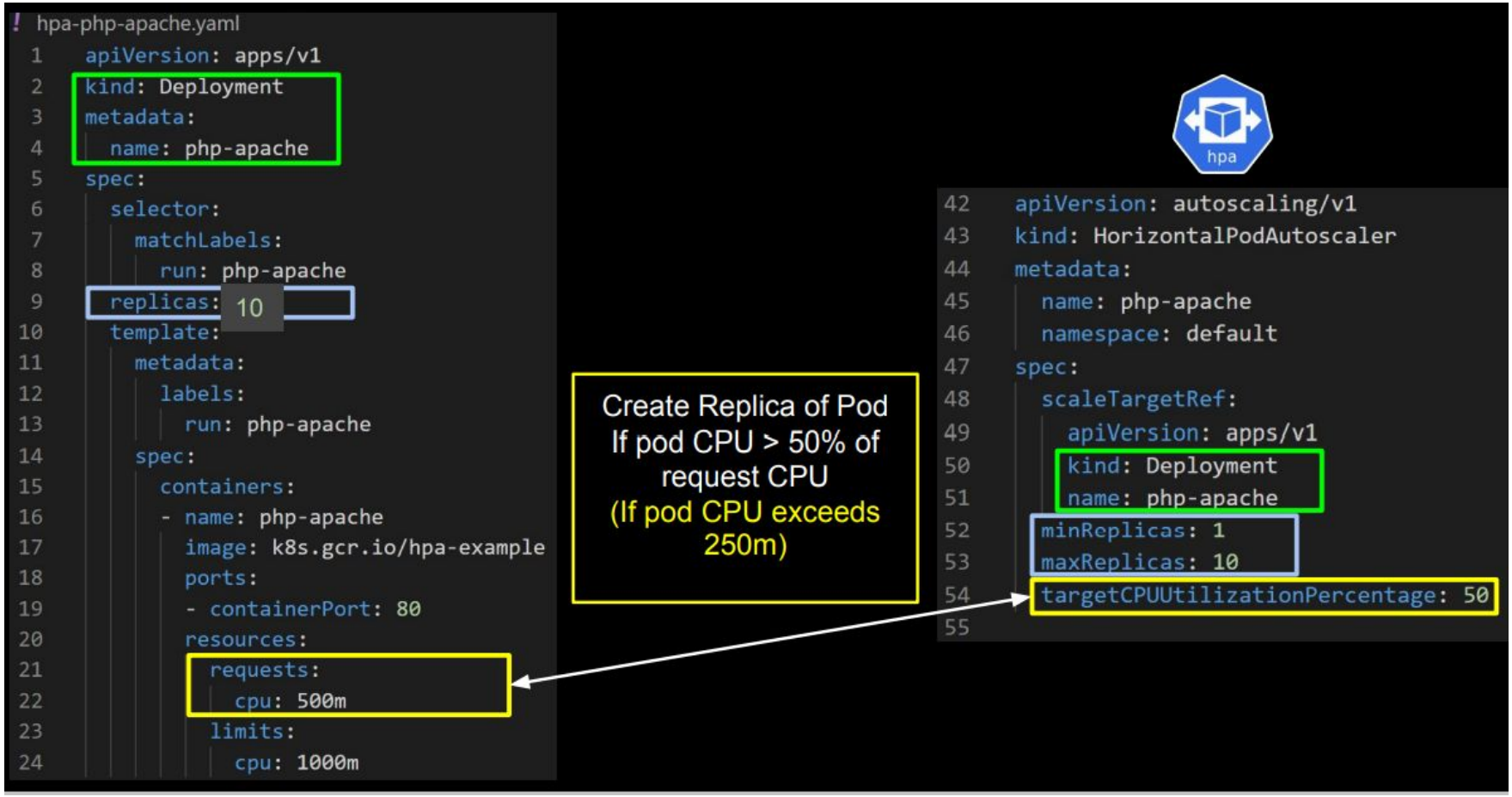
Create Replica of Pod
If pod CPU > 50% of request CPU
(If pod CPU exceeds 250m)


```yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
  namespace: default
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: php-apache
  minReplicas: 1
  maxReplicas: 10
  targetCPUUtilizationPercentage: 50
```

K8s HPA YAML


hpa-php-apache.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
name: php-apache
spec:
selector:
matchLabels:
run: php-apache
replicas: 1
template:
metadata:
labels:
run: php-apache
spec:
containers:
- name: php-apache
image: k8s.gcr.io/hpa-example
ports:
- containerPort: 80
resources:
requests:
cpu: 500m
limits:
cpu: 1000m
hpa
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
name: php-apache
namespace: default
spec:
scaleTargetRef:
apiVersion: apps/v1
kind: Deployment
name: php-apache
minReplicas: 1
maxReplicas: 10
targetCPUUtilizationPercentage: 50

Pod Request And Limit



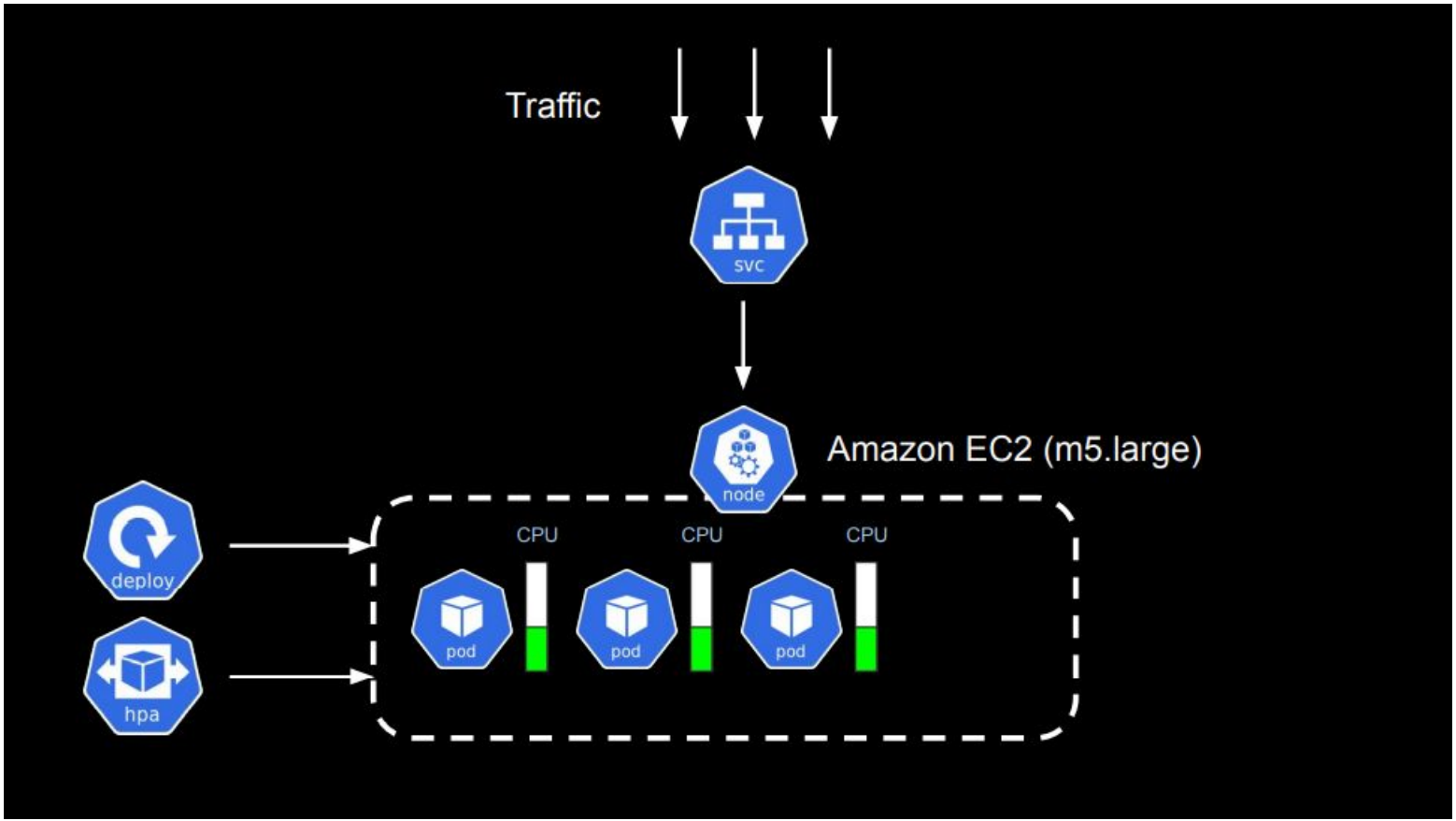
HPA Demo on EKS

- Install Metrics Server
- Deploy Deployment, Service, HPA
- Increase Load
- Pod scale!

Note - Need node larger than t3.micro

https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough

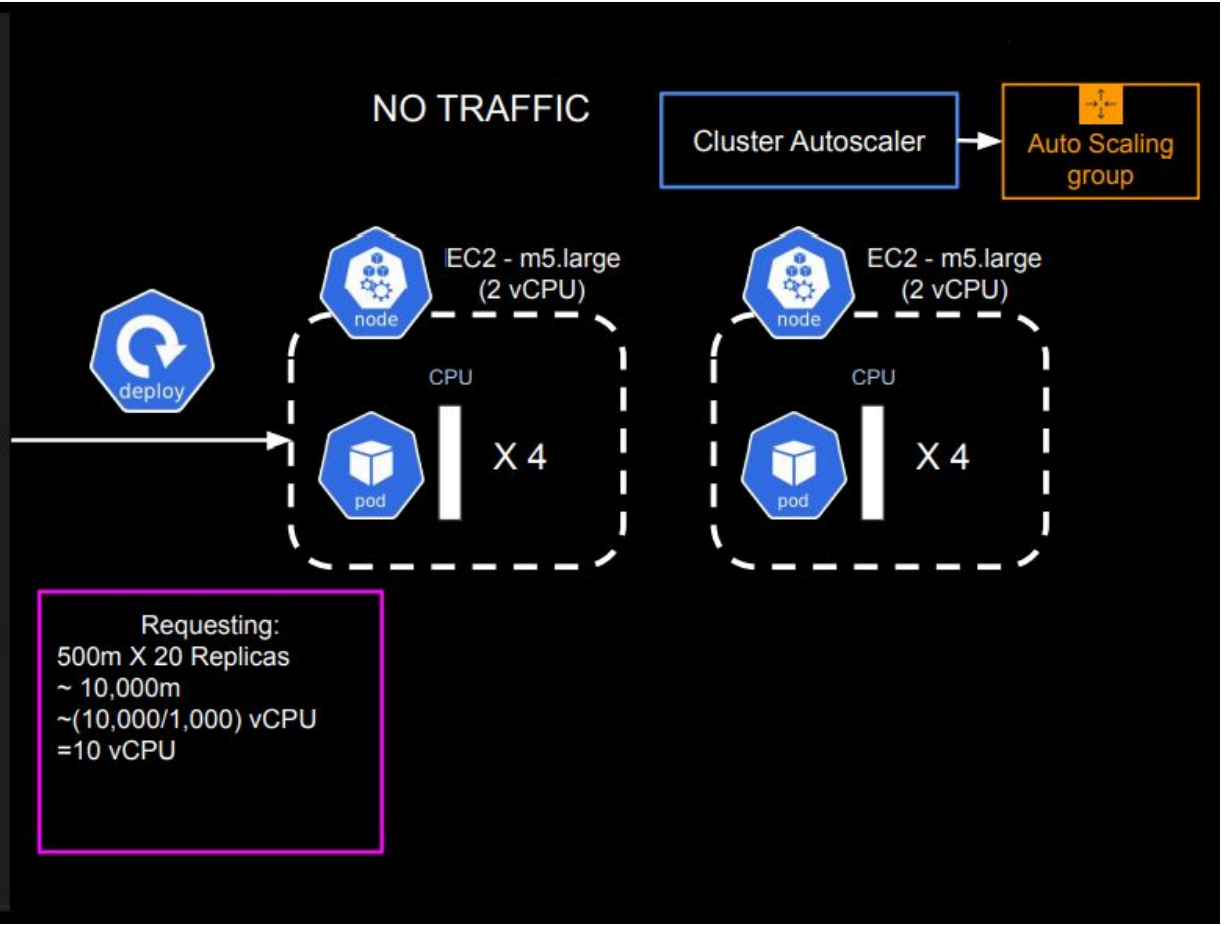
Horizontal Pod Autoscaler (HPA)



Cluster Autoscaler

cluster-autoscaler-deployment-1.yaml

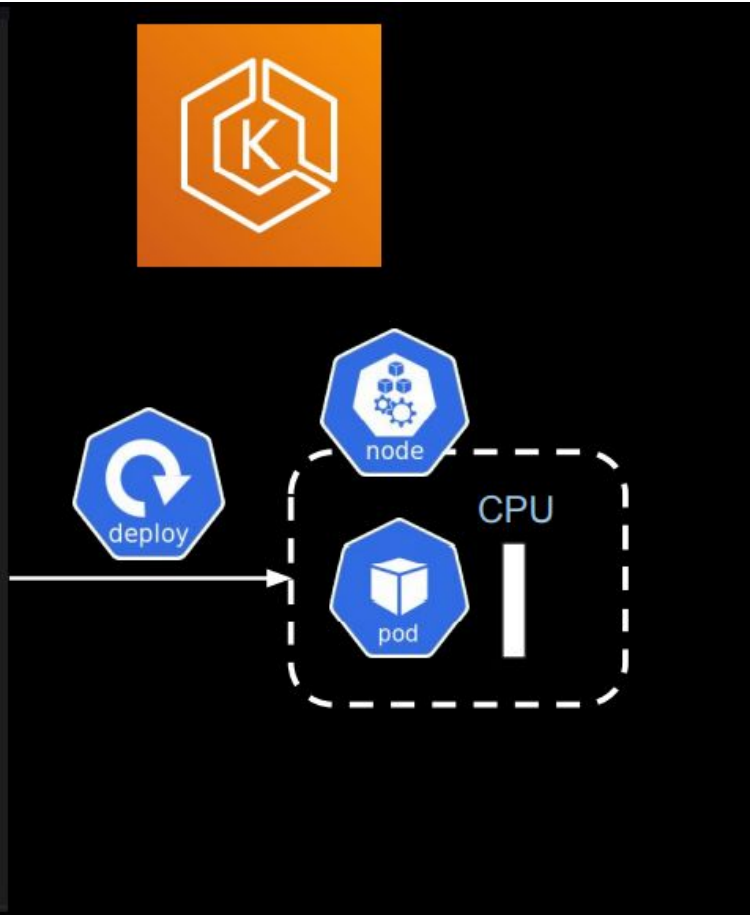
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: php-apache
spec:
  selector:
    matchLabels:
      run: php-apache
  replicas: 20
  template:
    metadata:
      labels:
        run: php-apache
    spec:
      containers:
      - name: php-apache
        image: k8s.gcr.io/hpa-example
        ports:
        - containerPort: 80
        resources:
          requests:
            cpu: 500m
          limits:
            cpu: 1000m
```



CLUSTER AUTOSCALER

cluster-autoscaler-deployment-1.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: php-apache
spec:
  selector:
    matchLabels:
      run: php-apache
  replicas: 20
  template:
    metadata:
      labels:
        run: php-apache
    spec:
      containers:
      - name: php-apache
        image: k8s.gcr.io/hpa-example
        ports:
        - containerPort: 80
        resources:
          requests:
            cpu: 500m
          limits:
            cpu: 1000m
```



Vertical Pod Autoscaler

Do NOT use this in Production!

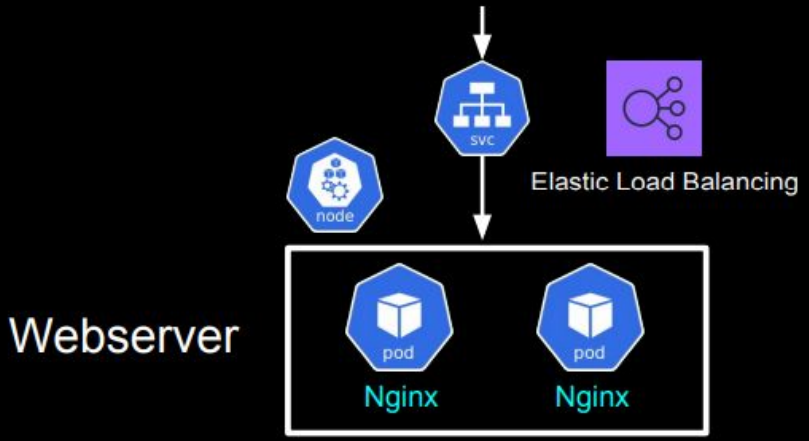
- Restarts PODs

Vertical Pod Autoscaler

- Vertical Vs Horizontal Scaling
  - Vertical - Going up in Size
  - Horizontal - Create more of same size
- Pods will go up or down in size (after restart!)
- Used in dev to determine optimal CPU and memory for the Pod
  - VPA recommends pod request, limit
  - Use the numbers for request, limits, HPA
  - VPA should not be used with HPA
- Accept VPA recommendation with grain of salt!

EKS Ingress

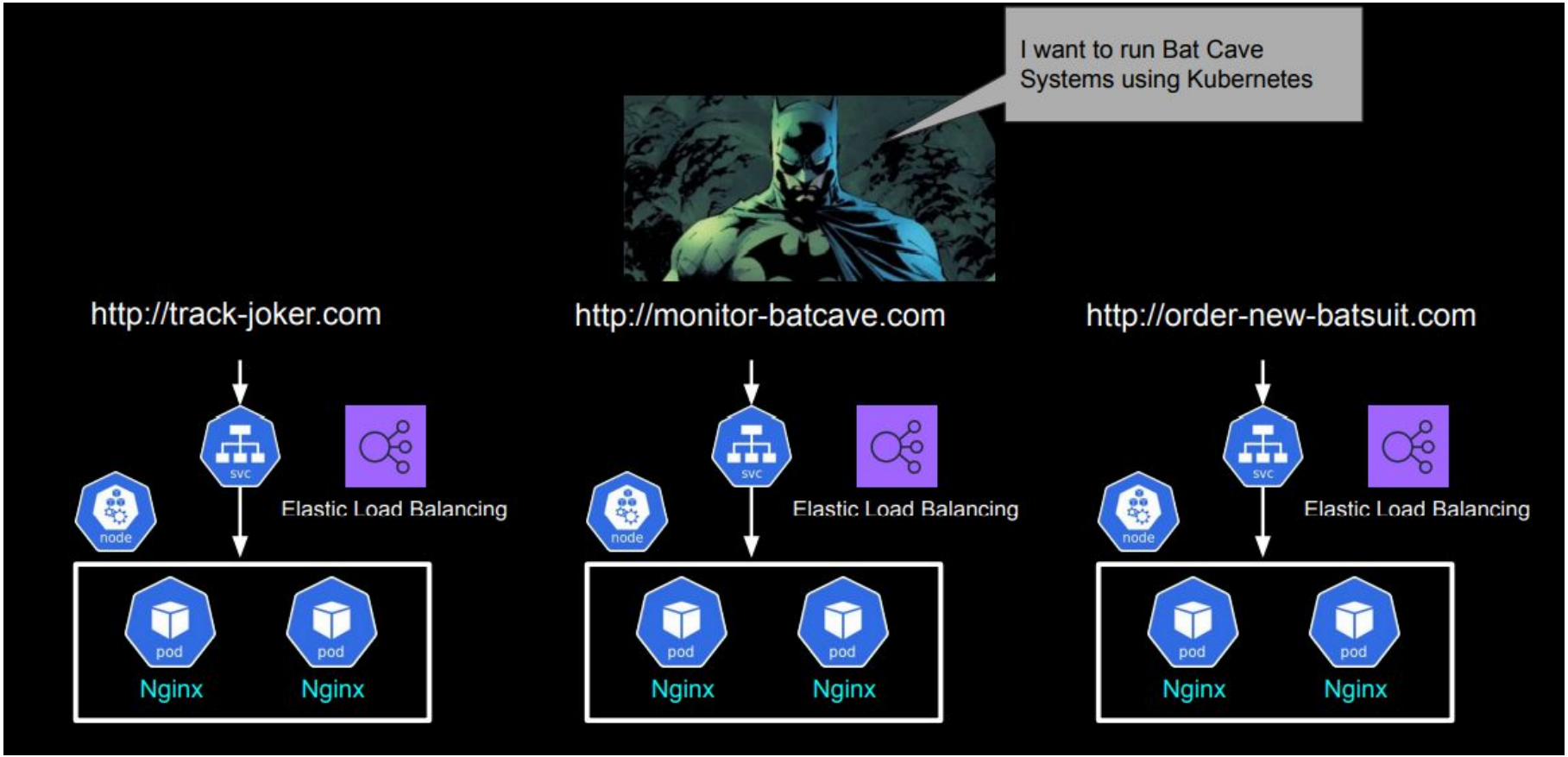
Ingress - What and Why?



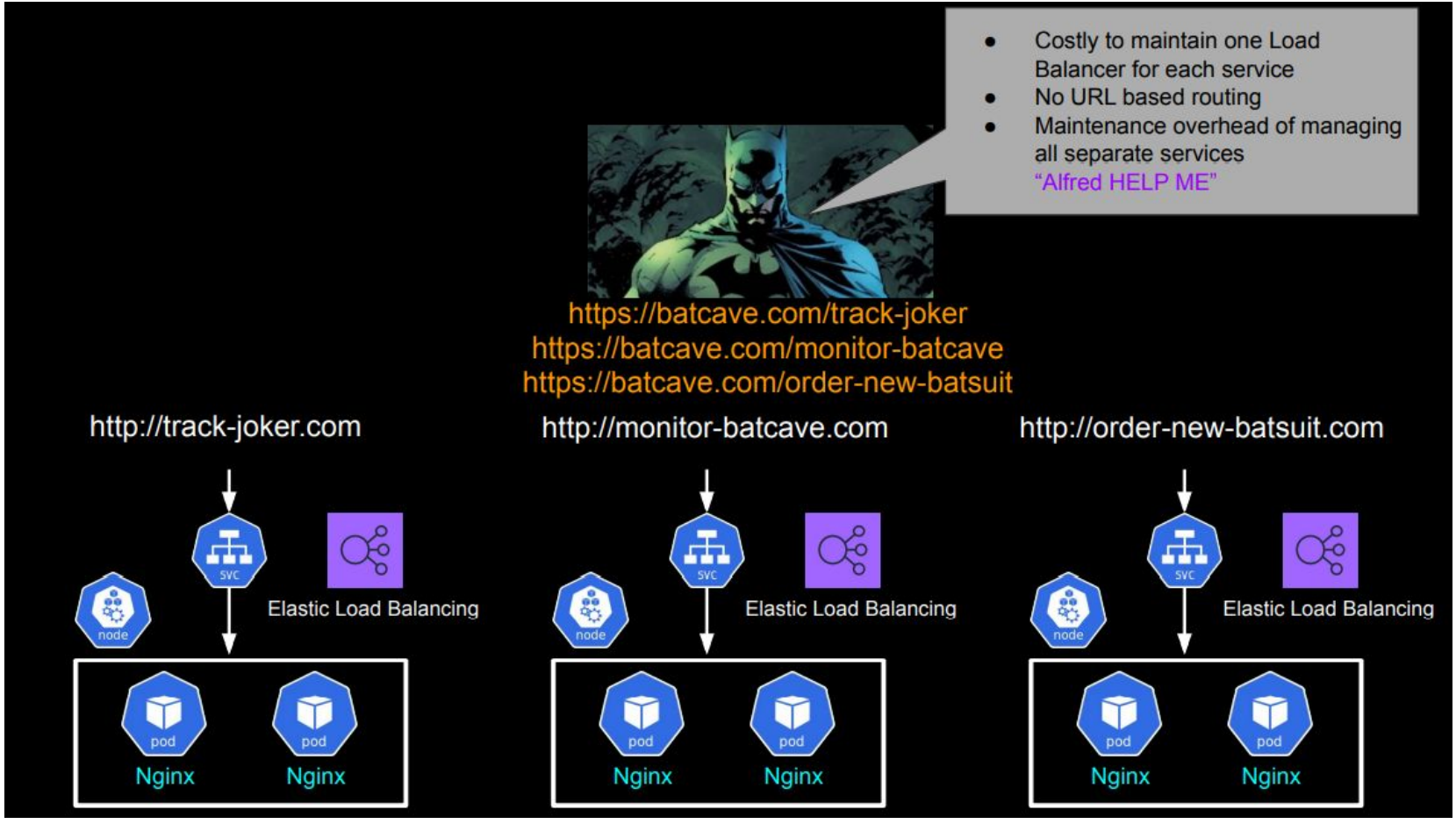
loadbalancer-service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: lb-service
  labels:
    app: lb-service
spec:
  type: LoadBalancer
  ports:
  - port: 80
  selector:
    app: frontend
```

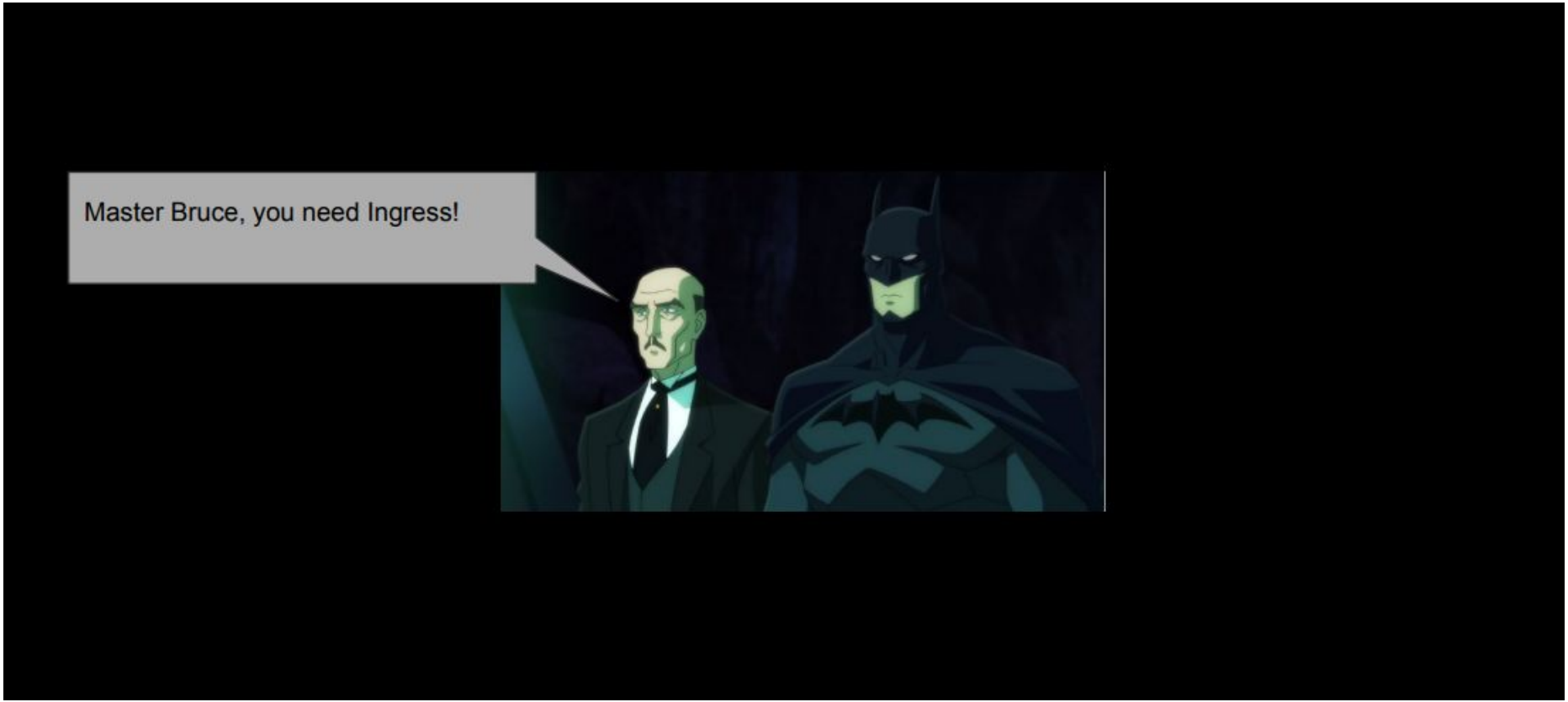
Ingress - What and Why?



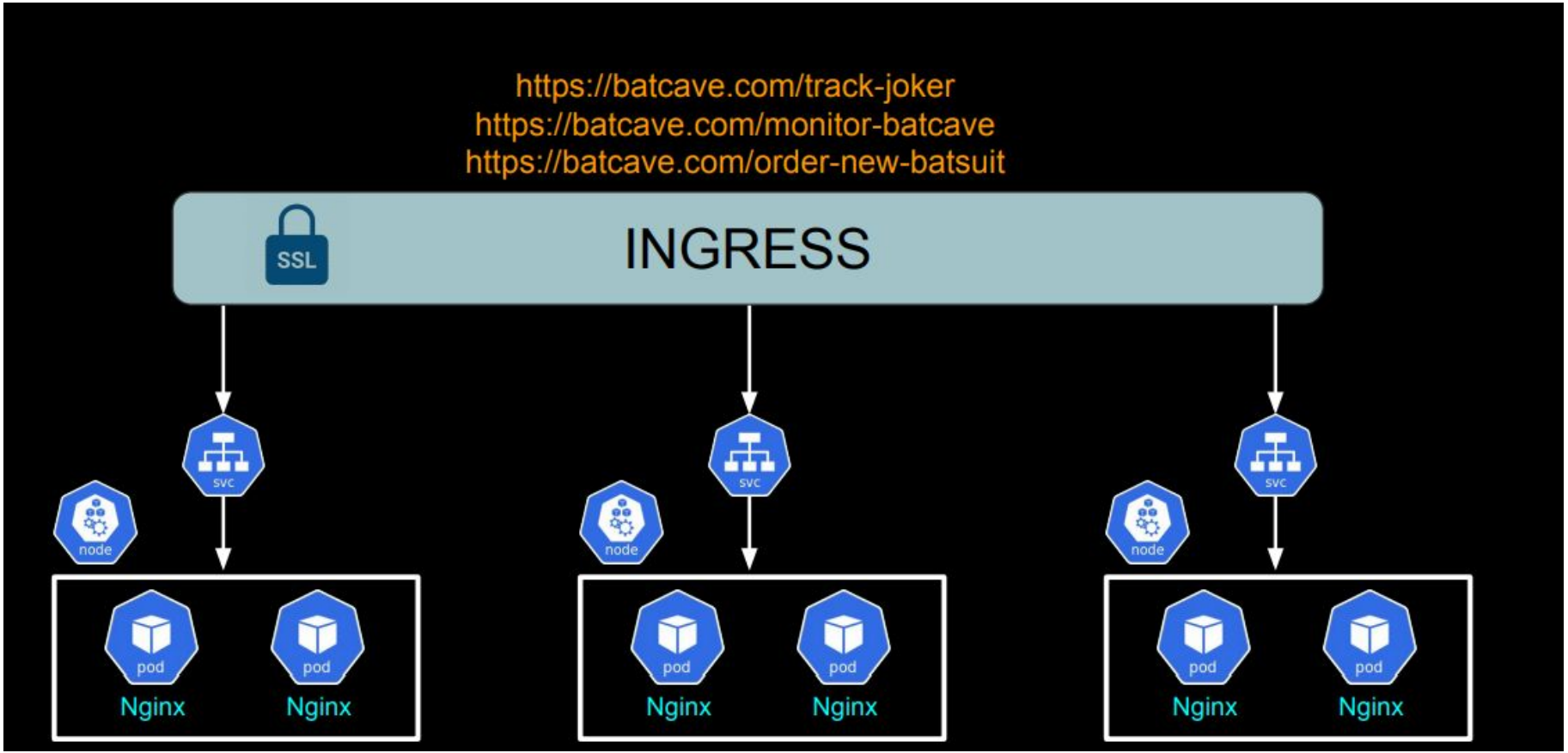
Ingress - What and Why?



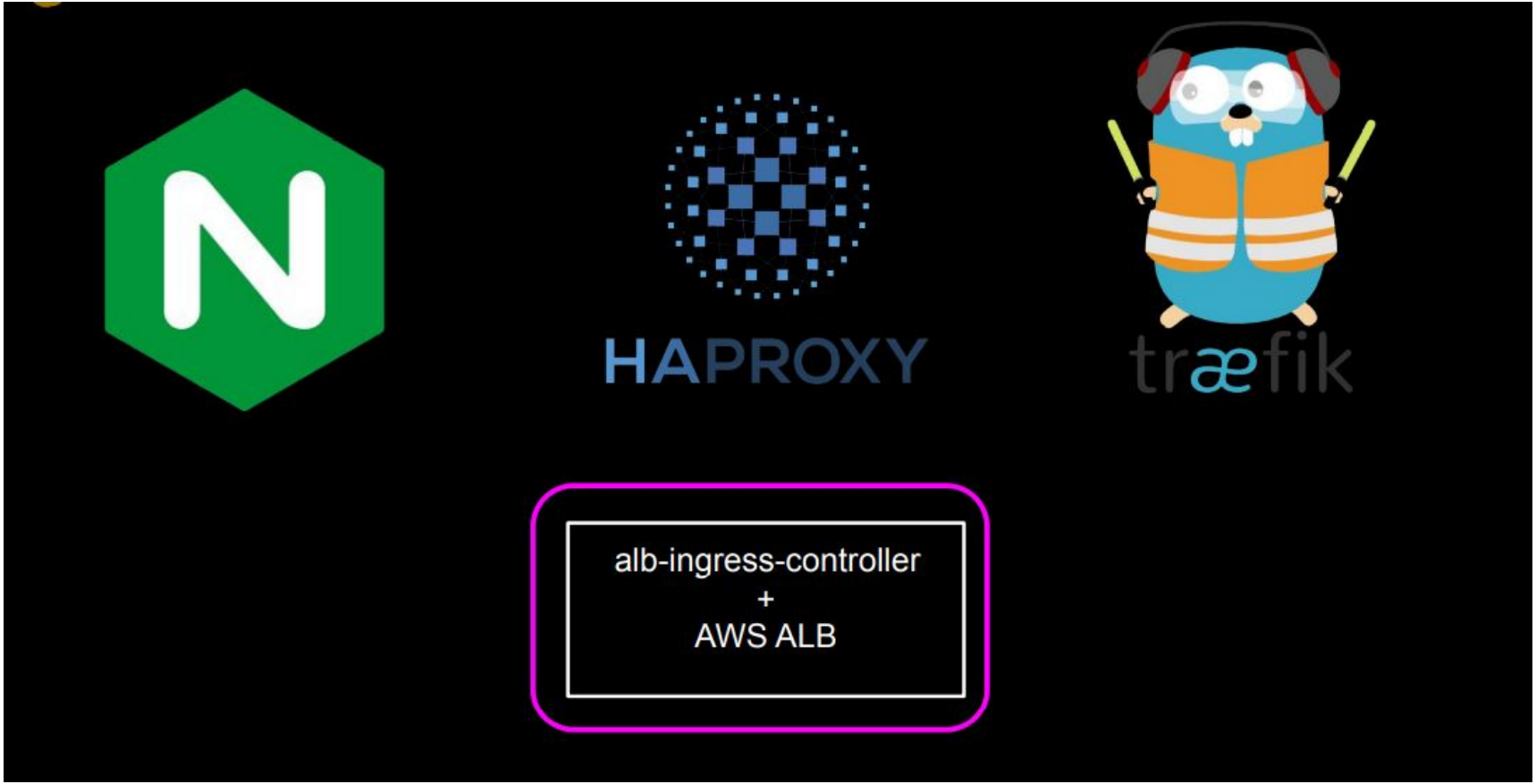
Ingress - What and Why?



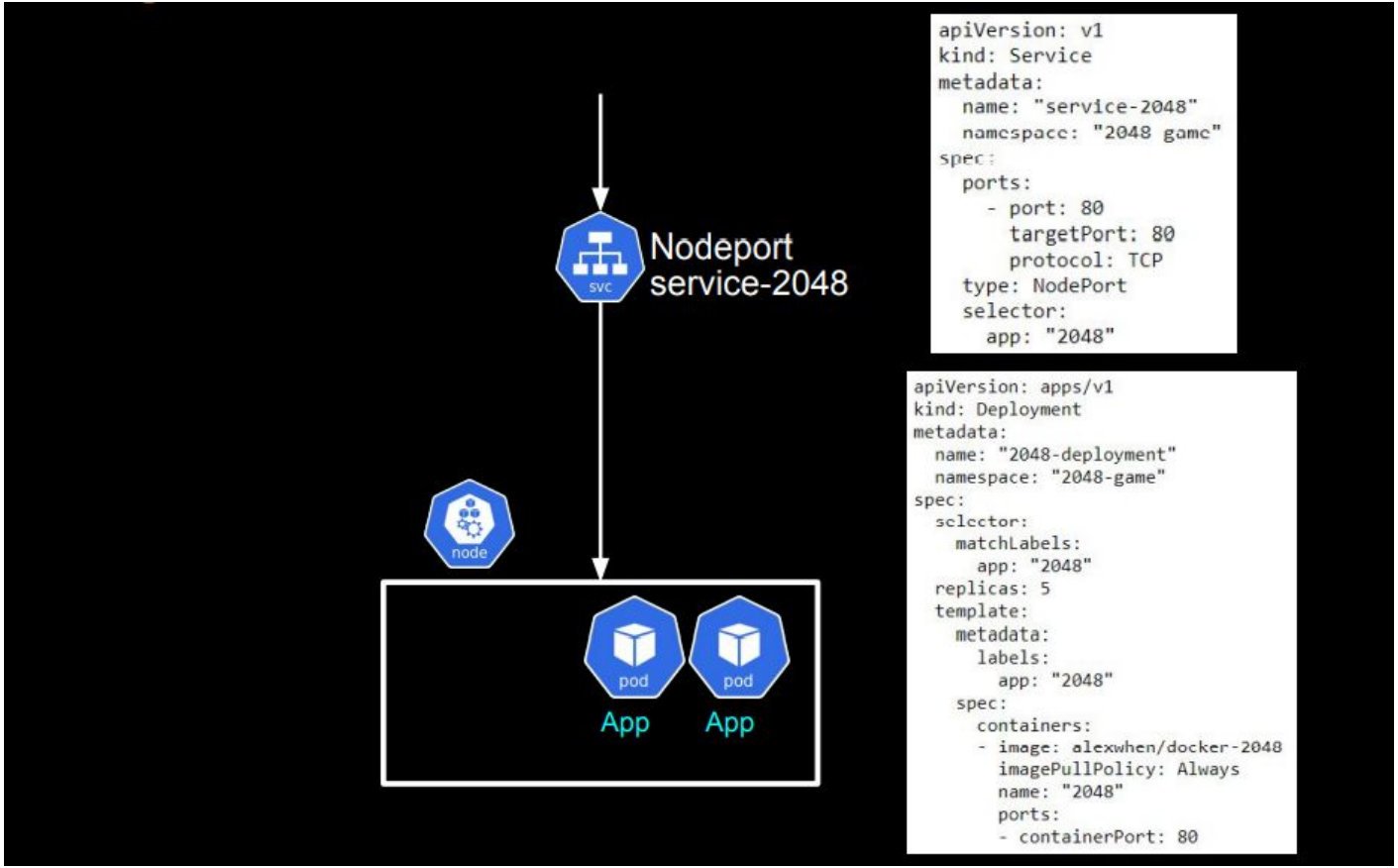
Ingress - What and Why?



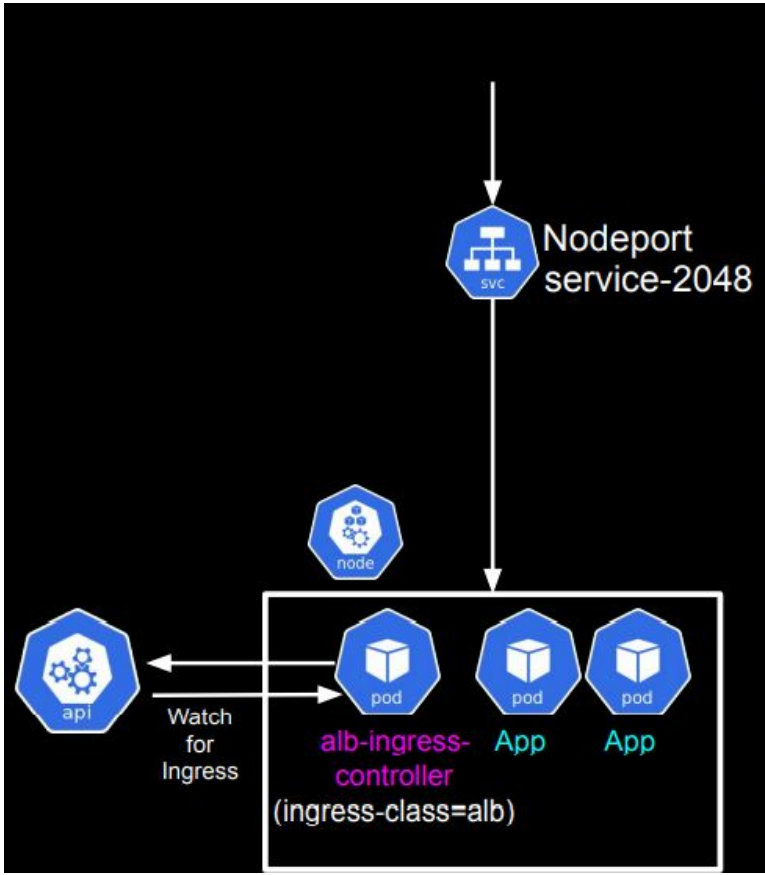
Ingress Controllers



ALB Ingress

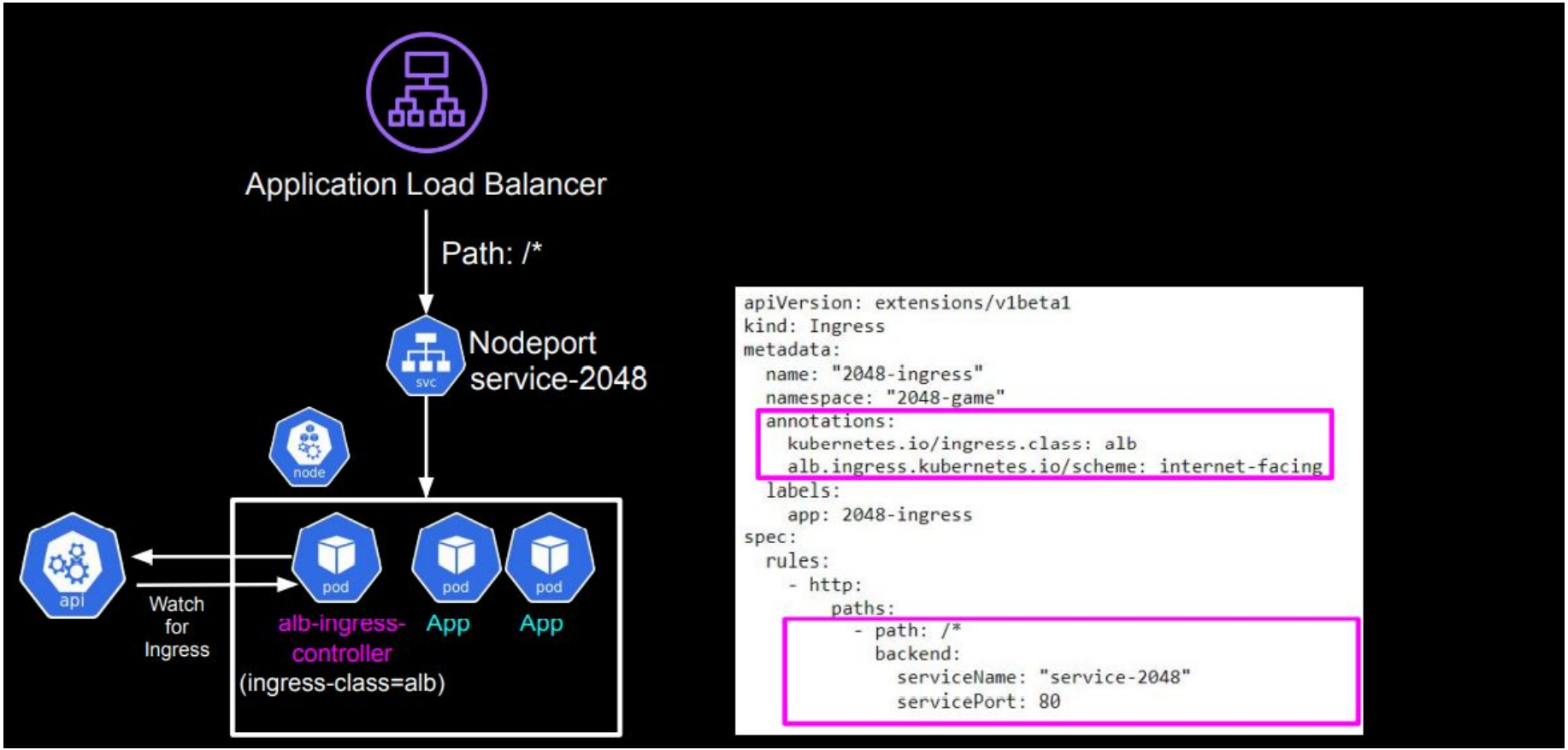

svc
Nodeport
service-2048
node
pod
pod
App
App
apiVersion: v1
kind: Service
metadata:
name: "service-2048"
namespace: "2048 game"
spec:
ports:
- port: 80
targetPort: 80
protocol: TCP
type: NodePort
selector:
app: "2048"
apiVersion: apps/v1
kind: Deployment
metadata:
name: "2048-deployment"
namespace: "2048-game"
spec:
selector:
matchLabels:
app: "2048"
replicas: 5
template:
metadata:
labels:
app: "2048"
spec:
containers:
- image: alexwhen/docker-2048
imagePullPolicy: Always
name: "2048"
ports:
- containerPort: 80

ALB Ingress Controller



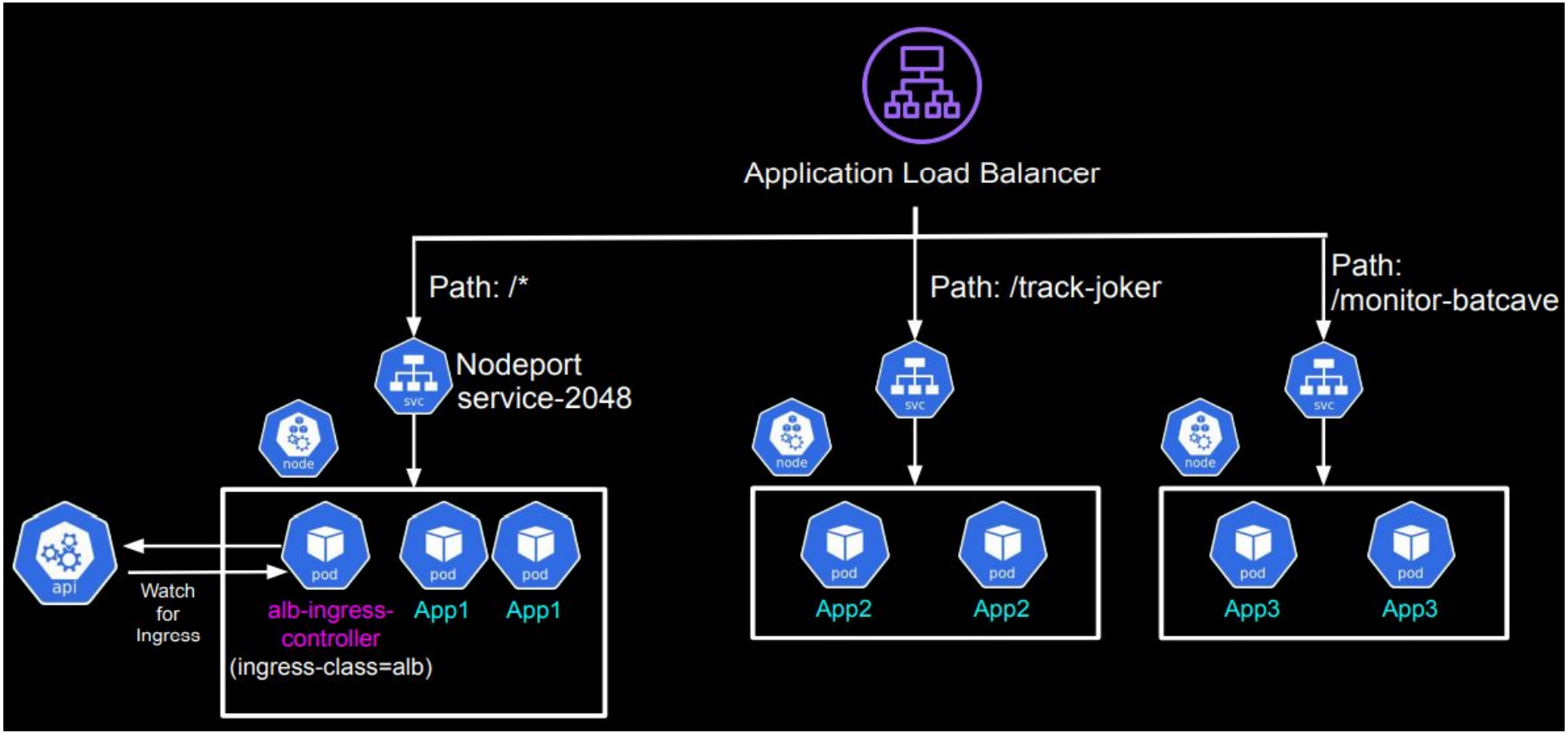

```
# Application Load Balancer (ALB) Ingress Controller Deployment Manifest.
# This manifest details sensible defaults for deploying an ALB Ingress Controller.
# GitHub: https://github.com/kubernetes-sigs/aws-alb-ingress-controller
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app.kubernetes.io/name: alb-ingress-controller
  name: alb-ingress-controller
  # Namespace the ALB Ingress Controller should run in. Does not impact which
  # namespaces it's able to resolve ingress resource for. For limiting ingress
  # namespace scope, see --watch-namespace.
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: alb-ingress-controller
  template:
    metadata:
      labels:
        app.kubernetes.io/name: alb-ingress-controller
    spec:
      containers:
        - name: alb-ingress-controller
          args:
            # Limit the namespace where this ALB Ingress Controller deployment will
            # resolve ingress resources. If left commented, all namespaces are used.
            # - --watch-namespace=your-k8s-namespace

            # Setting the ingress-class flag below ensures that only ingress resources with the
            # annotation kubernetes.io/ingress.class: "alb" are respected by the controller. You may
            # choose any class you'd like for this controller to respect.
            - --ingress-class=alb
```

Ingress resource




```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: "2048-ingress"
  namespace: "2048-game"
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/scheme: internet-facing
  labels:
    app: 2048-ingress
spec:
  rules:
    - http:
        paths:
          - path: /*
            backend:
              serviceName: "service-2048"
              servicePort: 80
```

ALB Ingress


Application Load Balancer
Path: /*
Path: /track-joker
Path: /monitor-batcave
Nodeport service-2048
svc
node
pod
api
Watch for Ingress
alb-ingress-controller
(ingress-class=alb)
App1
App2
App3

Ingress - What and Why?

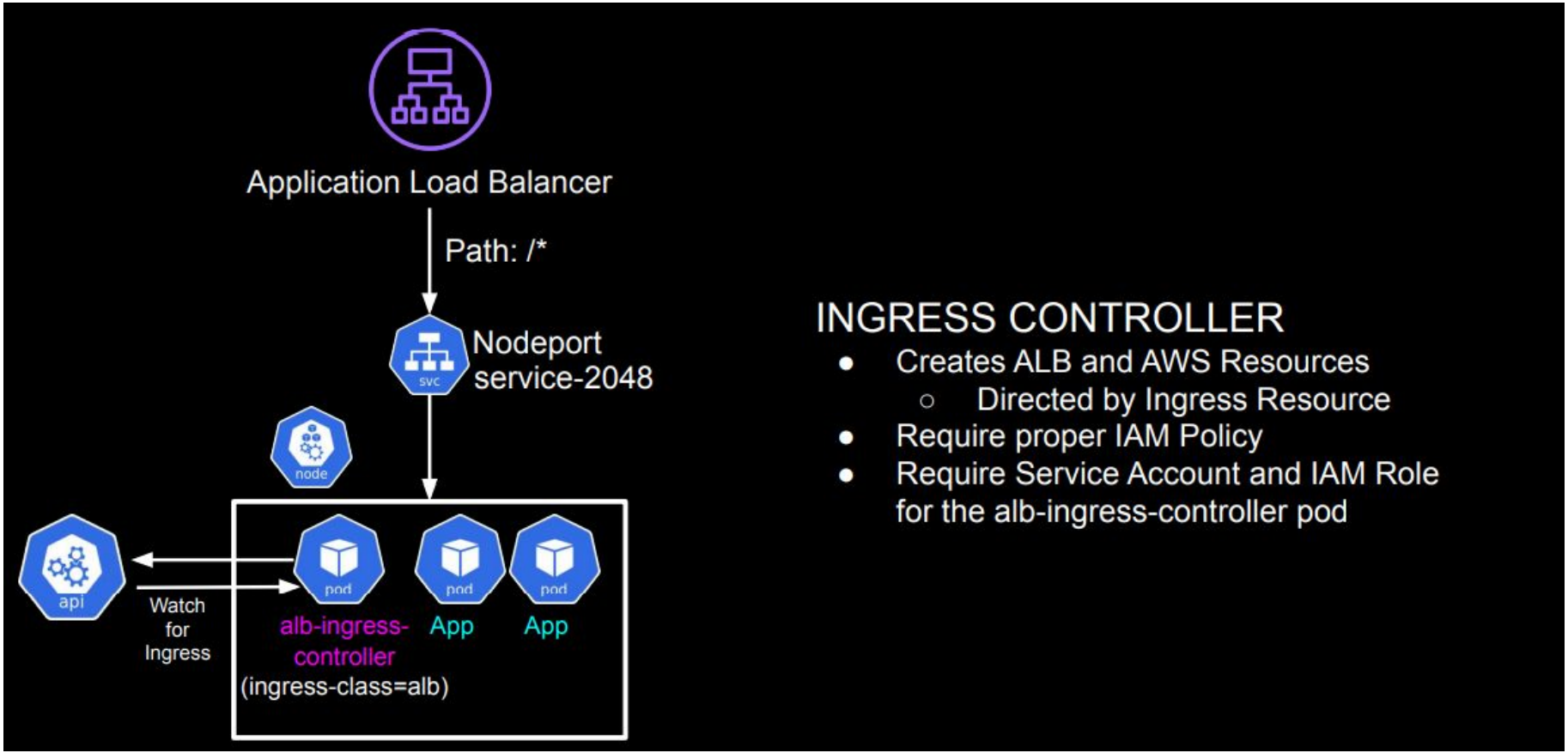
INGRESS CONTROLLER

- Monitors Ingress resources
- Creates necessary AWS resources for Ingress
  - Such as ALB for ALB Ingress Controller
- One Cluster can have more than one Ingress Controller!
  - Ingress Resource defines which Ingress Controller to use

INGRESS RESOURCE

- Selects which Ingress Controller to use
- Defines the URL Path and corresponding backend Service

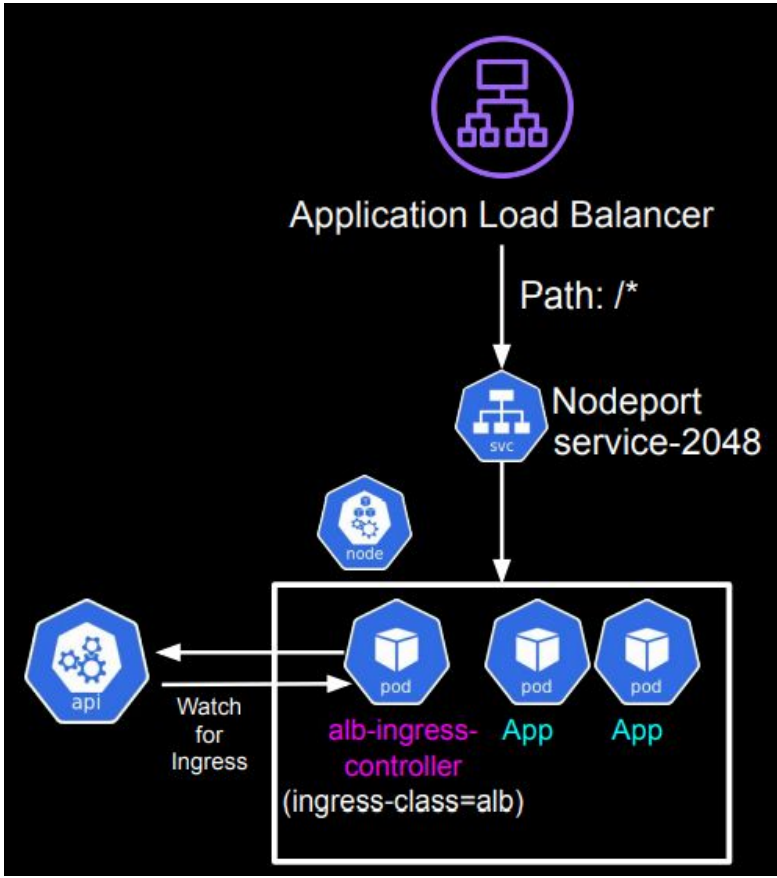
ALB Ingress



INGRESS CONTROLLER

- Creates ALB and AWS Resources
  - Directed by Ingress Resource
- Require proper IAM Policy
- Require Service Account and IAM Role for the alb-ingress-controller pod

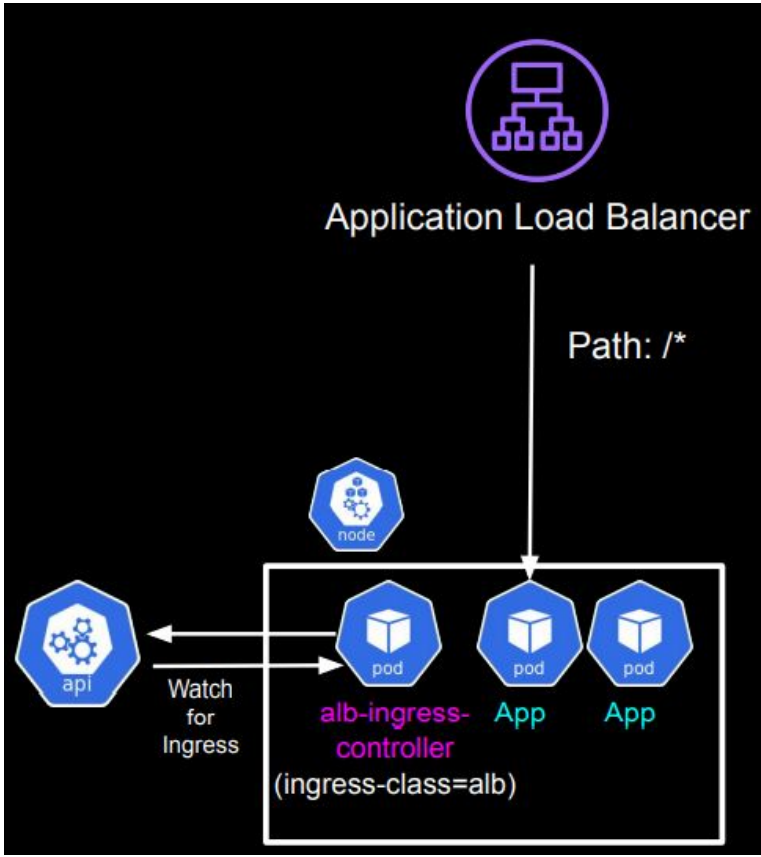
ALB Ingress



INGRESS TRAFFIC

- instance mode
  - ALB to Nodeport to pods
- ip mode
  - ALB to pods directly
  - Require secondary IP address on ENI as pode IP for networking plugin (AWS CNI plugin for kubernetes)
  - Saves one hop
  - alb-ingress-controller specific feature

ALB Ingress



INGRESS TRAFFIC

- instance mode
  - ALB to Nodeport to pods
- ip mode
  - ALB to pods directly
  - Require secondary IP address on ENI as pode IP for networking plugin (AWS CNI plugin for kubernetes)
  - Saves one hop
  - alb-ingress-controller specific feature

Thank You