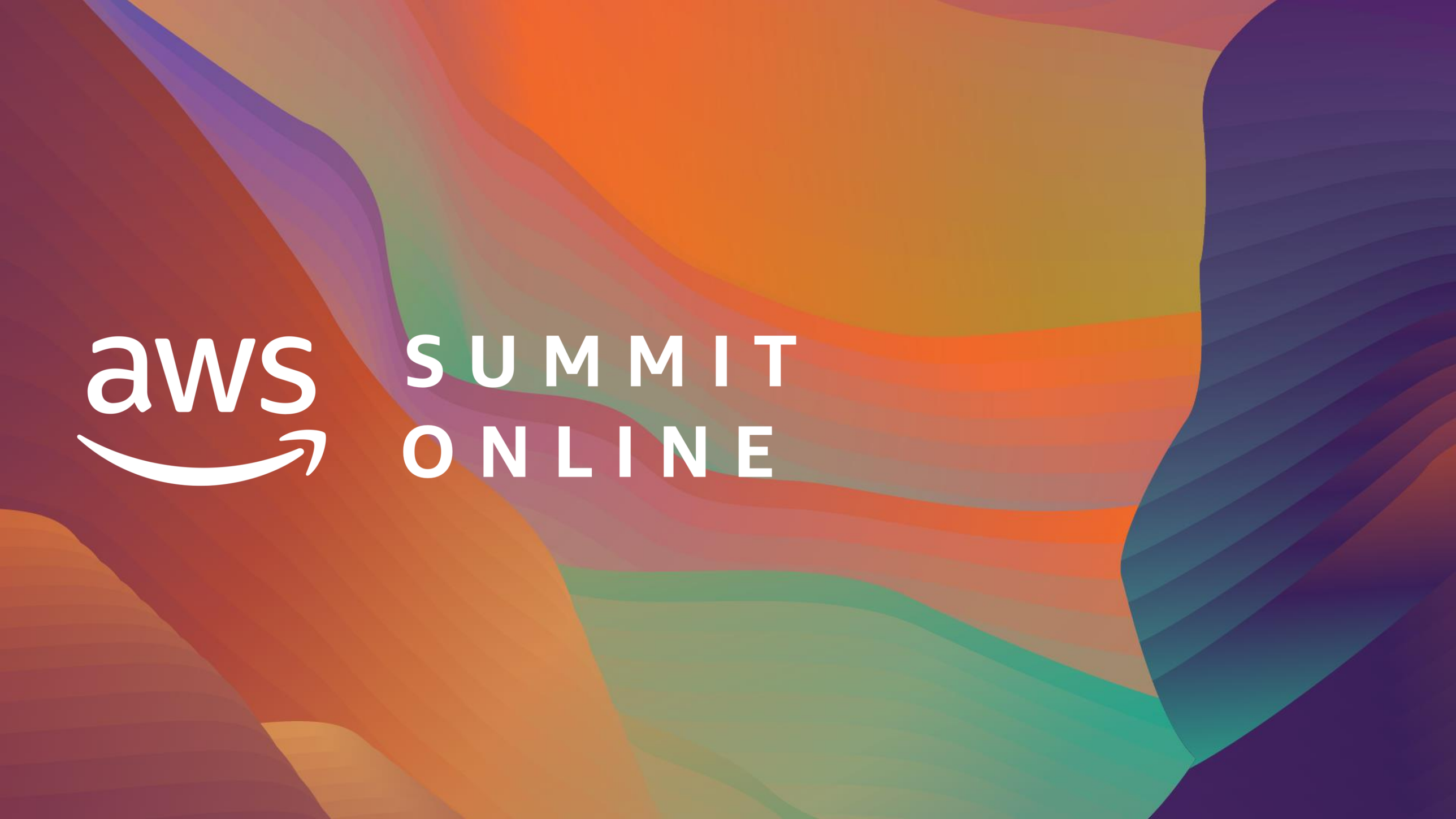

aws
SUMMIT
ONLINE

OPE09

The art of successful Kubernetes failures

Mitch Beaumont

Principal Solutions Architect
Amazon Web Services

aws SUMMIT ONLINE

“Everything fails,
all the time"

Dr Werner Vogals, CTO Amazon.com

How failures can be successful?

Using data to improve quality and reliability of our systems

Reducing the blast radius

How failures can be successful?

Using data to improve quality and reliability of our systems

Reducing the blast radius

Reducing the time to detection

How failures can be successful?

Using data to improve quality and reliability of our systems

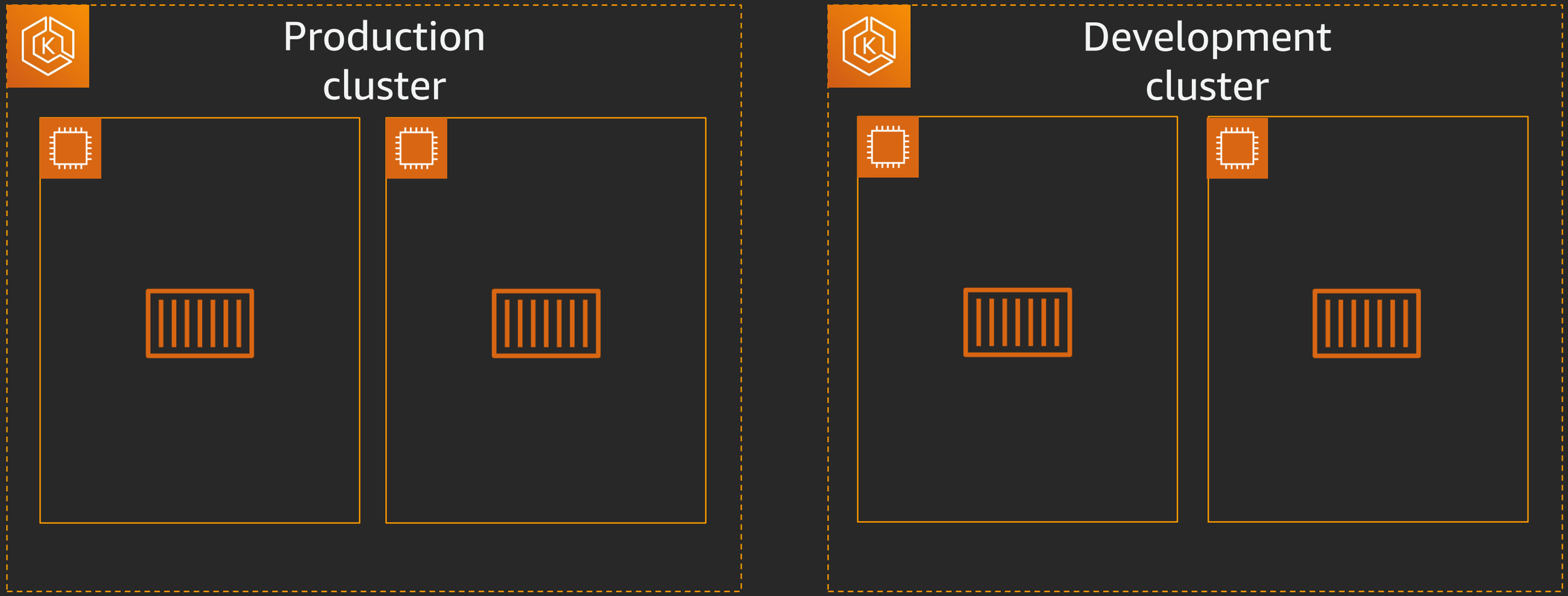
Reducing the blast radius

Reducing the time to detection

Reducing the time to mitigation

Reducing the blast radius

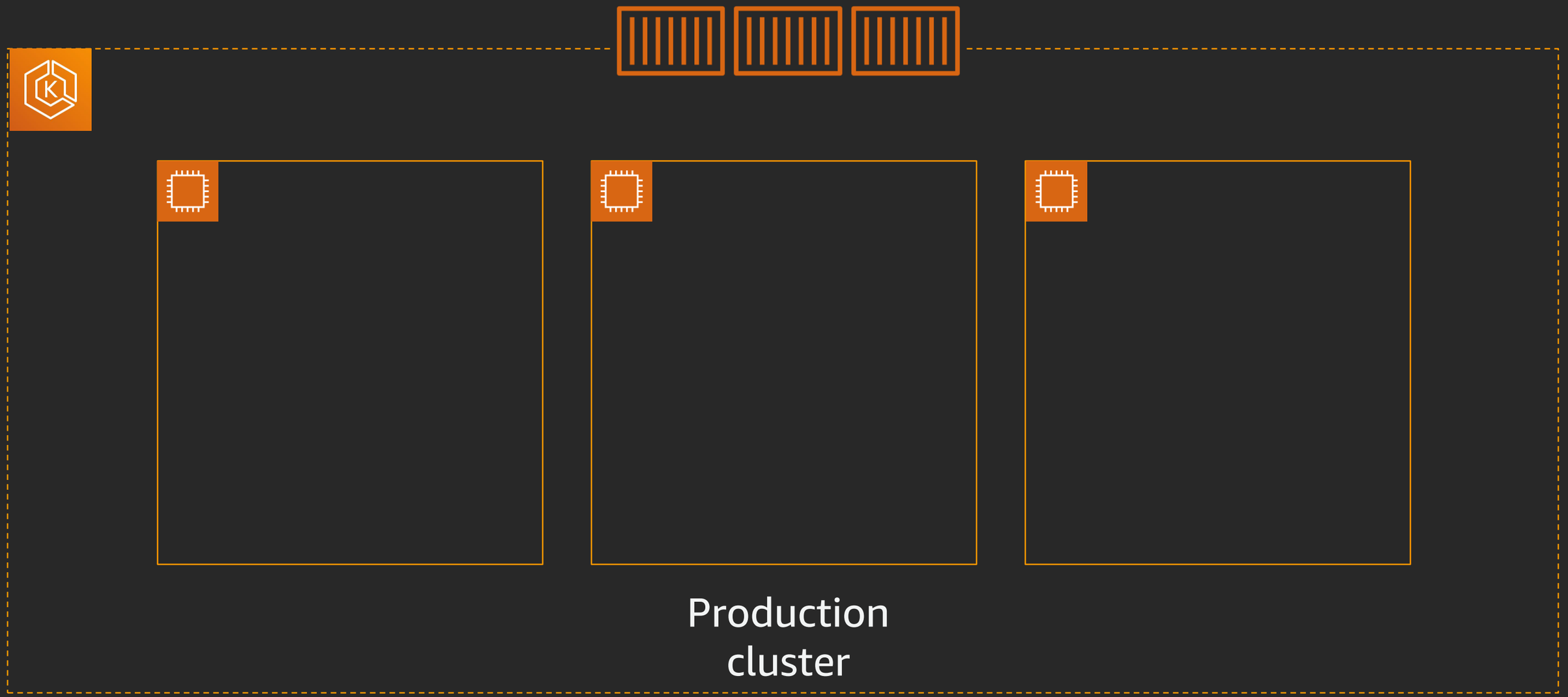
Cluster isolation



Cluster isolation

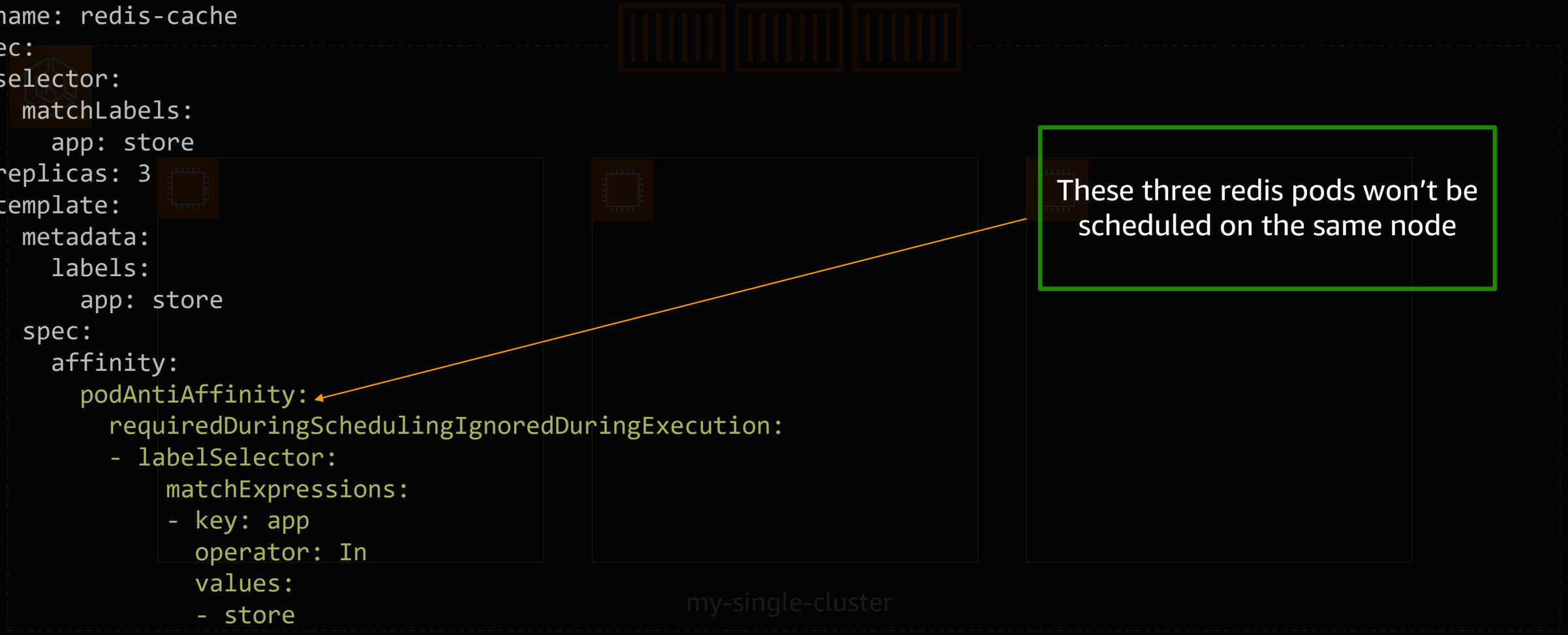
```
aws$ cat << EOF > cluster.yaml
apiVersion: eksctl.io/v1alpha5
 kind: ClusterConfig
 metadata:
  name: cloudwatch-cluster
   region: ap-southeast-2
 nodeGroups:
 - name: default
   instanceType: m5.large
   desiredCapacity: 3
   iam:
    withAddonPolicies:
     cloudWatch: true
     albIngress: true
     autoScaler: true
     appMesh: true
     xRay: true
 cloudWatch:
  clusterLogging:
   enableTypes: ["all"]

EOF

aws$ eksctl create cluster -f cluster.yaml
```

Scheduling (affinity and anti-affinity)



Scheduling (affinity and anti-affinity)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-cache
spec:
  selector:
    matchLabels:
      app: store
  replicas: 3
  template:
    metadata:
      labels:
        app: store
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - store
            topologyKey: "kubernetes.io/hostname"
      containers:
      - name: redis-server
        image: redis:3.2-alpine
```

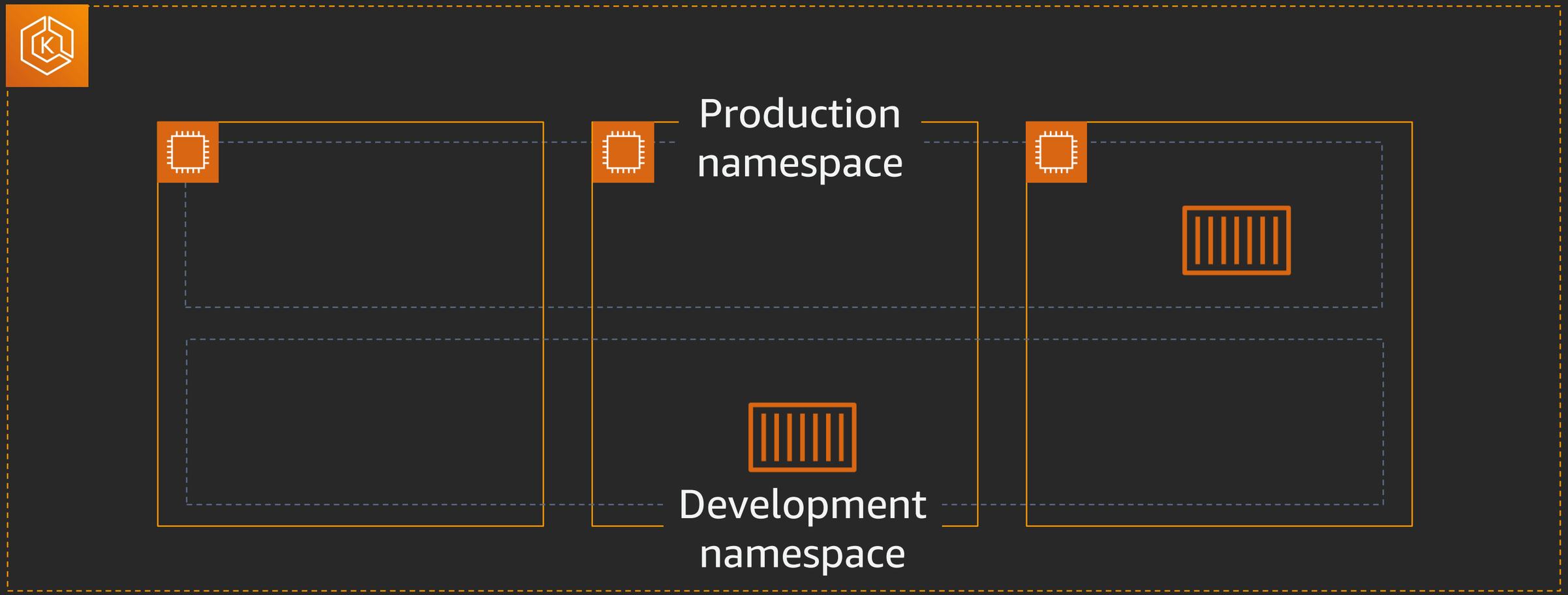


These three redis pods won't be scheduled on the same node

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-cache
spec:
  selector:
    matchLabels:
      app: store
  replicas: 3
  template:
    metadata:
      labels:
        app: store
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - store
            topologyKey: "kubernetes.io/hostname"
      containers:
      - name: redis-server
        image: redis:3.2-alpine
```

These three redis pods won't be scheduled on the same node

Namespaces



Namespaces

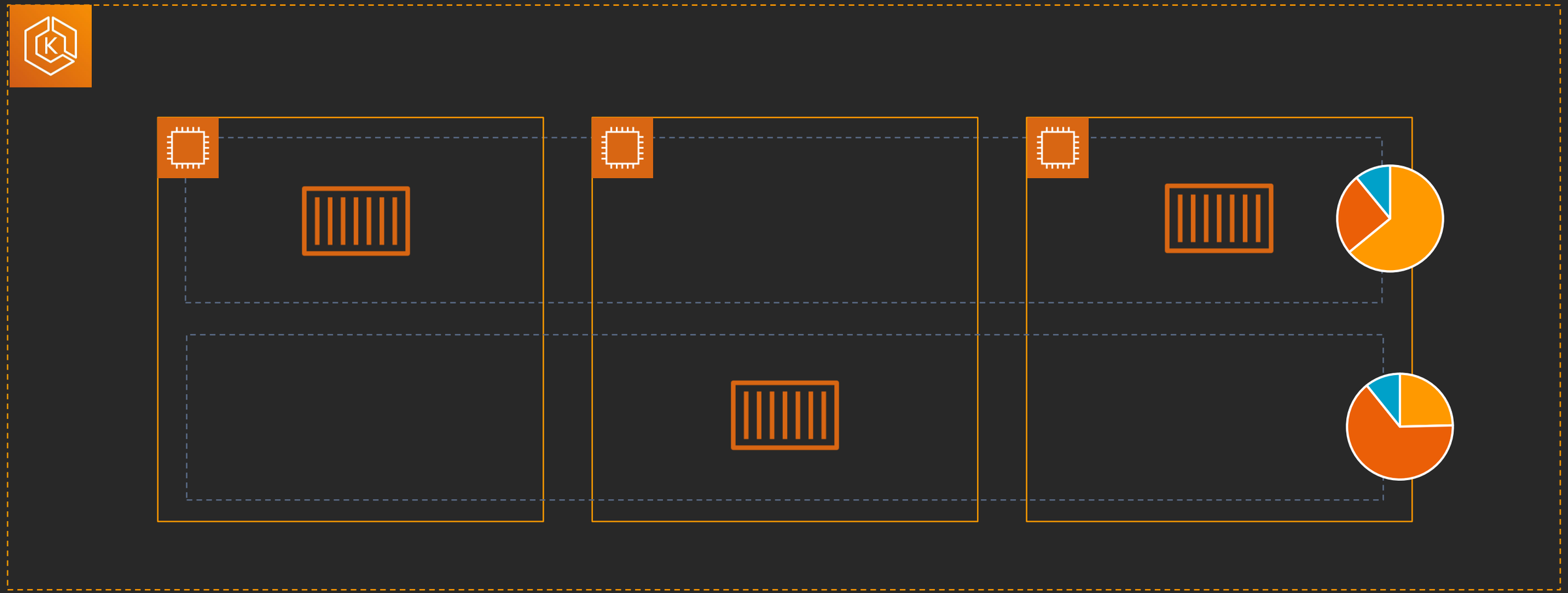
```
aws$ kubectl create namespace production-namespace
```

```
apiVersion: v1
 kind: Pod
 metadata:
  name: cats-pod
  namespace: production-namespace
  labels:
   app: cats-application
 spec:
  containers:
  - name: cats-container
    image: nginx:1.7.8
```

This pod will be deployed in to the production-namespace

```
aws$ kubectl apply -f pod.yaml
```

Resource quotas



Resource Quotas

```
aws$ kubectl create namespace dev-namespace
```

```
apiVersion: v1
 kind: ResourceQuota
 metadata:
  name: dev-namespace-quota
spec:
  hard:
   cpu: "10"
   memory: 20Gi
   pods: "10"
```

These limits will be applied to whichever namespace we apply this resource quota object to

```
aws$ kubectl apply -f dev-namespace-quota –n dev-namespace
```

A quick recap

Reducing the blast radius

Isolate workloads using multiple clusters

Affinity and anti-affinity rules can be configured to separate workloads

Use resource quotas to control resource usage

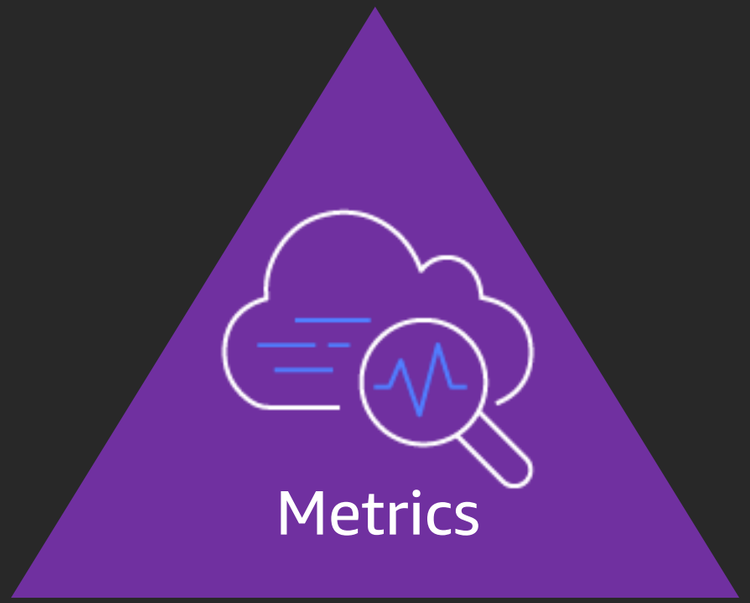
Reducing the time to detection

Detection

Definition

The action or process of identifying the presence of something concealed

Metrics



What: Numeric representation of data measured over intervals of time

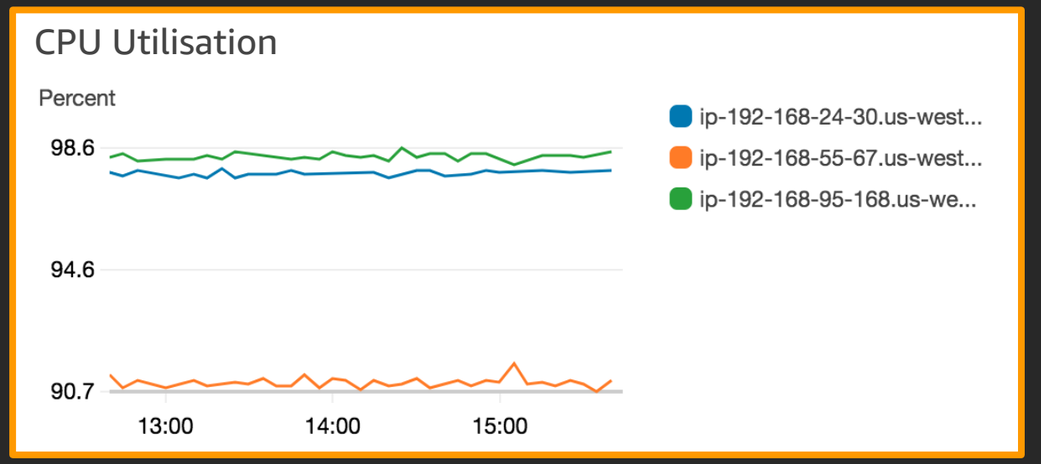
Why: Useful for identifying trends, mathematical modeling, and prediction

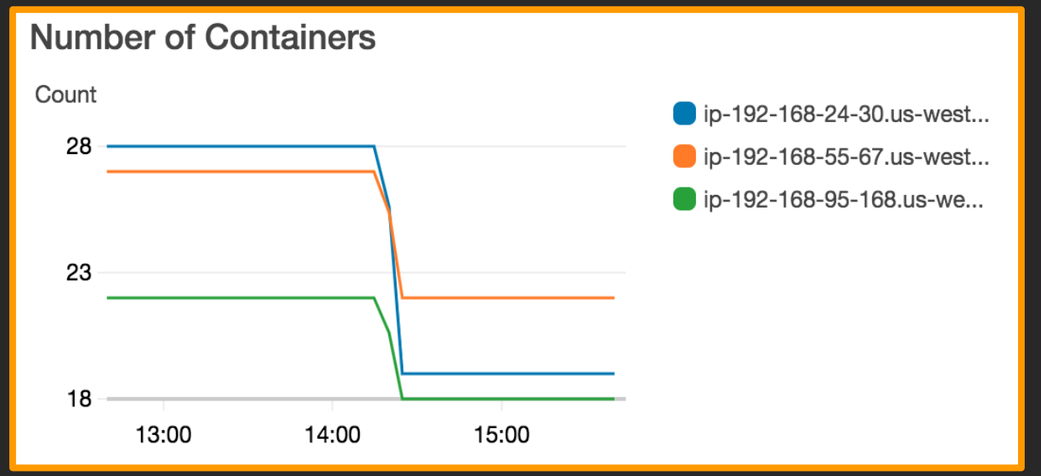
Metrics and Kubernetes

aws SUMMIT ONLINE

Things that produce metrics: the control plane

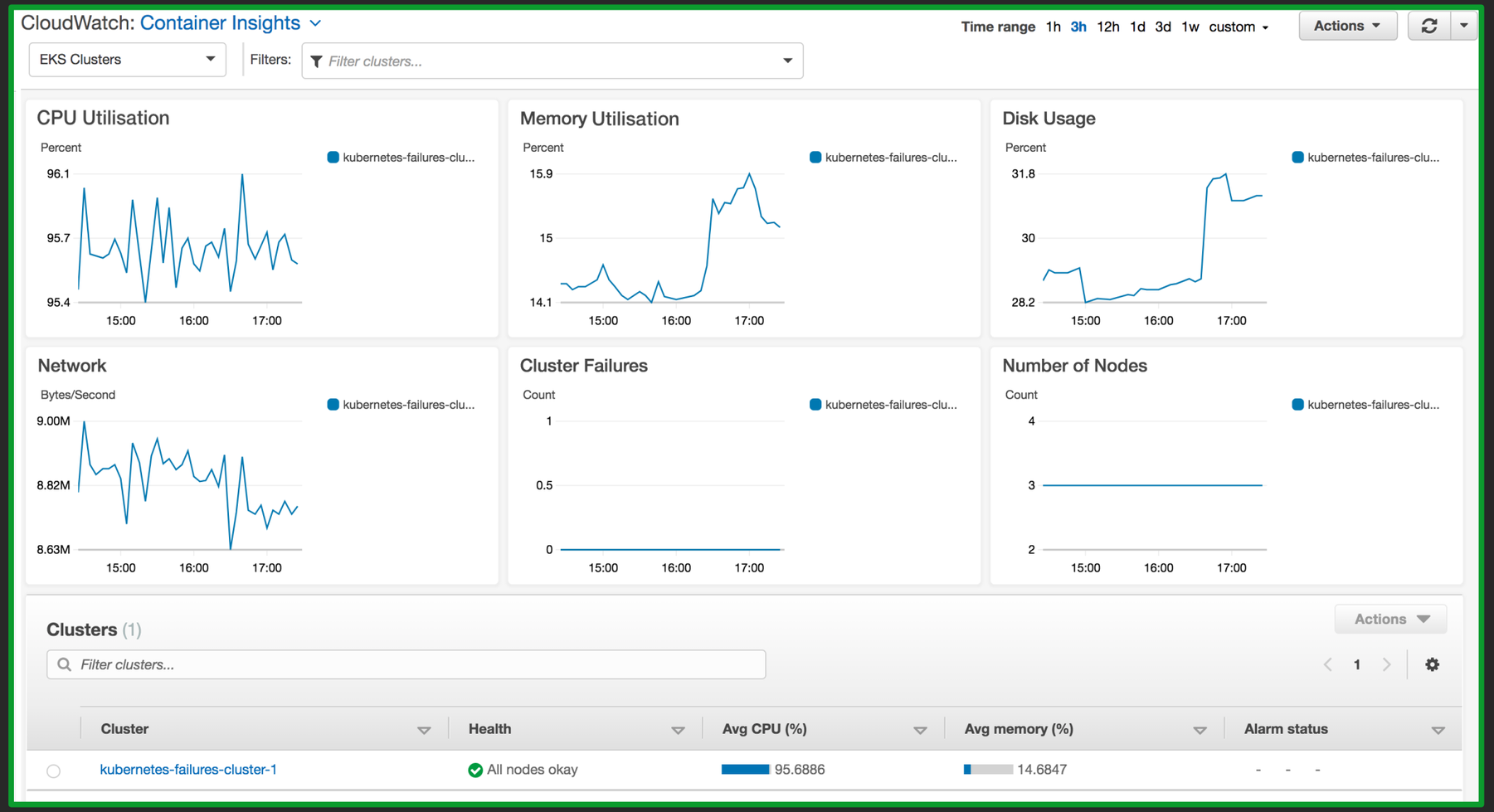
Things that produce metrics: the nodes



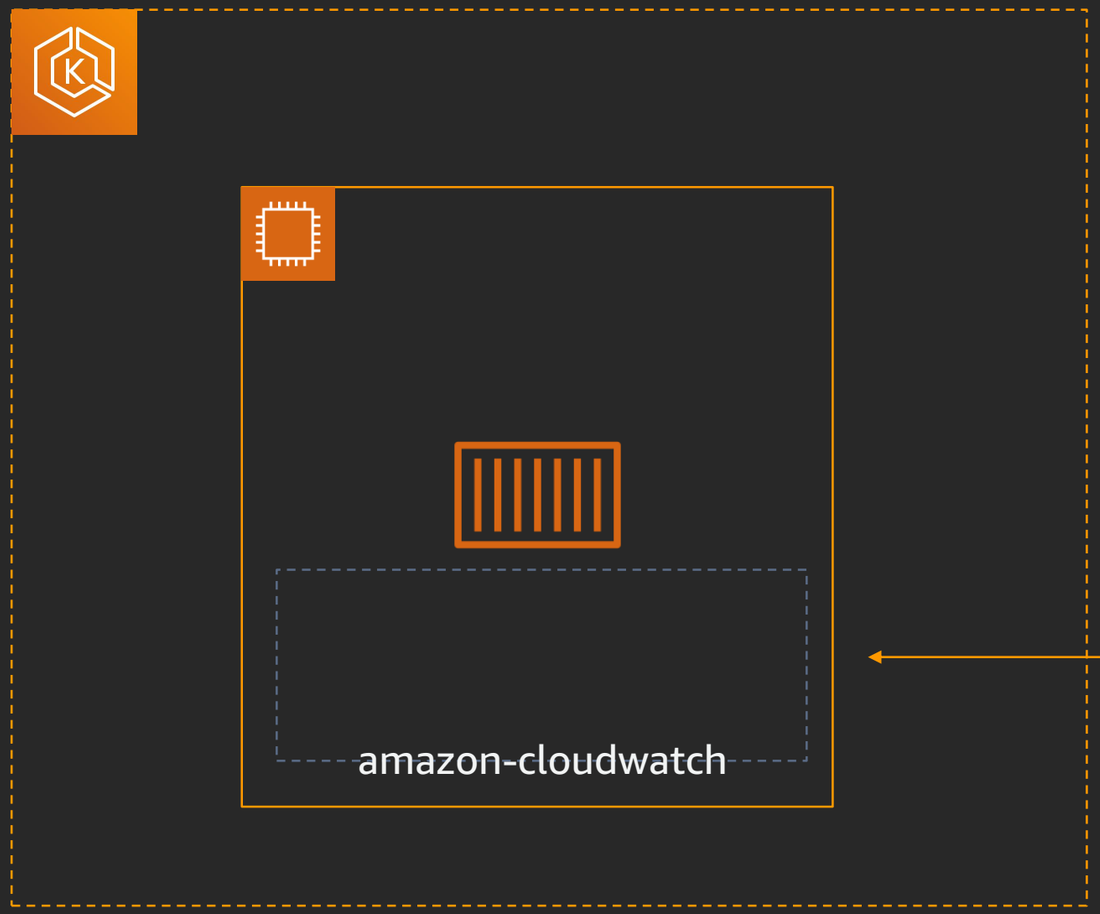


Things that produce metrics: The pods

Amazon CloudWatch Container Insights: Dashboard

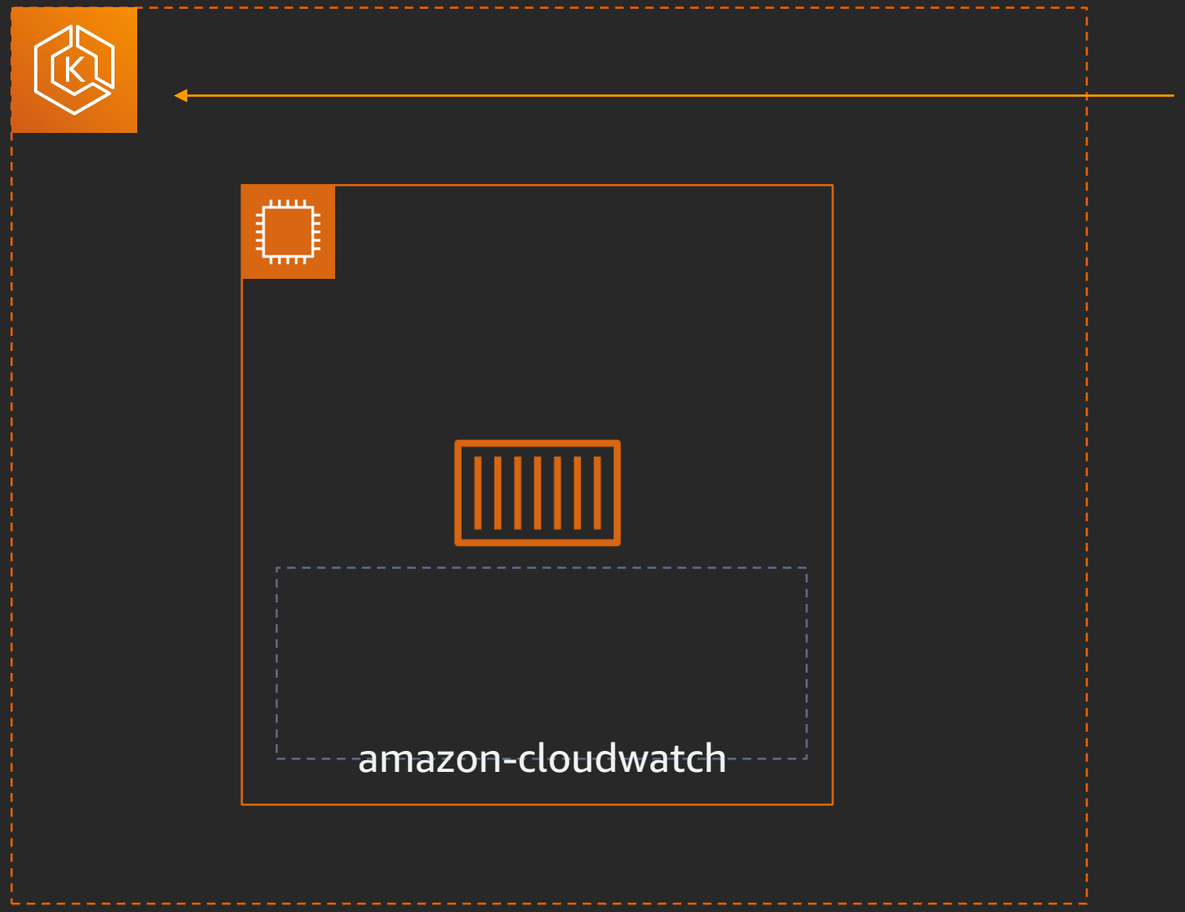


Amazon CloudWatch Container Insights



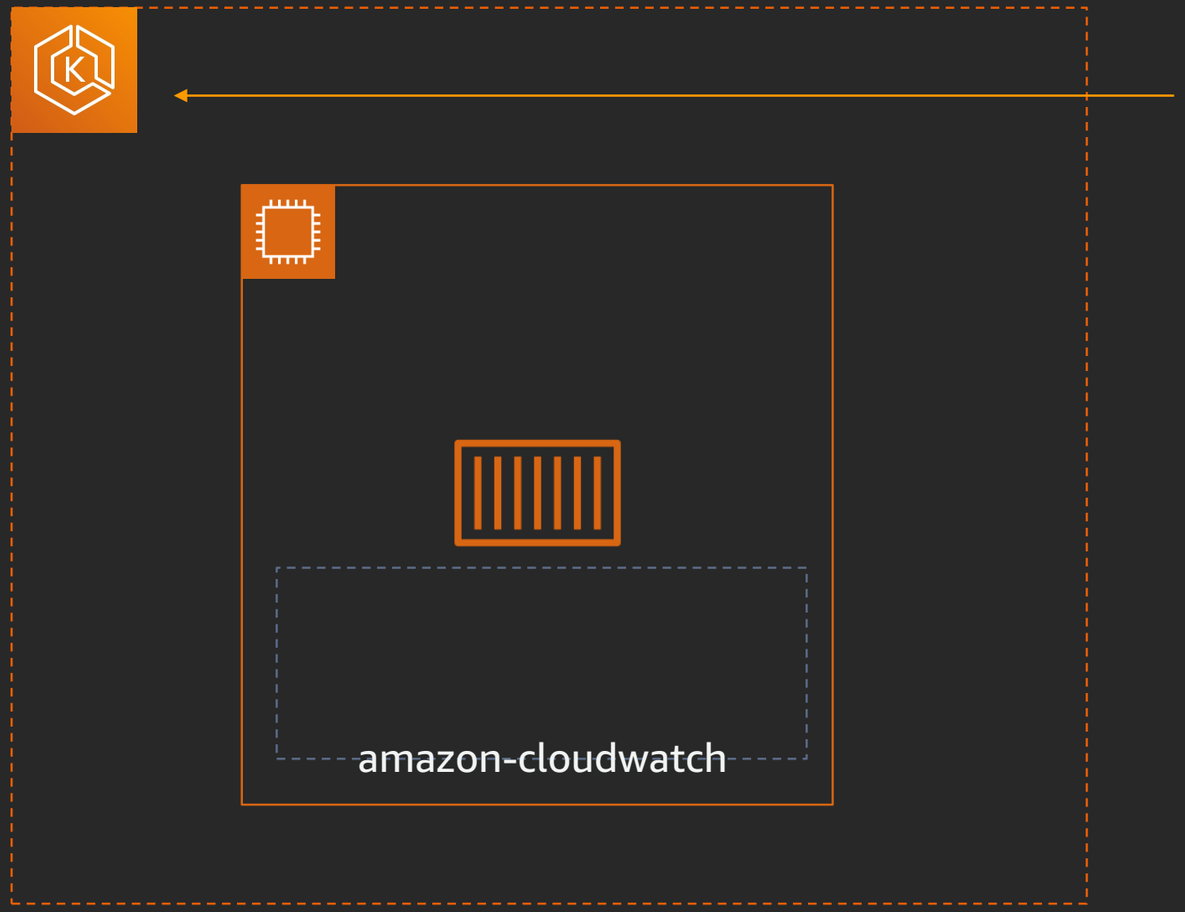

```
aws$ cat << EOF > cw-namespace.yaml

apiVersion: v1
 kind: Namespace
 metadata:
  name: amazon-cloudwatch
  labels:
   name: ap-southeast-2

EOF

aws$ kubectl apply -f cw-namespace.yaml
```

Amazon CloudWatch Container Insights



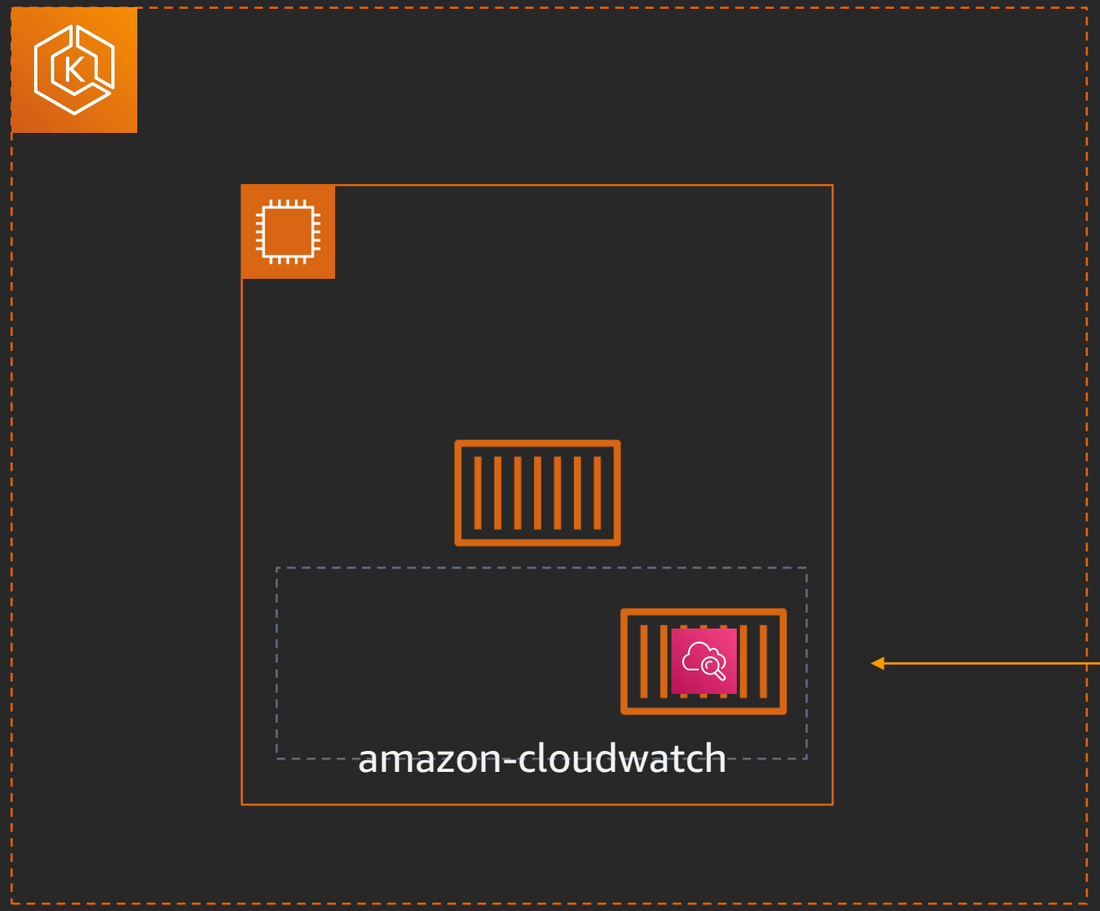

```
aws$ cat << EOF > cw-service.yaml

apiVersion: v1
kind: ServiceAccount
metadata:
  name: cloudwatch-agent
  namespace: amazon-cloudwatch

---
kind: ClusterRole
apiVersion: rbac.authorisation.k8s.io/v1
metadata:
  name: cloudwatch-agent-role
rules:
  - apiGroups: [""]
    resources: ["pods", "nodes", "endpoints"]
    verbs: ["list", "watch"]

... EOF

aws$ kubectl apply -f cw-service.yaml
```

Amazon CloudWatch Container Insights



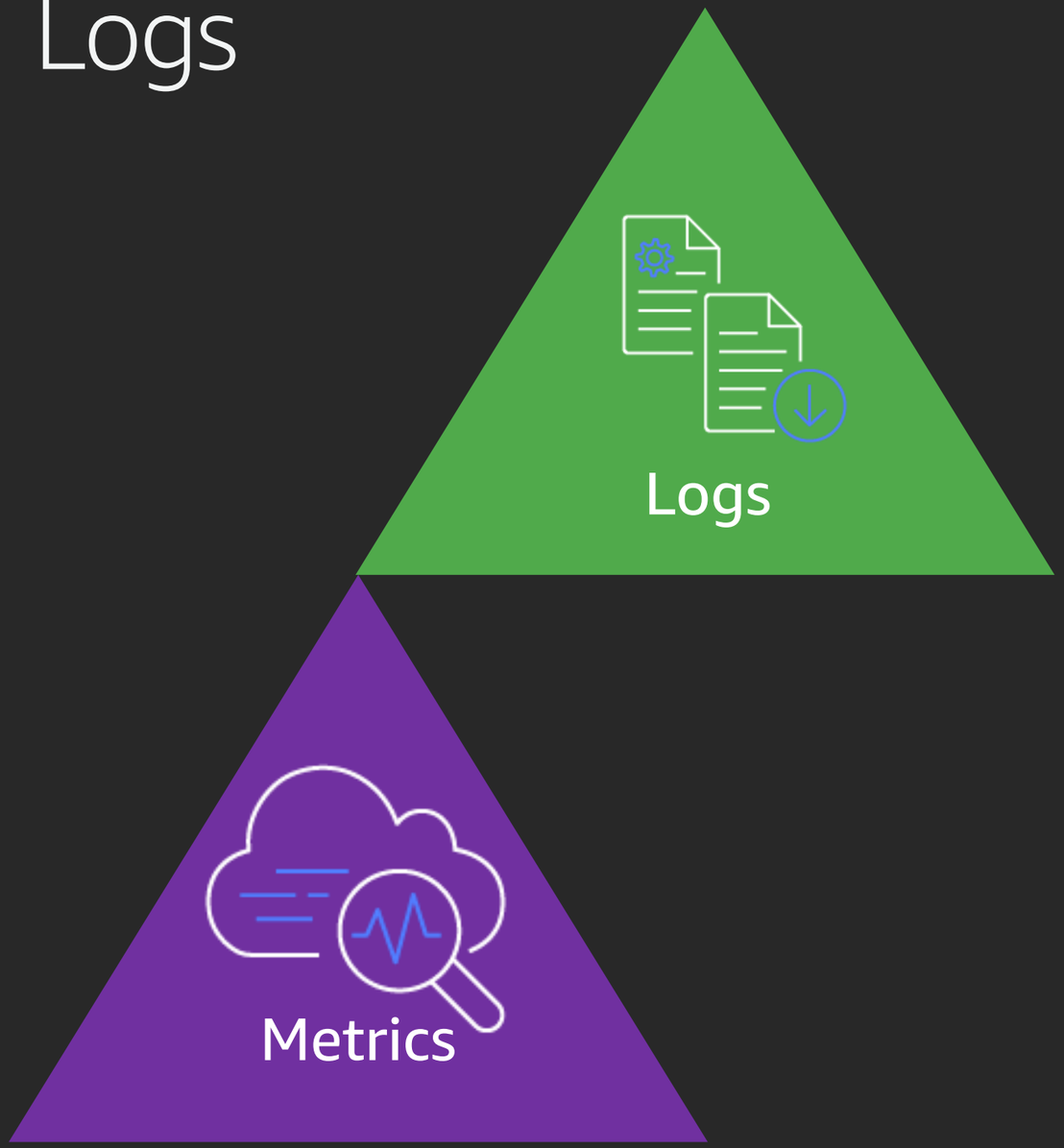

```
aws$ cat << EOF > cw-configmap.yaml

apiVersion: v1
data:
  cwagentconfig.json: |
    {
      "logs": {
        "metrics_collected": {
          "kubernetes": {
            "cluster_name": "{{cluster_name}}",
            "metrics_collection_interval": 60...

... EOF

aws$ kubectl apply -f cw-configmap.yaml
```

Amazon CloudWatch Container Insights




```
aws$ cat << EOF > cw-configmap.yaml

apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: cloudwatch-agent
  namespace: amazon-cloudwatch
spec:
  selector:
    matchLabels:
      name: cloudwatch-agent
... EOF

aws$ kubectl apply -f cw-configmap.yaml
```

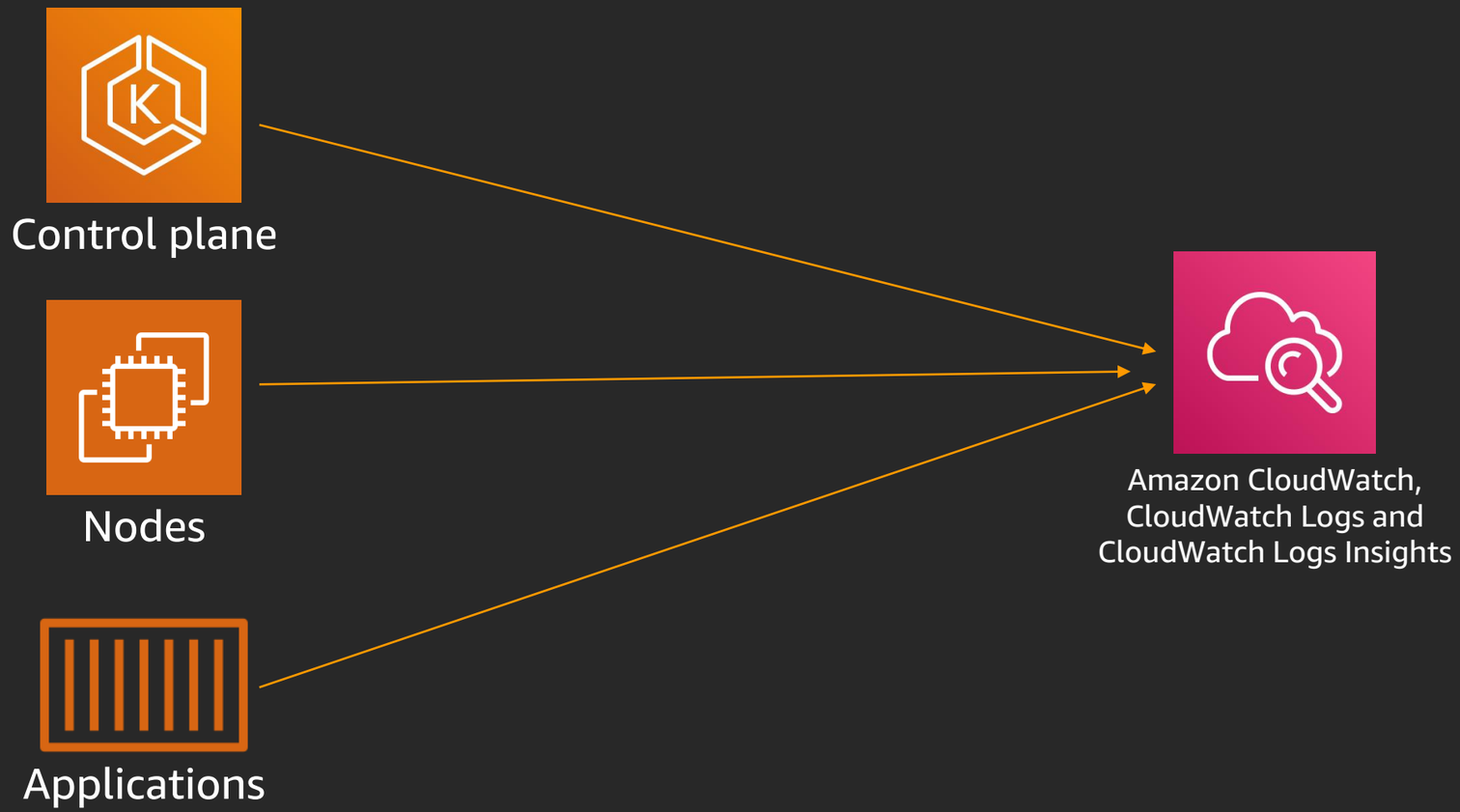
Logs



What: Immutable, timestamped record of discrete events that happened over time

Why: Useful for uncovering emergent and unpredictable behaviour

Collecting logs from your Kubernetes clusters

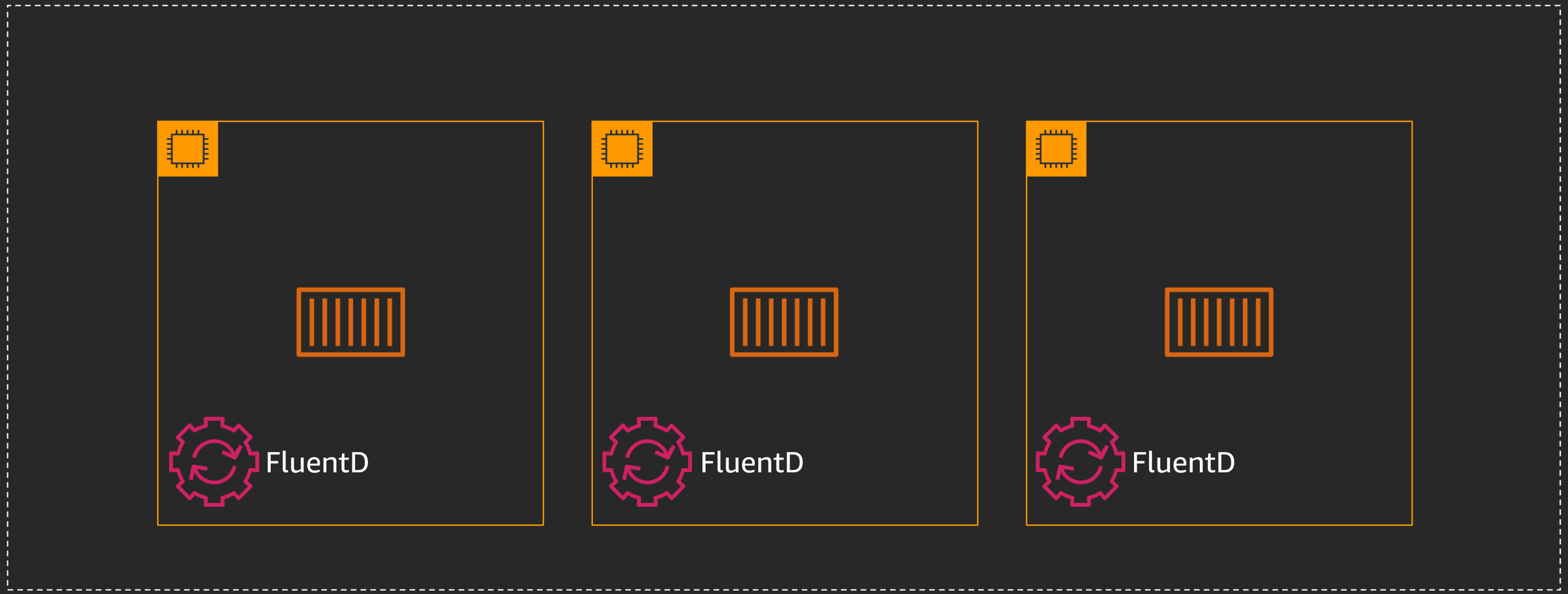


Enable control plane logging

```
aws$ cat << EOF > cluster.yaml

apiVersion: eksctl.io/v1alpha5
 kind: ClusterConfig
 metadata:
  name: cloudwatch-cluster
  region: ap-southeast-2
 nodeGroups:
 - name: default
   instanceType: m5.large
   desiredCapacity: 3
   iam:
    withAddonPolicies:
     cloudWatch: true
 cloudWatch:
  clusterLogging:
   enableTypes: ["all"]

EOF

aws$ eksctl create cluster -f cluster.yaml
```

enable and configure control plane logging on the cluster

Collecting logs from your nodes

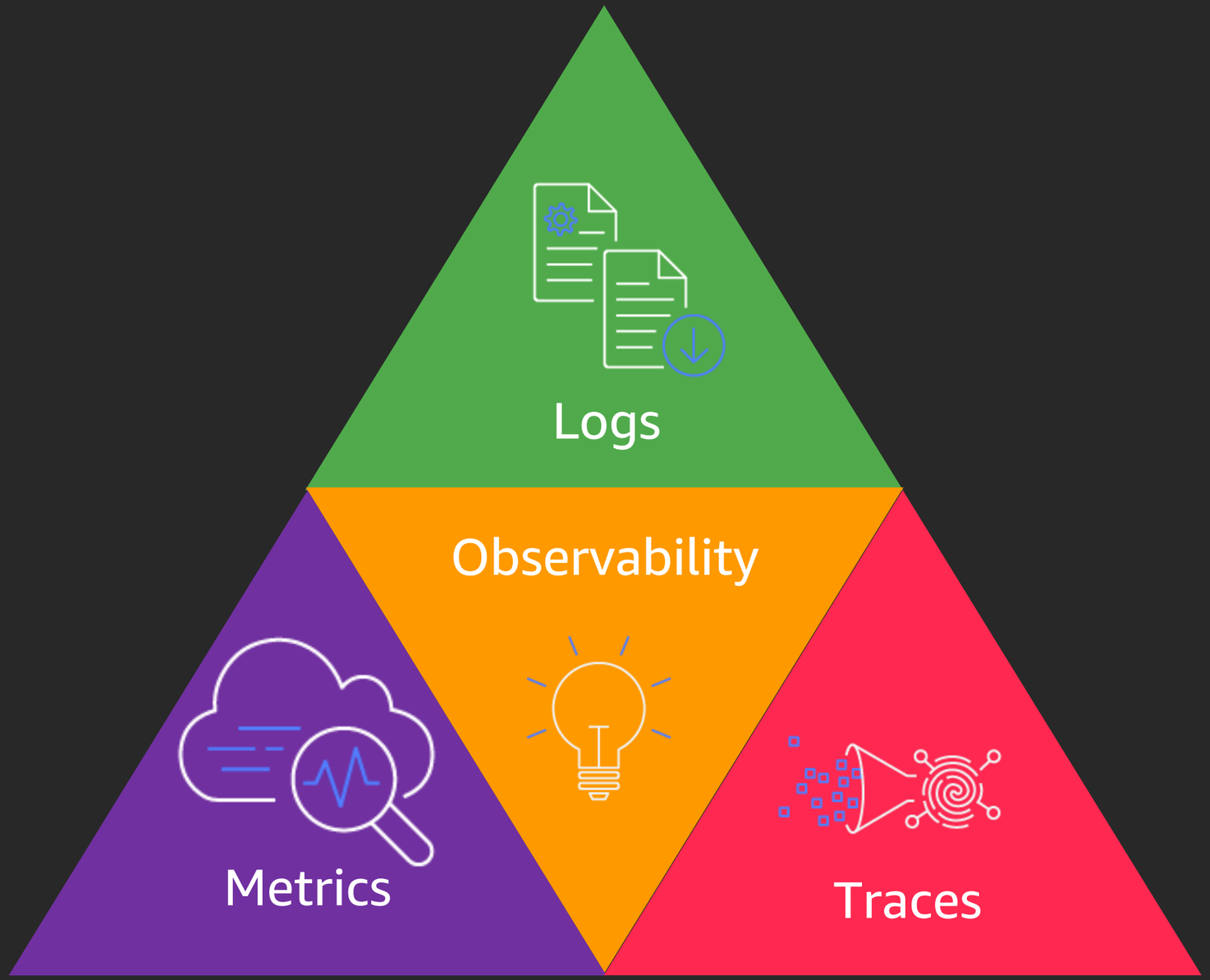


Amazon CloudWatch Logs Insights

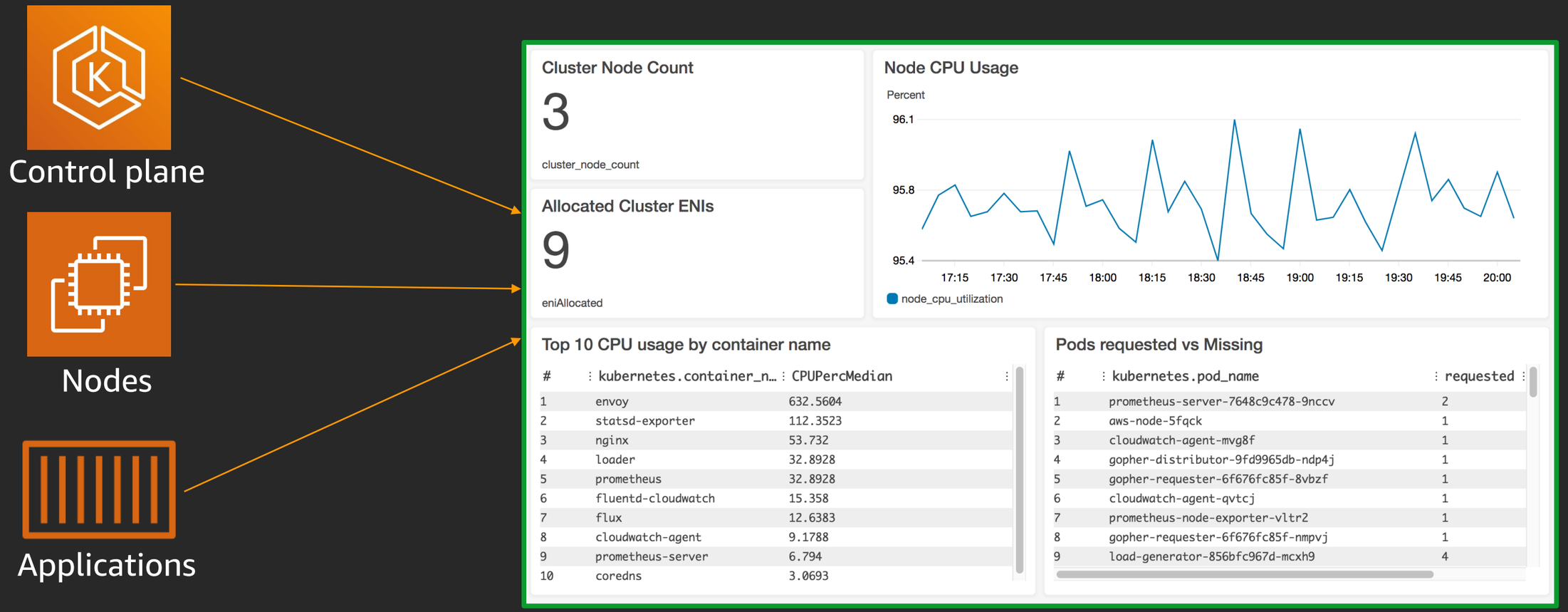
```
STATS avg(number_of_container_restarts) as avg_number_of_container_restarts by PodName
| SORT avg_number_of_container_restarts DESC
```

#	PodName	avg_number_of_container_restarts
1	aws-appmesh-grafana	0
2	coredns	0
3	aws-appmesh-inject	0
4	flux-memcached	0
5	flux	0
6	tiller-deploy	0
7	gopher-distributor	0
8	prometheus-kube-state-metrics	0

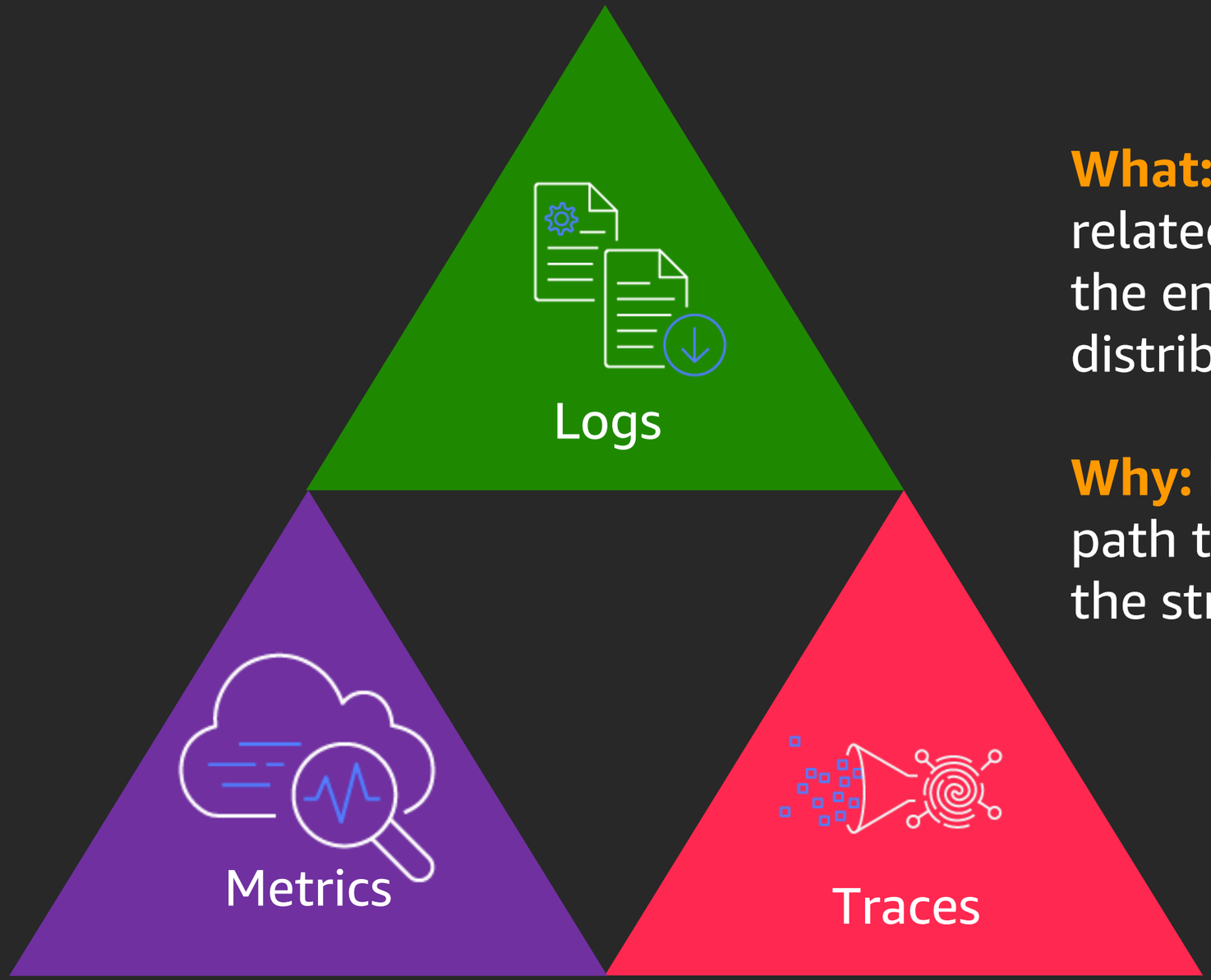
Observability is the goal



CloudWatch dashboards



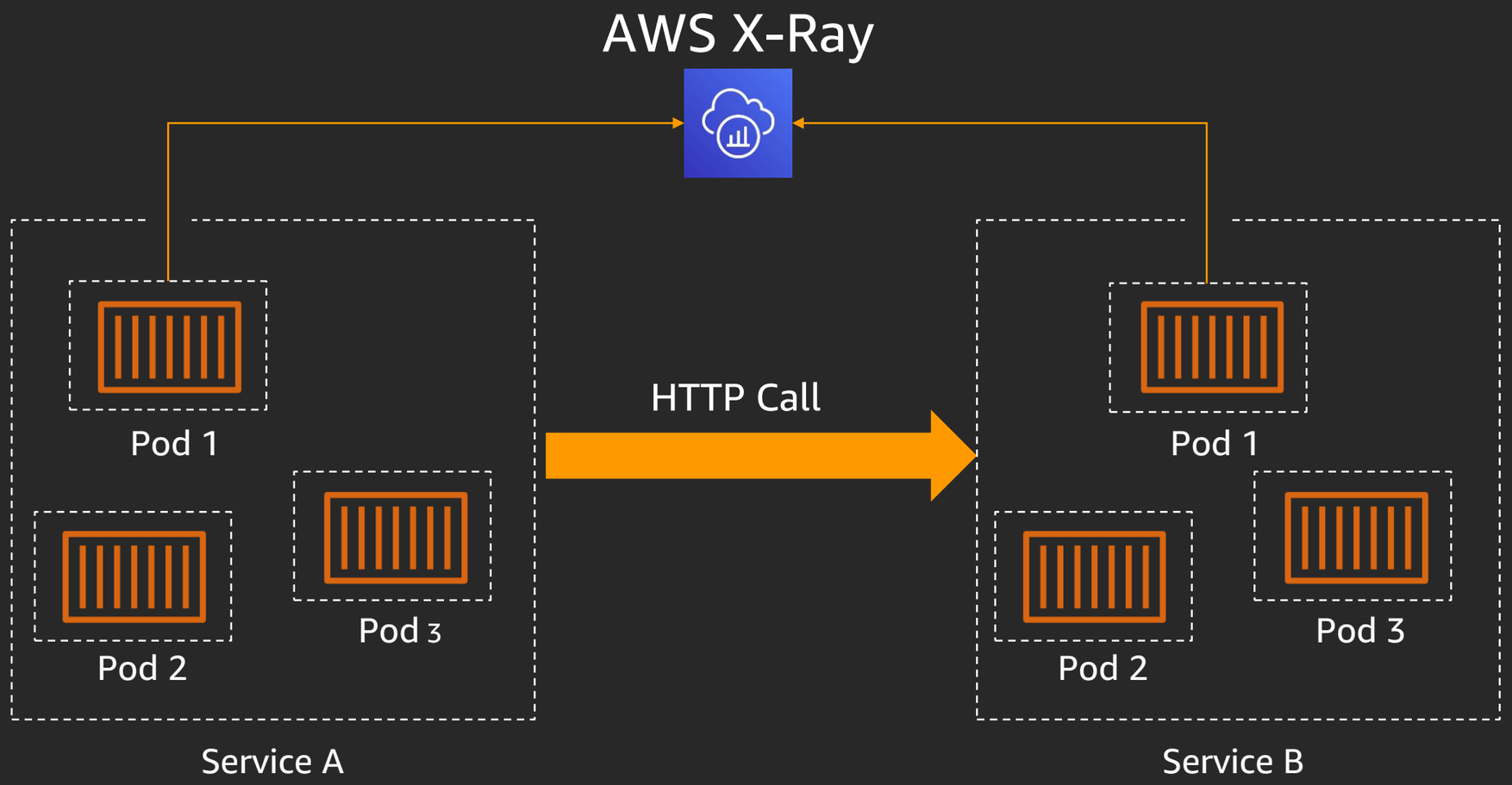
Traces



What: Representation of a series of related distributed events that encode the end-to-end request flow through a distributed system

Why: Provides visibility into both the path traversed by a request as well as the structure of a request

Tracing with AWS X-Ray

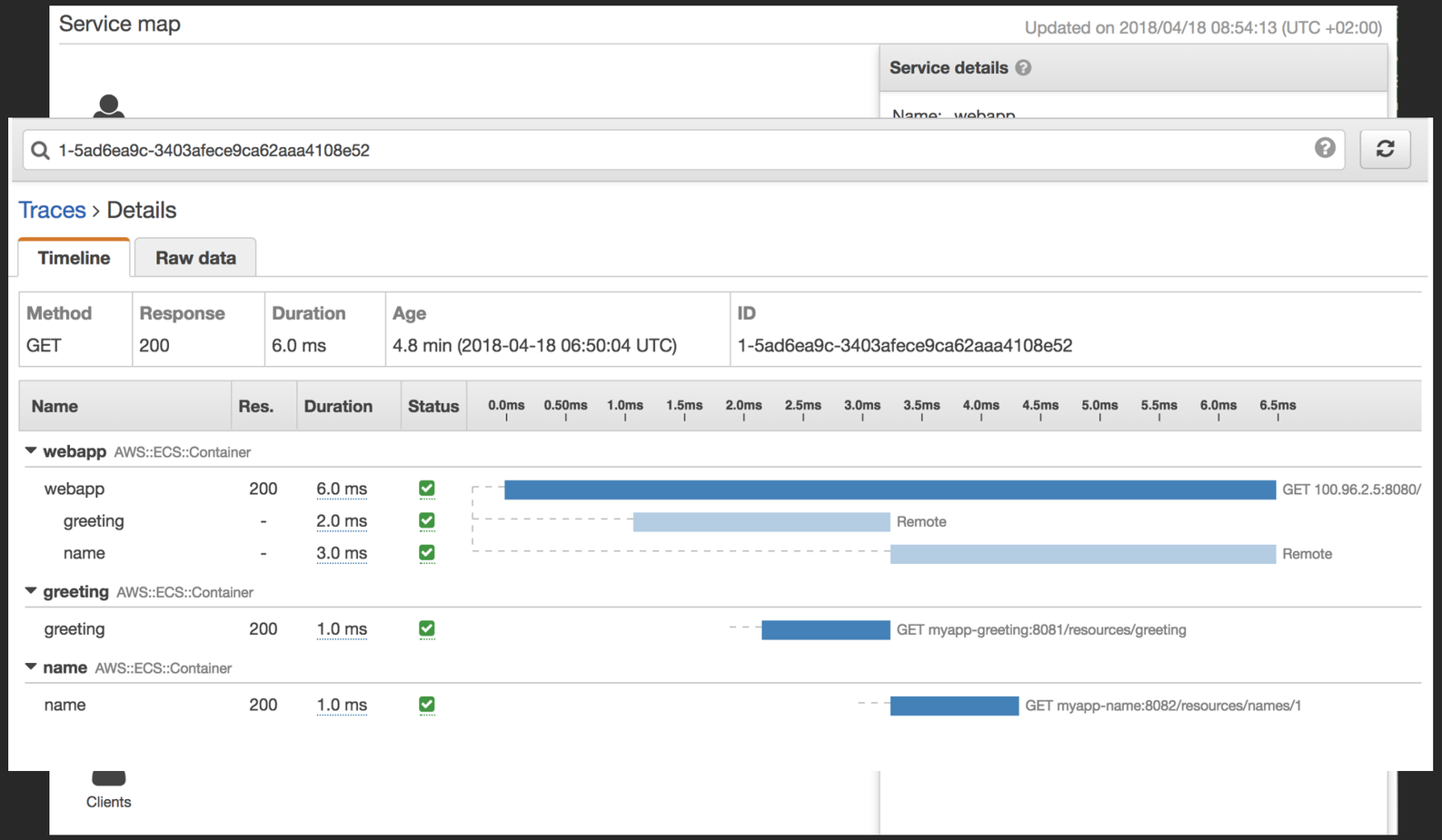


Tracing with AWS X-Ray

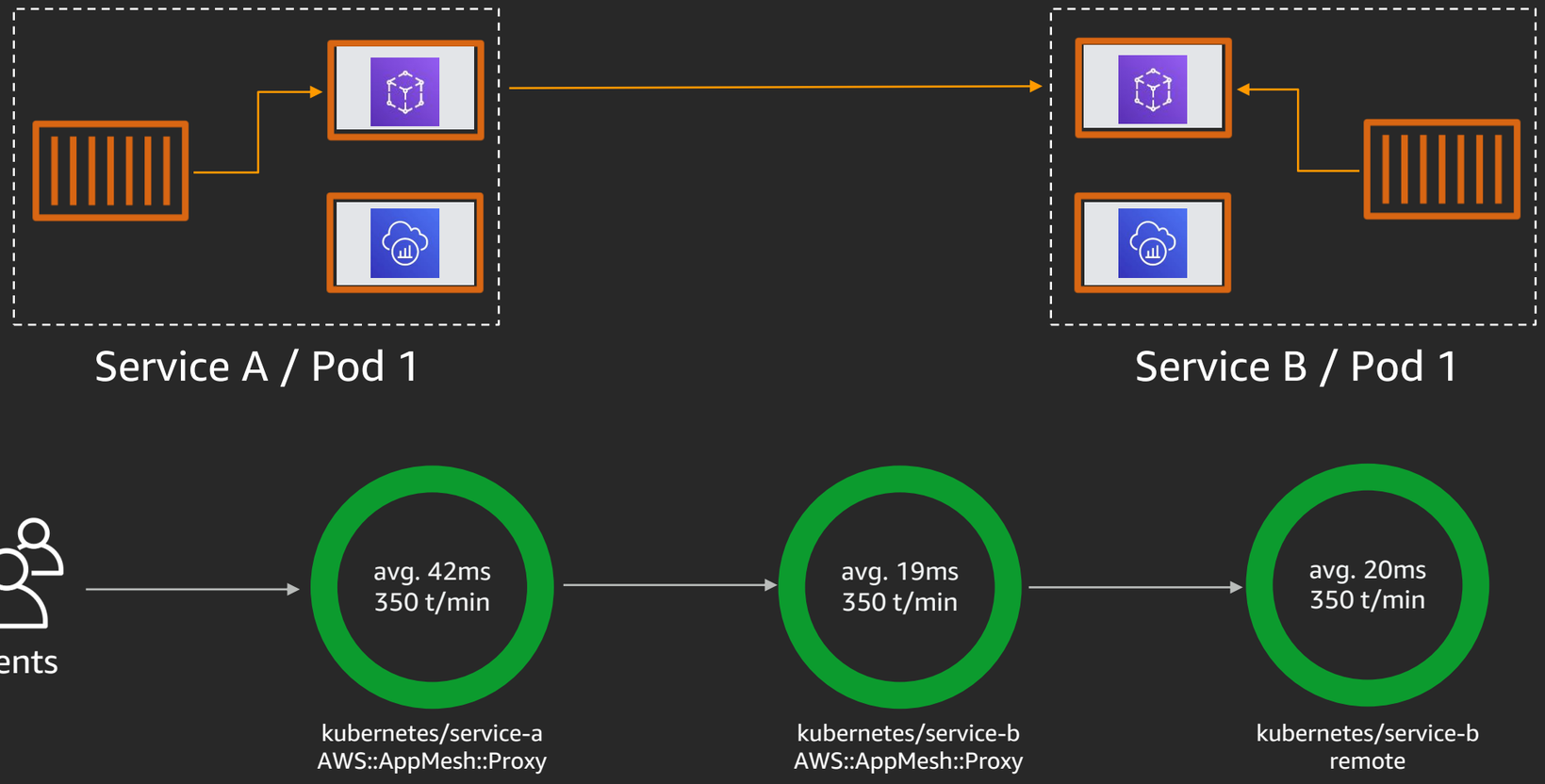
Configure IAM to allow pods running on nodes to send traces to X-Ray

Deploy the AWS X-Ray Daemon

Integrate AWS X-Ray SDK in to your application



AWS App Mesh and AWS X-Ray



A quick recap

Reducing the blast radius | Reducing the time to detection

Determine your health vs diagnostic metrics

Logs provide a rich source of events which can be used

Dive deeper using application level tracing

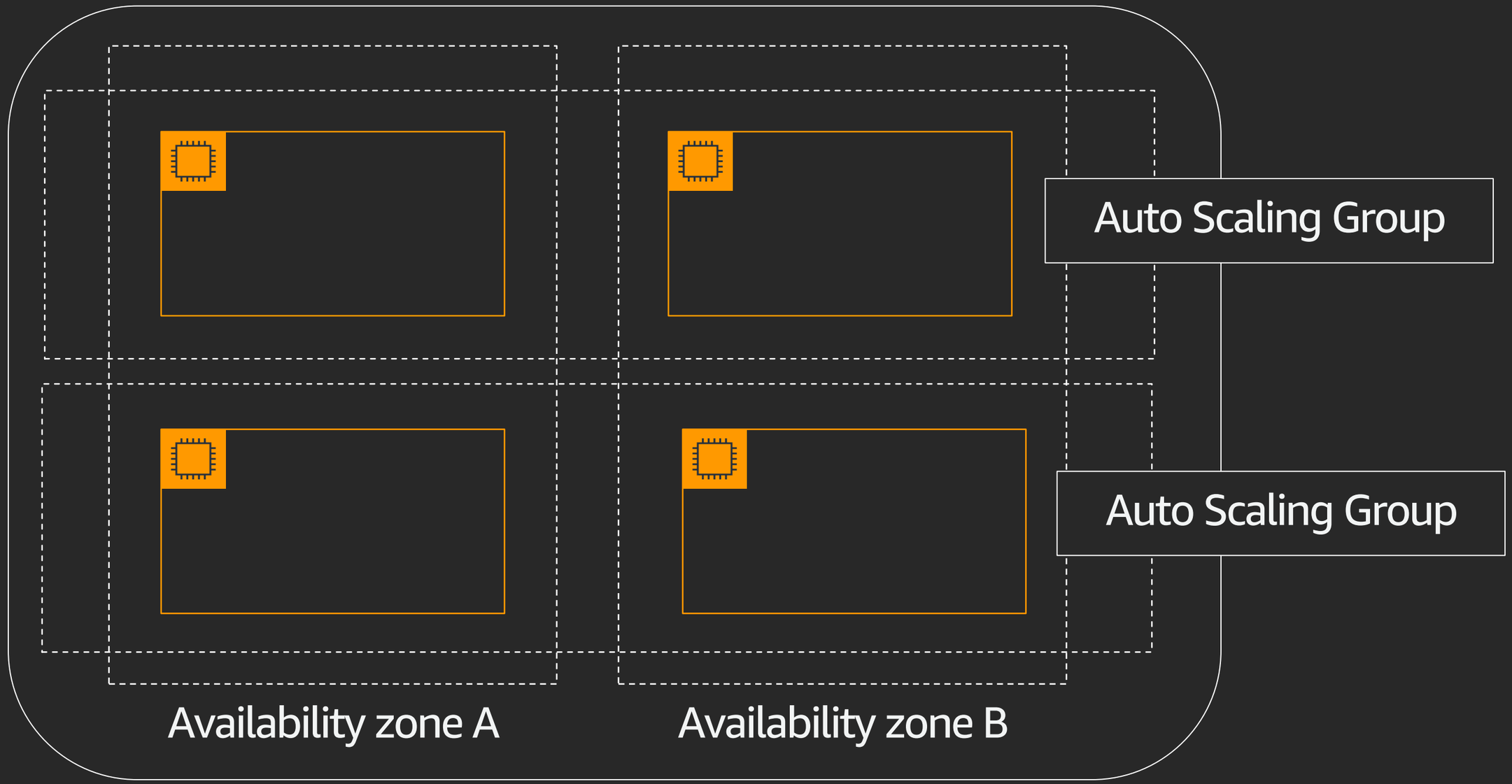
Reducing the time to mitigation

Mitigation

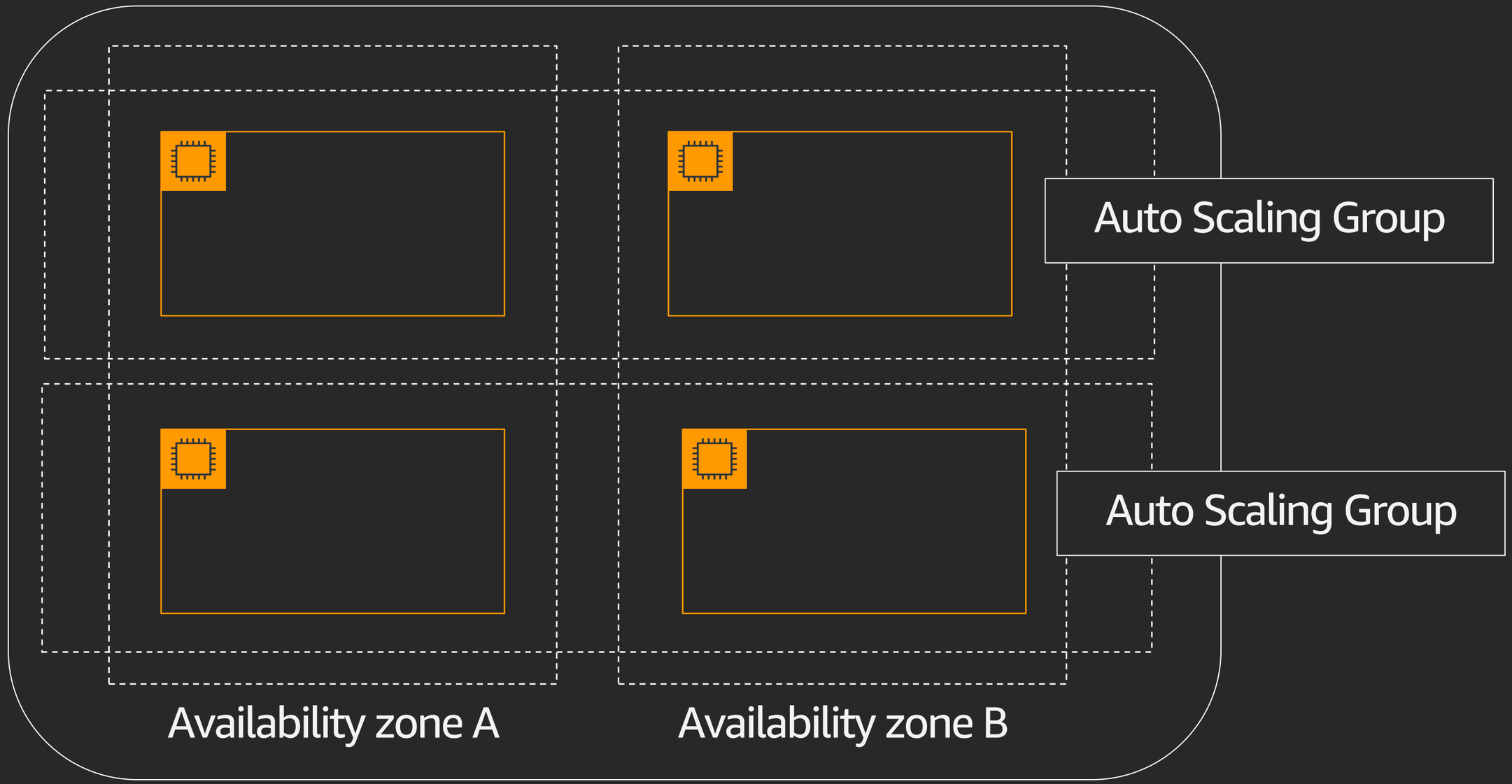
Definition

The action of reducing the severity, seriousness, or painfulness of something

Reducing with: Auto Scaling group

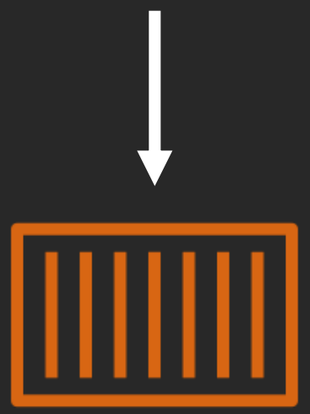


Reducing with: Auto Scaling group



Reducing with: health probes

HTTP GET



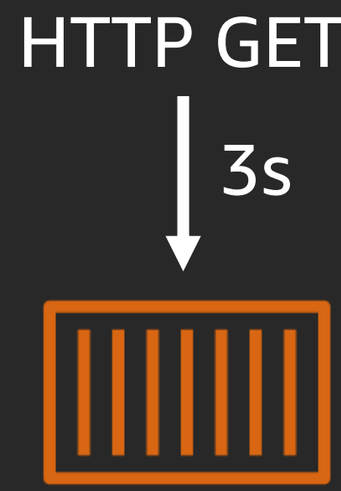
Readiness

Is my application ready to service requests?

```
aws$ cat << EOF > readinessProbe.yaml

readinessProbe:
 exec:
  command:
    - cat
    - /tmp/healthy
 initialDelaySeconds: 5
 periodSeconds: 5

EOF

aws$ kubectl apply -f readinessProbe.yaml
```

Reducing with: health probes

```
aws$ cat << EOF > livenessProbe.yaml

livenessProbe:
 httpGet:
  path: /healthz
  port: 8080
 initialDelaySeconds: 5
 periodSeconds: 5

EOF

aws$ kubectl apply -f livenessProbe.yaml
```



Liveness

Is my application healthy?

A quick recap

Reducing the blast radius | Reducing the time to detection | Reducing the time to mitigation

Leverage AWS native capabilities to automatically respond

Have a plan for cluster recovery

Use reediness and liveness probes reduce impact

Pulling it all together

Failing successfully with Kubernetes

What steps are your taking to reduce the blast radius?

How could you cut the time to detection in half?

How you could cut the time to migration in have?

Thank you!

Mitch Beaumont

beaumonm@amazon.com

aws SUMMIT ONLINE