Building a Sturdier Cloud with AWS EKS and Karpenter

August 2024

aws COMMUNITY DAY

Hi 👋

Elad Hirsch

Tech lead, CTO Office @ TERASKY

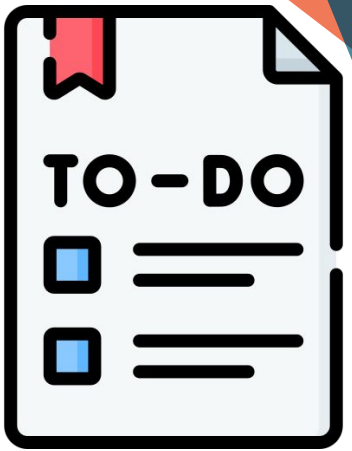
aws certified Solutions Architect PROFESSIONAL

Meet Dave

Senior Devops Engineer

TERASKY

- ❏ Schedule and decommission resources
- ❏ Rightly select your proper nodes
- ❏ Find underutilized and over provisioned clusters
- ❏ Split Between On-Demand & Spot Instances
- ❏ Prioritize Savings Plans and/or Reserved Instances
- ❏ Handle cost spikes or lake of proper availability



Tomorrow (*noun*):

```
/home$
```

Which of the following are you deploying on Kubernetes containers?

Kubernetes has grown exponentially in recent years due to its ability to support a vast range of workload types. According to our survey findings, the utilization of Kubernetes is no longer limited to microservices; over 60% of respondents are tapping into data ingestion, cleansing, and analytics with programs like Apache Spark. This trend further confirms that Kubernetes is maturing as an industry standard, with tech leaders deploying a wide variety of workloads on the platform.

A. Databases or data cache (e.g., PostgreSQL, MongoDB, Redis) (59%)

B. AI/ML software (e.g., Python, Tensorflow, Pytorch) (54%)

C. Web servers (e.g., NGINX) (58%)

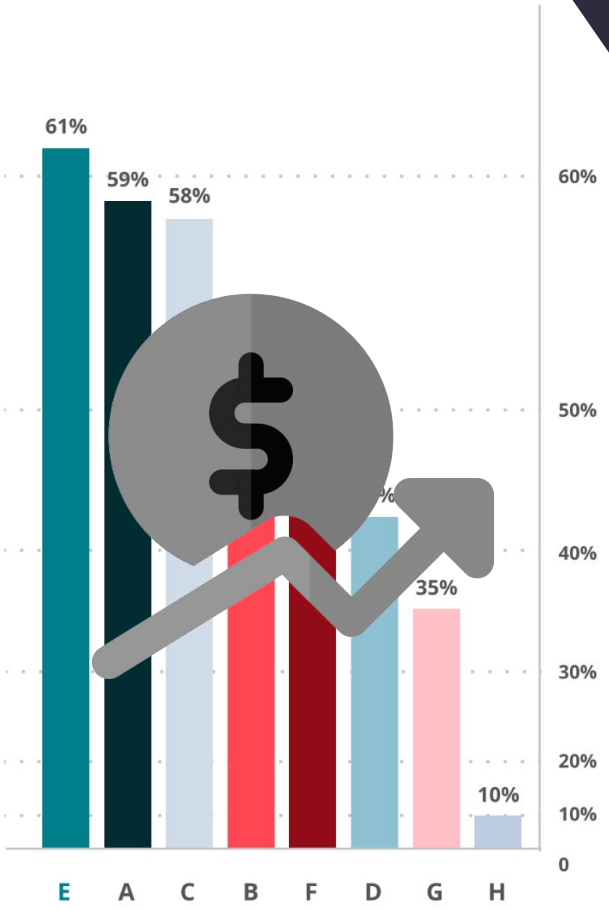
D. Logging and monitoring (e.g., Elastic, Splunk) (42%)

E. Data ingestion, cleansing, and analytics (e.g., Apache Spark) (61%)

F. Programming languages (e.g., Node.js, Java) (48%)

G. Application servers (35%)

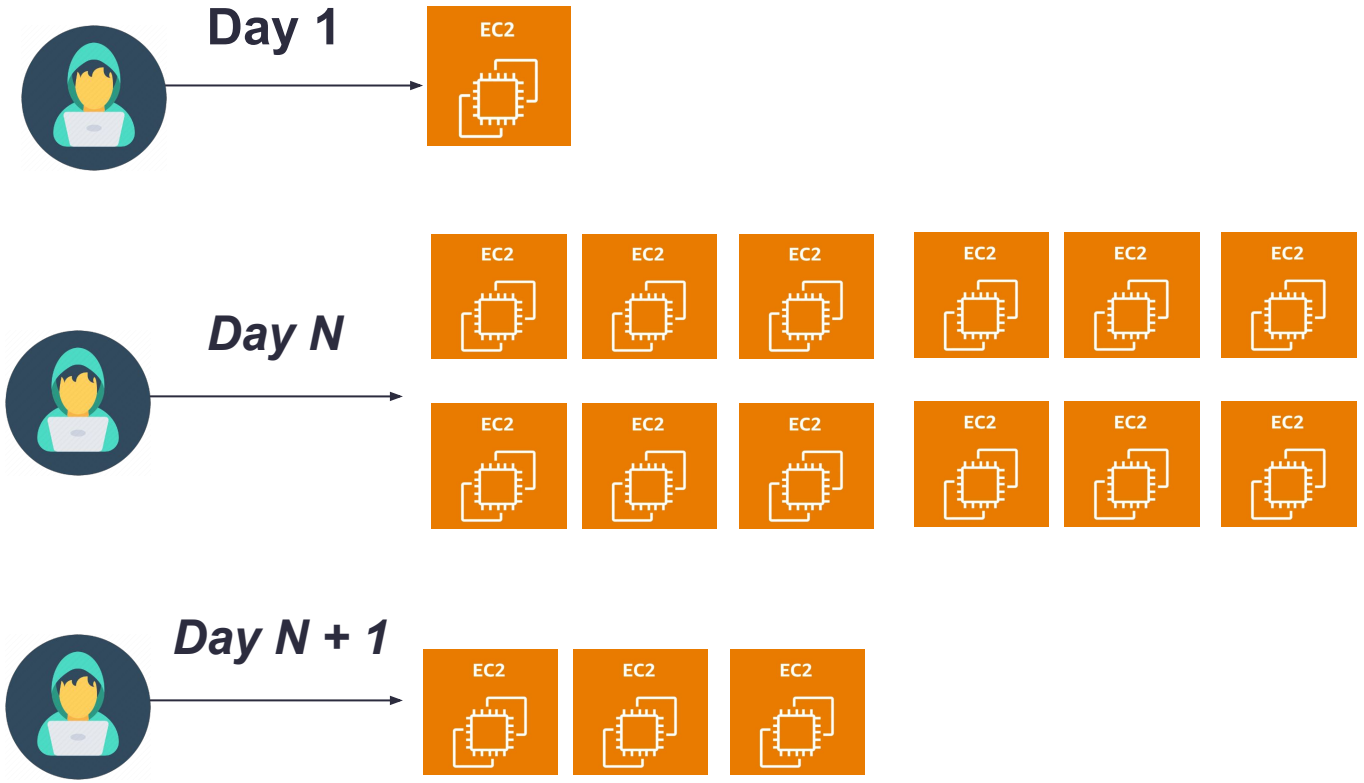
H. Message broker services (10%)



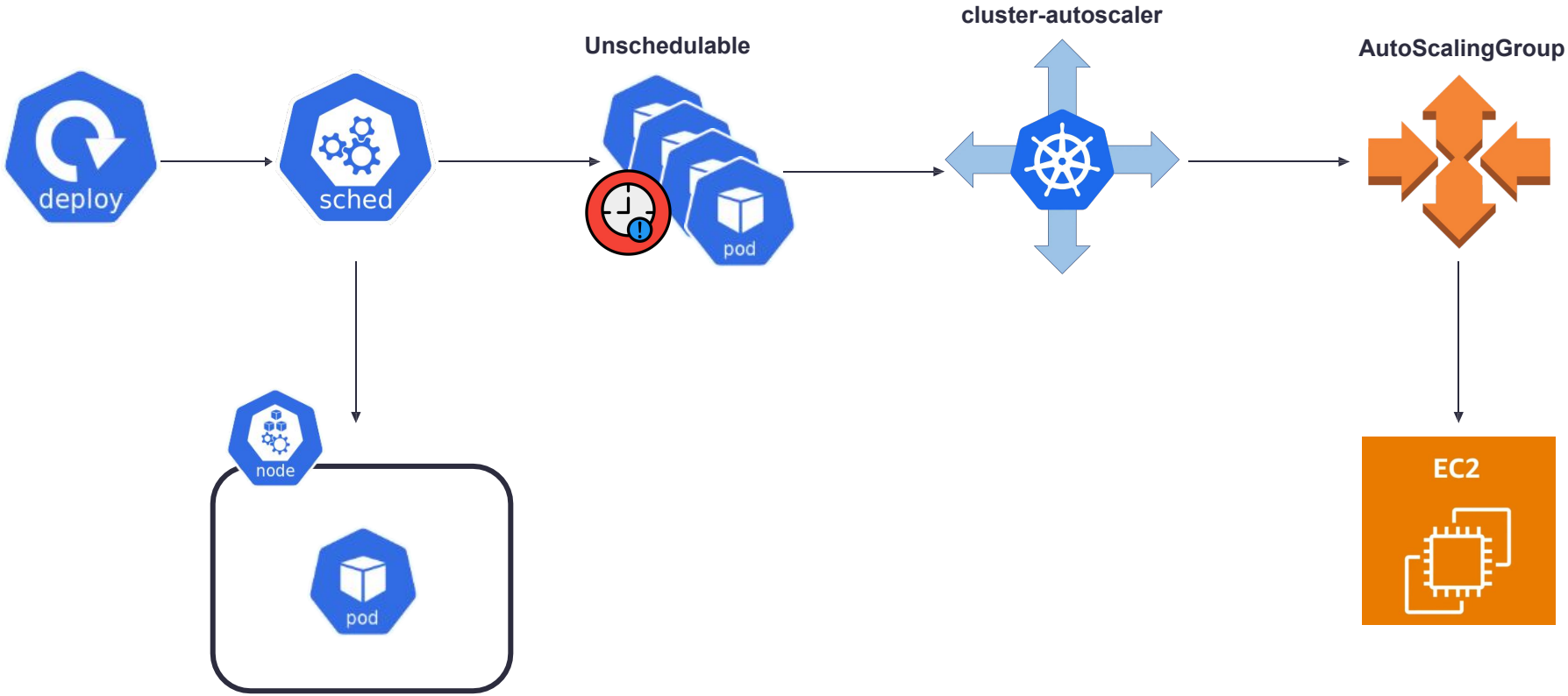
This makes Dave

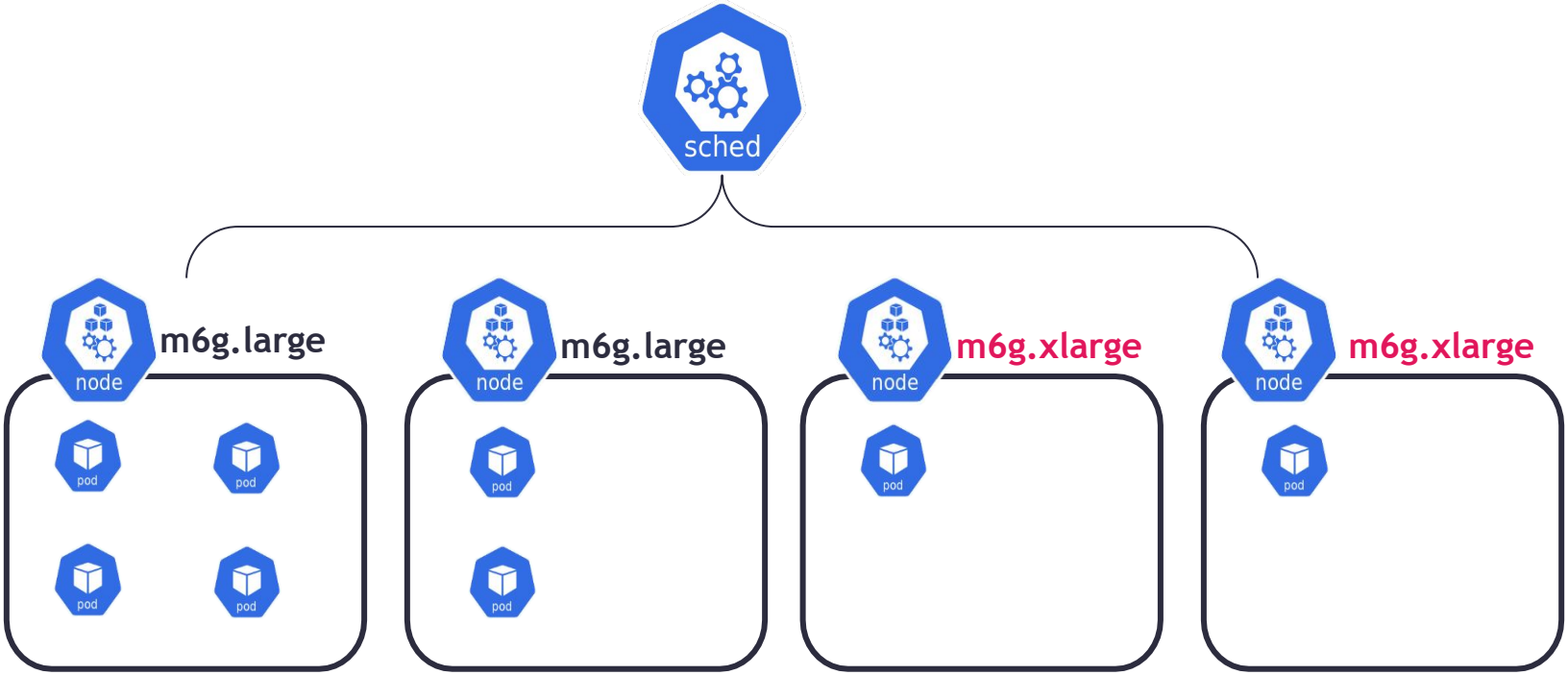
VERY FRUSTRATED

Our goal is to scale while staying within our budget

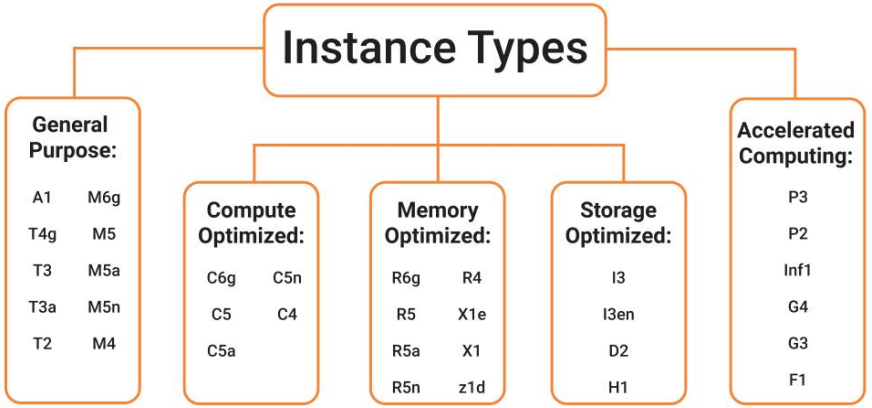


Cluster autoscaler




sched
node
m6g.large
pod
pod
pod
pod
node
m6g.large
pod
pod
node
m6g.xlarge
pod
node
m6g.xlarge
pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: resource-demo
spec:
  containers:
  - name: demo-container
    image: nginx:latest
    resources:
      requests:
        cpu: "500m"
        memory: "256Mi"
        ephemeral-storage: "1Gi"
      limits:
        cpu: "1"
        memory: "512Mi"
        ephemeral-storage: "2Gi"
  nodeSelector:
    kubernetes.io/hostname: gpu-node
  tolerations:
  - key: "nvidia.com/gpu"
    operator: "Exists"
    effect: "NoSchedule"
  containers:
  - name: gpu-container
    image: nvidia/cuda:10.0-base
    resources:
      limits:
        nvidia.com/gpu: 1
```



Karpenter simplifies Kubernetes infrastructure with the right nodes at the right time

CNCF donation? #756

Closed **elsesiy** opened this issue on Dec 10, 2021 · 6 comments

elsesiy commented on Dec 10, 2021

Are there any plans to donate this project to the CNCF or host it in any outside organization?
Right now karpenter only supports aws, however I anticipate the community will add support for other clouds soon. Having the project under a well-known umbrella could help drive adoption.

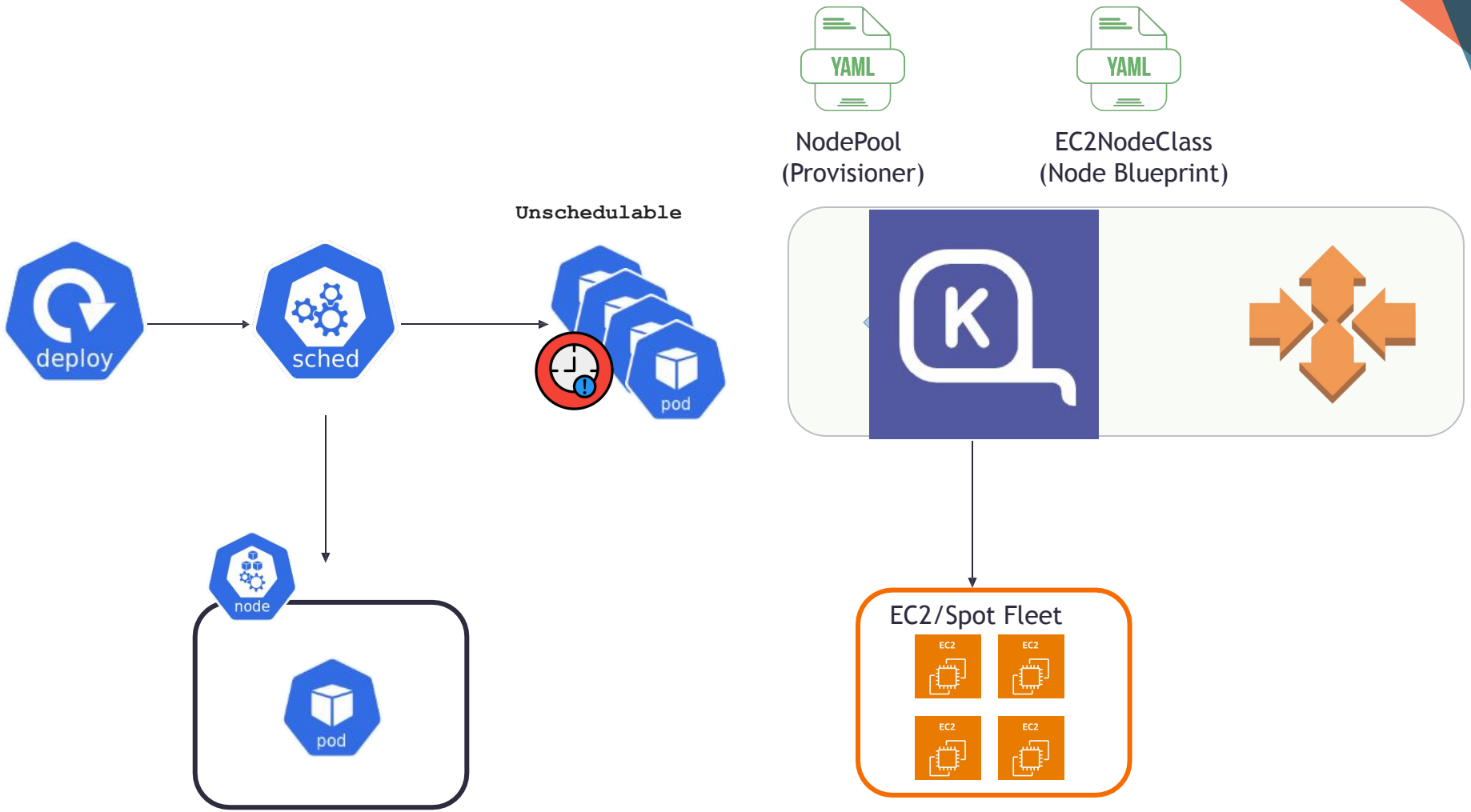
Community Note

- Please vote on this issue by adding a 👍 reaction to the original issue to help the community and maintainers prioritize this request
- Please do not leave "+1" or "me too" comments, they generate extra noise for issue followers and do not help prioritize the request
- If you are interested in working on this issue or have submitted a pull request, please leave a comment

👍 73

elsesiy added the kind/feature label on Dec 10, 2021

ellistarn added neutrality kind/support and removed kind/feature labels on Dec 10, 2021


YAML
YAML
NodePool
(Provisioner)
EC2NodeClass
(Node Blueprint)
Unschedulable
deploy
sched
pod
node
pod
EC2/Spot Fleet
EC2
EC2
EC2
EC2

NodePool

Pod Scheduling Configuration

- Define requirements and preferences for node selection, such as instance types, architectures, and capacity types.

Resource Allocation

- Specify resource requests and limits to ensure efficient utilization of node resources.

Taints and Tolerations

- Set node taints and pod tolerations to control which pods can be scheduled on which nodes.

Auto-scaling Parameters

- Configure scaling behavior, including minimum and maximum node counts, and target utilization to optimize resource usage and cost.

```yaml
apiVersion: karpenter.sh/v1beta1
kind: NodePool
metadata:
  name: default
spec:
  weight: 10
  limits:
    cpu: 1000
    memory: 1000Gi
    nvidia.com/gpu: 2
  template:
    spec:
      requirements:
        - key: karpenter.k8s.aws/instance-family
          operator: In
          values: ["m6g", "r5"]
        - key: karpenter.k8s.aws/instance-size
          operator: NotIn
          values: ["nano", "micro"]
        - key: topology.kubernetes.io/zone
          operator: In
          values: ["us-east-2a", "us-east-2b"]
        - key: kubernetes.io/arch
          operator: In
          values: ["amd64", "arm64"]
        - key: karpenter.sh/capacity-type
          operator: In
          values: ["spot", "on-demand"]
```

In order for a pod to run on a node defined in this NodePool, it must tolerate nvidia.com/gpu in its pod spec

```yaml
apiVersion: karpenter.sh/v1beta1
kind: NodePool
metadata:
  name: gpu
spec:
  disruption:
    consolidationPolicy: WhenUnderutilized
  template:
    spec:
      requirements:
      - key: node.kubernetes.io/instance-type
        operator: In
        values: ["p3.8xlarge", "p3.16xlarge"]
      taints:
      - key: nvidia.com/gpu
        value: "true"
        effect: NoSchedule
```

Node Classes

Instance Configuration

- Define AMI family, instance type, and instance profile for node identity and permissions.

Network Settings

- Select subnets and security groups to attach to nodes, ensuring proper networking and security configurations.

Storage Options

- Configure block device mappings and instance store volumes for customized storage needs.

Custom Tags and Metadata

- Apply custom tags and set metadata options for nodes, allowing for detailed management and tracking of instances.

```yaml
apiVersion: karpenter.k8s.aws/v1beta1
kind: EC2NodeClass
metadata:
  name: default
spec:
  amiFamily: AL2
  role: "KarpenterNodeRole-${NAME}"
  instanceProfile: "KarpenterNodeInstanceProfile-${NAME}"
  tags:
    team: team-a
  blockDeviceMappings:
    - deviceName: /dev/xvda
      ebs:
        volumeSize: 100Gi
        volumeType: gp3
        encrypted: true
  associatePublicIPAddress: true
status:
  subnets:
    - id: subnet-0a462d98193ff9fac
  securityGroups:
    - id: sg-041513b454818610b
  amis:
    - id: ami-01234567890123456
```

We need to ensure that pods are scheduled onto the appropriate nodes provisioned by Karpenter

NodePool Provisioner

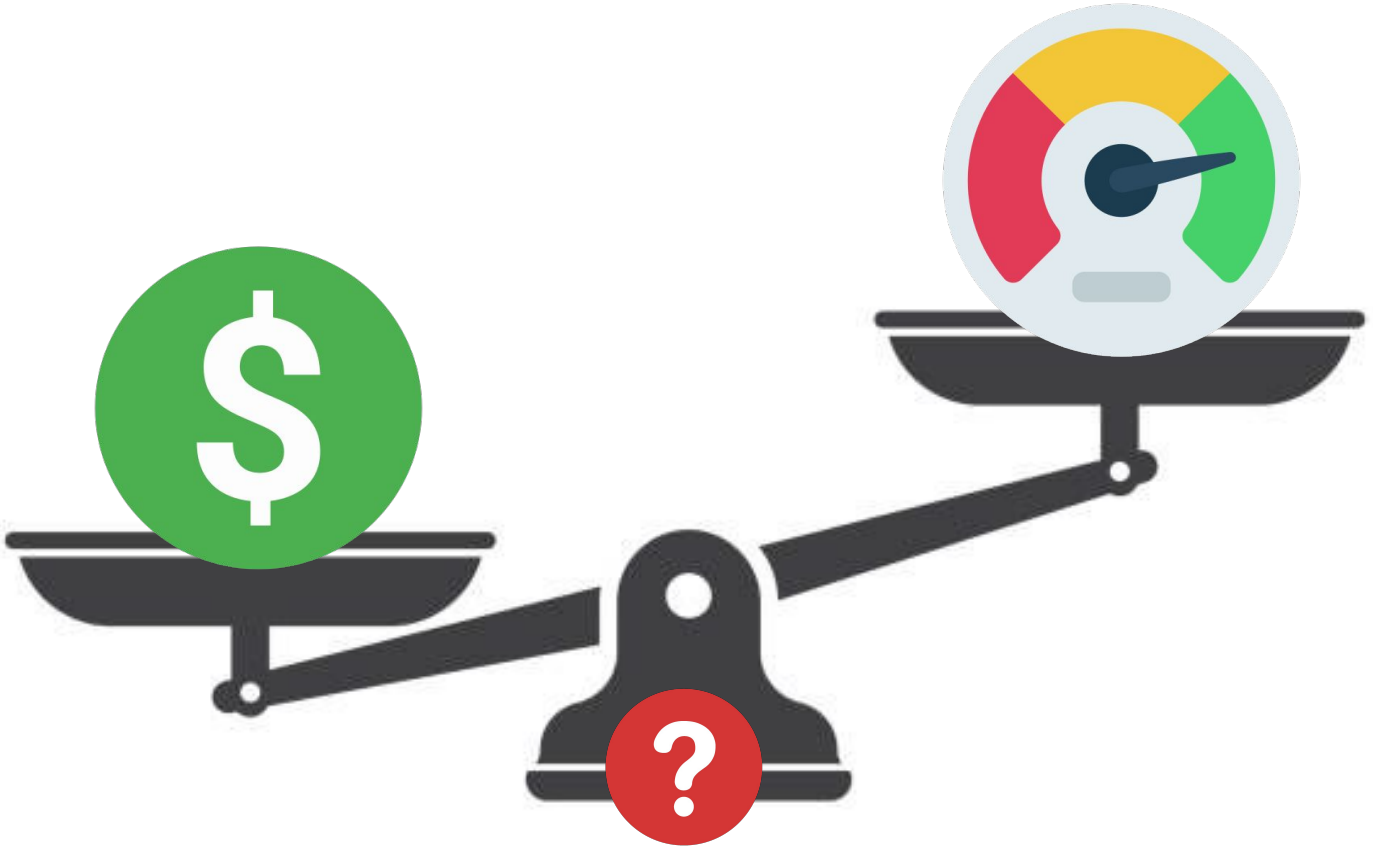
Capacity type

Architecture

Region

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
  labels:
    app: example-pod
spec:
  containers:
    - name: example-container
      image: nginx
      resources:
        requests:
          memory: "128Mi"
          cpu: "500m"
  nodeSelector:
    karpenter.sh/capacity-type: spot
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: kubernetes.io/arch
                operator: In
                values: ["amd64", "arm64"]
      preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 1
          preference:
            matchExpressions:
              - key: topology.kubernetes.io/zone
                operator: In
                values: ["us-east-1a", "us-east-1b"]
  topologySpreadConstraints:
    - maxSkew: 1
      topologyKey: topology.kubernetes.io/zone
      whenUnsatisfiable: DoNotSchedule
      labelSelector:
        matchLabels:
          app: example-pod
```

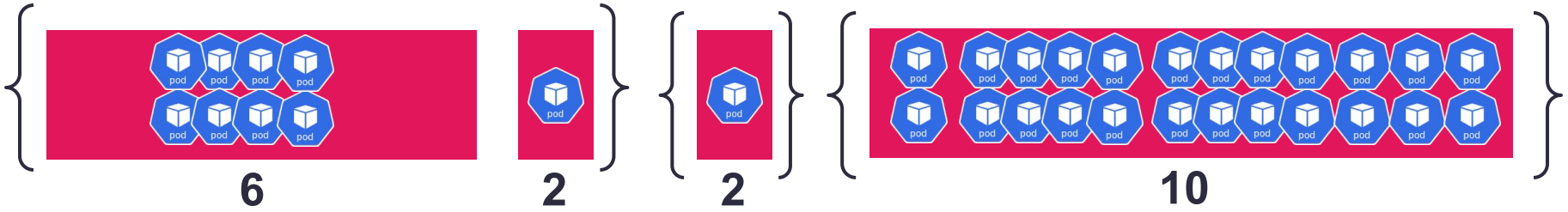
Our goal is to scale while staying within our budget



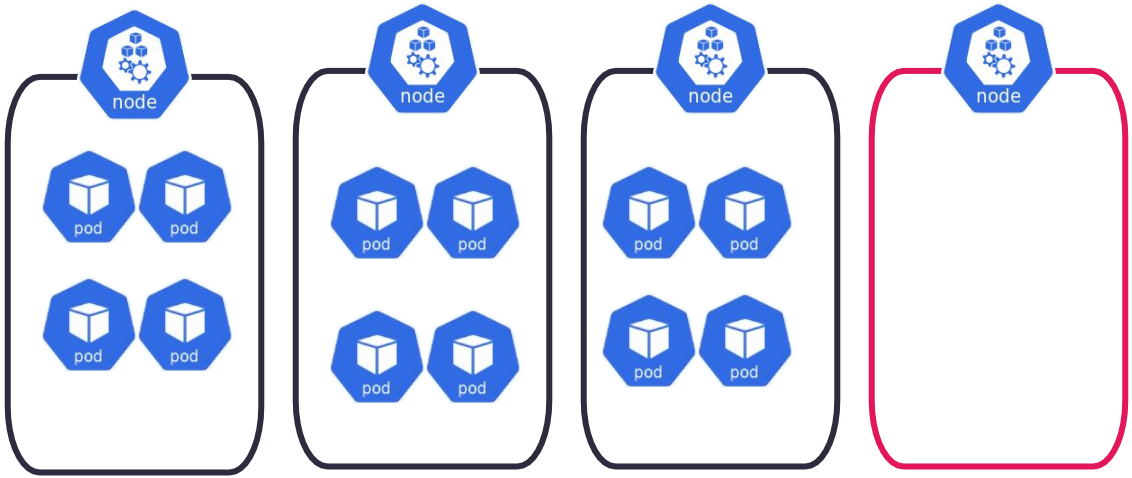
Node scheduling - Batching

BATCH_IDLE_DURATION = 2 sec

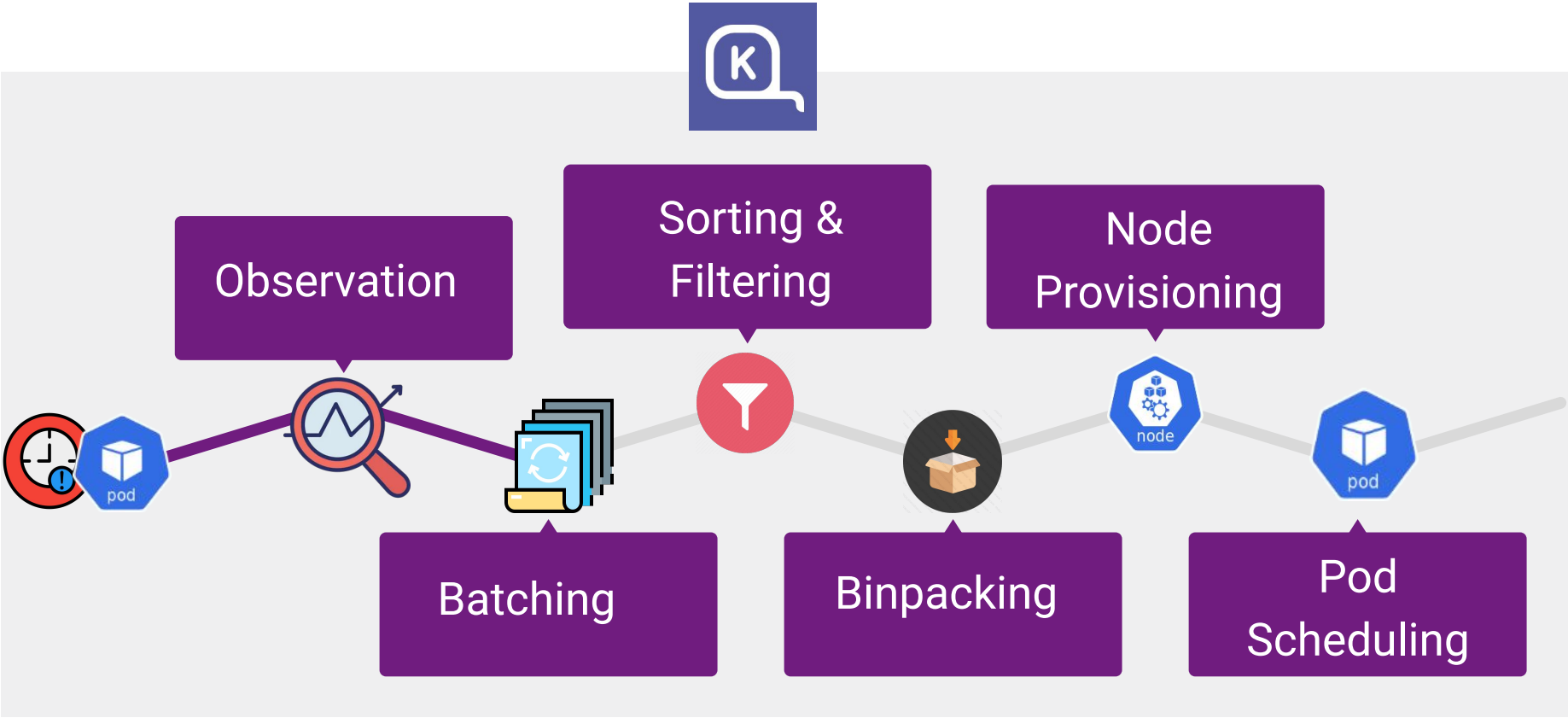
BATCH_MAX_DURATION = 10 sec

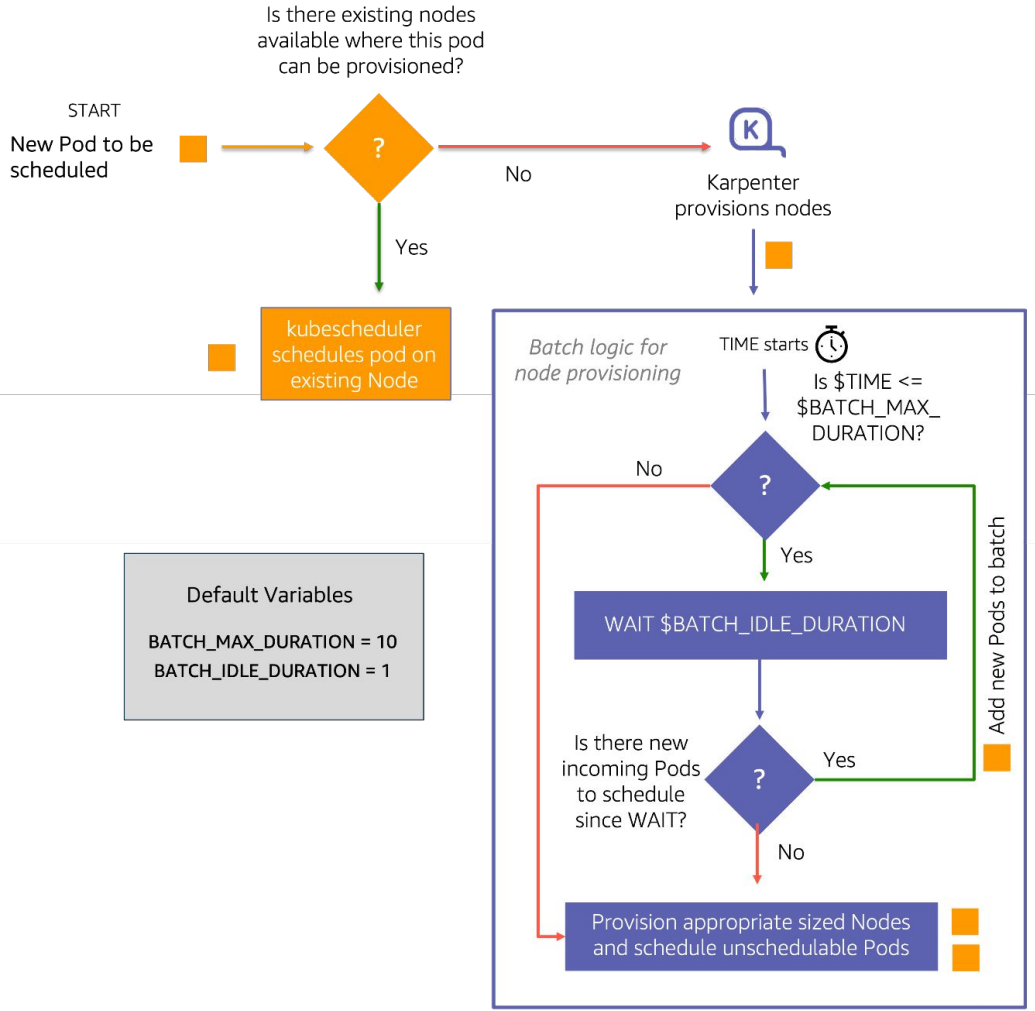


Node consolidation - Bin Packing

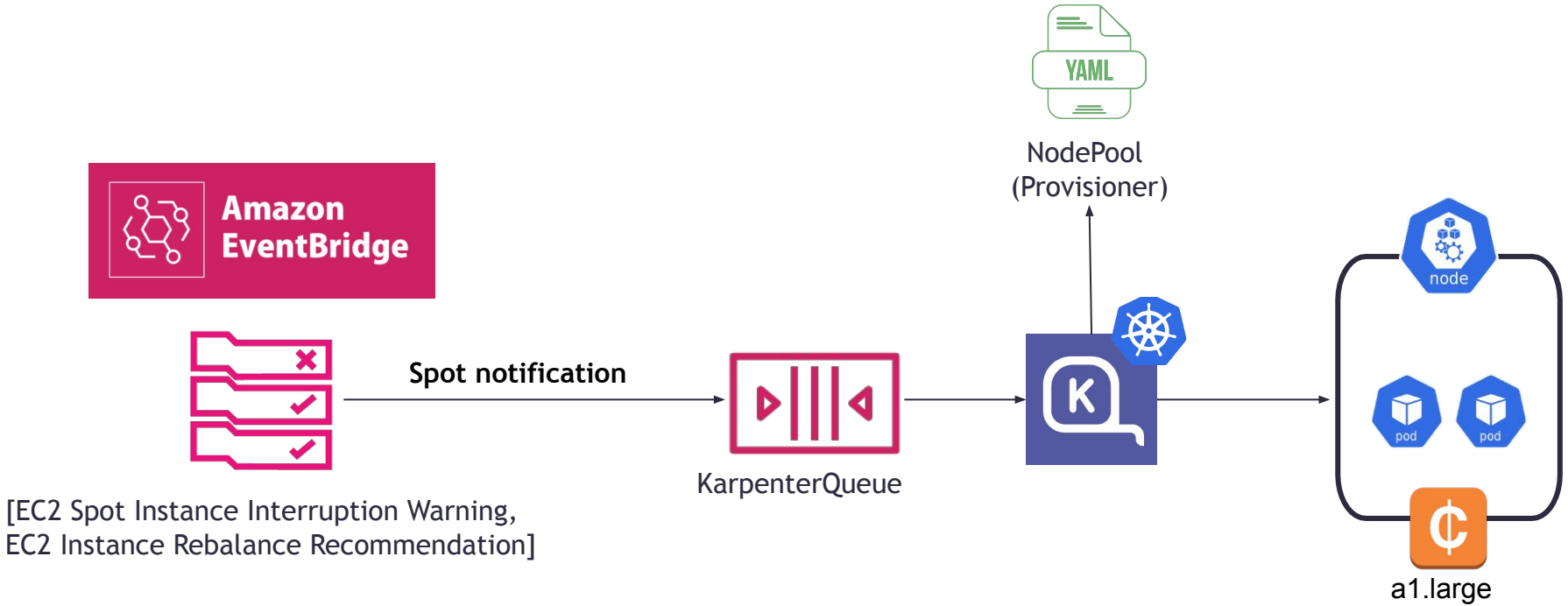



```
spec:
  disruption:
    consolidationPolicy: WhenUnderutilized
```


Observation
Sorting & Filtering
Node Provisioning
pod
node
pod
Batching
Binpacking
Pod Scheduling


Is there existing nodes available where this pod can be provisioned?
START
New Pod to be scheduled
?
No
Karpenter provisions nodes
Yes
kubescheduler schedules pod on existing Node
Batch logic for node provisioning
TIME starts
Is $TIME <= $BATCH_MAX_DURATION?
No
?
Yes
WAIT $BATCH_IDLE_DURATION
Is there new incoming Pods to schedule since WAIT?
?
Yes
Add new Pods to batch
No
Provision appropriate sized Nodes and schedule unschedulable Pods
Default Variables
BATCH_MAX_DURATION = 10
BATCH_IDLE_DURATION = 1

Spot handling with Karpenter



Node disruption

- **Budget Constraints -** Allocate resources within predefined limits to maintain high availability
- **Voluntary -** Planned activities such as maintenance, scaling, or configuration changes initiated by administrators
- **Involuntary** - Unplanned events like hardware failures, spot instance interruptions, or unexpected terminations by the cloud provider
- **Drifted -** Nodes are disrupted when they deviate from their desired configuration specified in *NodePool or EC2NodeClass*

```yaml
apiVersion: karpenter.sh/v1beta1
kind: NodePool
metadata:
  name: default
spec:
  disruption:
    consolidationPolicy: WhenUnderutilized
    expireAfter: 720h # 30 * 24h = 720h
    budgets:
    - nodes: "20%"
    - nodes: "5"
    - nodes: "0"
      schedule: "@daily"
      duration: 10m
```


DEMO

Alex Kestner • 2nd

Sr. Product Manager at AWS Elastic Kubernetes Servi...

4d •

+ Follow

Nearly three years ago, as part of the Amazon EKS team, Ellis Tarn and I launched Karpenter, a scrappy open-source Kubernetes cluster autoscaler. Today, it's a comprehensive Kubernetes compute management solu ...see more

Announcing Karpenter 1.0 - AWS

aws.amazon.com

475

23 comments

Like Comment Repost Send

Karpenter 1.0

Feature	Details
Disruption Controls by Reason	Allows you to set specific rules for when and why nodes can be disrupted, such as when they are underutilized or empty
Consolidation Policy Renamed	The policy for removing underutilized or empty nodes is now called WhenEmptyOrUnderutilized to clarify its purpose
ConsolidateAfter Control	Introduces a delay option for consolidating nodes, giving you more control over when underutilized nodes are removed
Termination Grace Period	Sets a maximum time for how long a node can be in the shutdown process before it is forcefully terminated to ensure compliance with security policies

Making Dave

HAPPY

TERASKY

TERASKY

Don't tell anyone, but you still need to rightsize your Kubernetes workloads