
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

CON326

# Are you ready? Essential Strategies for Kubernetes Adoption

Eswar Bala (he/him)

Director, EKS Service Team
AWS Kubernetes

Rick Sostheim (he/him)

Principal Engineer
AWS Kubernetes

# Agenda

- Optimizing organizations for success
- EKS and Kubernetes
  - How systems work and fail and what to do about it

# Optimizing organizations for success

# Culture:

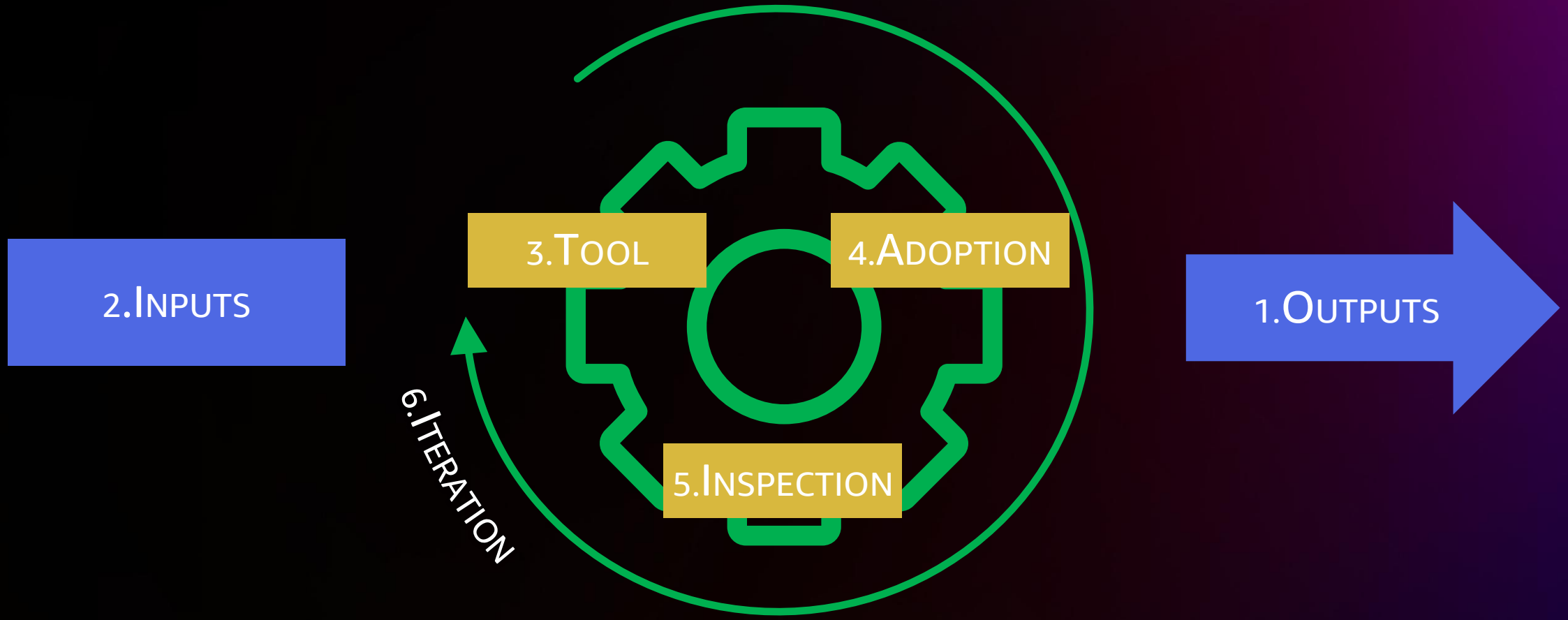
A set of beliefs, morals, methods, institutions and a collection of human knowledge that is dependent on the transmission of these characteristics to younger generations

Merriam-Webster’s Dictionary

# Mechanisms

- Recurring problems
- Solutions – sustain and scale
- Sustain  culture
- Examples inside Amazon
  - Operational reviews
  - Correction of Errors
  - Product development

# Mechanisms – In practice

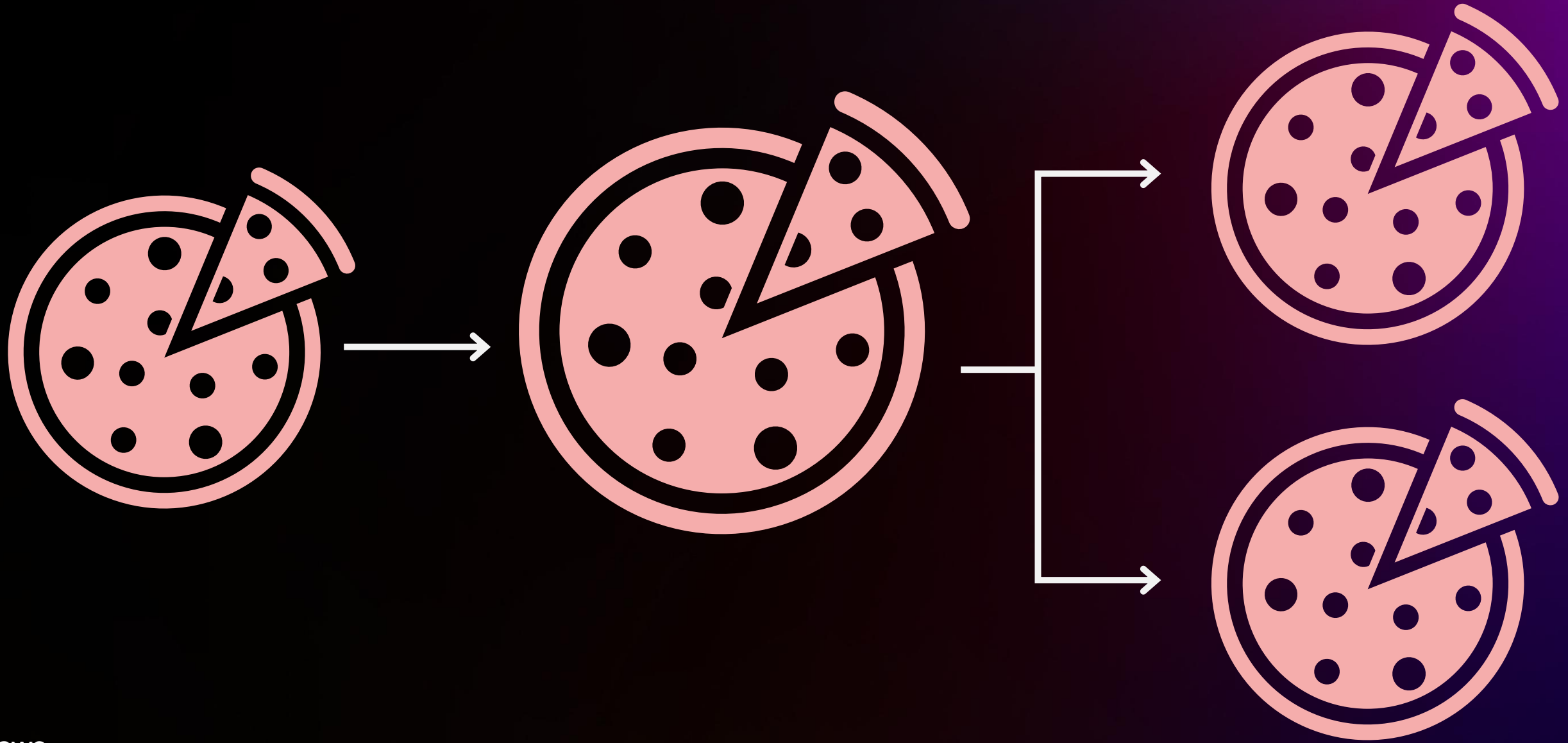


# Mechanisms – Organizational levers

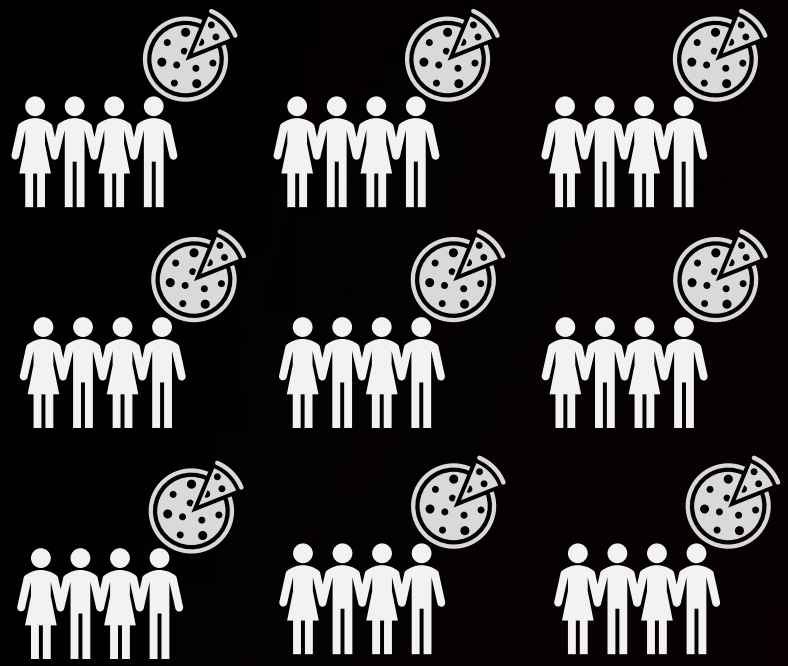
# Organizational structure

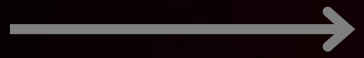
- Small (two-pizza) teams
- Ownership  (decentralized) / autonomy
- Local decisions
- Lowers cost of experimentation and failure

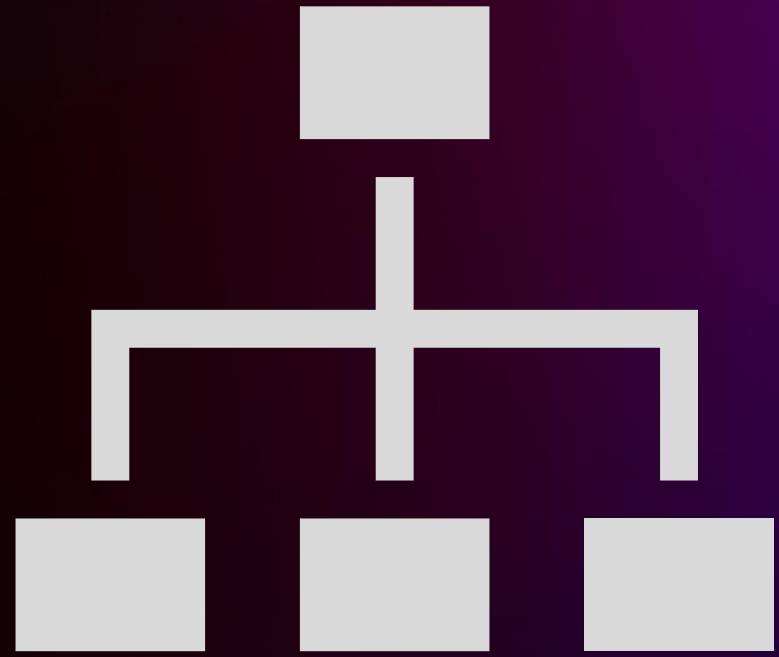
# Two-pizza team growth



# Two-pizza teams at scale



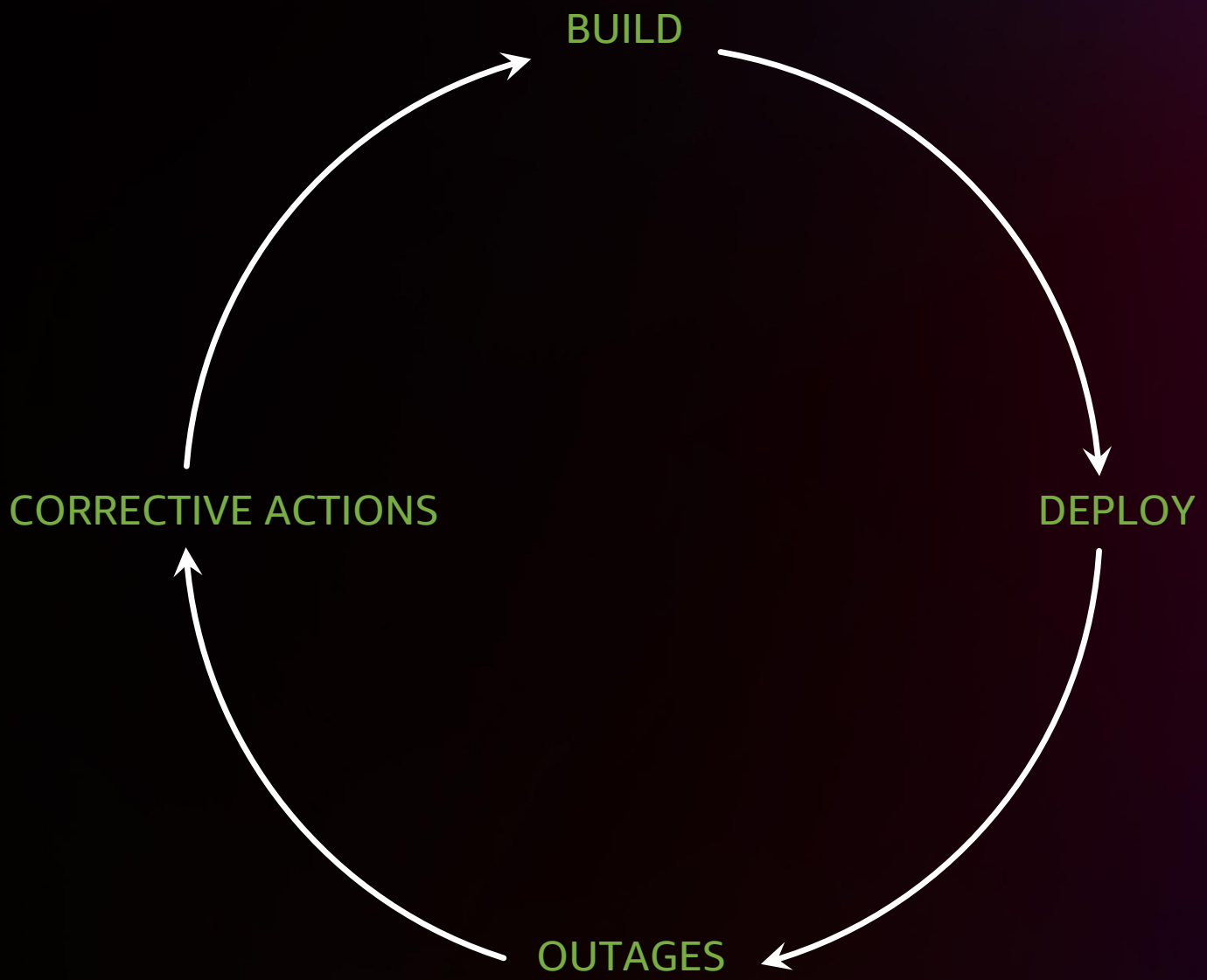




# AWS operational culture

- You Build, You Operate
- Broad operational responsibility
- Learn from failures

# Correction of Errors (COE)



# COE – In practice

- Best practices
- Tool improvements
- COE contents
  - Customer Impact
  - Metrics and Data
  - Root Cause Analysis – 5 Whys
  - Detection and Mitigation
  - Lessons Learned
  - Action Items

# Operational Readiness Reviews (ORR)

- Recognize failure patterns and avoid repetition
- Best operational practices
- Learn and correct before production
- Achieve and maintain operational excellence

# ORR – In practice

- Checklist of questions
- Derived from past failures
- Structure
  - Architecture
  - Release quality
  - Event management
- Multiple variations – service/feature/enhancement

# Deployment consistency

# Deployment consistency – Best practices

- Integration / pre-prod testing
- Canary / Fractional deployments
- Rollback alarms
- Metrics anomaly detection

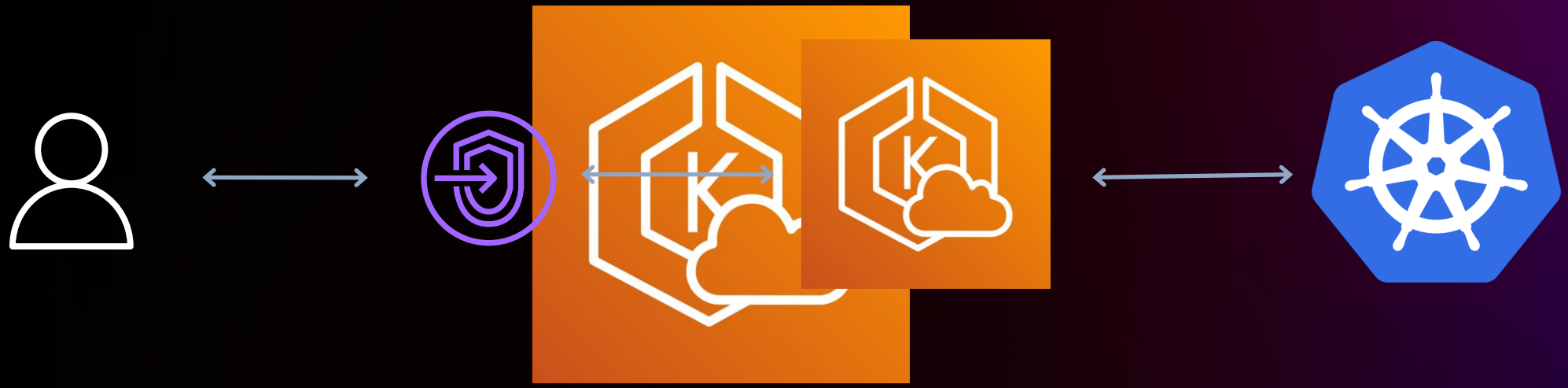
# EKS and Kubernetes

“A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.”

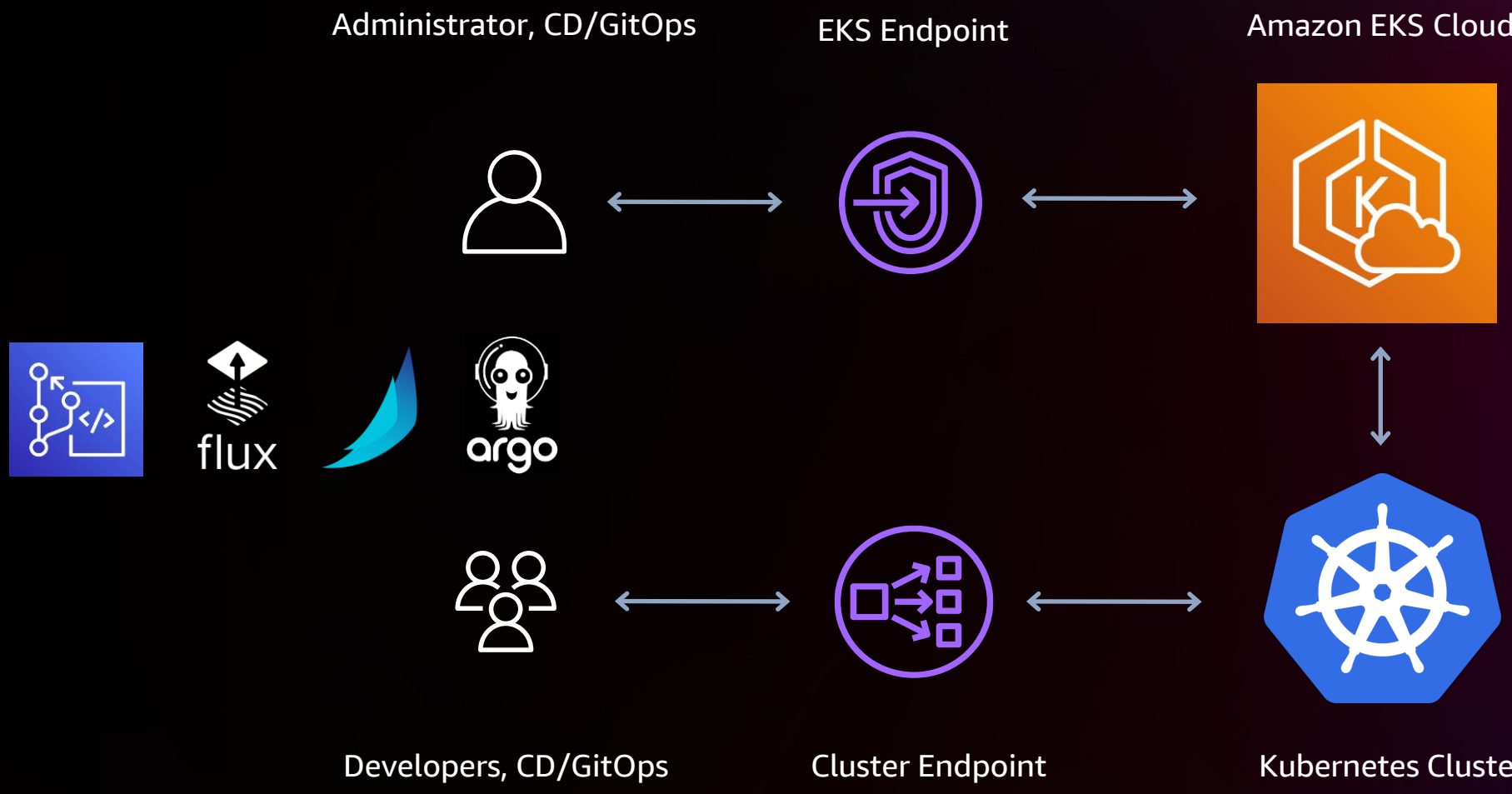
**Leslie Lamport, 2013 Turing Award**

Computer Scientist and Mathematician, MIT

# EKS and Kubernetes overview



# EKS and Kubernetes overview



# EKS and Kubernetes overview

EKS Regional Endpoint
eks.us-west-2.amazonaws.com

```
$ aws eks list-clusters

$ aws eks create-cluster --name prod <…>

$ aws eks describe-cluster --name prod

$ aws eks get-token --cluster-name prod

$ aws eks create-nodegroup --cluster-name prod
      --nodegroup-name frontend <…>
```

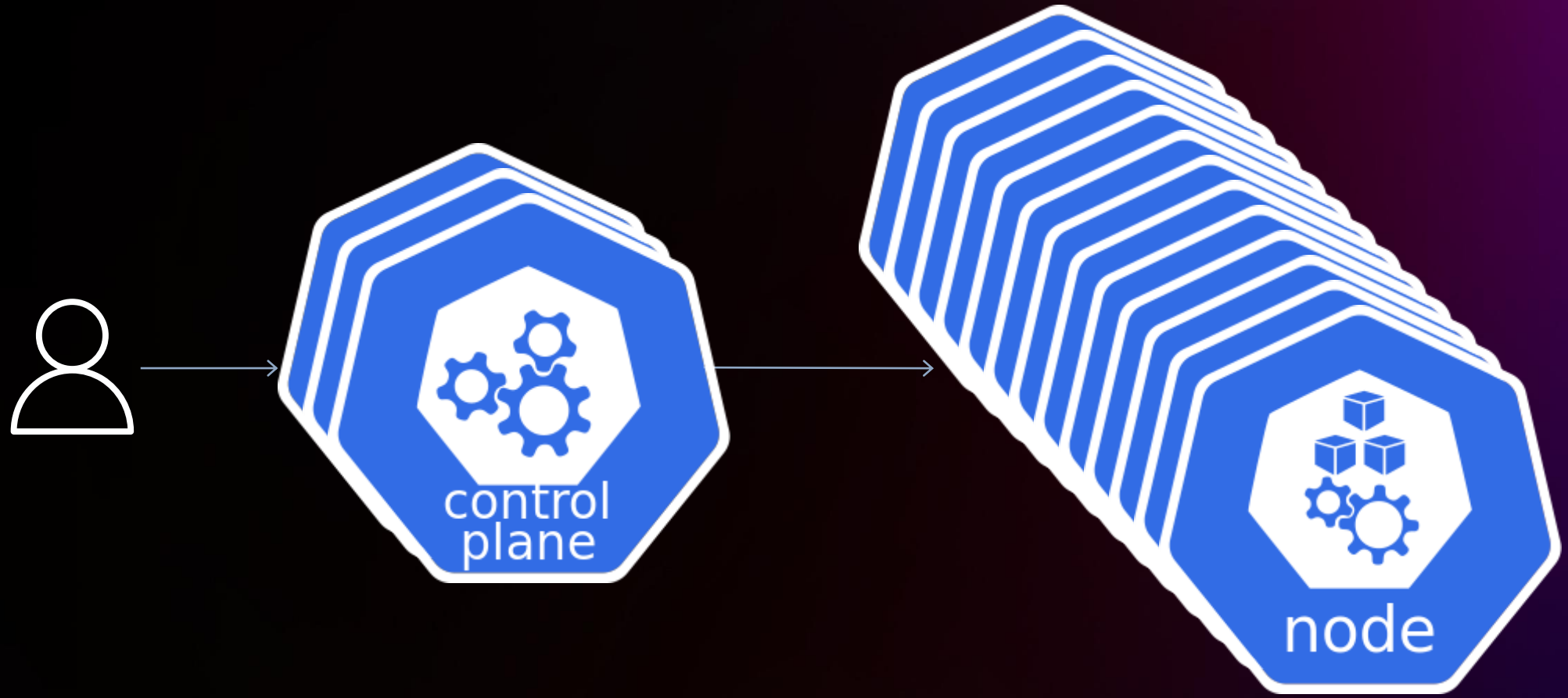
Single Cluster API Server NLB Endpoint
https://1122334455667788OOAABB.gr7.us-west-2.eks.amazonaws.com

```
$ kubectl apply -f my_application.yml

$ kubectl describe nodes my-node

$ kubectl get pods --all-namespaces

$ kubectl exec --stdin --tty my-pod -- /bin/sh

$ kubectl create deployment nginx --image=nginx

$ kubectl rollout restart deployment/frontend
```

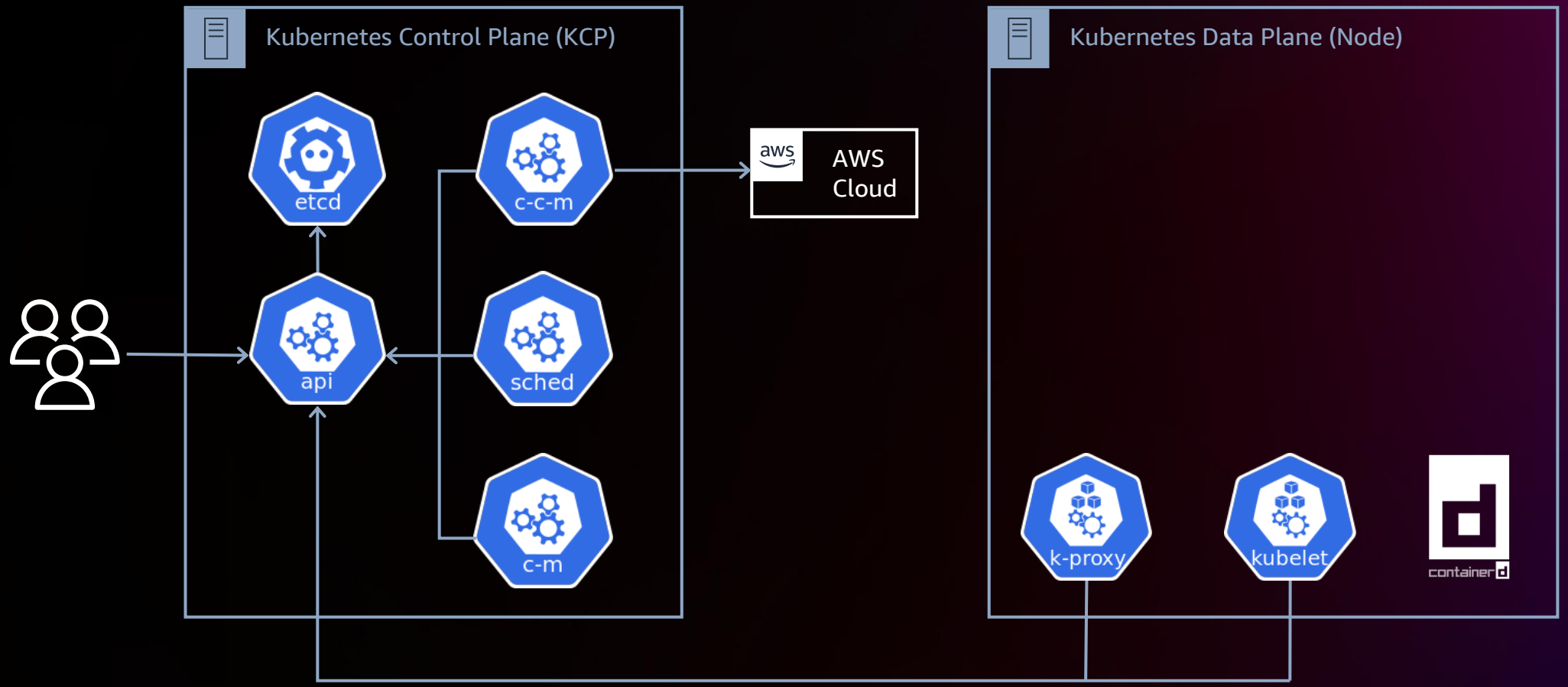
# How systems work

# Kubernetes high-level overview

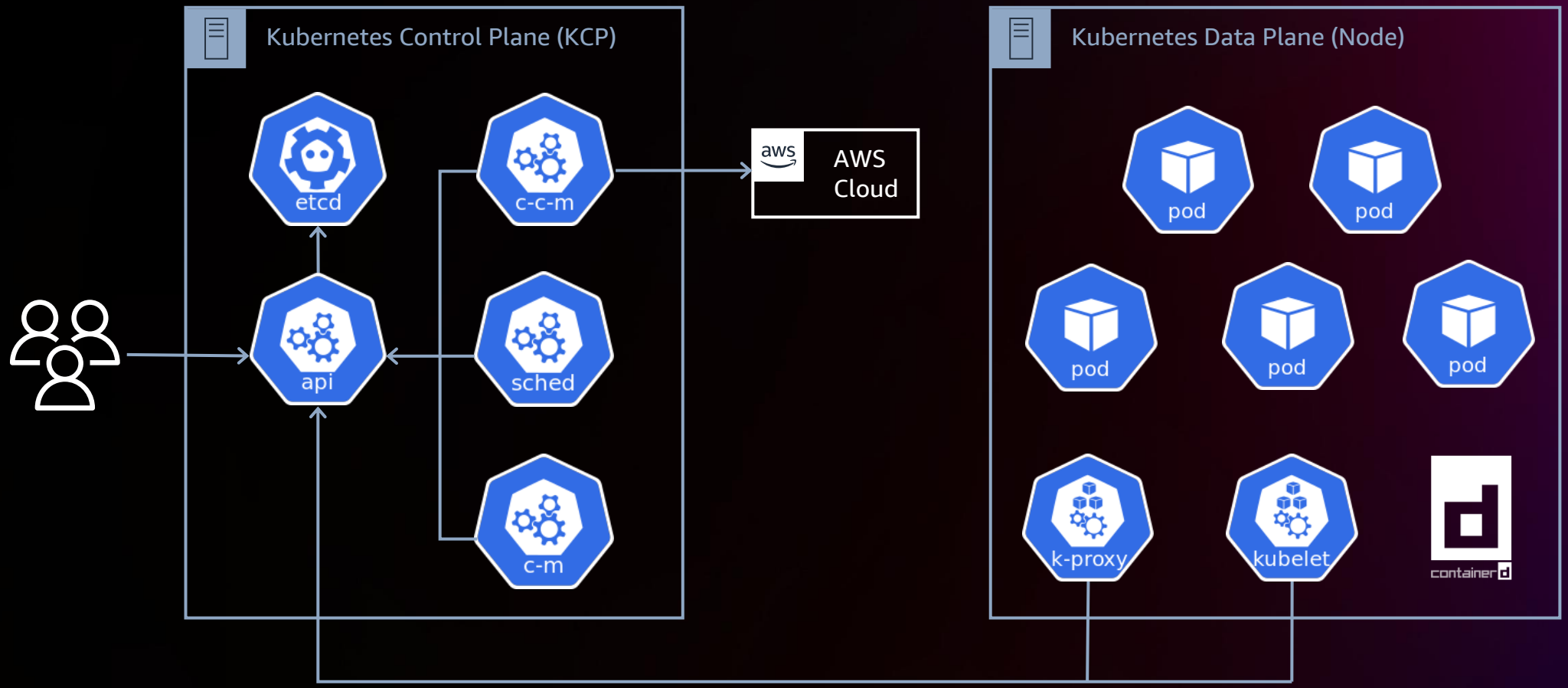
# Kubernetes high-level overview



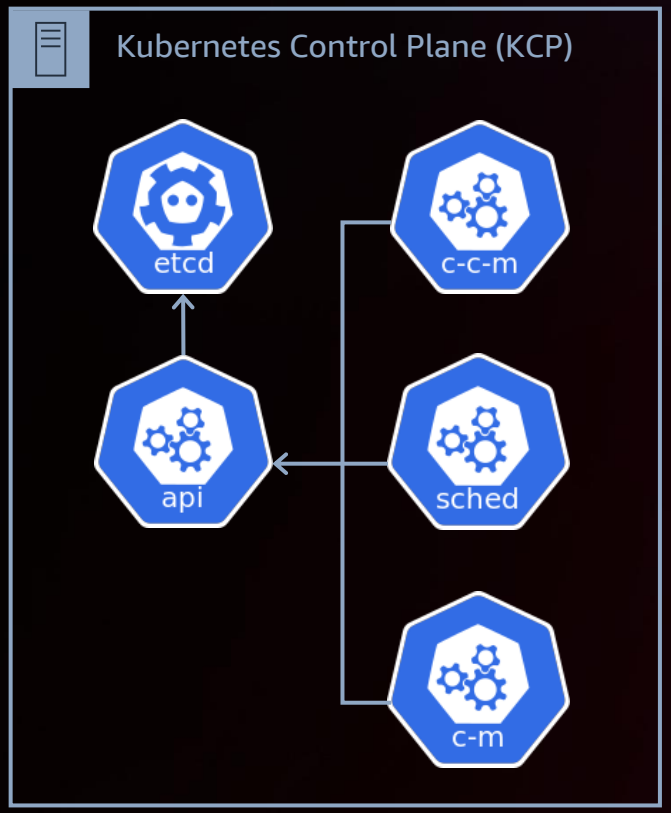
# Kubernetes high-level overview



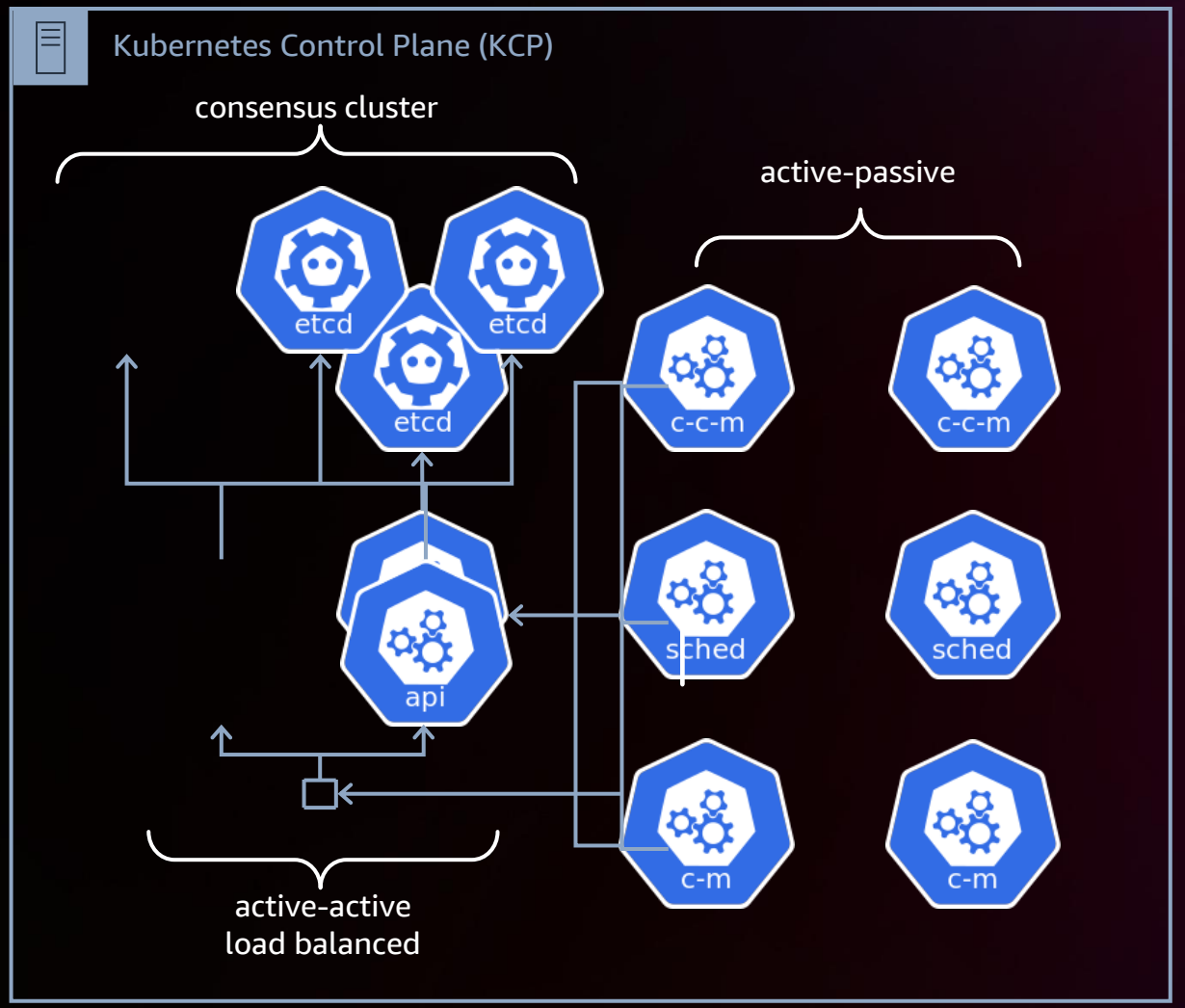
# Kubernetes high-level overview



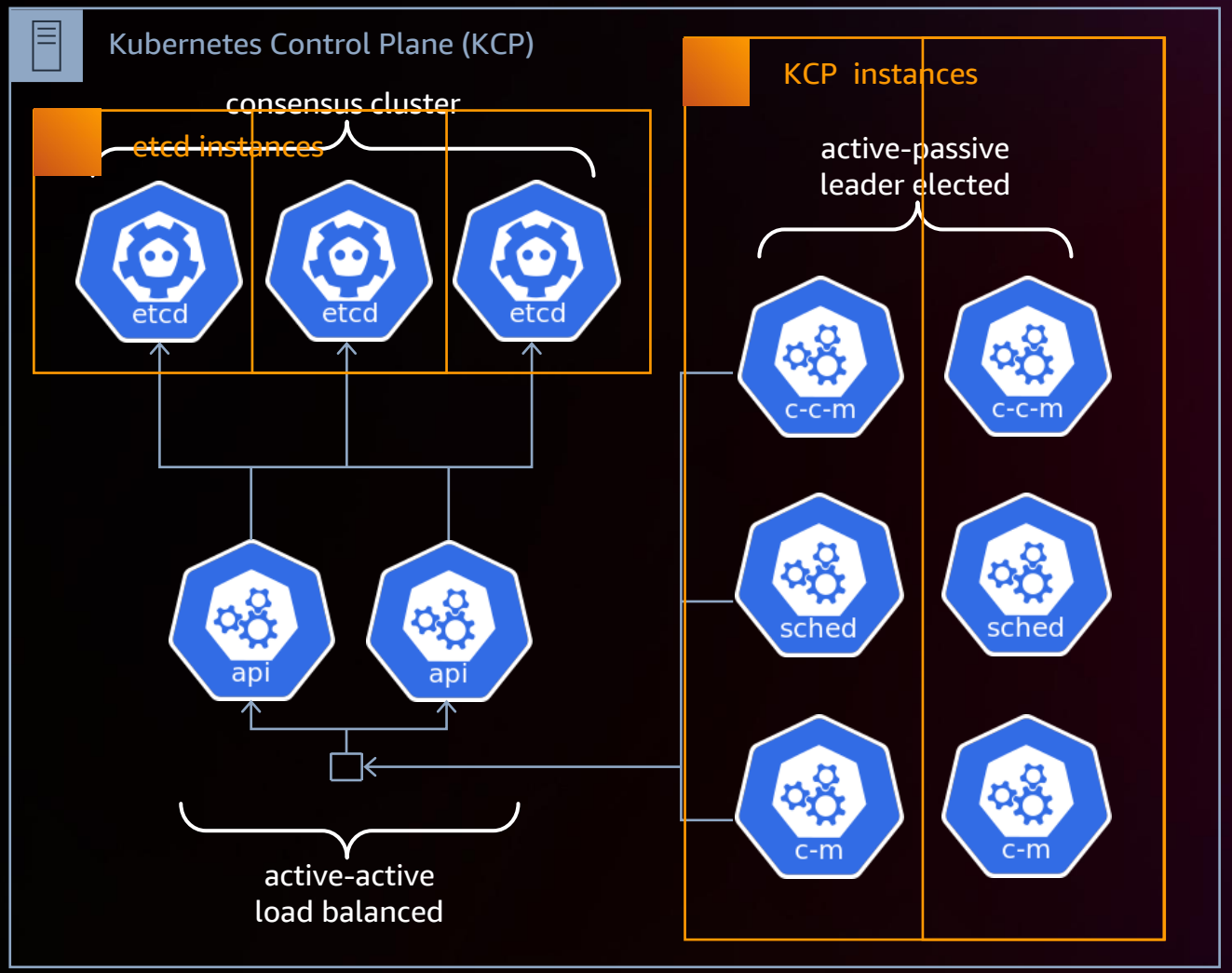
# Kubernetes high-level overview



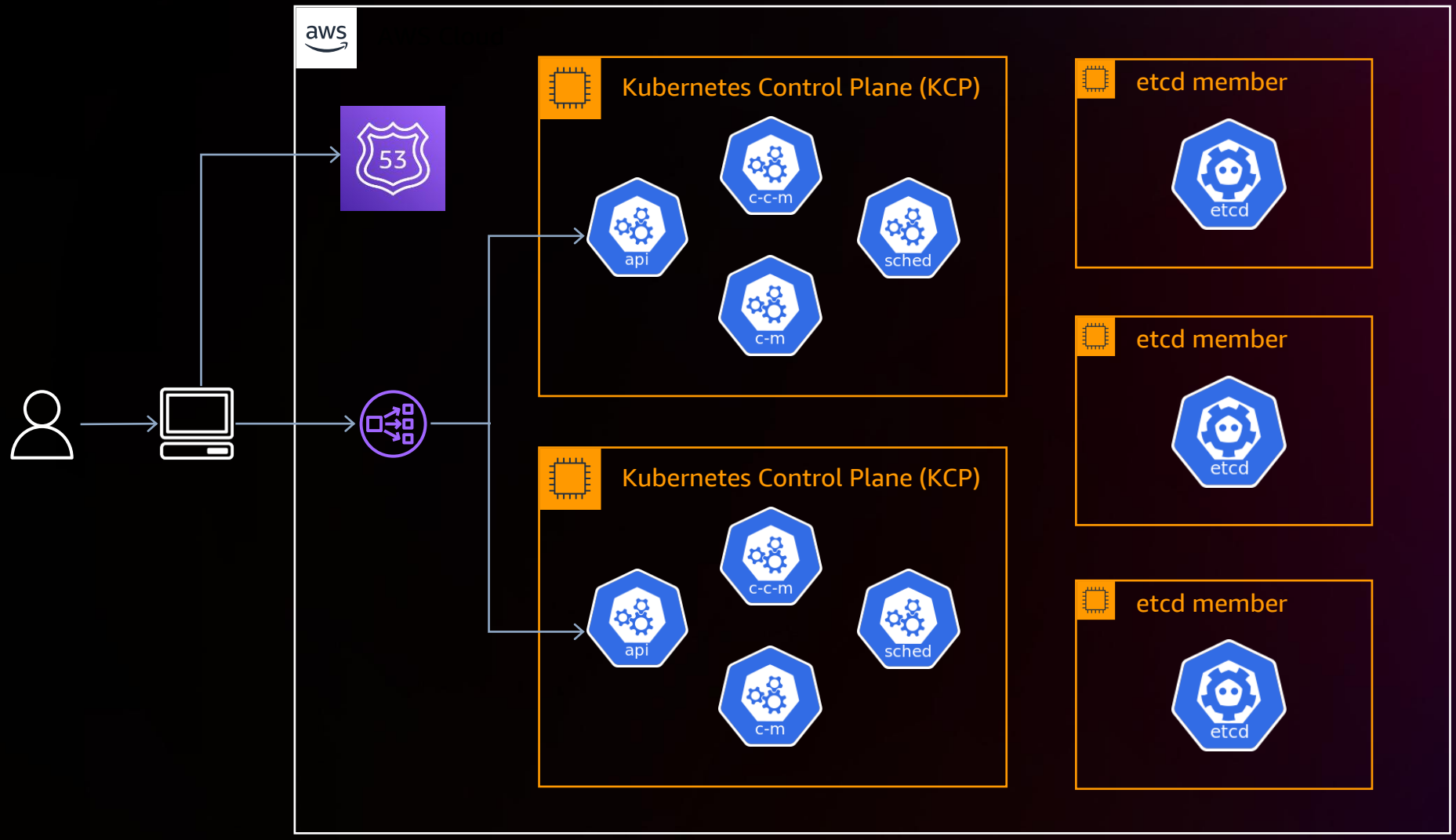
# Kubernetes high-level overview



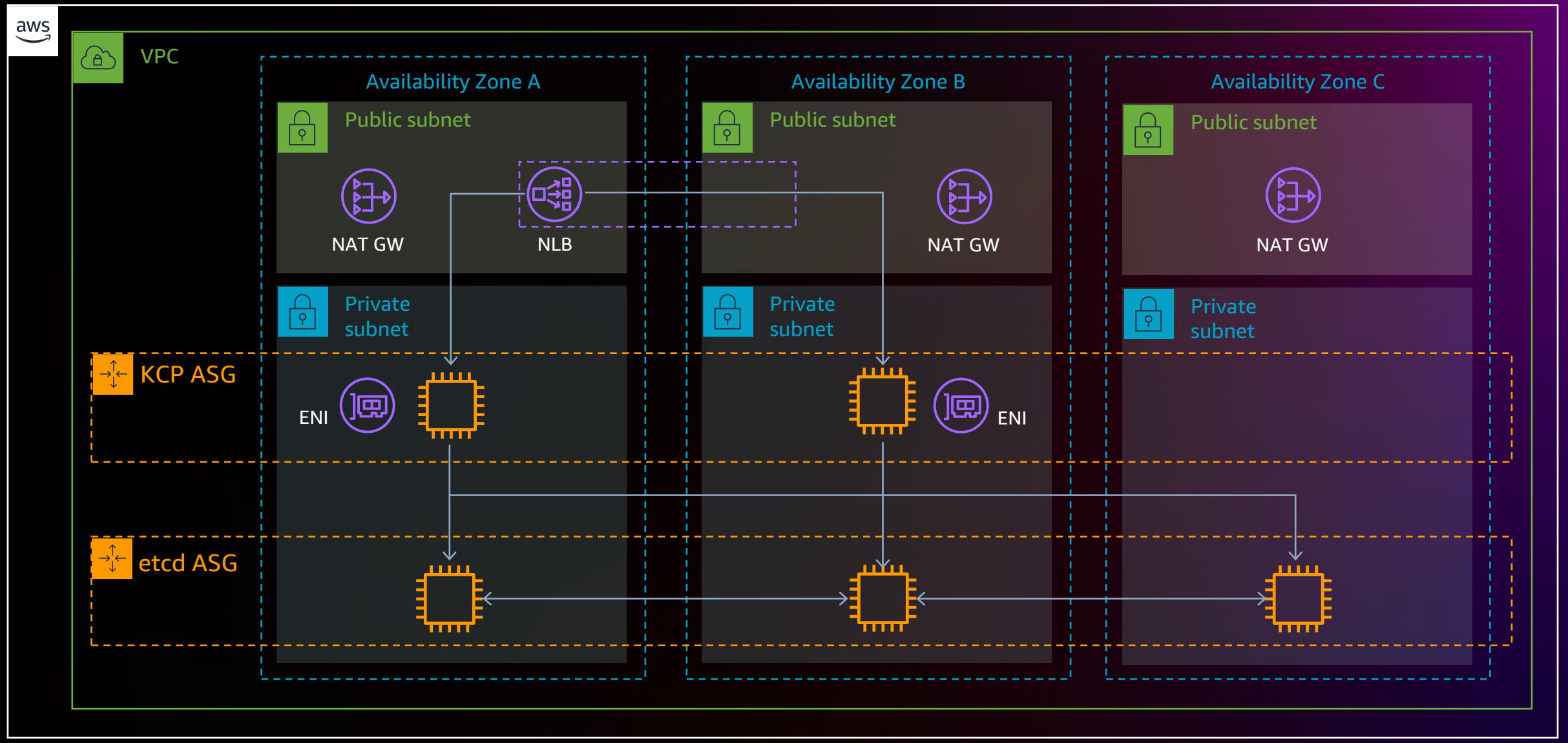
# Kubernetes high-level architecture in EKS



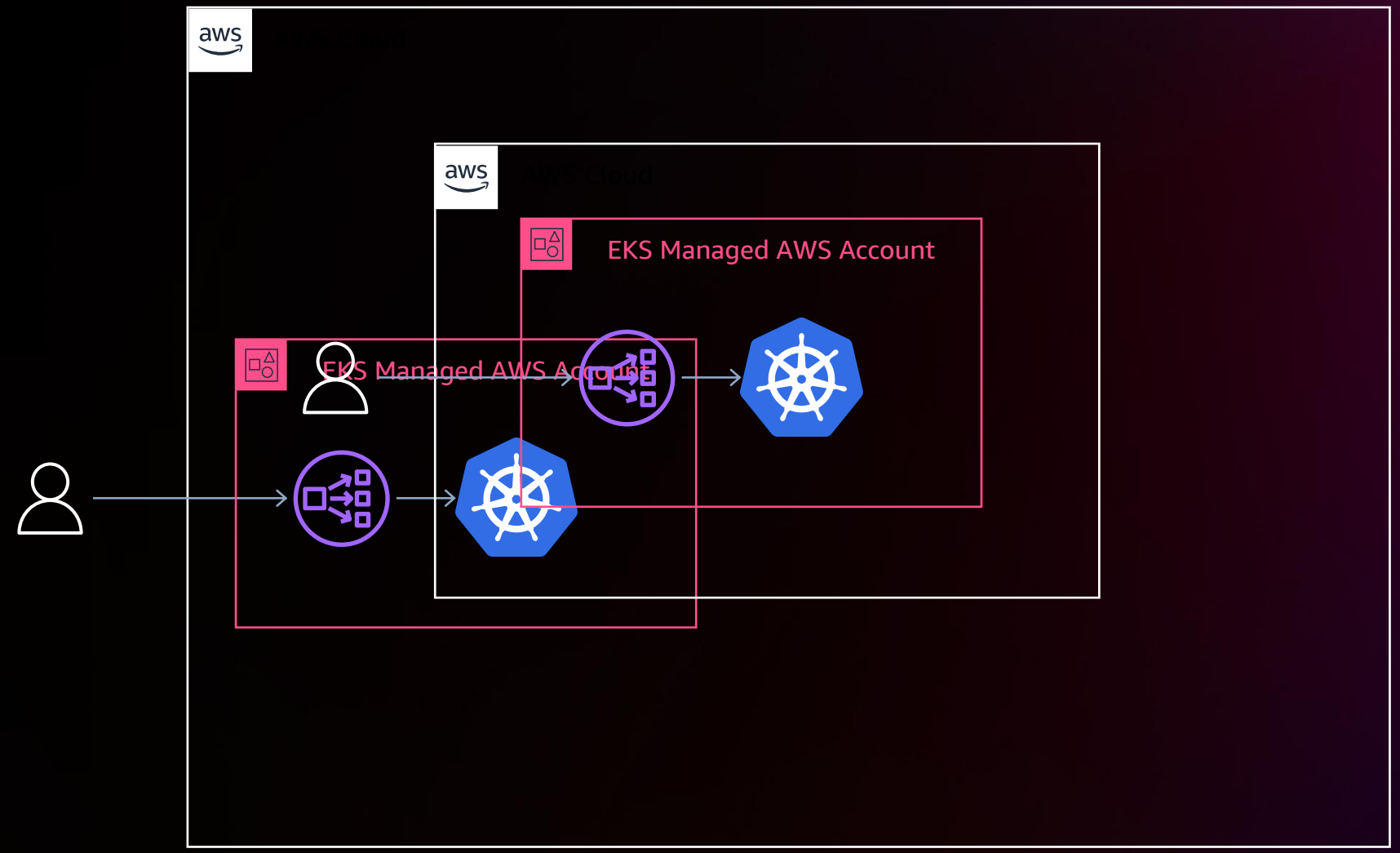
# EKS high-level architecture – Control Plane



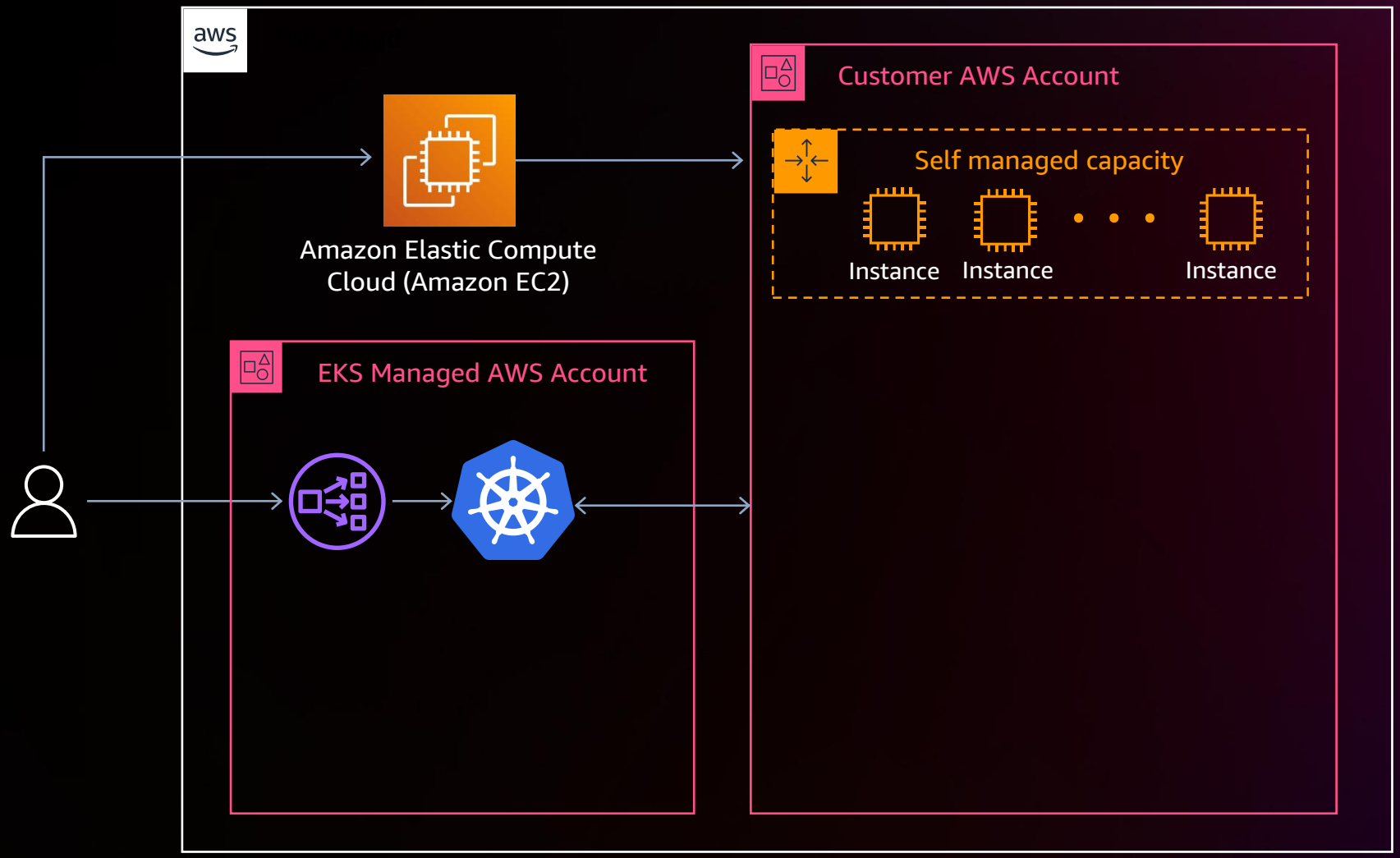
# EKS Architecture – Control Plane



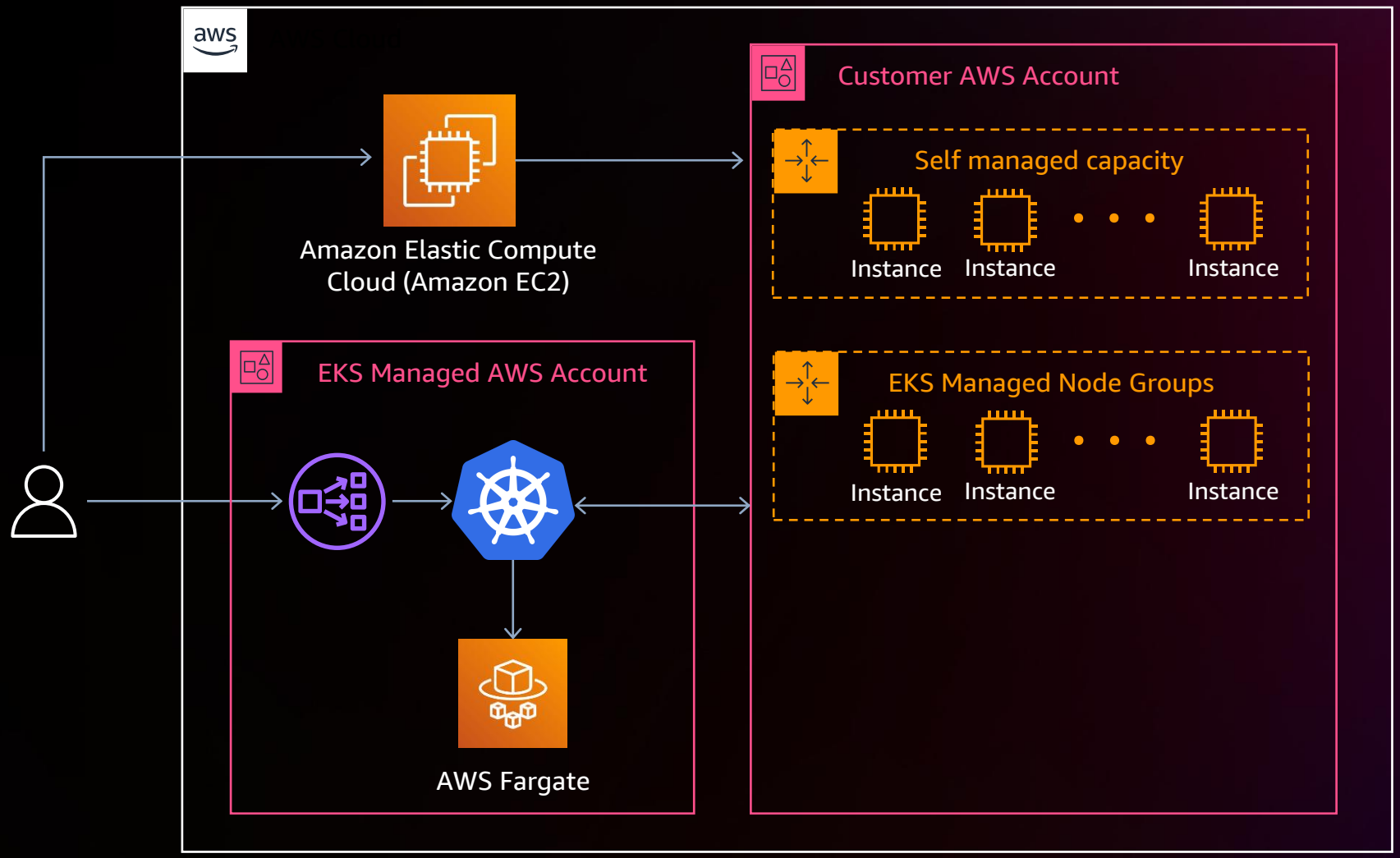
# EKS high-level architecture- Control Plane



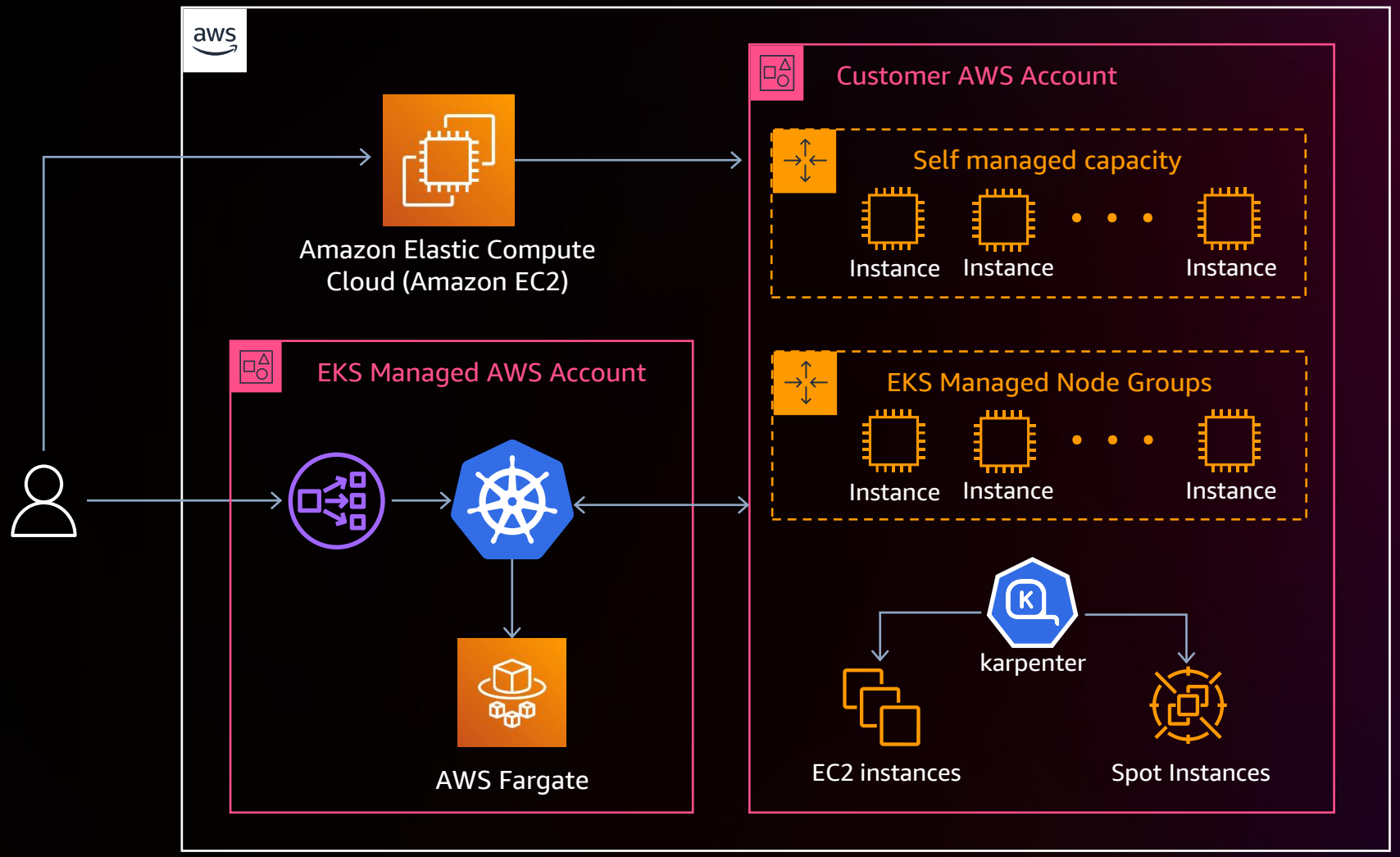
# EKS high-level architecture – Data Plane



# EKS high-level architecture – Data Plane

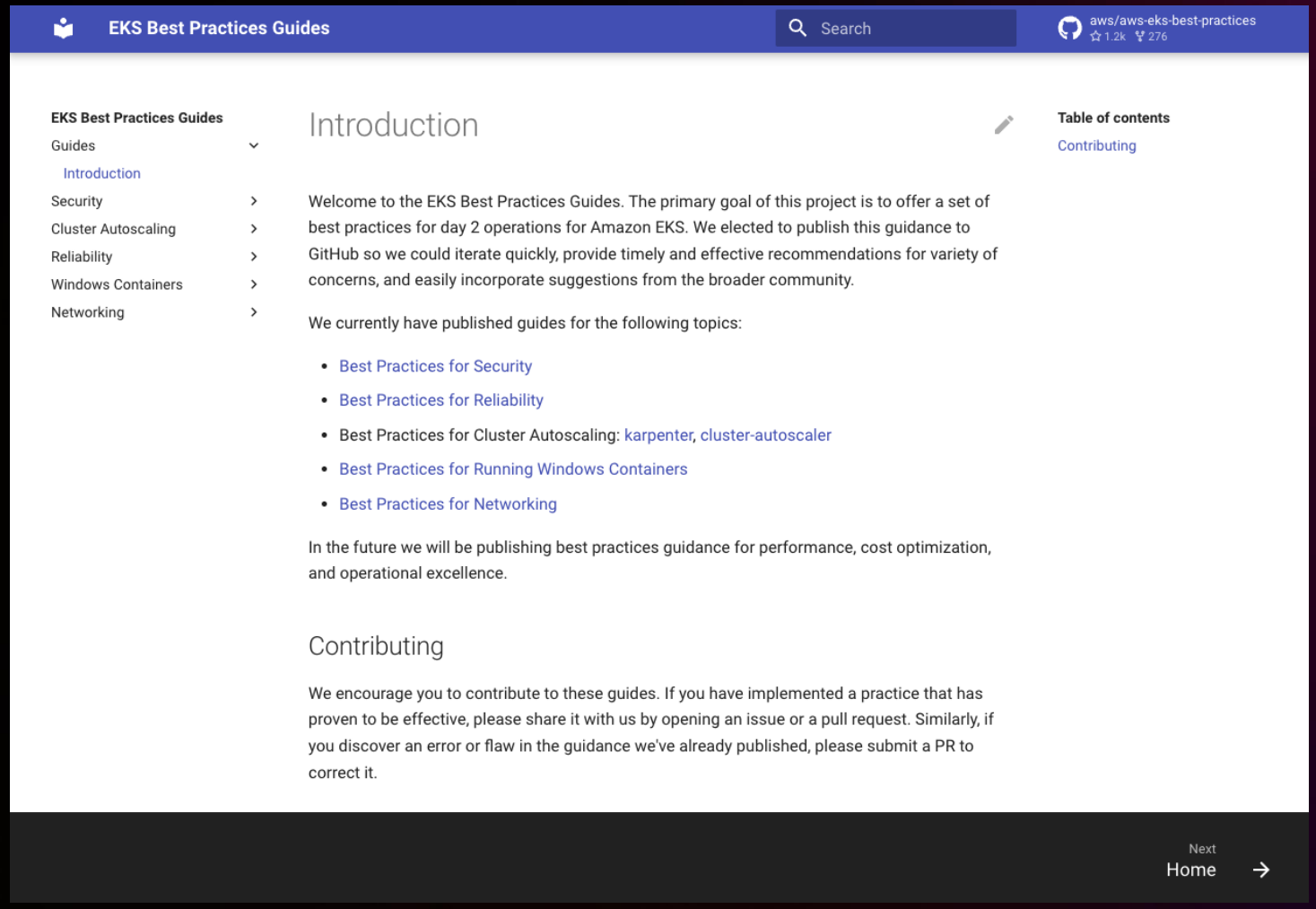


# EKS high-level architecture – Data Plane



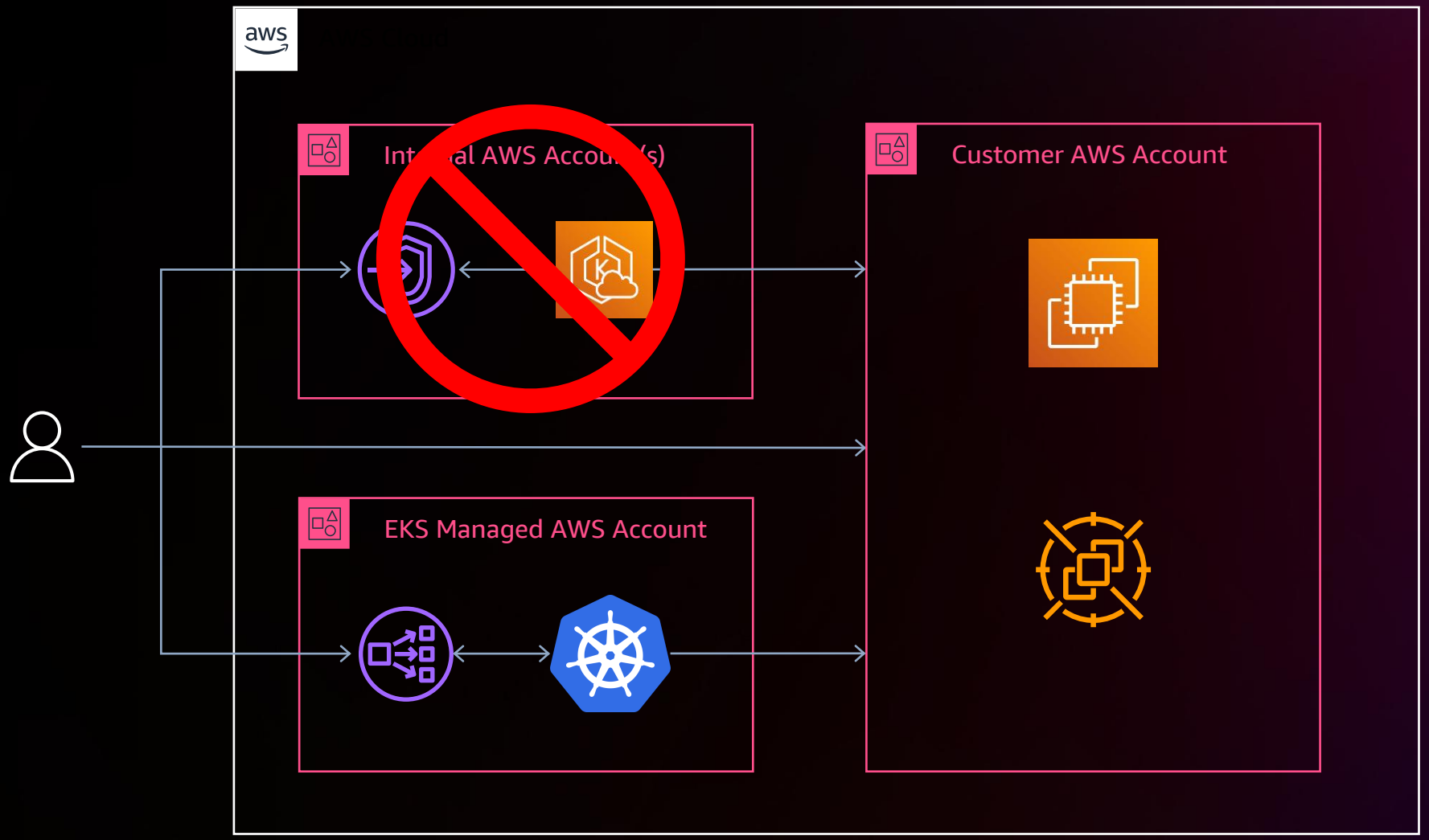
# EKS Best Practices Guides

Visit: https://aws.github.io/aws-eks-best-practices/



# How systems fail

# Independent failure domains

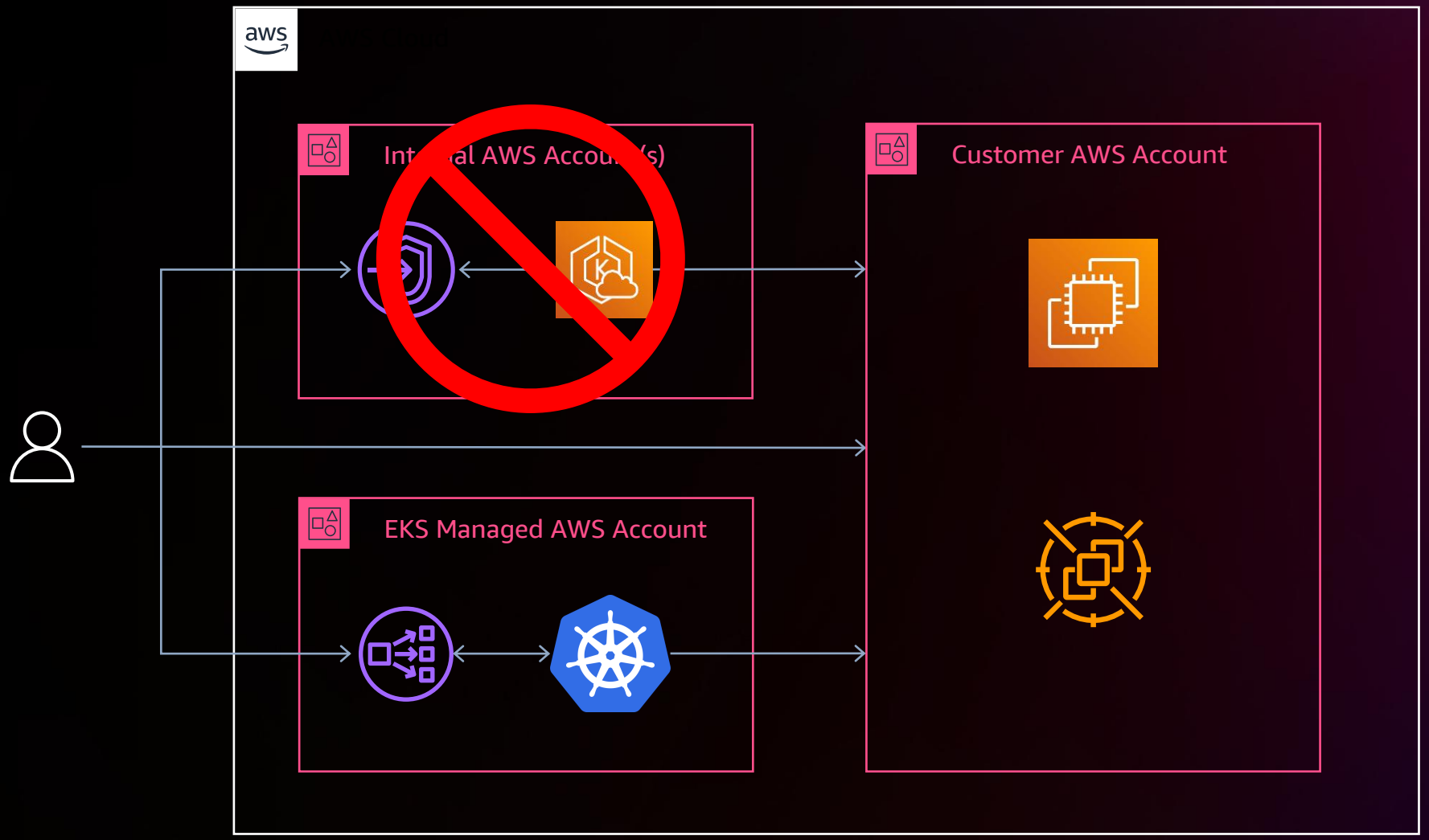


# EKS and Kubernetes interactions

```
$ aws eks list-clusters

$ aws eks create-cluster --name prod <...>

$ aws eks describe-cluster --name prod

$ aws eks get-token --cluster-name prod

$ aws eks create-nodegroup --cluster-name prod
    --nodegroup-name frontend
```

```
$ kubectl apply -f my_application.yml

$ kubectl describe nodes my-node

$ kubectl get pods --all-namespaces

$ kubectl exec --stdin --tty my-pod -- /bin/sh

$ kubectl create deployment nginx --image=nginx

$ kubectl rollout restart deployment/frontend
```

# Independent failure domains



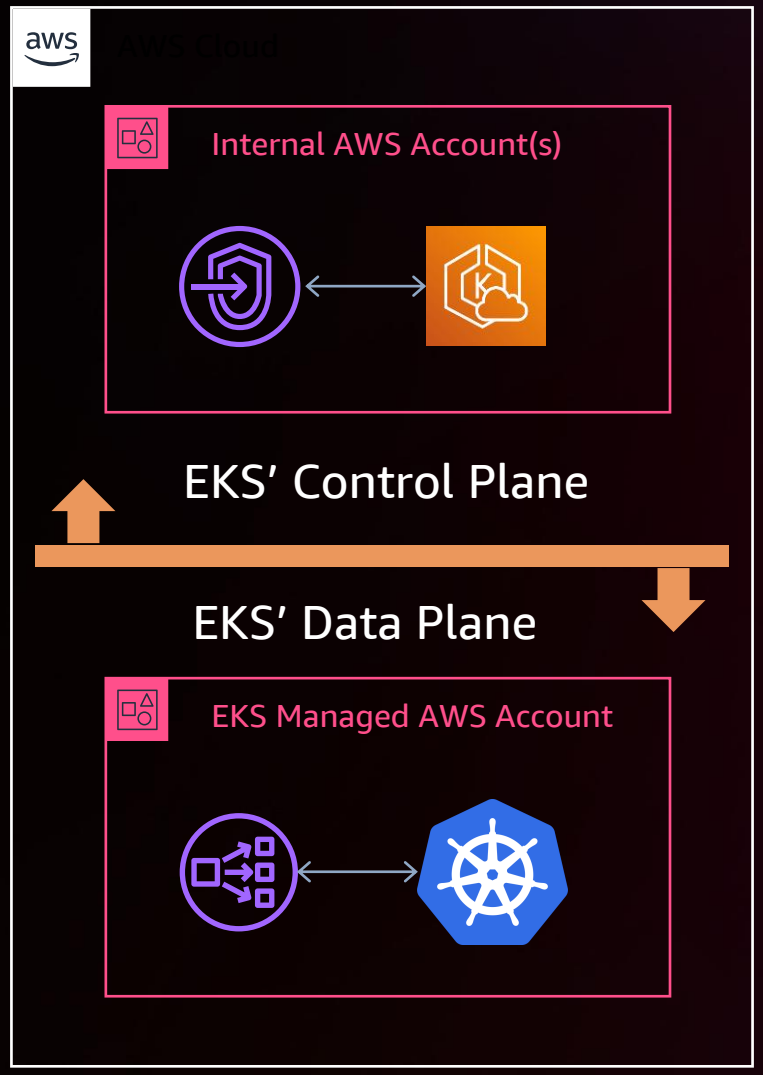
“In a statically stable design, the overall system keeps working when a dependency becomes impaired. Perhaps the system doesn’t see any updated information… However, everything it was doing before the dependency became impaired continues to work.”

**Becky Weiss and Mike Furr**

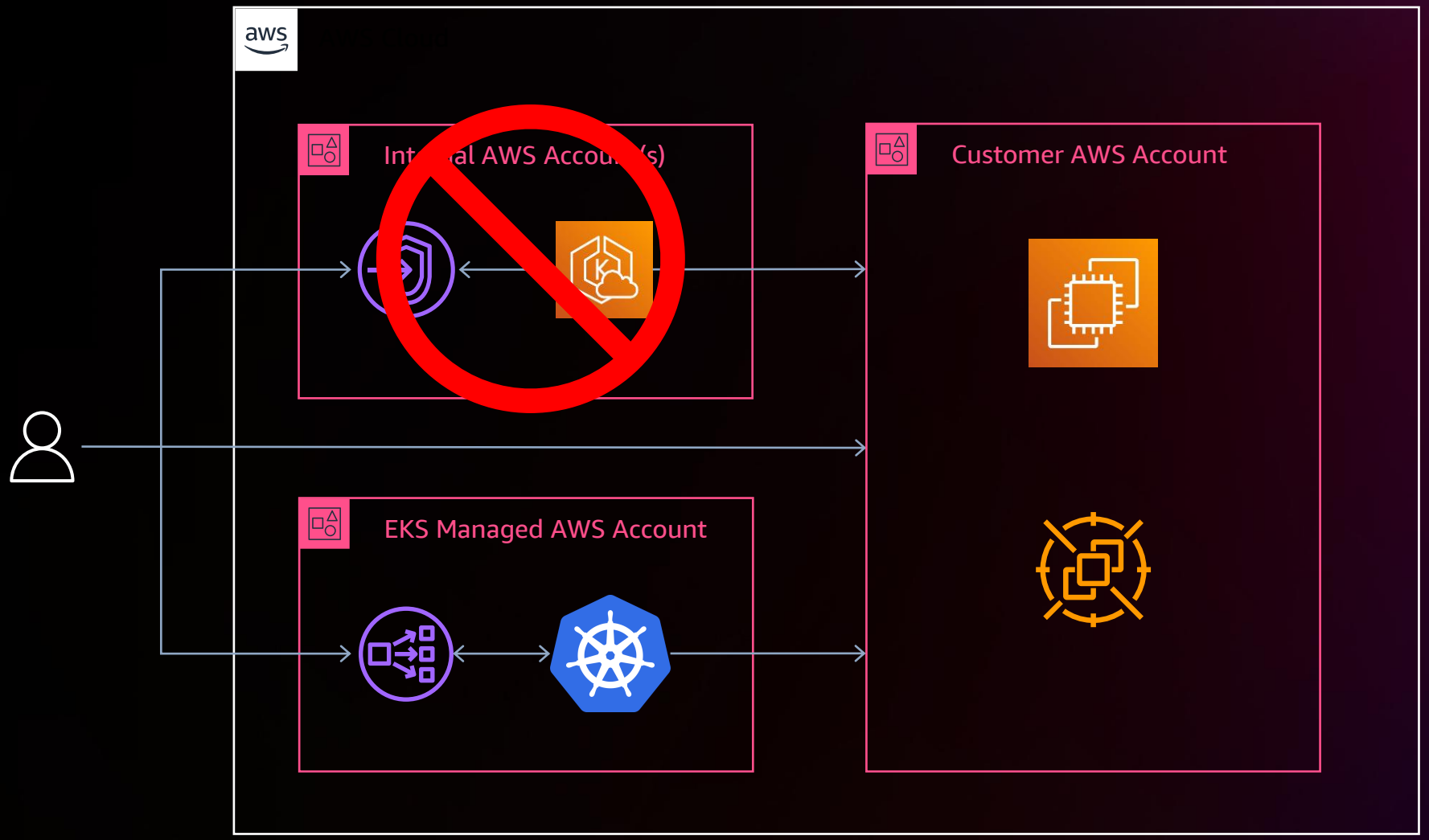
AWS Sr. Principal Engineers

**The Amazon Builders' Library: Static stability using Availability Zones**

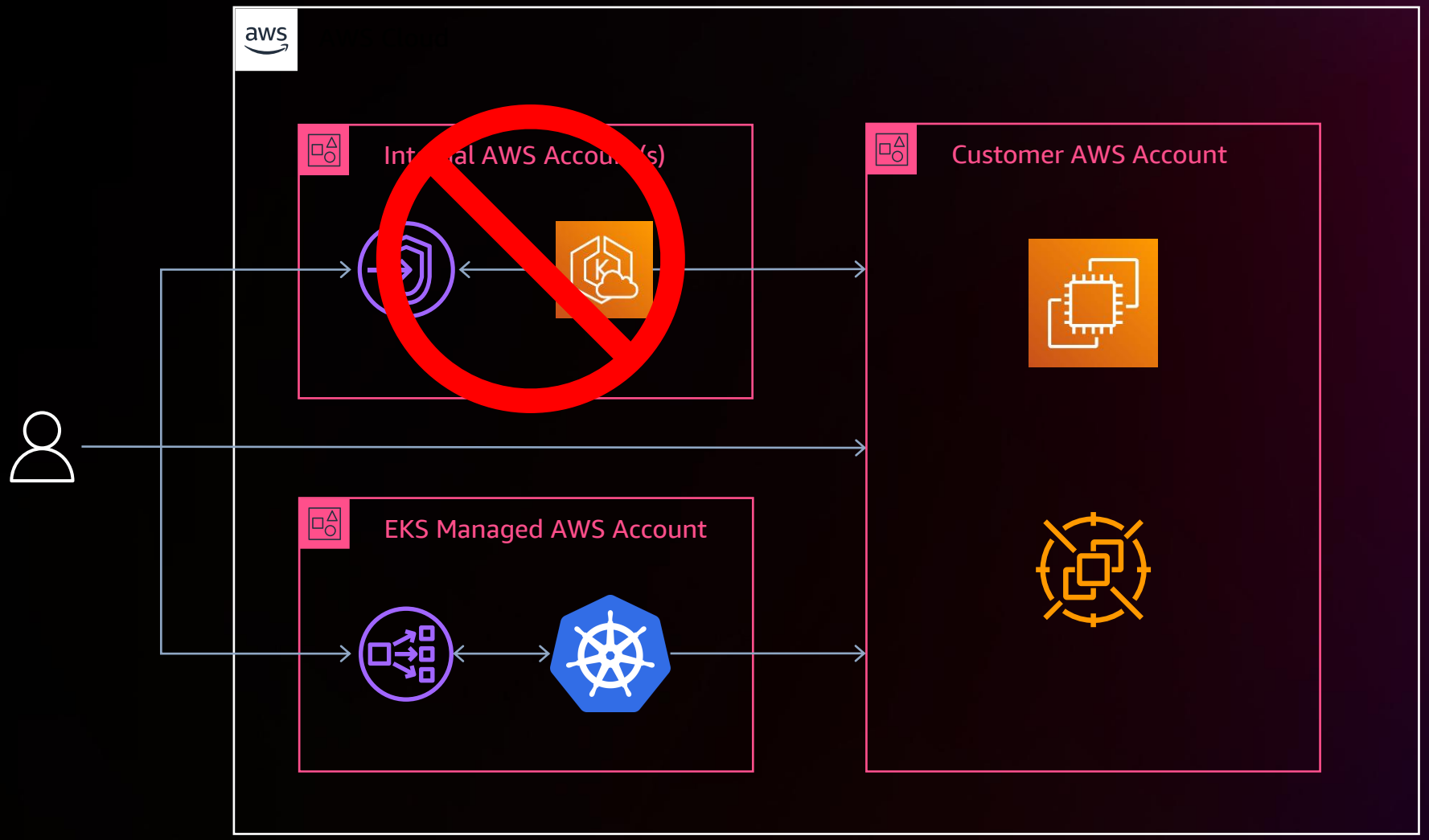
# EKS Control Plane and Data Plane



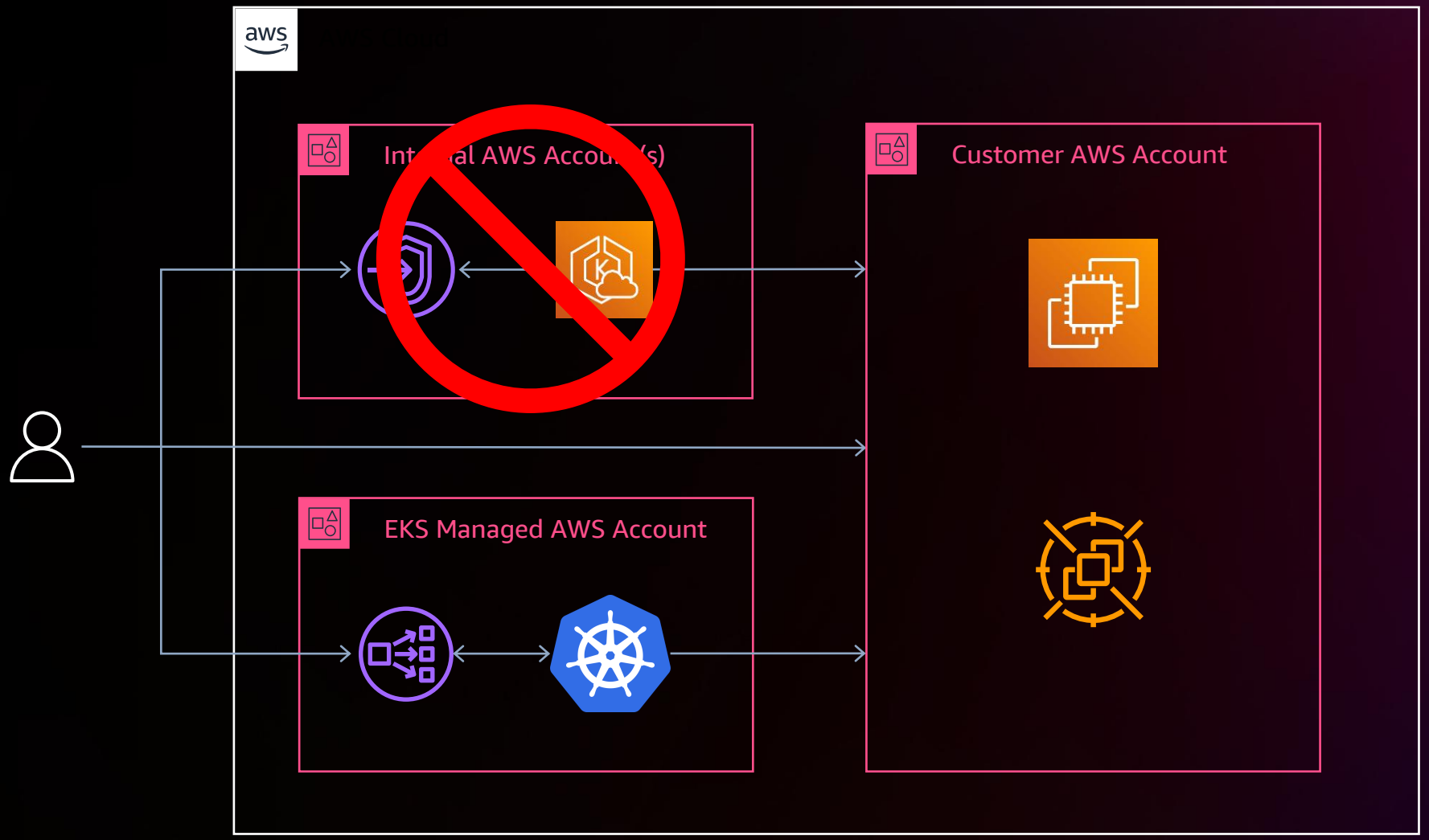
# Independent failure domains



# Independent failure domains



# Independent failure domains



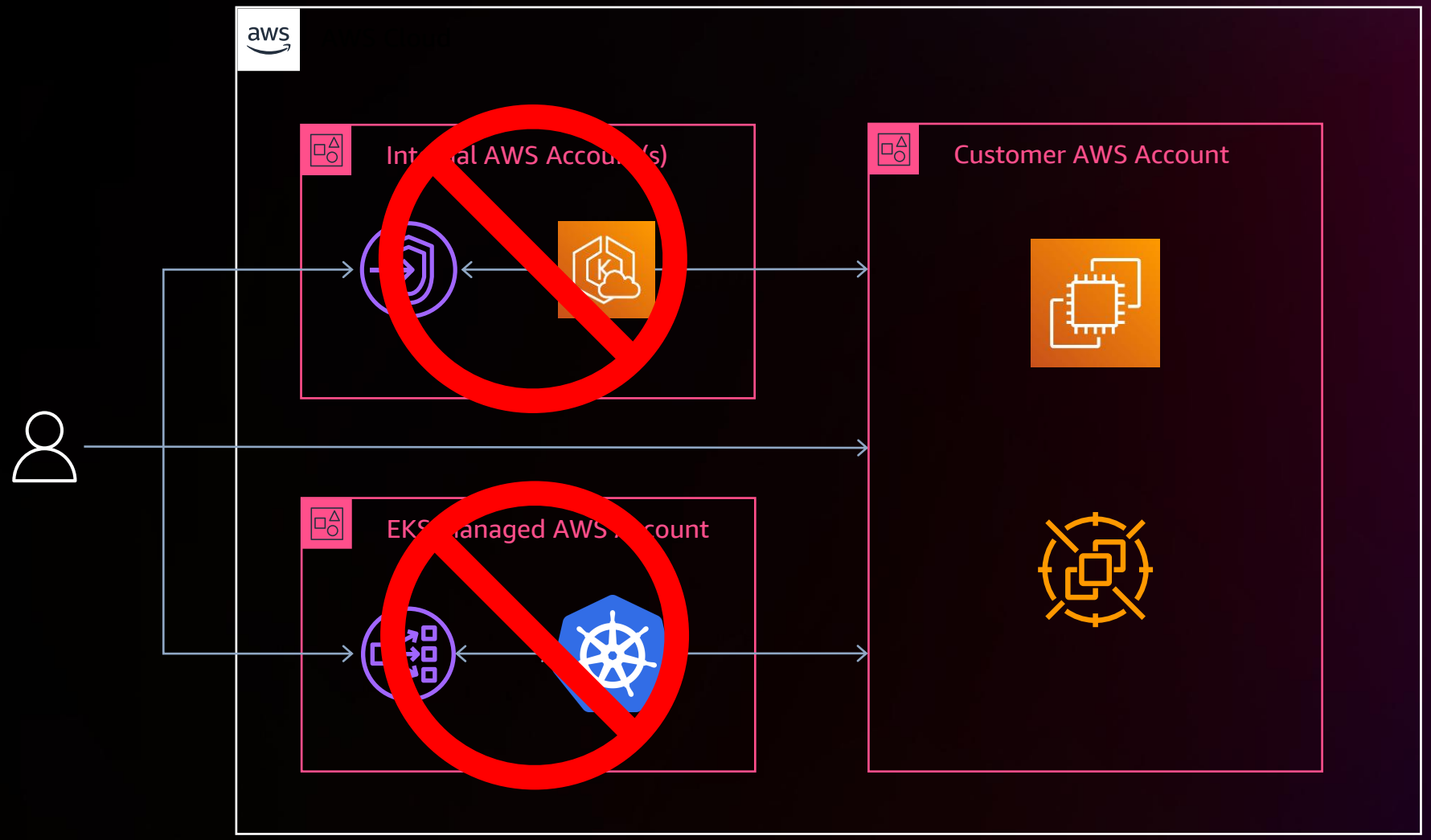
The Amazon Builders' Library

# Timeouts, retries, and backoff with jitter

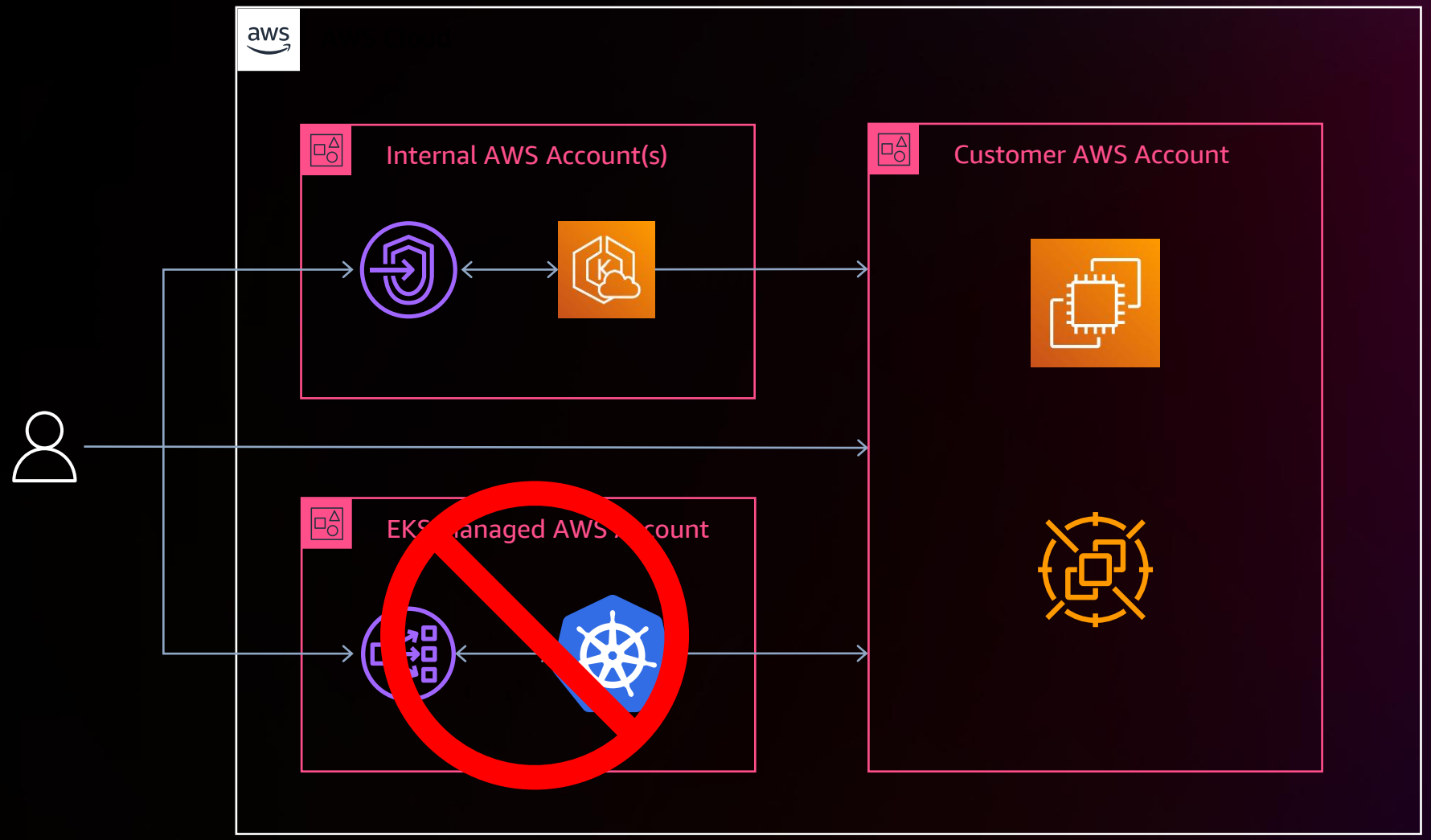
**Marc Brooker**

AWS Sr. Principal Engineer

# Kubernetes Control Plane failures



# Kubernetes Control Plane failures

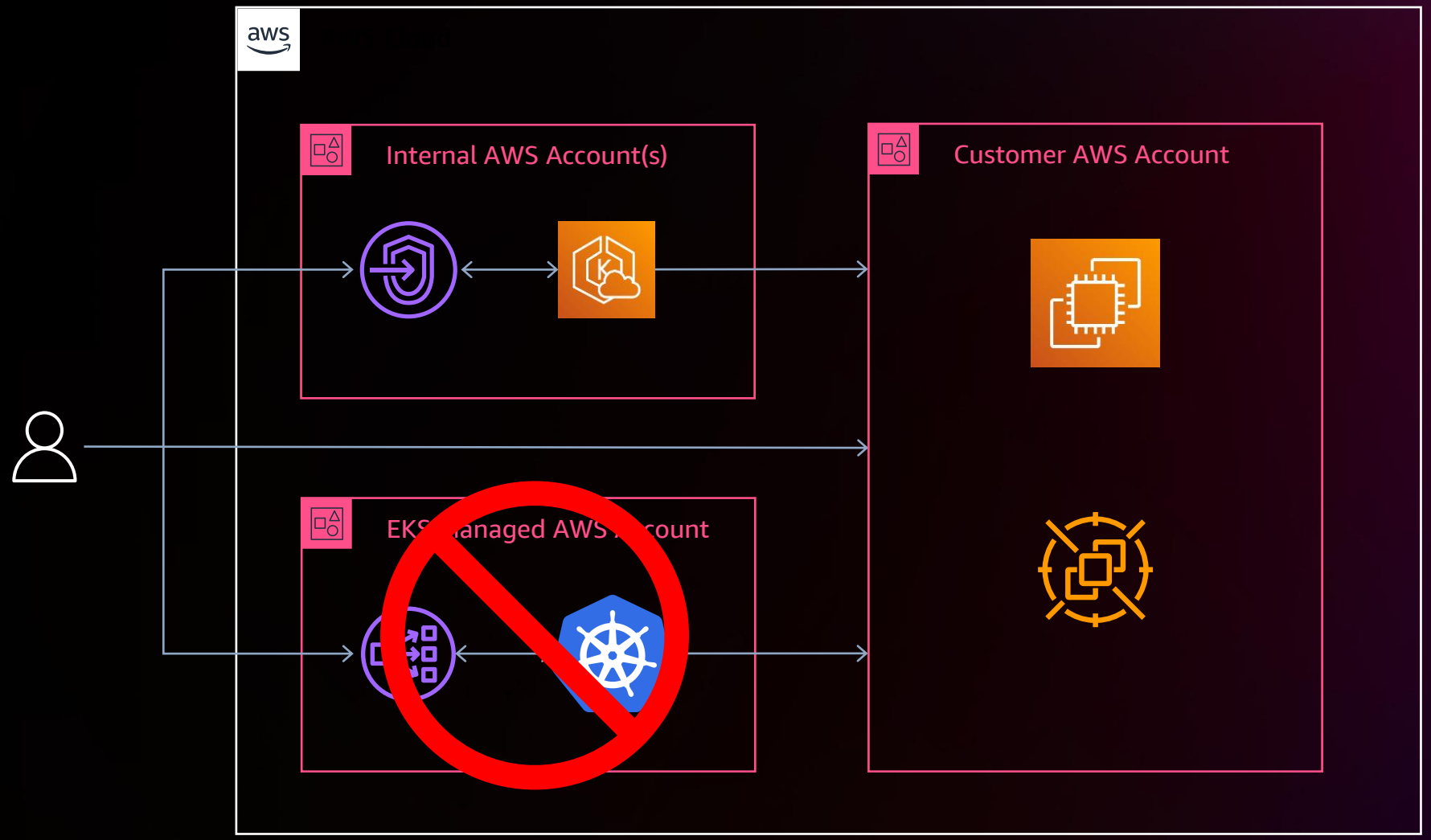


# EKS and Kubernetes interactions

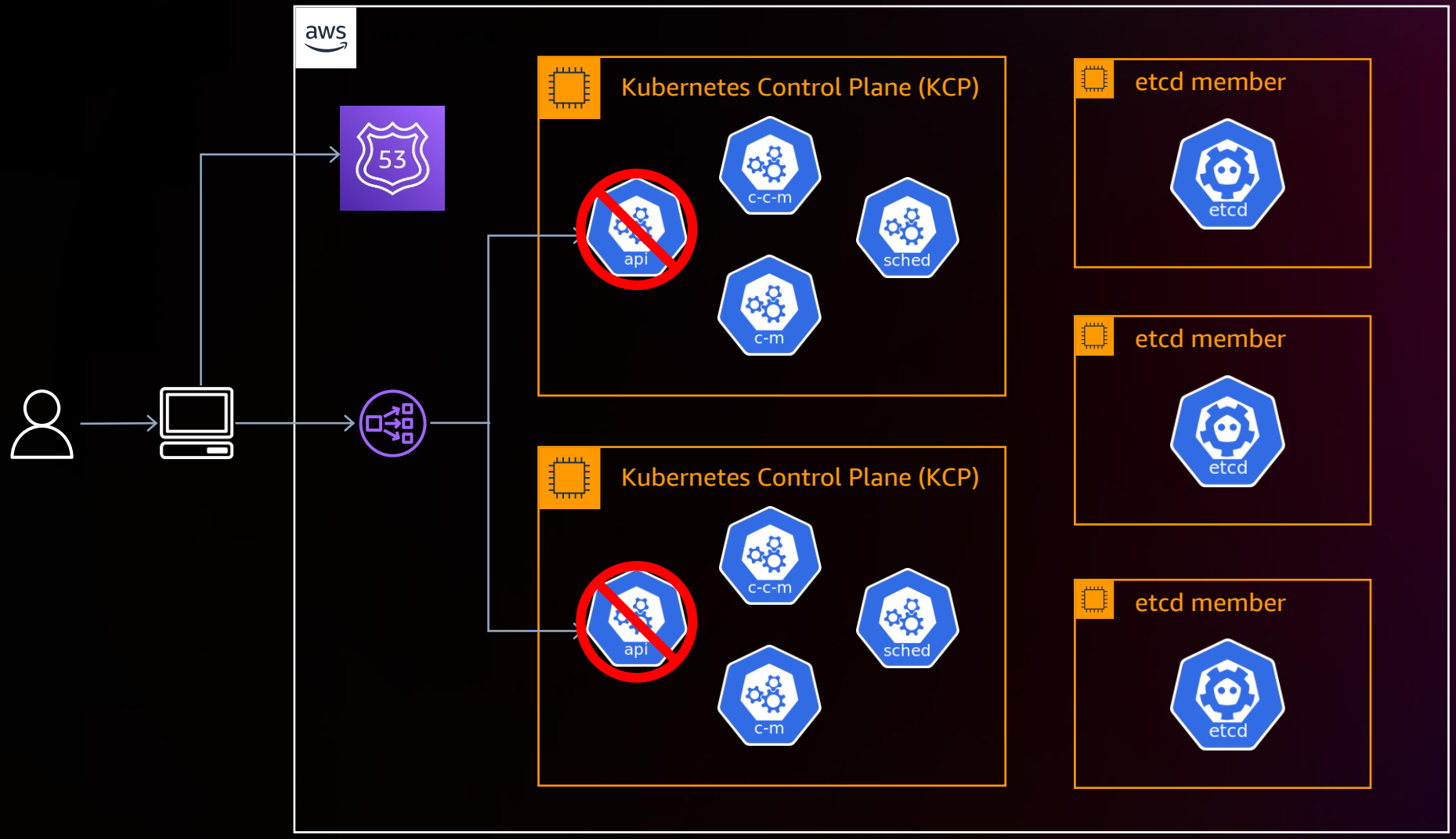
```
$ aws eks list-clusters

$ aws eks create-cluster --name prod <…>

$ aws eks describe-cluster --name prod

$ aws eks get-token --cluster-name prod

$ aws eks create-nodegroup --cluster-name prod
      --nodegroup-name frontend <…>
```

```
$ kubectl appl    my_applic    n.yml

$ kubectl  esc    e nodes my-node

$ kubec    get pods    ll-namespaces

$ kubec    exec --stdin -    v my-pod     /bin/sh

$ kubectl    eate deployment n    x    mage=nginx

$ kubectl roll    restart d    yment/frontend
```

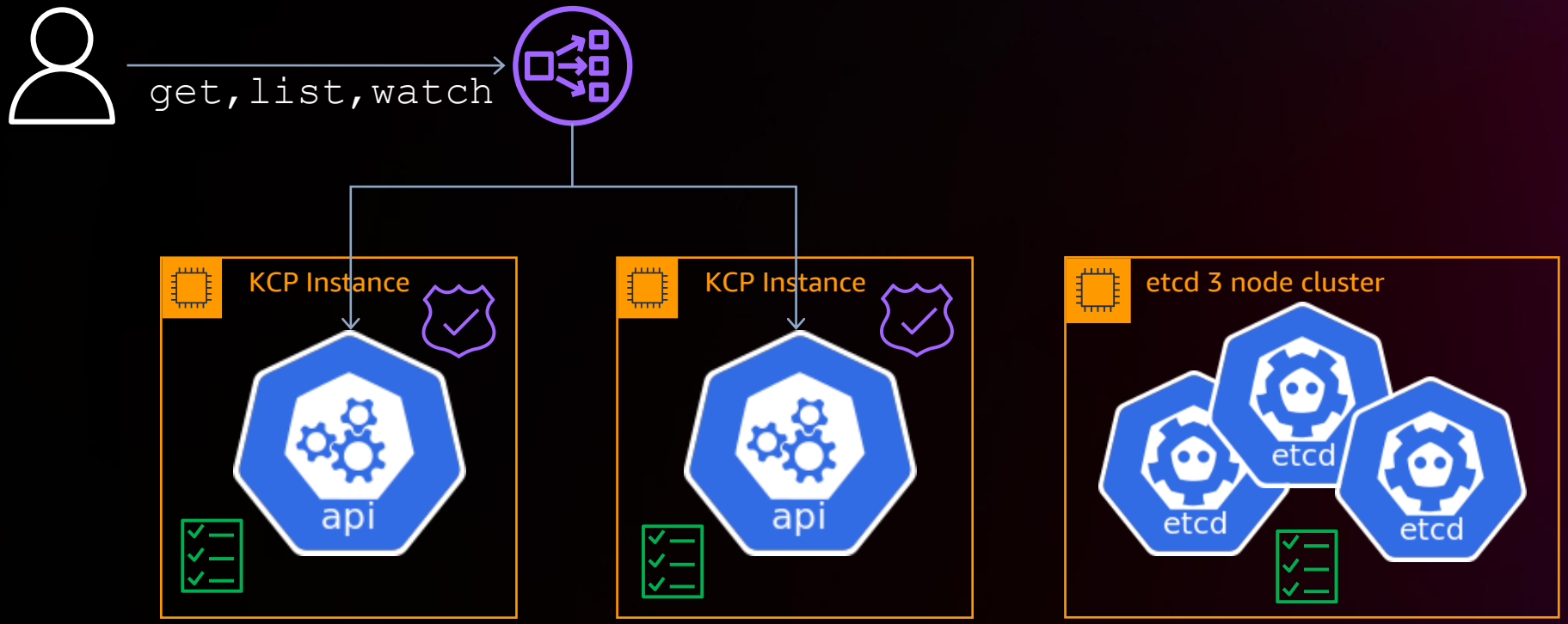
# Kubernetes Control Plane failures



# Control Plane and shared fate

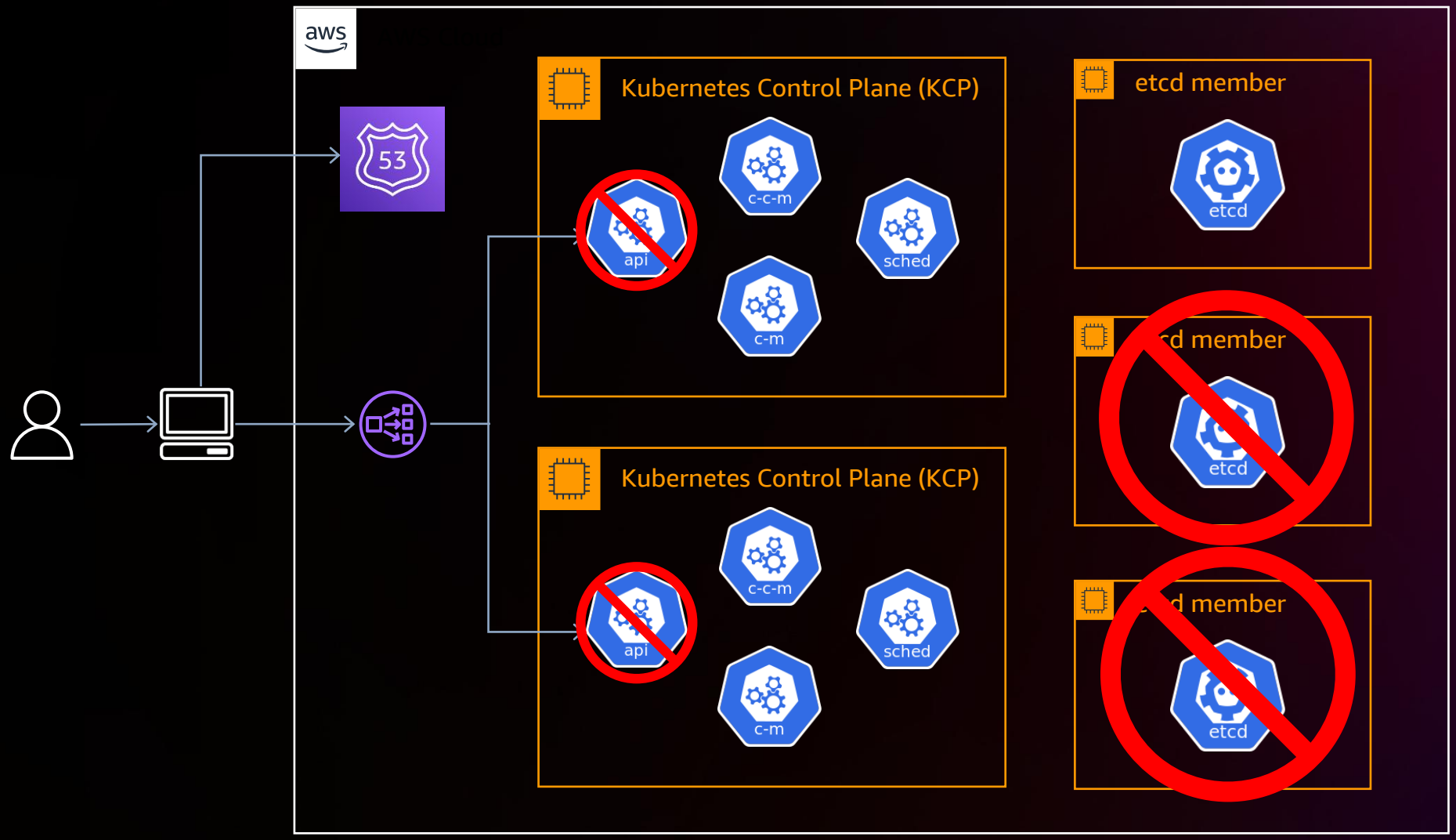


# Control Plane and shared fate

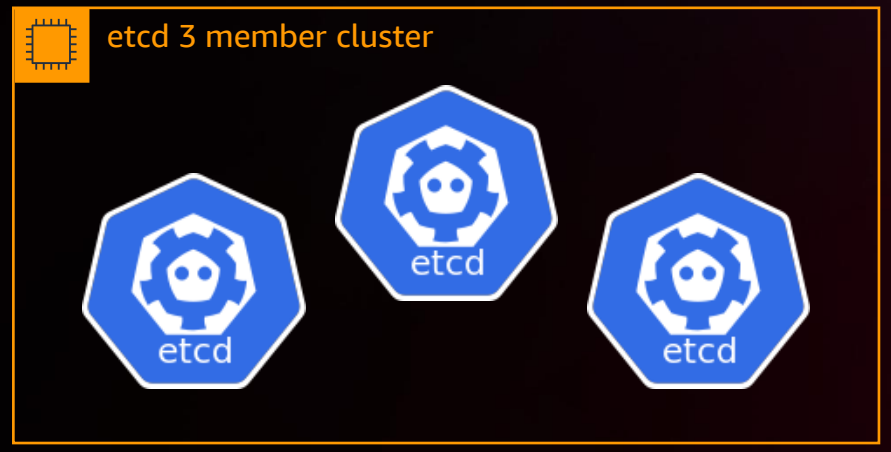



EKS Internal dashboards

# Kubernetes Persistent Data (etcd)



# Kubernetes Persistent Data (etcd)



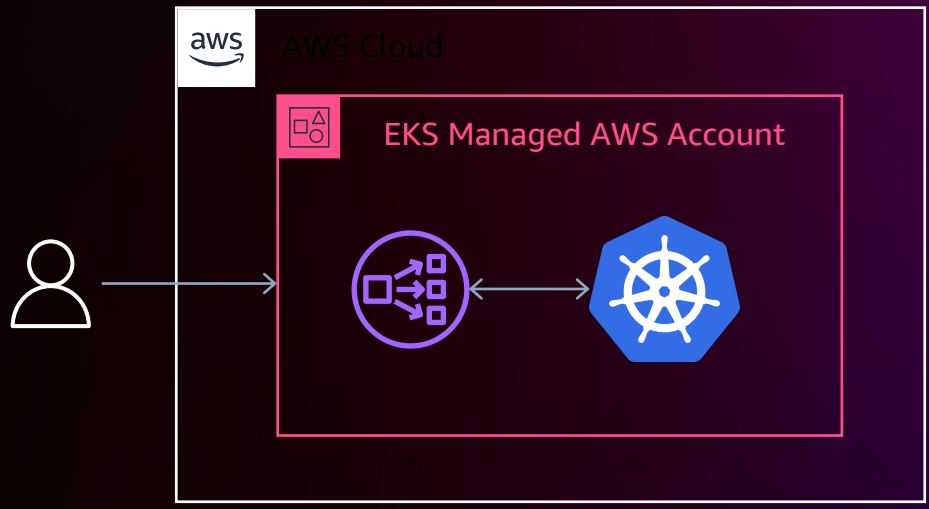

Track the etcd used storage size: `etcd_db_total_size_in_bytes`

# Kubernetes Cluster backup, restore, migration

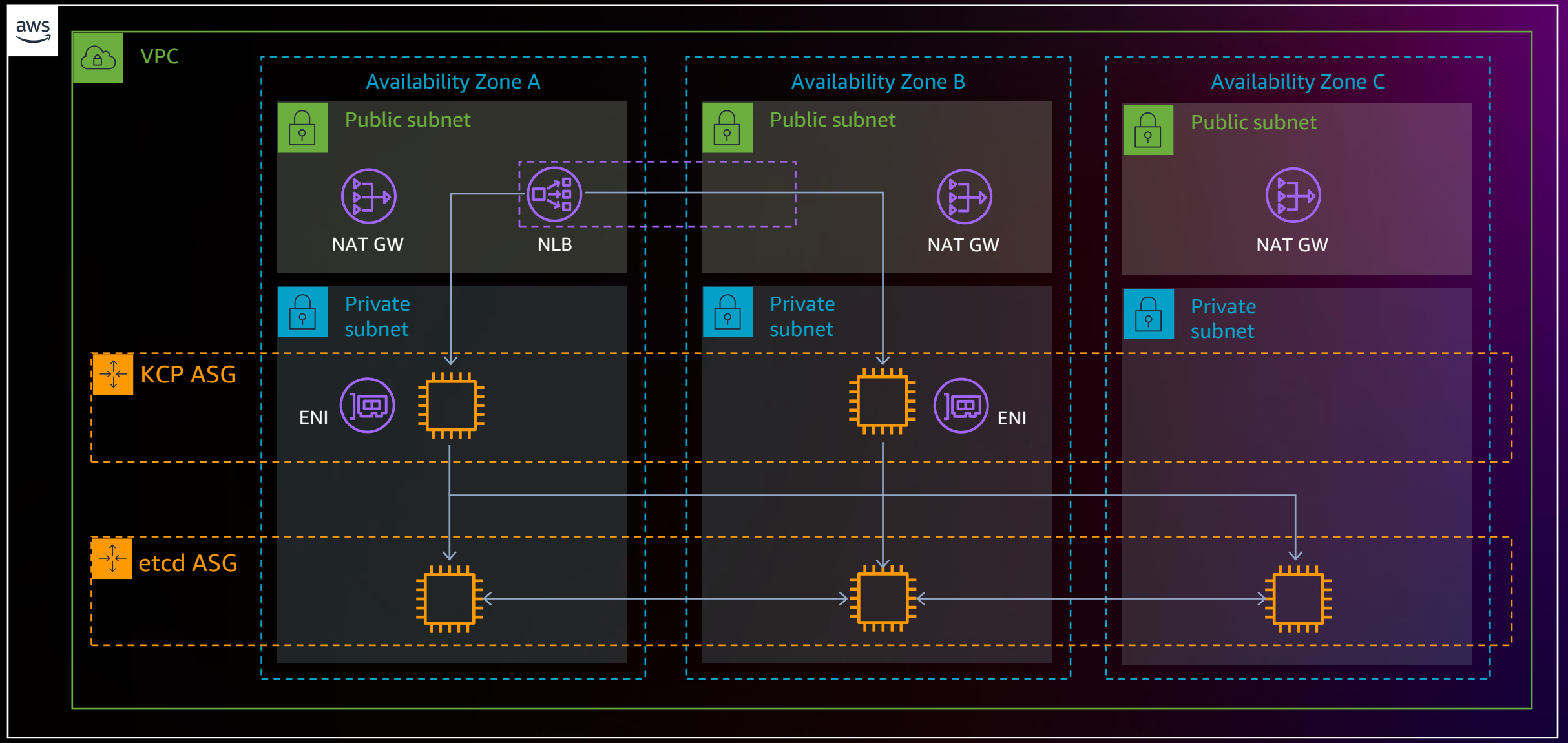
AWS Solutions

Open source solutions

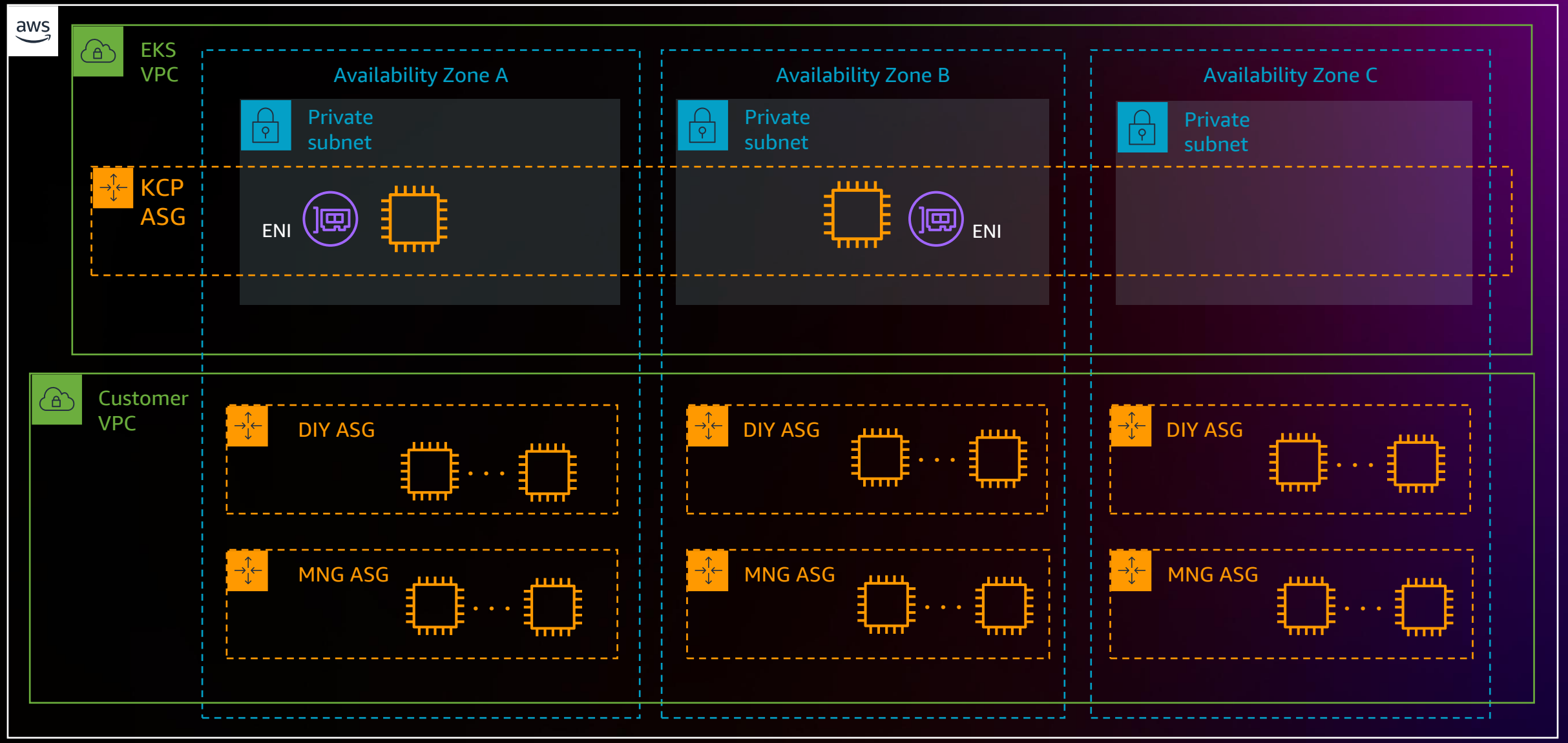

Commercial third-party solutions



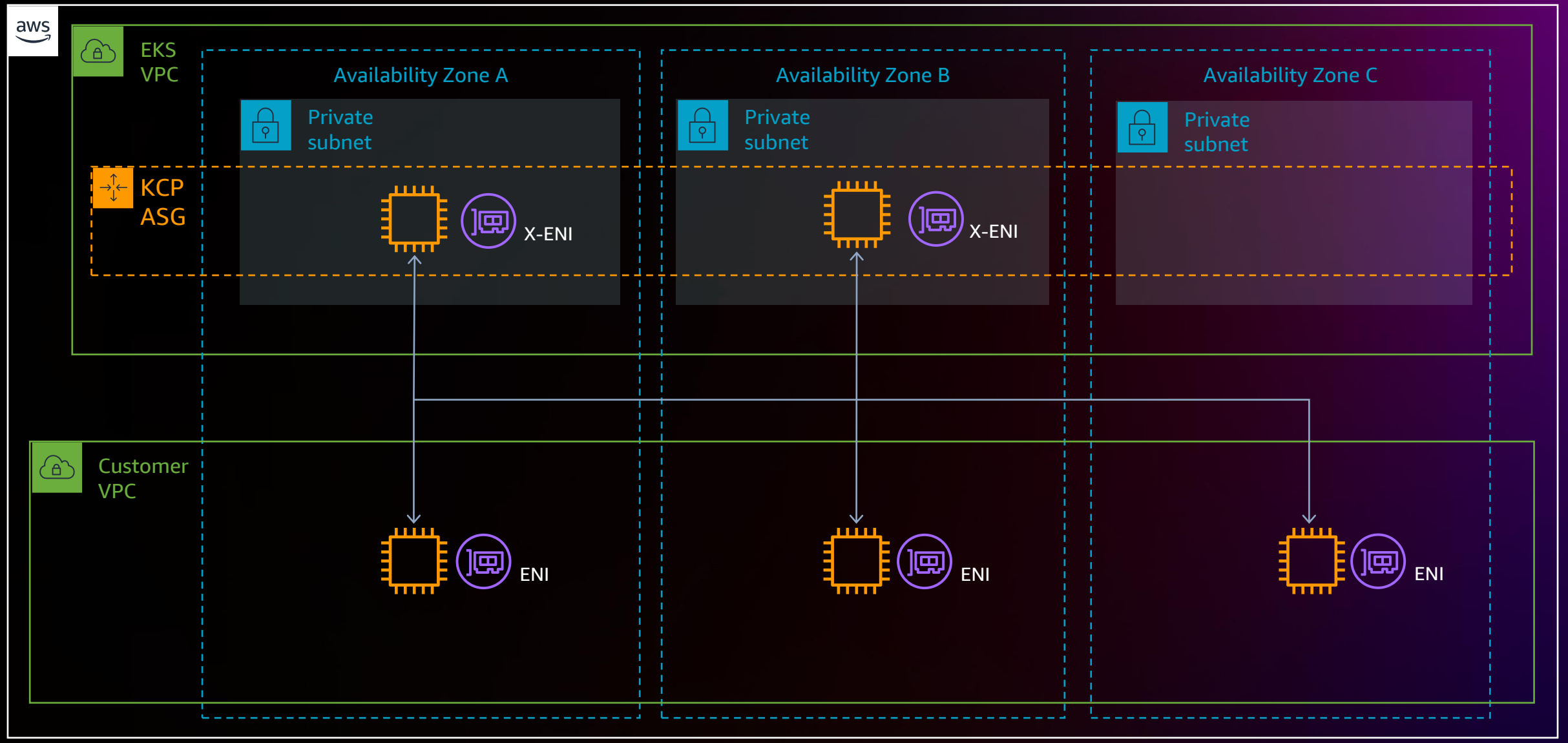
# EKS Architecture – Control Plane



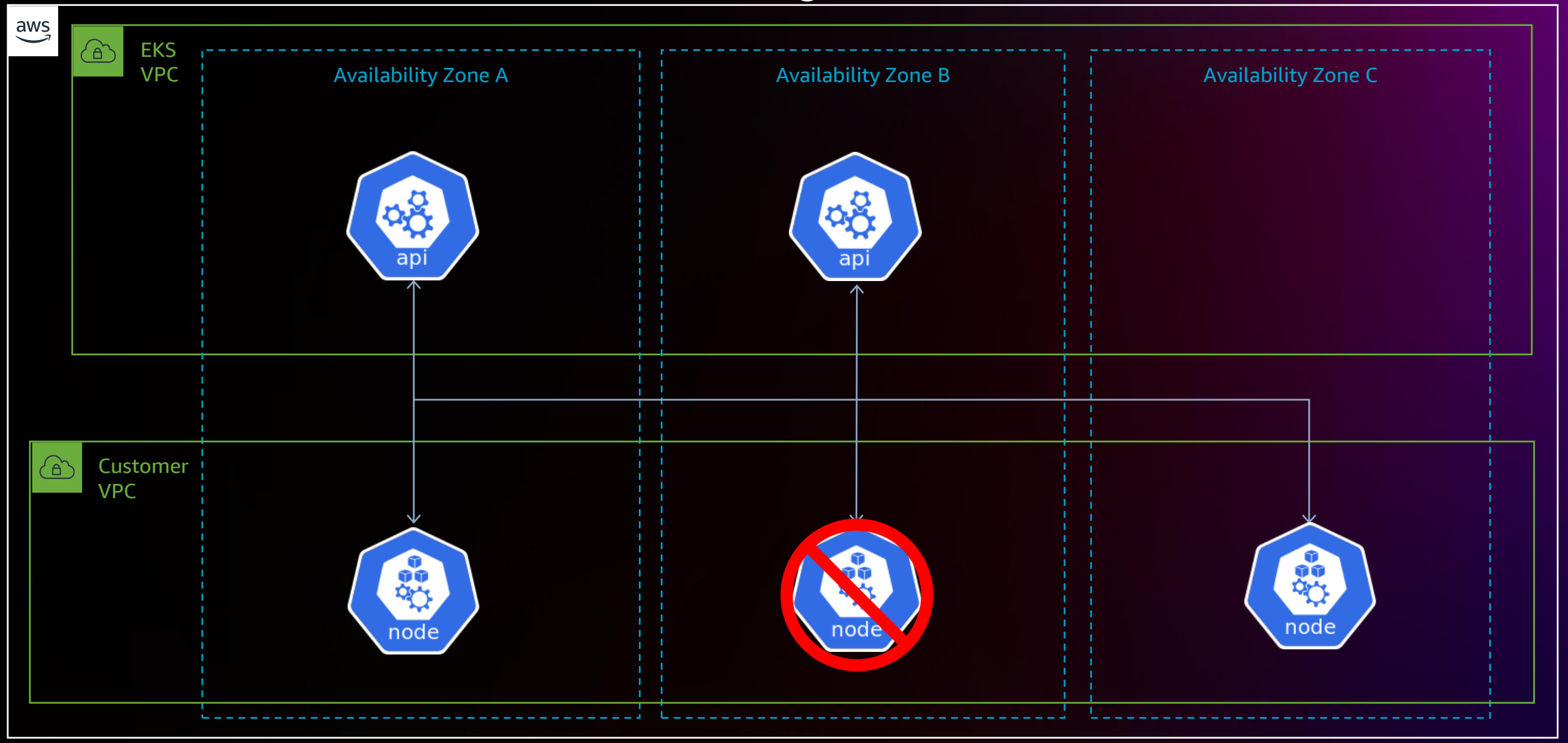
# EKS Control Plane – Data Plane communication



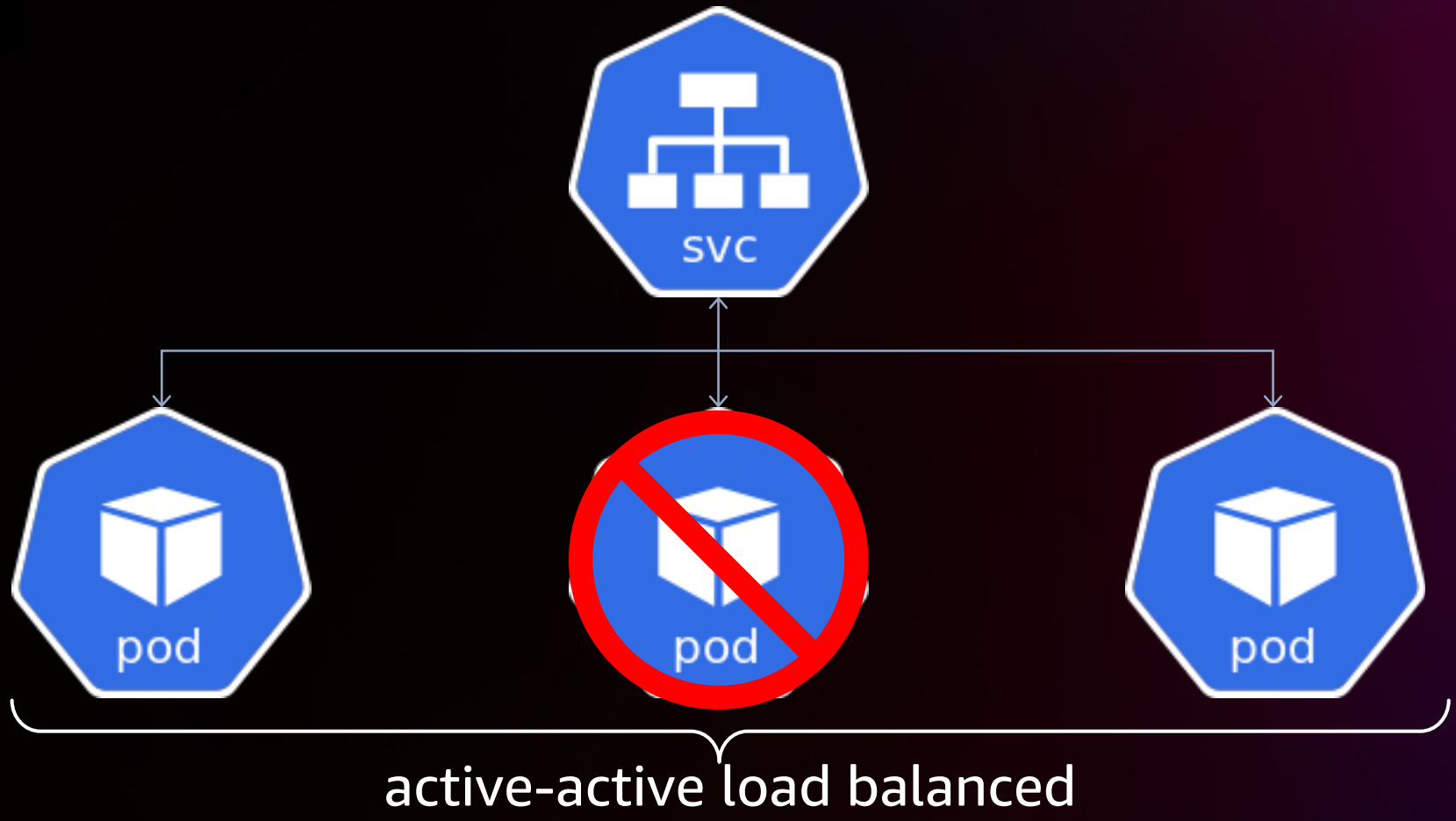
# EKS Control Plane – Data Plane communication



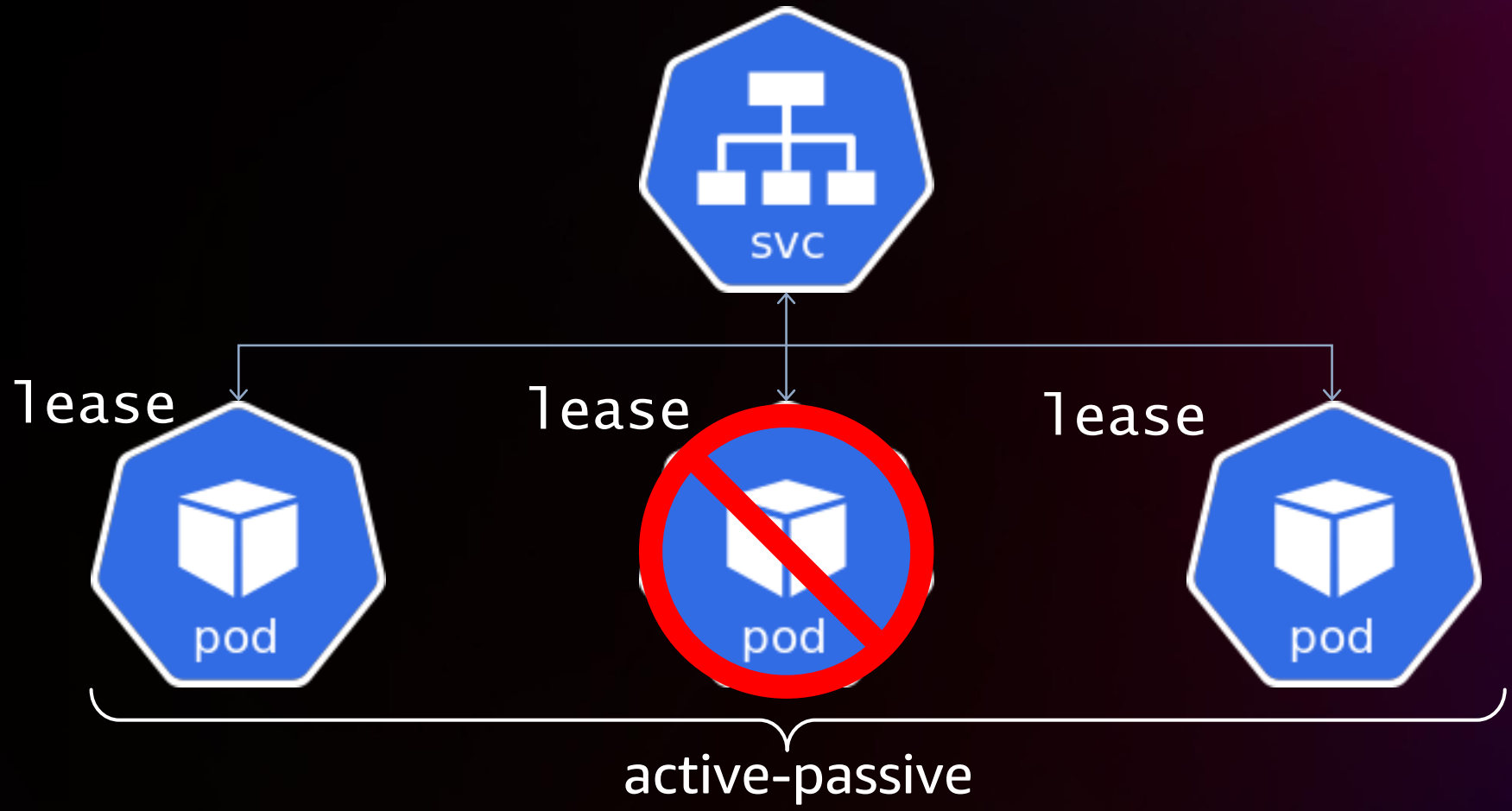
# Data Plane static stability



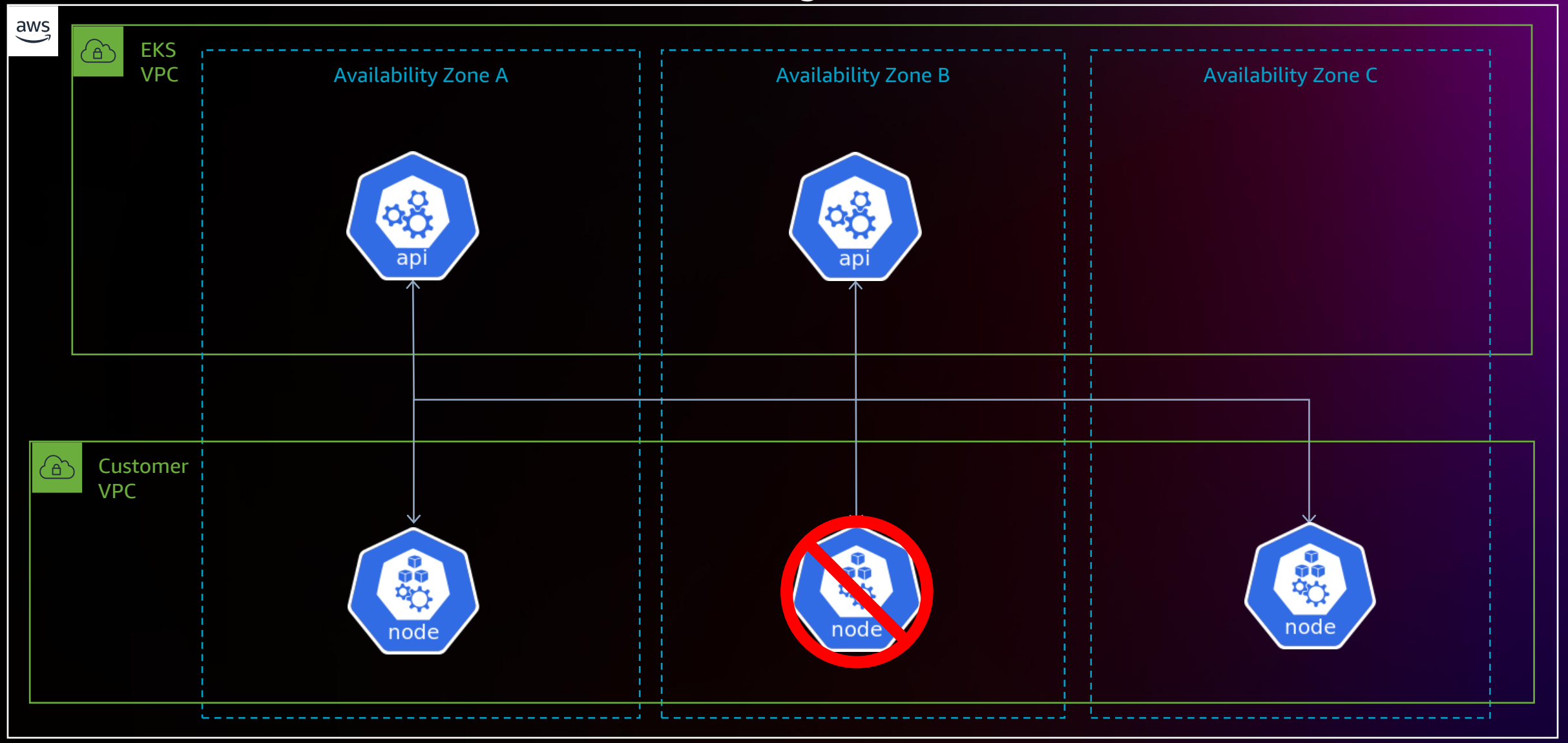
# Application resilience



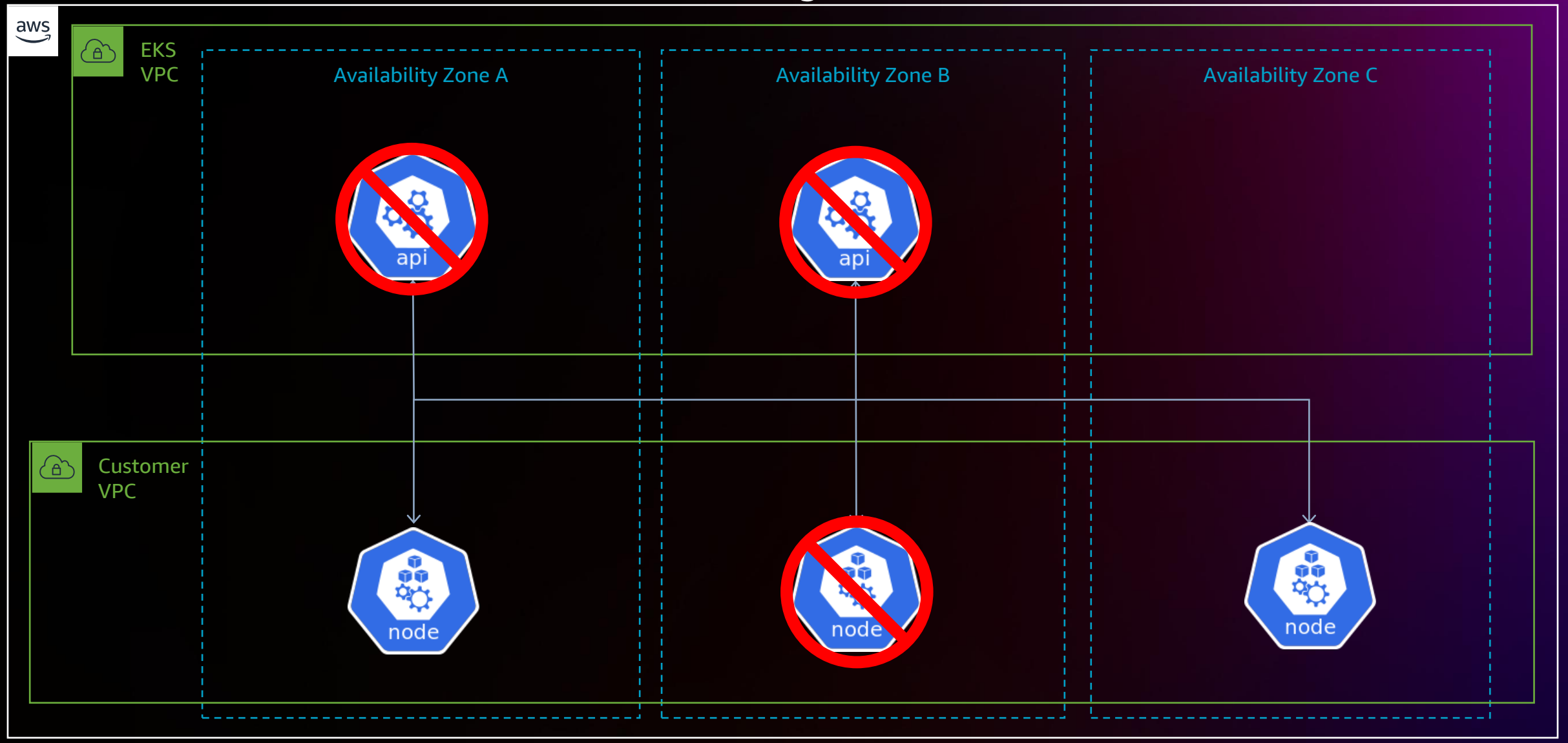
# Application resilience



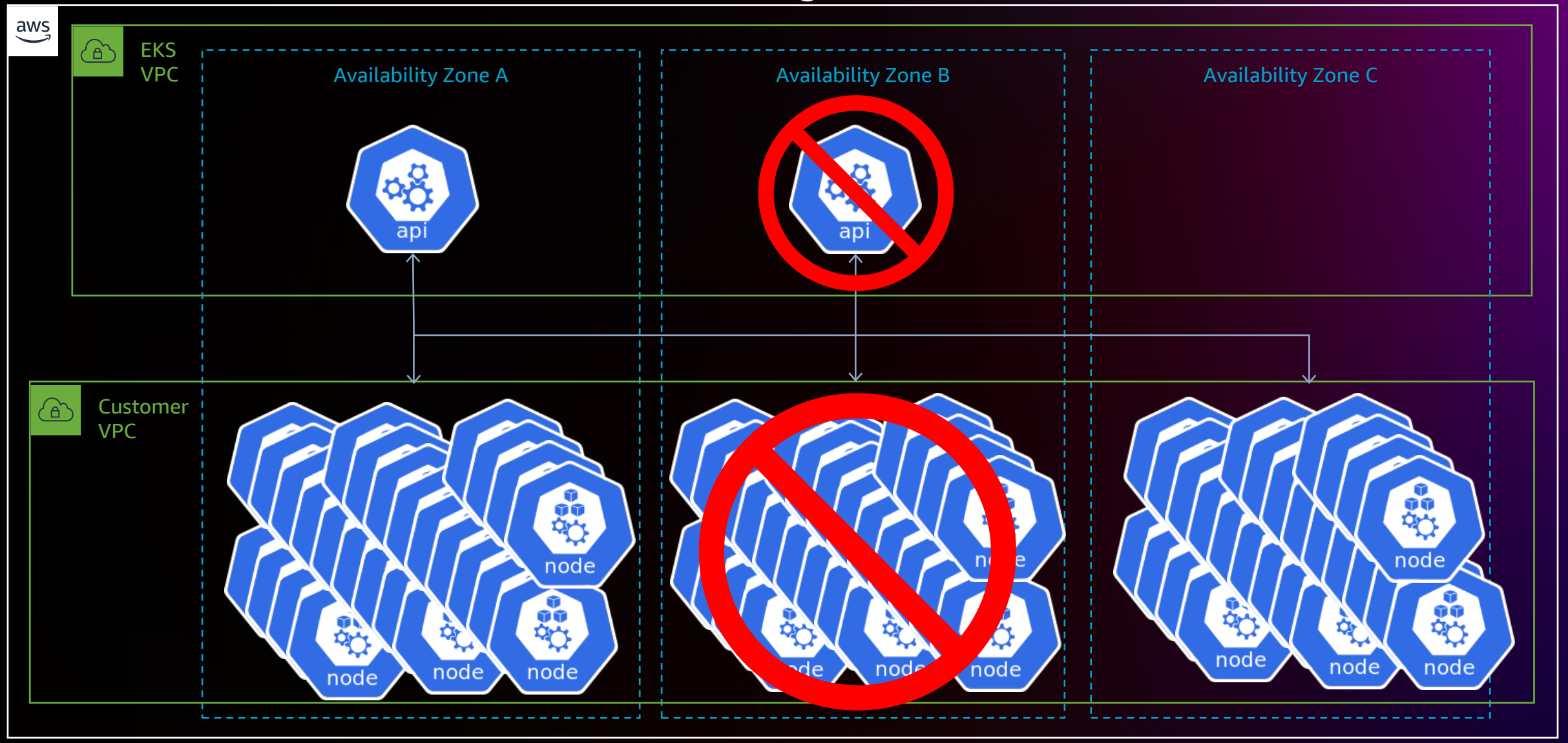
# Data Plane static stability



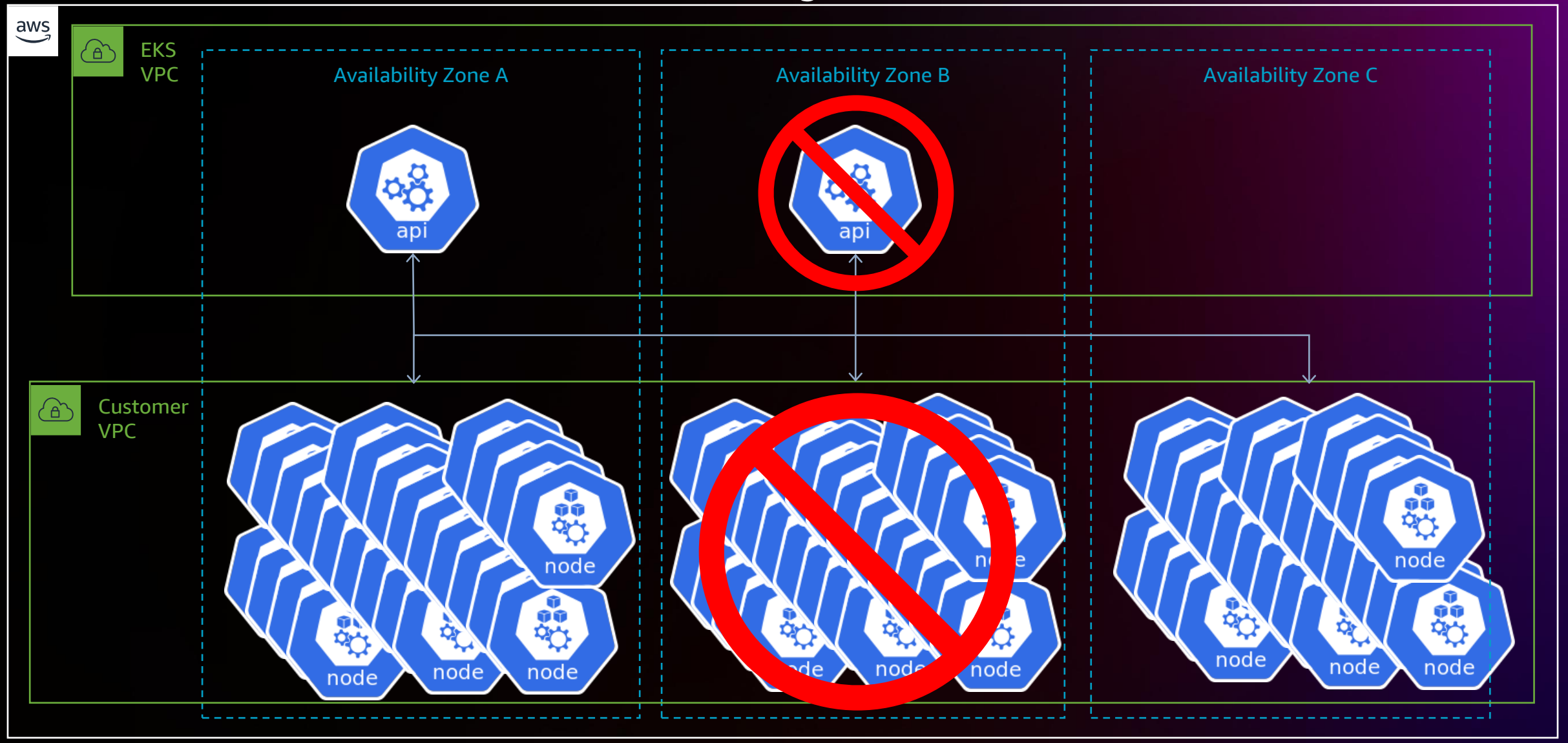
# Data Plane static stability



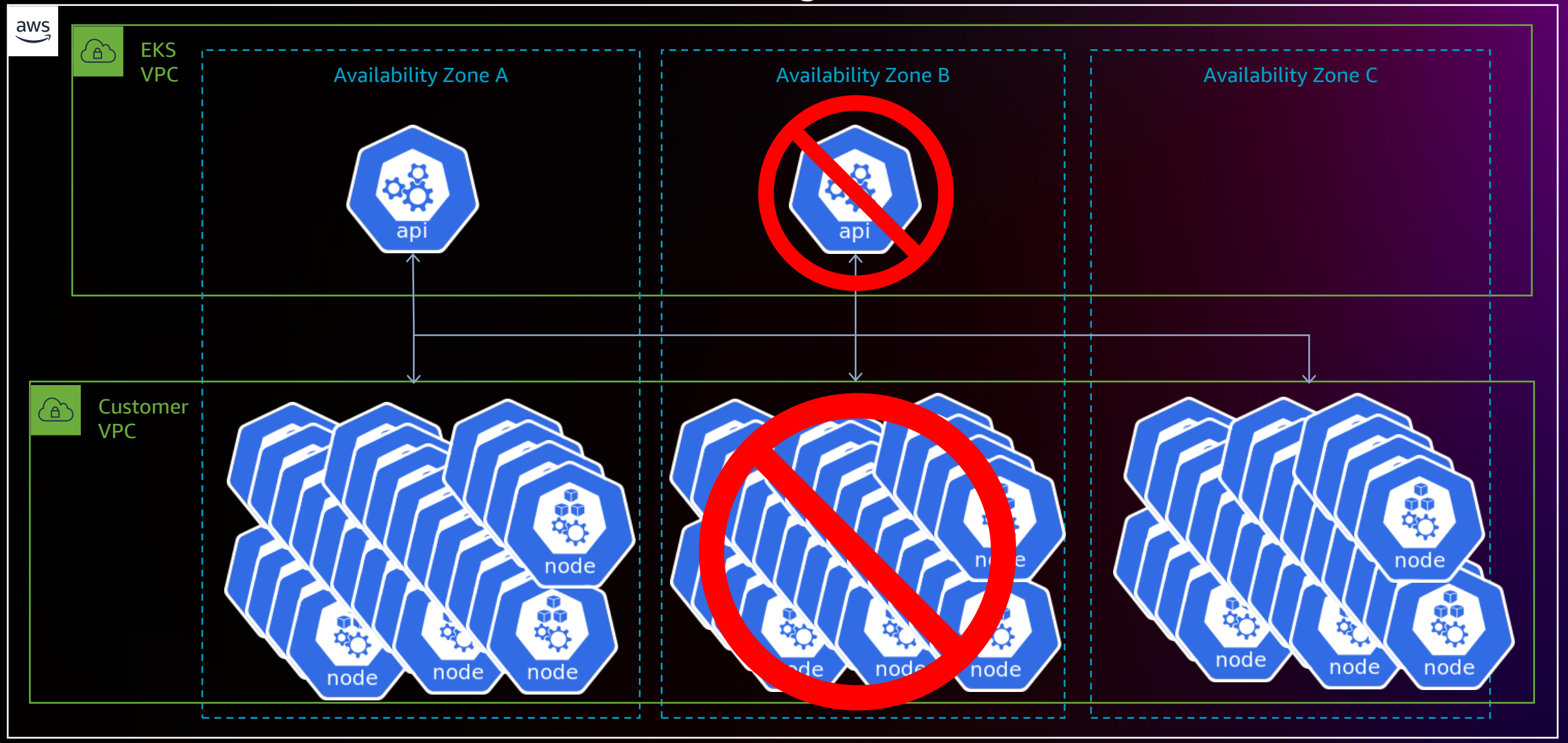
# Data Plane static stability



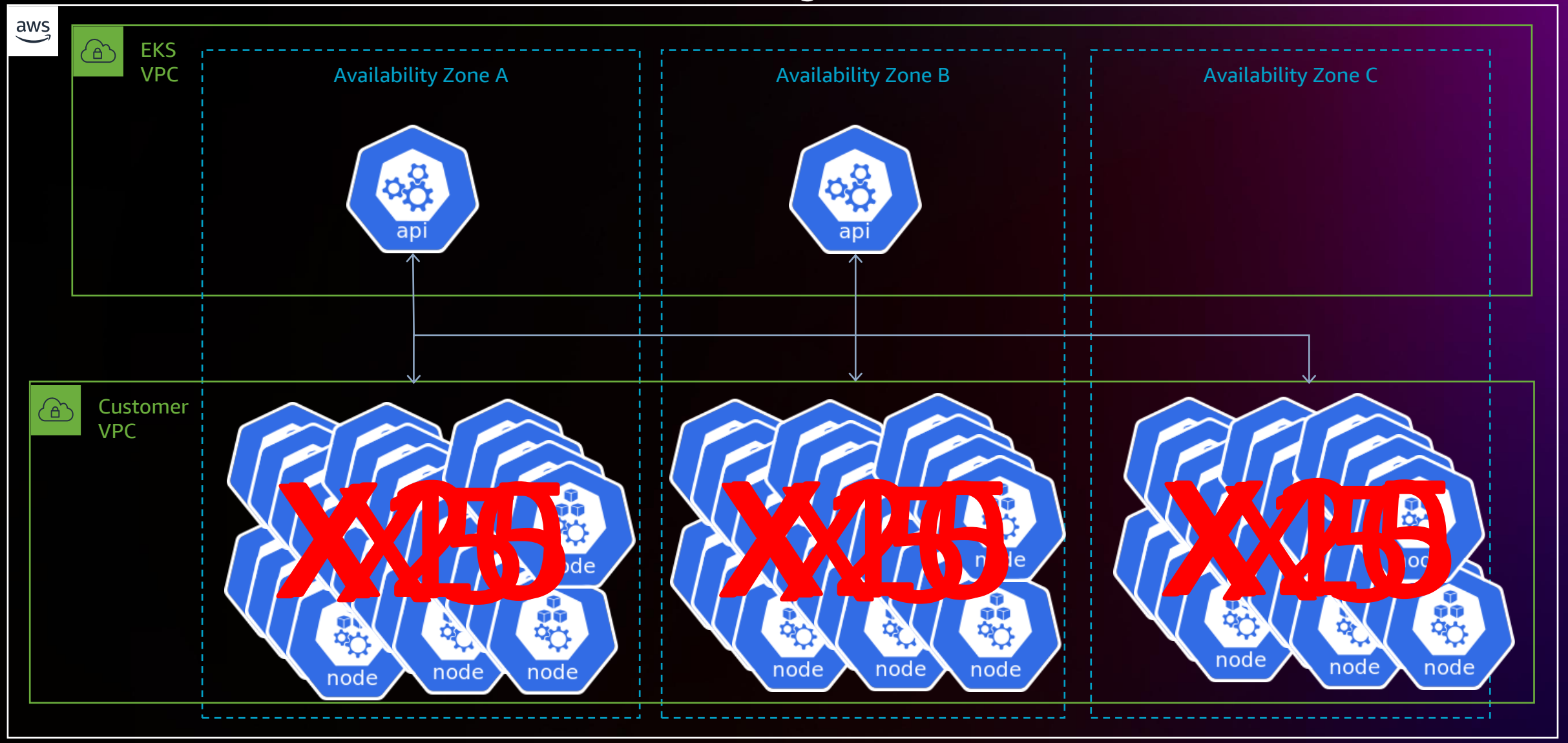
# Data Plane static stability



# Data Plane static stability



# Data Plane static stability



The Amazon Builders' Library

# Avoiding overload in distributed systems by putting the smaller service in control

**Joe Magerramov**

AWS Sr. Principal Engineer

# Further Reading

Amazon Builders' Library

EKS Best Practices

COE

ORR

# Thank you!

Eswar Bala
beswar@amazon.com

Rick Sostheim
sostheim@amazon.com

Please complete the session survey in the **mobile app**