Workshop: Smart Scaling with Karpenter

While we are waiting...

Each AWS account will be provided

Option 1) Run Karpenter workshop on EKS

https://tinyurl.com/KarpenterOnEKS

1. Follow “At an AWS event” instruction
2. Skip “Click here to open Workshop Studio login”, just copy one of URLs on a new browser
    - https://tinyurl.com/EKS-200 (try it first)
    - https://tinyurl.com/EKS-400 (try it if EKS-200 reaches the max capacity)
    - https://tinyurl.com/EKS-600 (try it if both EKS-200, EKS-400 reach the max capacity)
3. Continue to “Open IDE Environment to..”, wait for the instruction after the presentation

Option 2) Run Karpenter workshop on AKS

https://tinyurl.com/KarpenterOnAKS

1. Follow 1_aks_cluster_creation_and_install_karpenter.md
2. Best to make it through at least applying the required role assignments post cluster creation

You need your own Azure Account

Speakers

Praseeda Sathaye

Principal SA - Containers/Open Source
CNCF Ambassador
AWS

Charlie McBride

SDE2 - Kubernetes
Microsoft Azure

Chance Lee

Senior SA - Containers
AWS

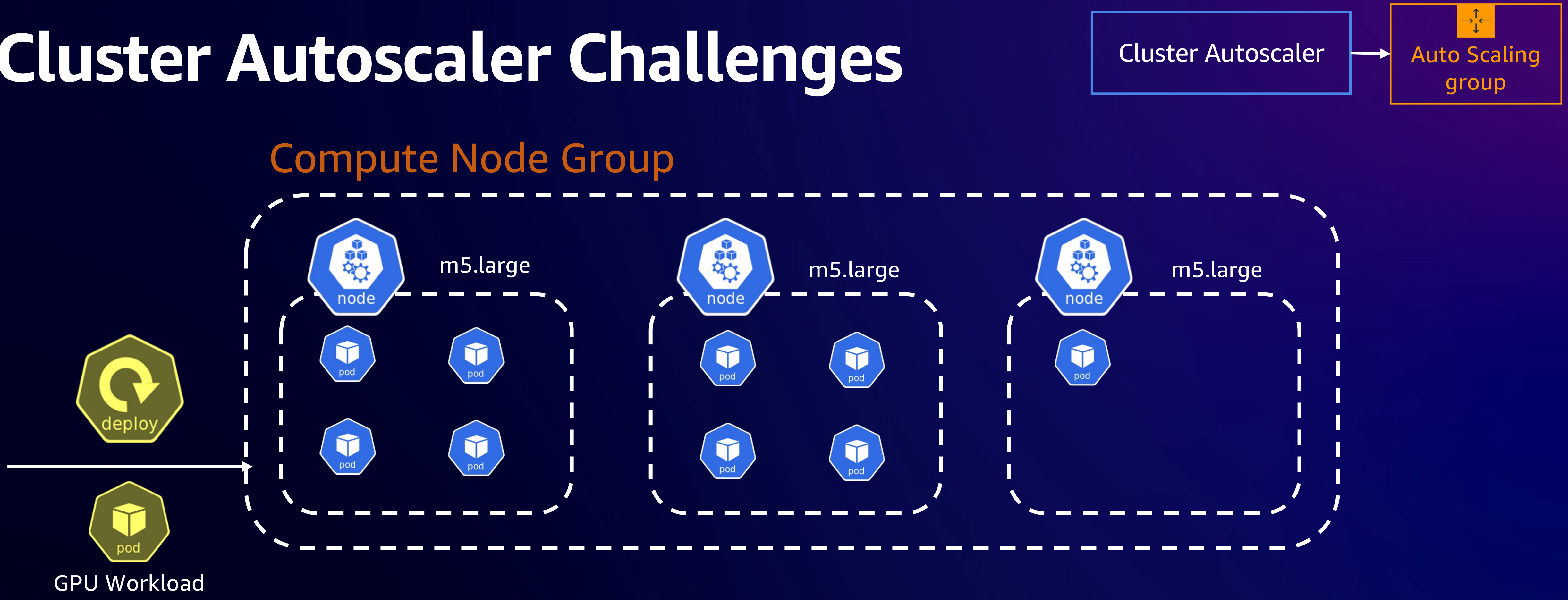
Wilson Darko

Product Manager - Kubernetes
Microsoft Azure

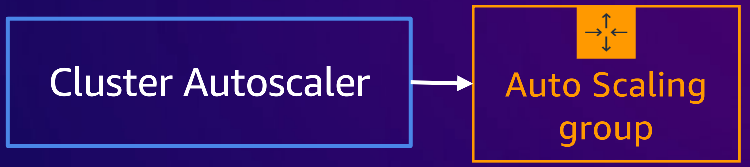
Rajdeep Saha

Principal SA - Containers/Serverless
AWS

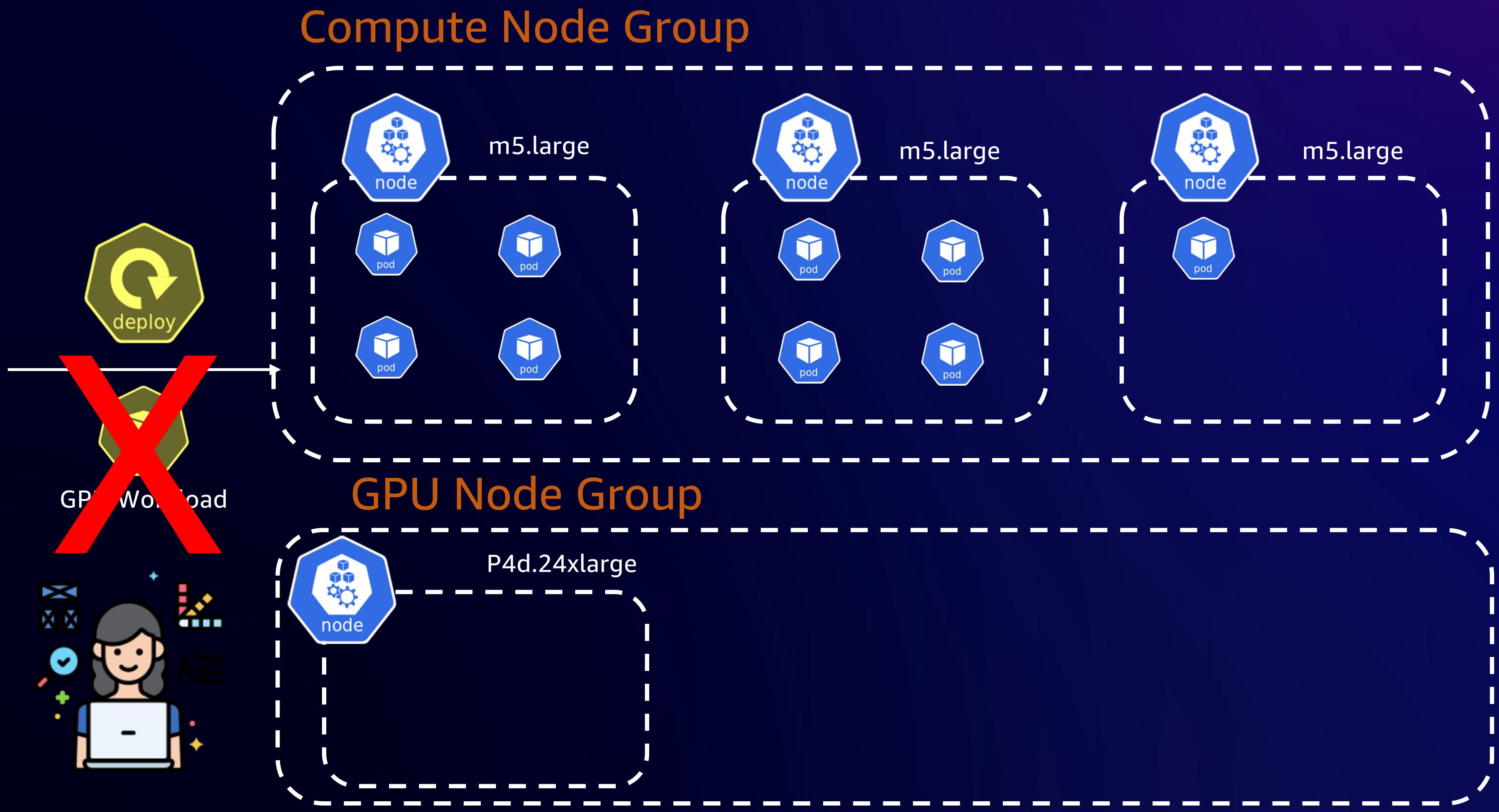
Cluster Autoscaler Challenges



Cluster Autoscaler Node Group



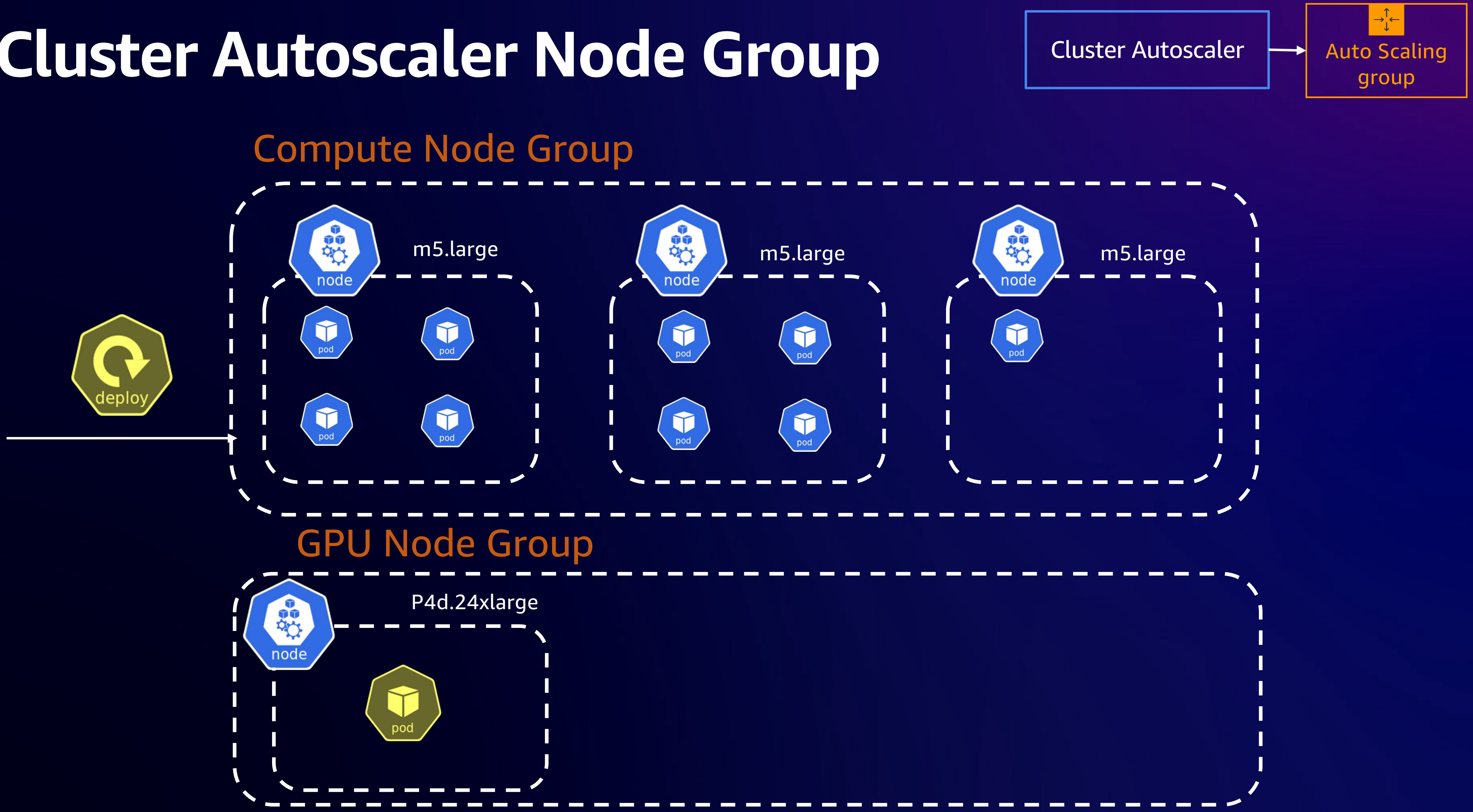
Cluster Autoscaler → Auto Scaling group



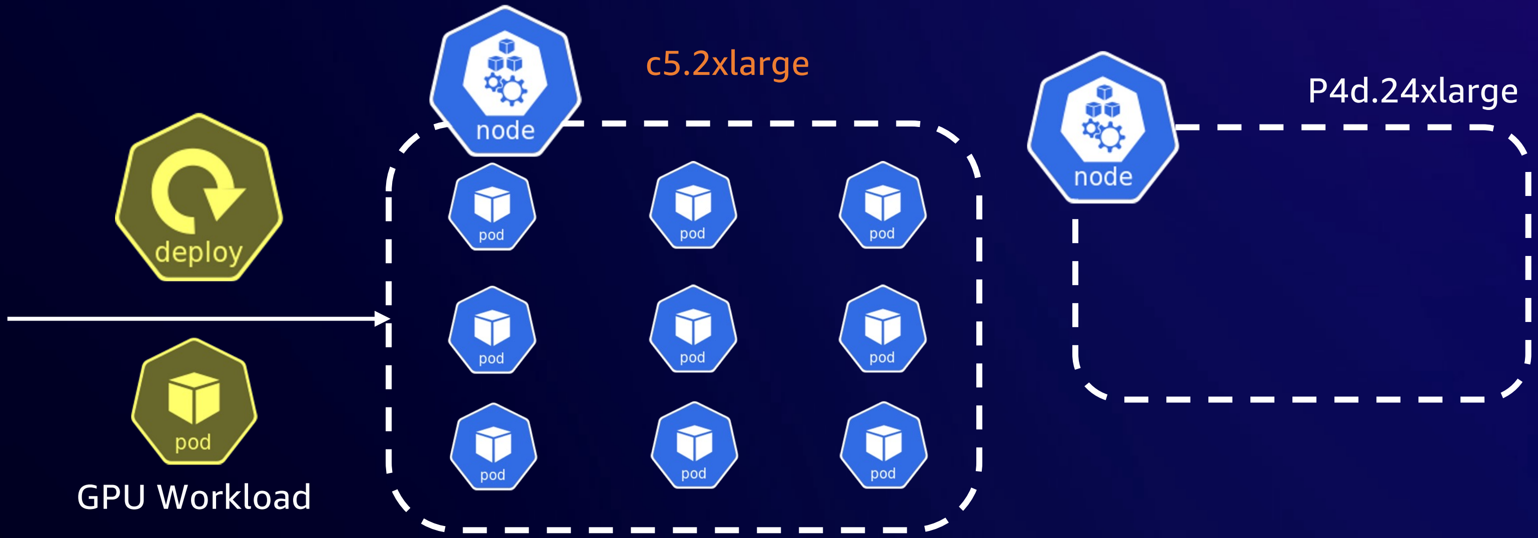
Compute Node Group

GPU Node Group

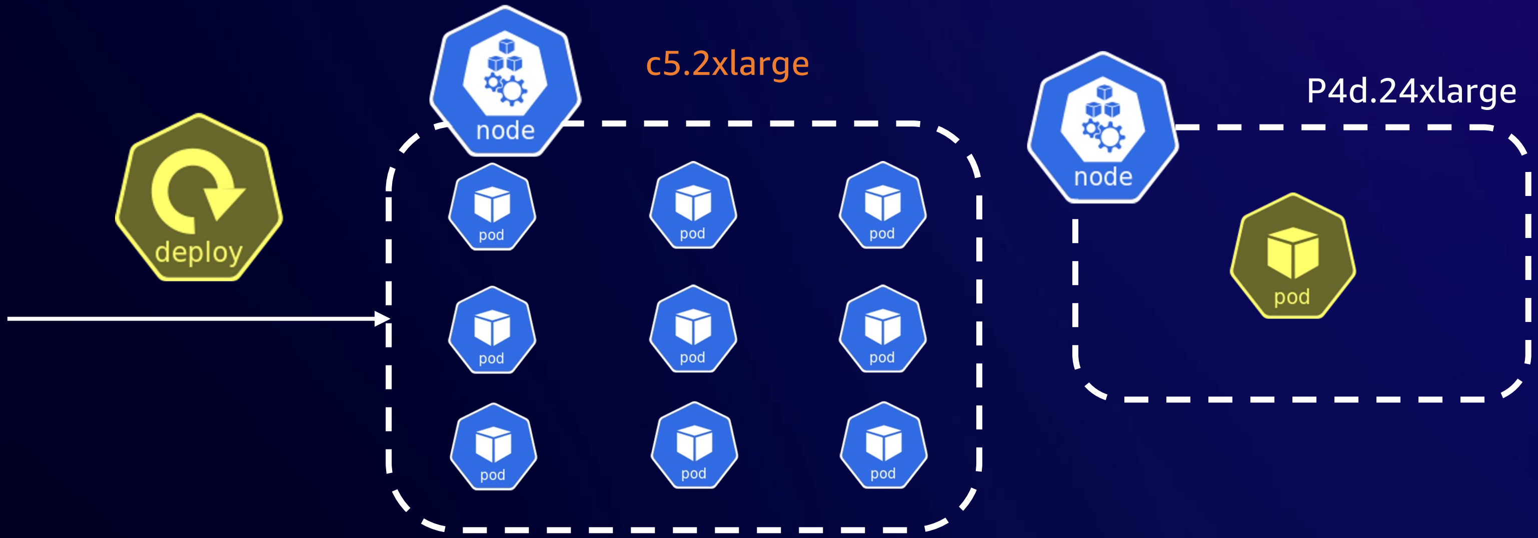
Cluster Autoscaler Node Group



Karpenter – CNCF SIG Autoscaling Project

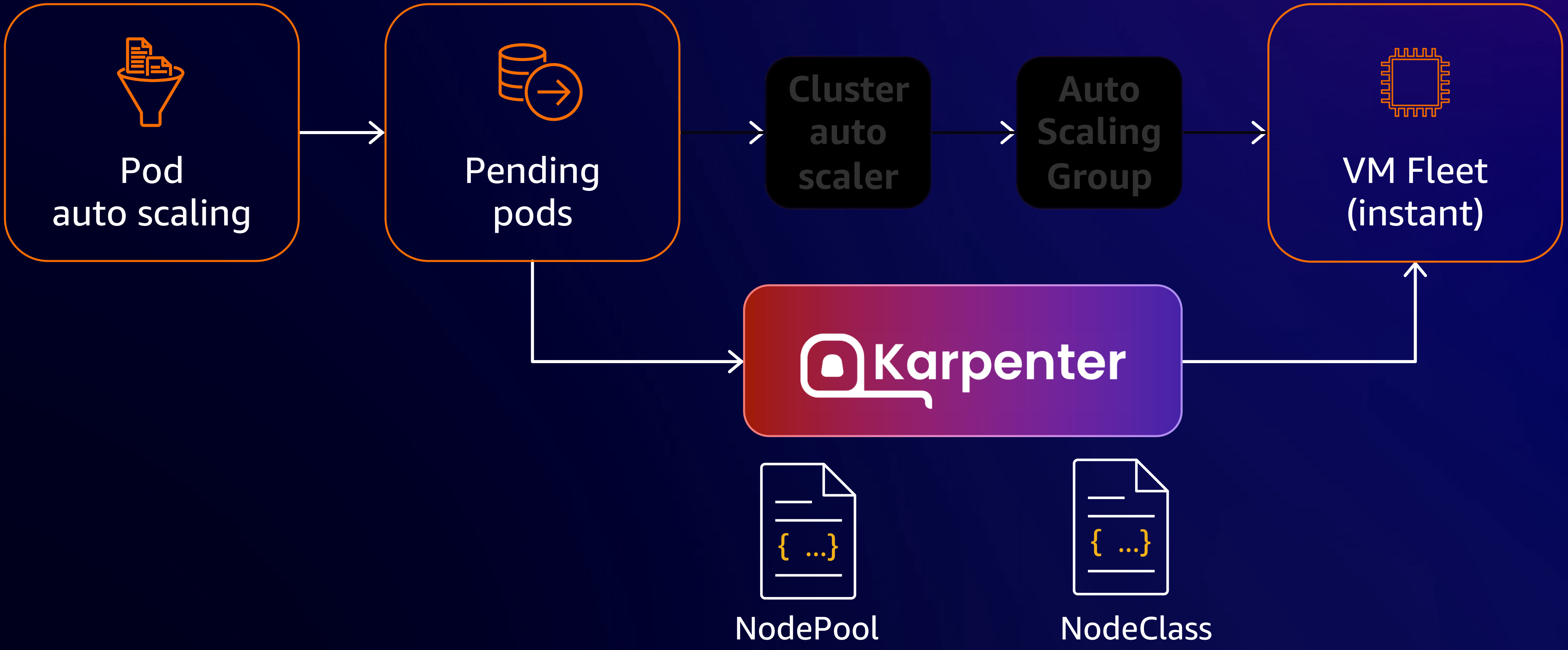


Karpenter – CNCF SIG Autoscaling Project



- Provision appropriate instances based on podspec without separate nodegroups
- Faster than Cluster Autoscaler

How Karpenter works



Karpenter does more than scaling

Cost optimization

Supports diverse workloads including ML and generative AI

Helps upgrade and patching

Total data plane implementation

Karpenter is part of Kubernetes (OSS)

Compute flexibility

Instance type flexibility

- Attribute-based requirements → sizes, families, generations, CPU architectures
- No list → picks from all instance types
- Limits how many VM instances this NodePool can provision
- Prioritizes cost

AZ flexibility

- Provision in any AZ
- Provision in specified AZs

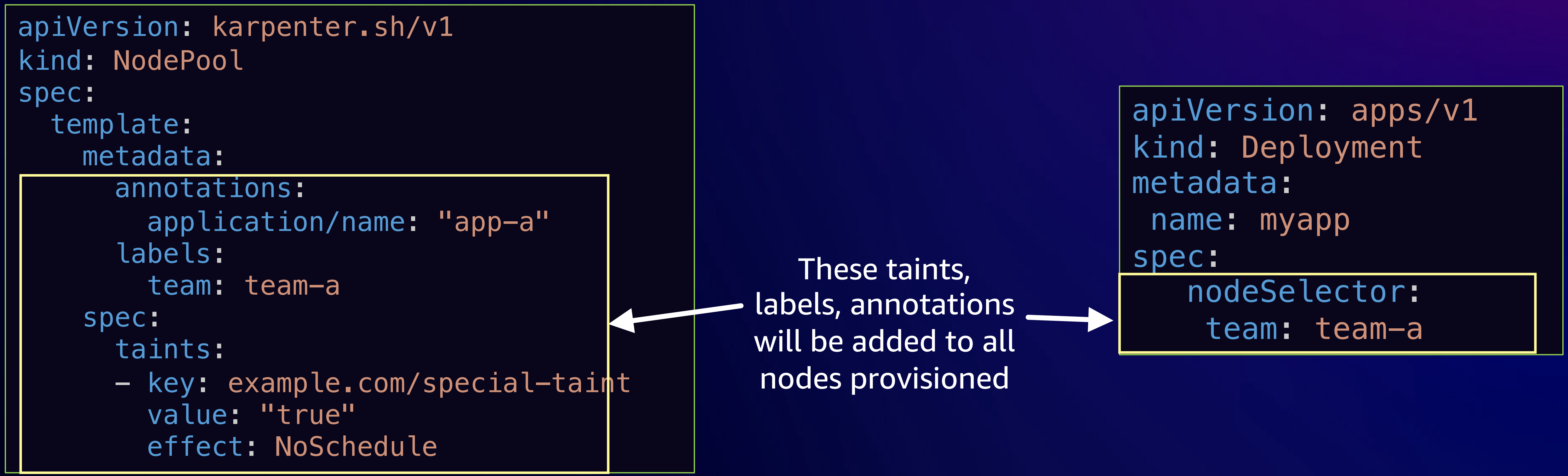
```yaml
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: default
spec:
  template:
    spec:
      requirements:
      - key: karpenter.k8s.aws/instance-category
        operator: In
        values: ["c","m","r","t"]
      - key: karpenter.k8s.aws/instance-size
        operator: NotIn
        values: ["nano","micro","small”,"medium"]
      - key: karpenter.k8s.aws/instance-hypervisor
        operator: In
        values: ["nitro"]
      - key: topology.kubernetes.io/zone
        operator: In
        values: ["us-west-2a","us-west-2b"]
      - key: kubernetes.io/arch
        operator: In
        values: ["amd64","arm64"]
limits:
    cpu: 100
```

Karpenter works with Kubernetes scheduling

Standard K8s pod scheduling mechanisms

- Node selectors
- Node affinity
- Taints and tolerations
- Topology spread

User-defined annotation, labels, taints



```yaml
apiVersion: karpenter.sh/v1
kind: NodePool
spec:
  template:
    metadata:
      annotations:
        application/name: "app-a"
      labels:
        team: team-a
    spec:
      taints:
      - key: example.com/special-taint
        value: "true"
        effect: NoSchedule
```

These taints, labels, annotations will be added to all nodes provisioned

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: myapp
spec:
   nodeSelector:
    team: team-a
```

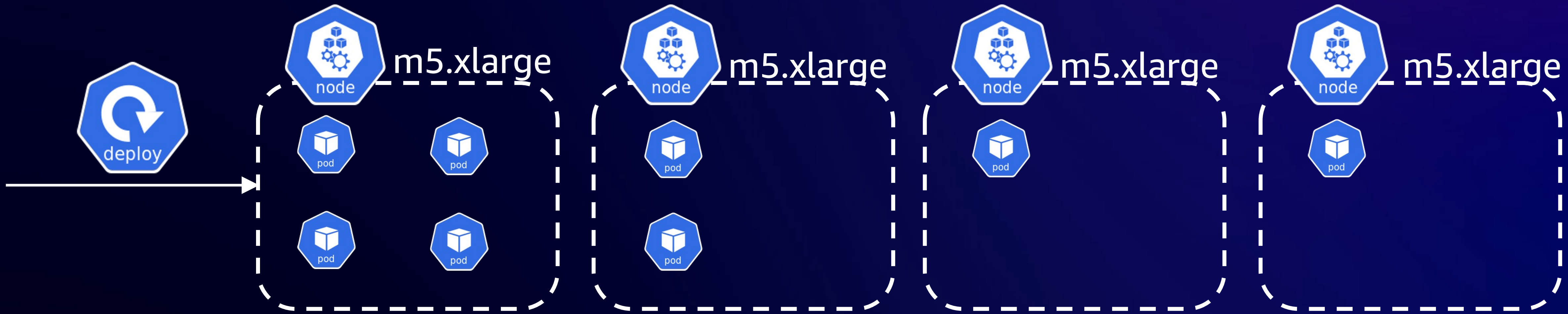
Use labels to schedule pods for different apps

Sample well-known labels added to nodes

Label	Example
topology.kubernetes.io/zone	us-east-2a
node.kubernetes.io/instance-type	g4dn.8xlarge
kubernetes.io/os	linux
kubernetes.io/arch	amd64
karpenter.sh/capacity-type	spot
karpenter.k8s.aws/instance-hypervisor	nitro
karpenter.k8s.aws/instance-encryption-in-transit-supported	true
karpenter.k8s.aws/instance-category	g

And more . . .

Node Disruption - Consolidation



Consolidation – Reducing number of nodes or replacing nodes for optimal bin-packing

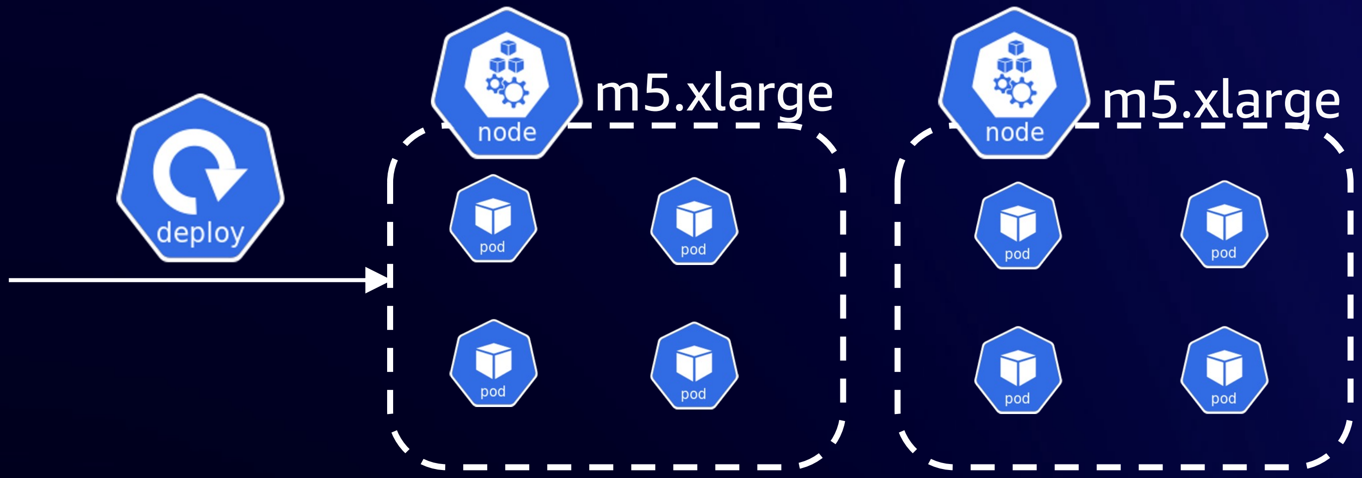
Consolidation Policies: WhenEmpty or WhenEmptyOrUnderutilized

Optional - ConsolidateAfter

```yaml
apiVersion: karpenter.sh/v1
kind: NodePool
spec:
  disruption:
    consolidationPolicy: WhenEmptyOrUnderutilized
```

Karpenter optimization with ConsolidationPolicy

Without consolidateAfter (by default set to 0)

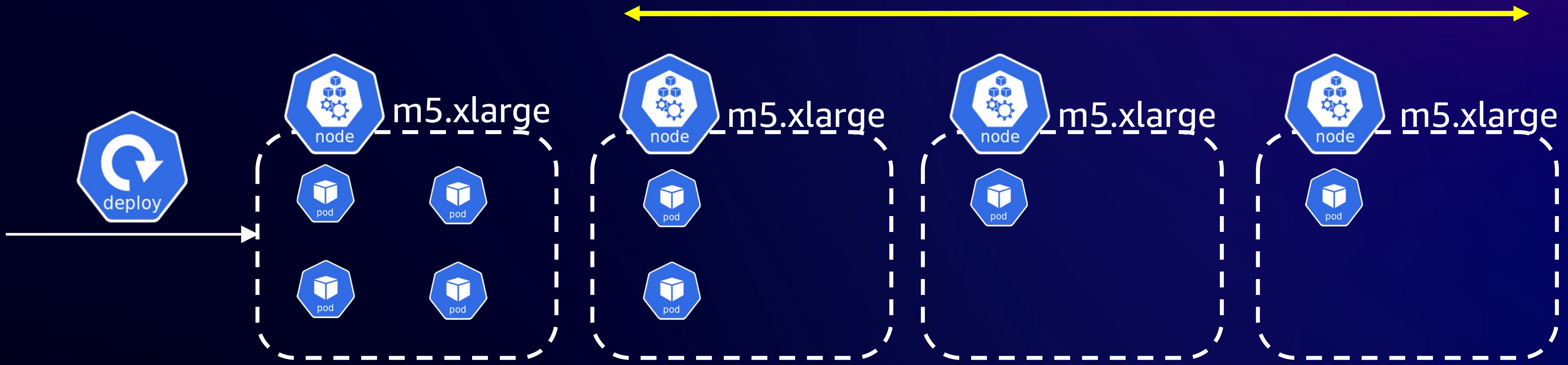


This could result in high node churn

consolidateAfter with WhenEmptyOrUnderutilized

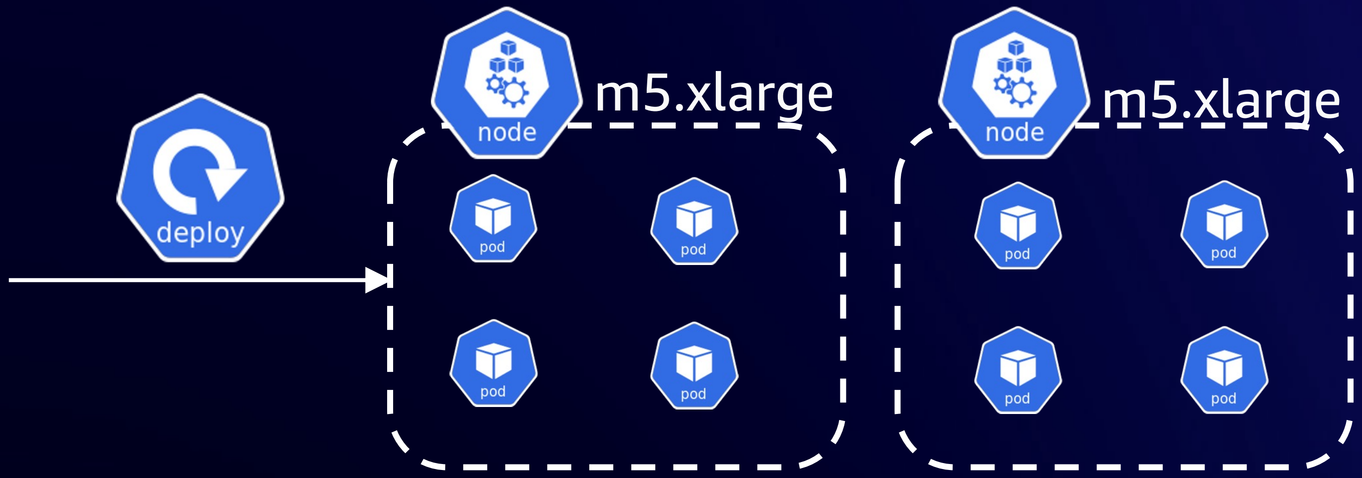
With consolidateAfter

Will wait for 1 hour after last pod is added or removed to consolidate



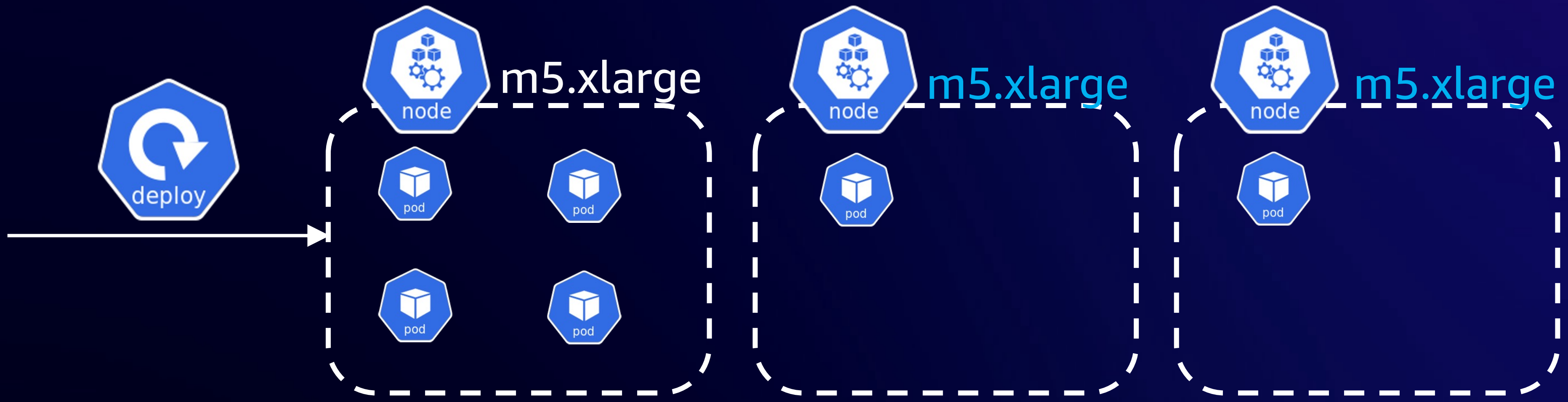
```
apiVersion: karpenter.sh/v1
kind: NodePool
spec:
  disruption:
    consolidationPolicy: WhenEmptyOrUnderutilized
    consolidateAfter: 1h
```

Karpenter optimization with ConsolidationPolicy



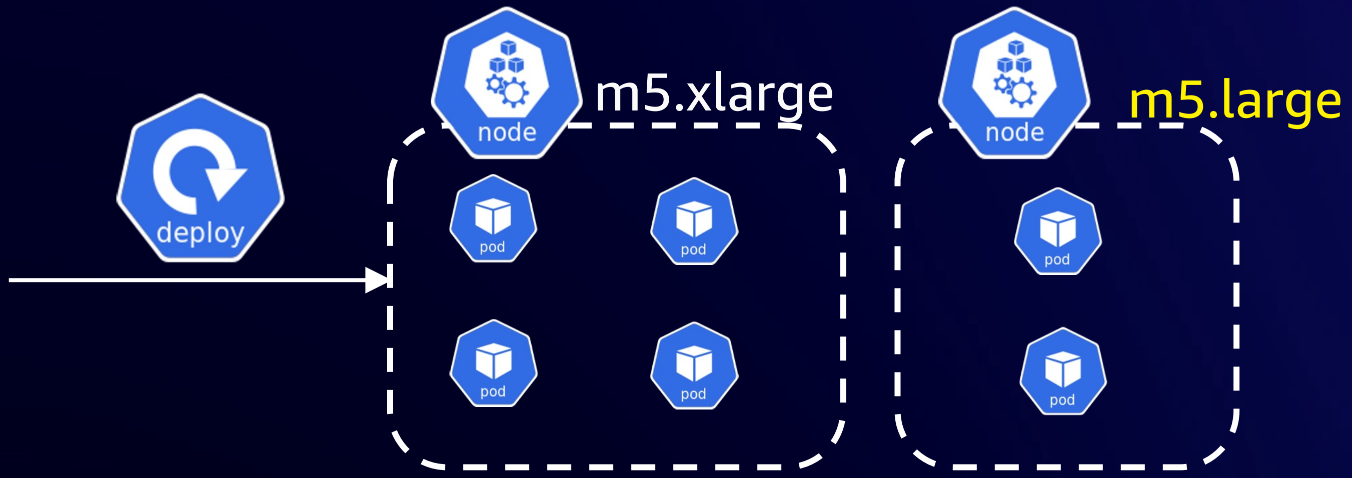
Better utilization of worker nodes – reduced cost

Karpenter optimization



Enable consolidation

Karpenter optimization – Pick cheaper nodes



Better selection of worker nodes – reduced cost

Correcting nodes experiencing Drift

Drift – detects and corrects NodeClaim’s which no longer match their owning NodePool, and/or NodeClass specifications.

```
  kind: NodePool
  …
  spec:
    template:
      spec:
        requirements:
          - key:
karpenter.sh/capacity-type
            operator: In
            values:  [“spot”]
…
```

```
  kind: NodePool
  …
  spec:
    template:
      spec:
        requirements:
          - key: karpenter.sh/capacity-
type
            operator: In
            values:  [“on-demand”]
…
```

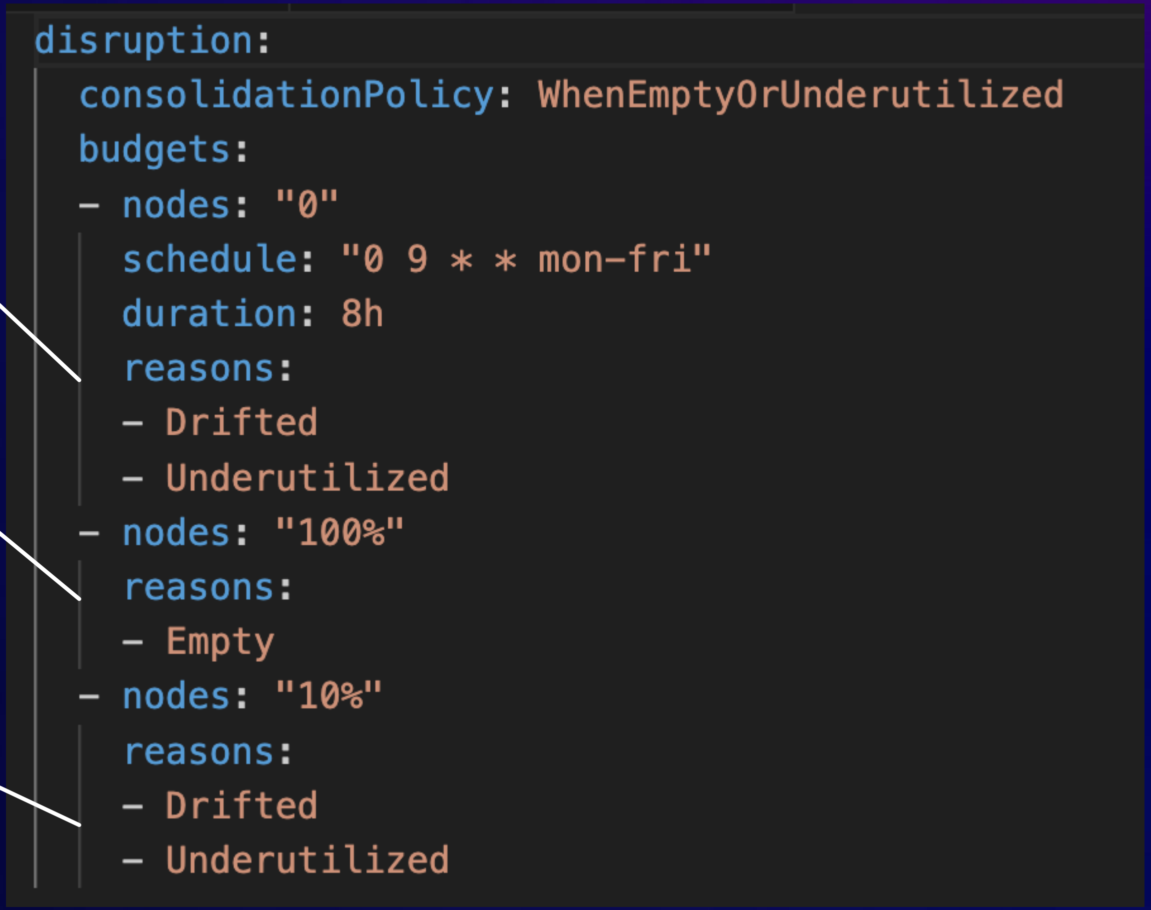
Control Disruption Budget

Note that Underutilized not equals Empty

- 0 nodes can be disrupted due to Drift or Underutilized from Monday to Friday from 9:00 UTC for 8 hours
- Empty nodes can still be disrupted

- 100% of Empty nodes can be disrupted any time

- 10% of nodes can be disrupted when Drifted or Underutilized at any time
- The rules are cumulative i.e. the first and second rules allow all empty nodes to be disrupted at any time
- Outside the business hours set by the first rule, drifted or underutilized (but NOT empty) nodes can be disrupted at 10% at any time



```yaml
disruption:
  consolidationPolicy: WhenEmptyOrUnderutilized
  budgets:
  - nodes: "0"
    schedule: "0 9 * * mon-fri"
    duration: 8h
    reasons:
    - Drifted
    - Underutilized
  - nodes: "100%"
    reasons:
    - Empty
  - nodes: "10%"
    reasons:
    - Drifted
    - Underutilized
```

Workshop

While we are waiting...

Each AWS account will be provided

Option 1) Run Karpenter workshop on EKS

https://tinyurl.com/KarpenterOnEKS

1. Follow “At an AWS event” instruction
2. Skip “Click here to open Workshop Studio login”, just copy one of URLs on a new browser
   - https://tinyurl.com/EKS-200 (try it first)
   - https://tinyurl.com/EKS-400 (try it if EKS-200 reaches the max capacity)
   - https://tinyurl.com/EKS-600 (try it if both EKS-200, EKS-400 reach the max capacity)
3. Continue to “Open IDE Environment to..”, wait for the instruction after the presentation

Option 2) Run Karpenter workshop on AKS

https://tinyurl.com/KarpenterOnAKS

1. Follow 1_aks_cluster_creation_and_install_karpenter.md
2. Best to make it through at least applying the required role assignments post cluster creation

You need your own Azure Account

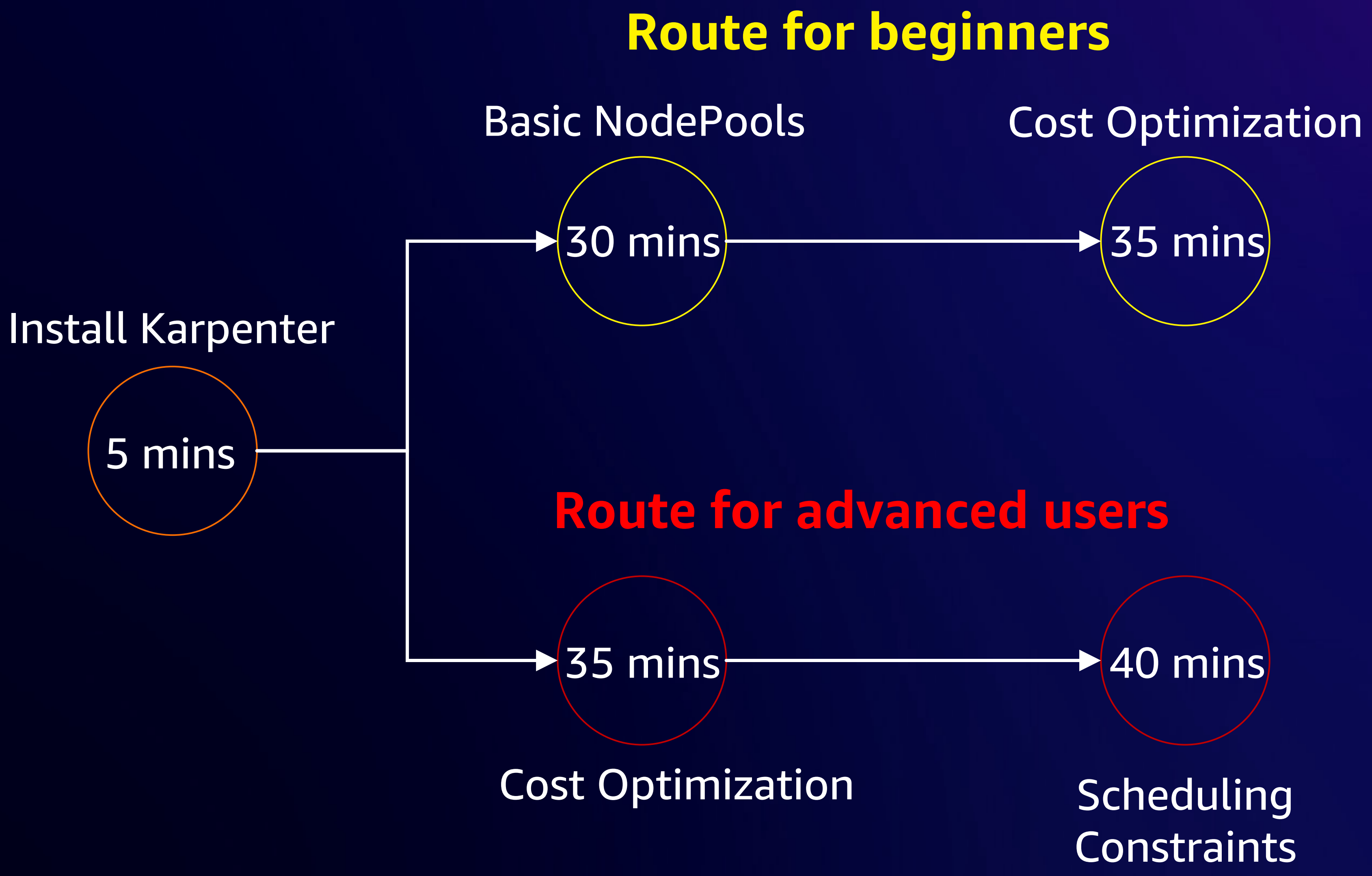
Before you start on AWS

You have access to an AWS account with everything needed to complete this session.

The AWS account is only available for the duration of this session. **You will lose access to the account once the session is complete.**

Review the terms and conditions of the event. **Do not upload any personal or confidential information to the account.**

Choose your adventure



Route for beginners

Install Karpenter

5 mins

Basic NodePools

30 mins

Cost Optimization

35 mins

Route for advanced users

35 mins

Cost Optimization

40 mins

Scheduling Constraints

Thank you!

Please complete the session survey in the mobile app

Rajdeep Saha

in cloudwithraj

▶ cloud with raj

X cloudwithraj

Chance Lee

in hellochance

Chancelee0111

Praseeda Sathaye

in praseedas

Charlie McBride

in charliedmcb

charliedmcb

Wilson Darko

in wilsondarko