
AWS
re:Invent

CMP402

Scaling Kubernetes for less with Amazon EC2 Spot Instances

Isaac Vallhonrat

Sr. Specialist Solutions Architect, EC2 Spot

AWS

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

Amazon EC2 Spot Instances overview

Running Kubernetes nodes on EC2 Spot Instances



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon EC2 Spot Instances overview



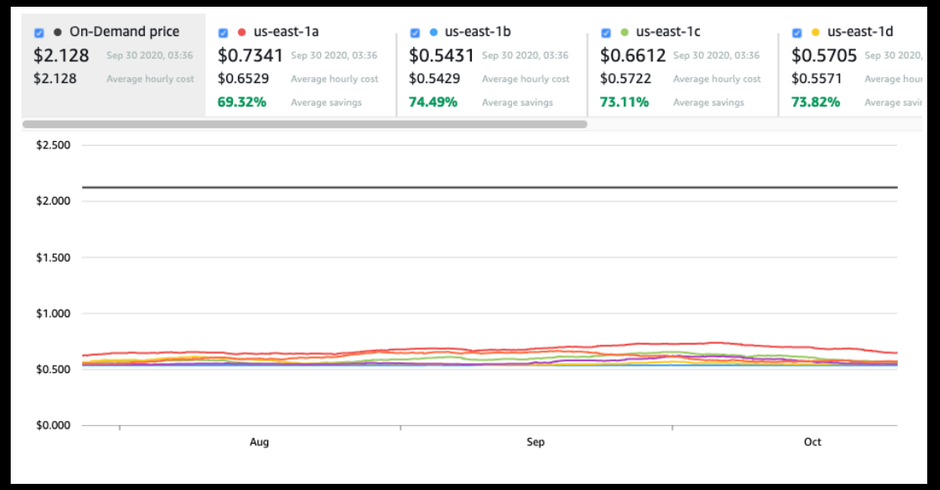
AWS re:Invent

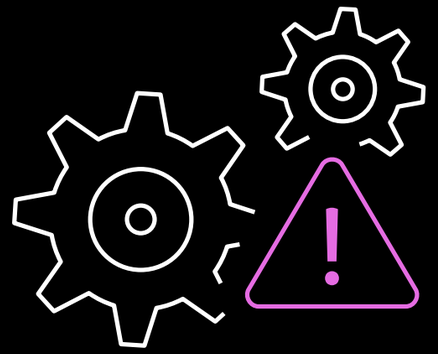
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What are EC2 Spot Instances?

Spot Instances are a spare EC2 capacity made available to customers as a purchase option with steep discounts







Spare EC2 capacity

Same infrastructure as On-Demand

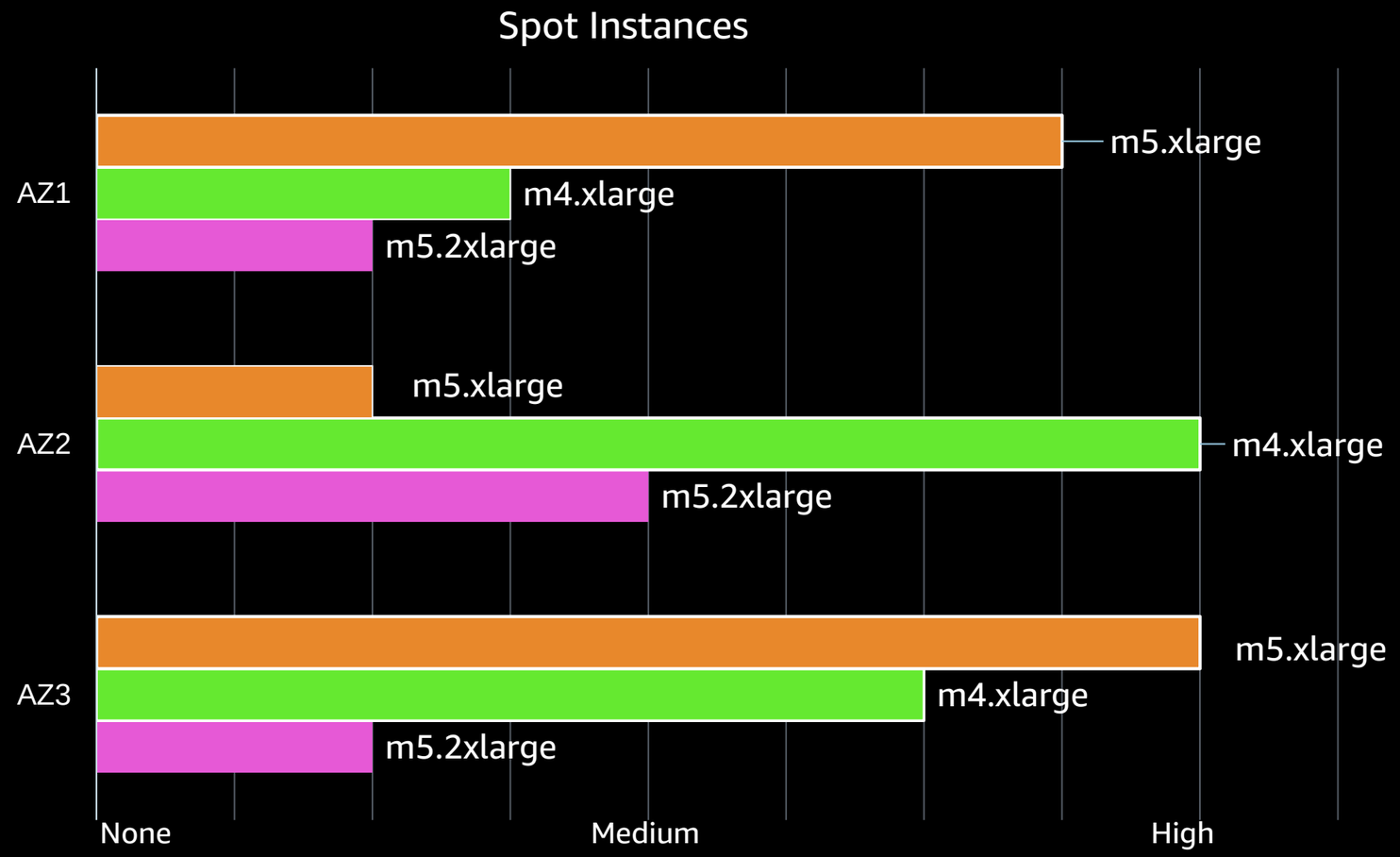
Up to 90% less than On-Demand pricing

Spot price is based on long-term supply and demand trends of EC2 instances (no bidding)

Interruptible

Spot Instances can be interrupted with a 2-minute warning if EC2 needs the capacity back

Understanding EC2 Spot capacity pools



Each instance type

in each Availability Zone

in every Region

is a separate Spot capacity pool

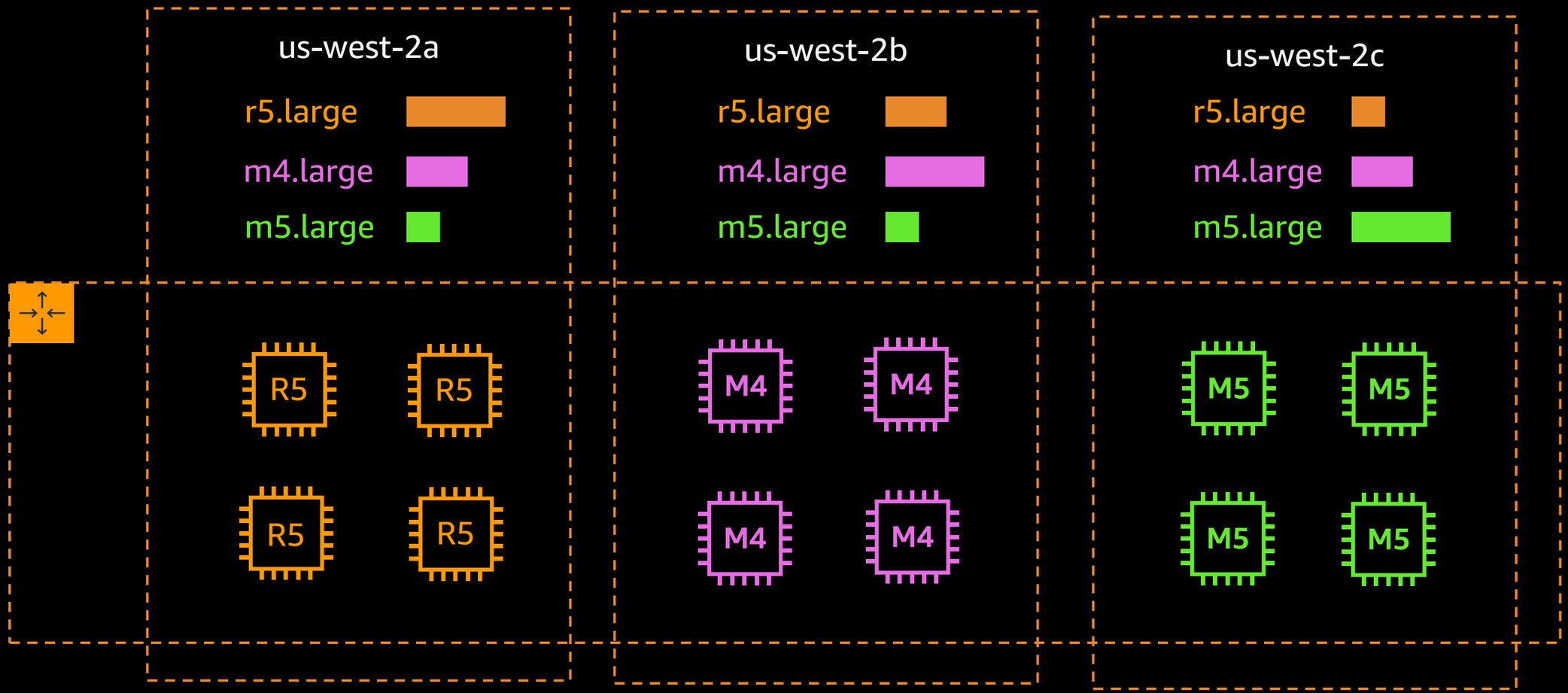
*Example figures

Availability Zone and instance type flexibility (e.g., family, generation, size) are key to provisioning and maintaining your target capacity

Launch instances from optimal pools with ASG

`Desired capacity: 12` `OnDemandBaseCapacity: 0` `OnDemandPercentageAboveBaseCapacity: 0`

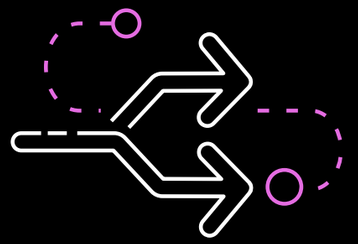
`Overrides: ["r5.large", "m4.large", "m5.large"]` `SpotAllocationStrategy: capacity-optimized`



Amazon EC2 Auto Scaling Capacity Rebalancing

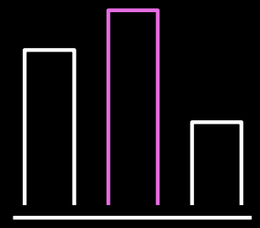
Automatically attempt to maintain desired capacity through the Spot Instance lifecycle

New!

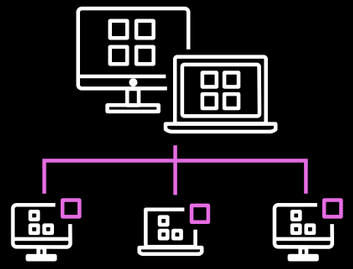


Attempt to replace Spot Instances before interruption, leveraging the **EC2 instance rebalance recommendation**

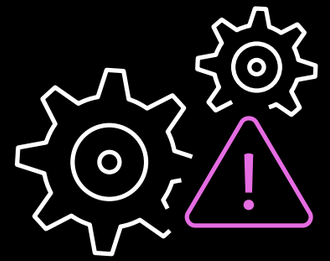
New!



Leverage the capacity-optimized allocation strategy to launch replacement capacity

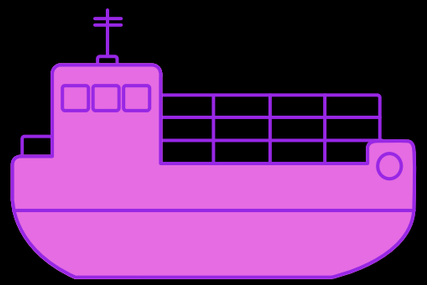


Automatically deregister from the load balancer



Execute actions before instance termination with lifecycle hooks

Why containers are a good fit to run on Spot



- Containerized applications are often stateless and fault-tolerant; they can normally run on Spot Instances with minimal engineering effort
- Spot Instances can be used with Amazon ECS or Amazon EKS to run any fault-tolerant workload

Delivery Hero

Delivery Hero is a leading online food ordering and delivery marketplace. The company operates delivery fleets in 43 countries, transporting more than 4 million food orders a day.

“Our experience running Amazon EKS on Amazon EC2 Spot Instances was eye-opening. It has become a big cost saver and freed our time and energy to focus on business growth instead.”

Vojtech Vondra
Senior Director of Engineering, Logistics
Delivery Hero

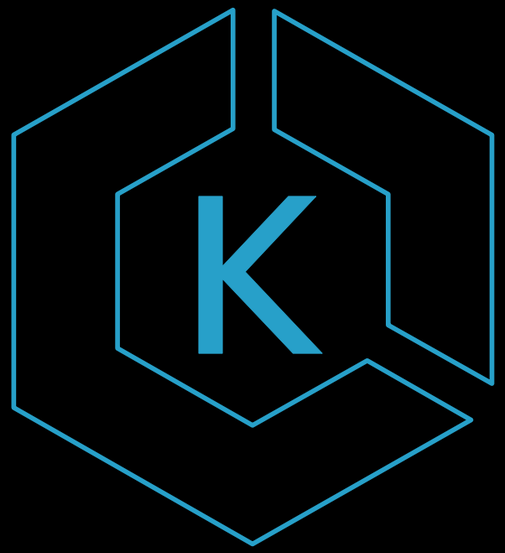
Running Kubernetes nodes on Spot Instances



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon EKS



Amazon EKS

Upstream Kubernetes and certified Kubernetes conformant

- Uses standard Kubernetes APIs
- Works with community tools

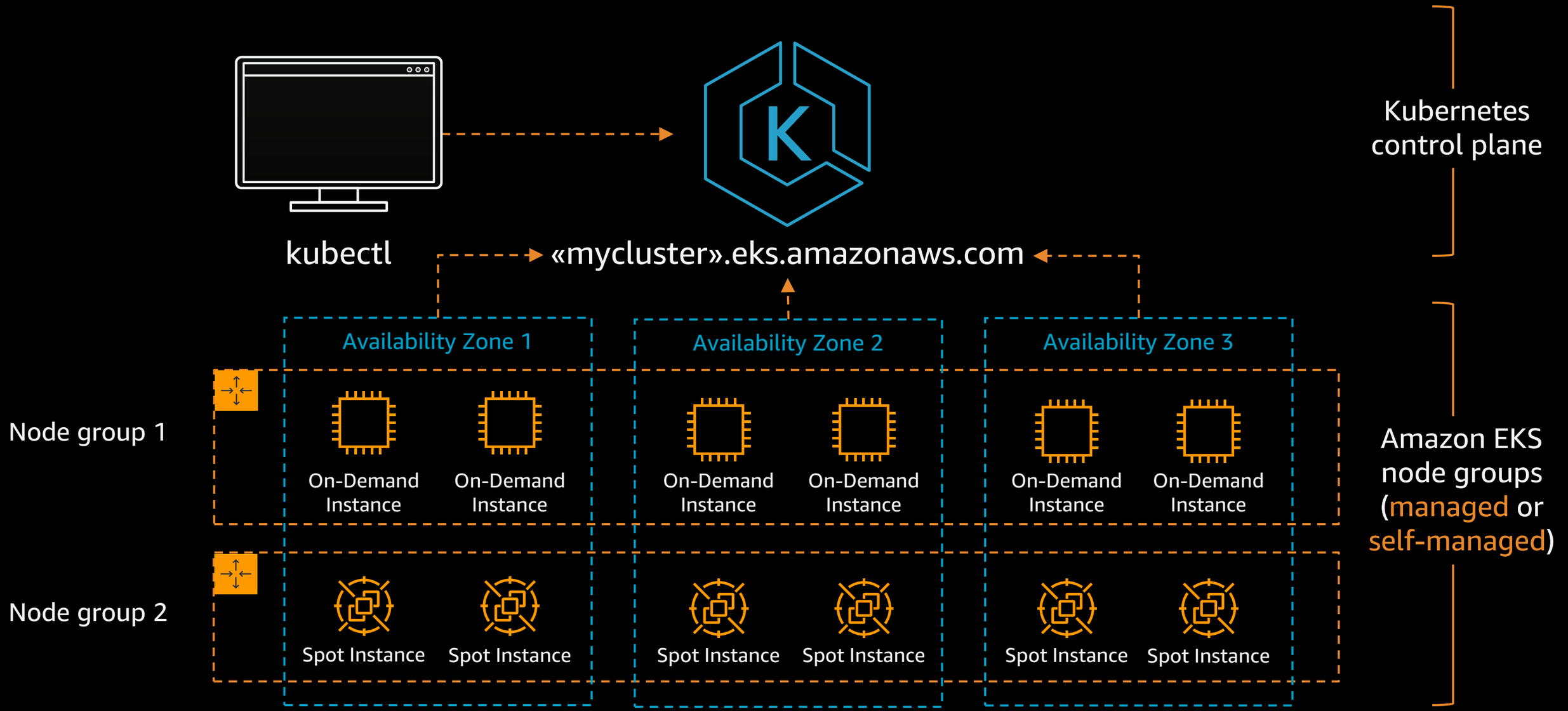
Highly available

- Amazon EKS runs the Kubernetes control plane across multiple Availability Zones for all clusters
- Backed by a 99.95% service-level agreement

Integrated with other AWS services

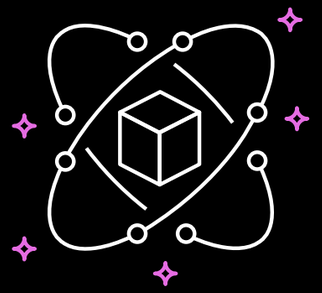
- VPC networking
- Elastic Load Balancing
- AWS Identity and Access Management (IAM)
- Amazon CloudWatch
- And more

A high-level view of an Amazon EKS cluster



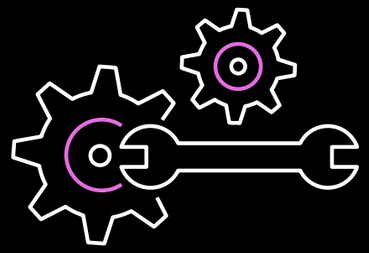
Amazon EKS managed node groups with EC2 Spot Instances

New!



Automatic node provisioning

- Support for mixed instance types
- Amazon EC2 Auto Scaling group provisioning with capacity-optimized allocation strategy
- Automatically configures Kubernetes labels and AWS tags (e.g., cluster autoscaler)



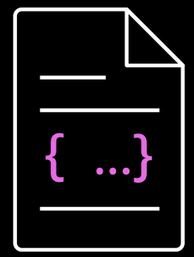
Managed node update and termination

Managed node draining during node termination and updates



Spot Instance lifecycle management

Managed Spot Instance draining integrated with Amazon EC2 Auto Scaling Capacity Rebalancing



Launch template support

Custom configuration and custom AMIs for your Kubernetes nodes

Configuring your Spot-managed node group



my-cluster-definition.yaml

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
    name: my-eks-cluster
    region: us-west-2
managedNodeGroups:
      - name: my-spot-managed-node-group
        desiredCapacity: 2
        spot: true
        instanceTypes:
            - m5.xlarge
            - m5d.xlarge
            - m5a.xlarge
            - m5ad.xlarge
            - m4.xlarge
```



Select same-size instance types (4 vCPUs and 16 GiB memory) for consistent scaling performance with Kubernetes Cluster Autoscaler

```
$ eksctl create nodegroup -f my-cluster-definition.yaml
```

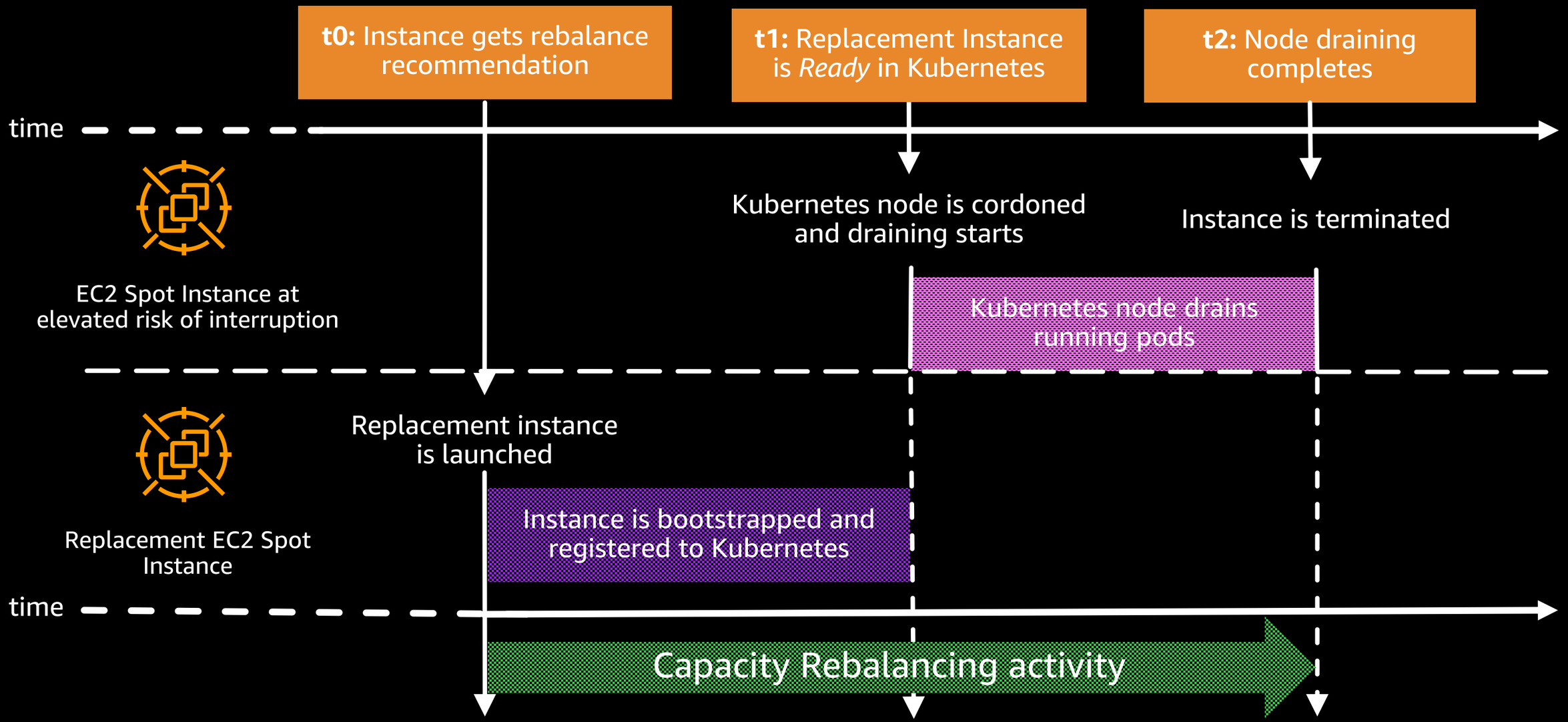
ec2-instance-selector

A CLI tool and go library that recommends instance types on the basis of resource criteria

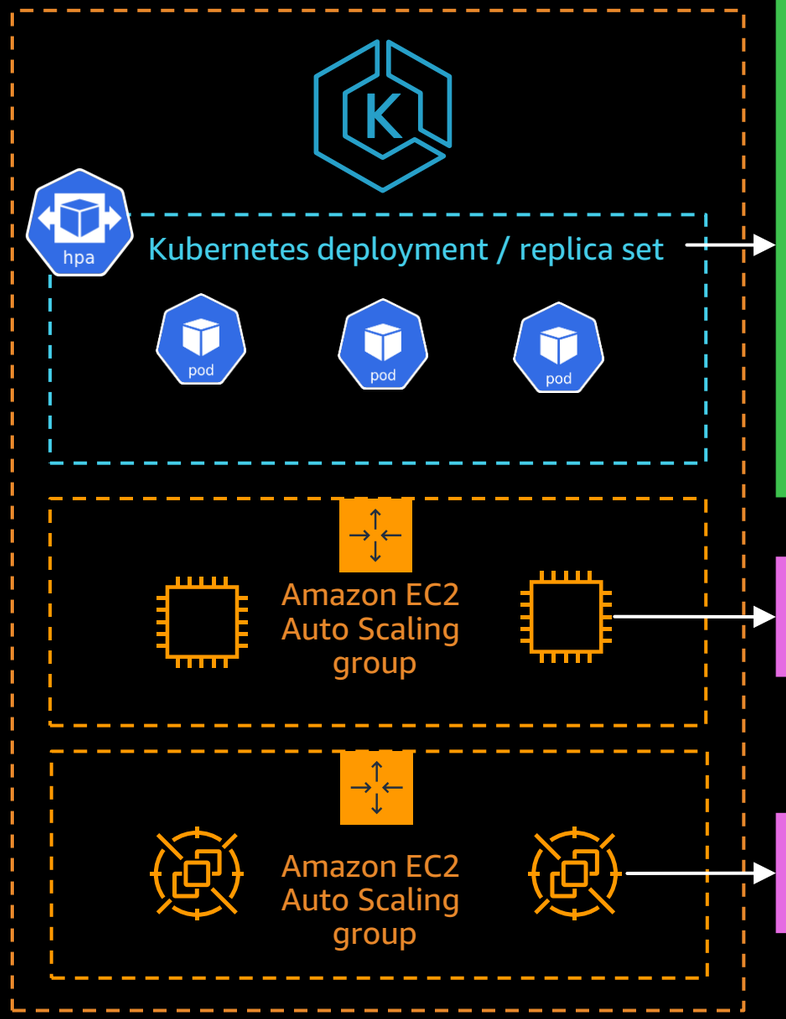
```
$ ec2-instance-selector --vcpus 4 --memory 16 --cpu-architecture x86_64 --gpus-max 0 -o table
Instance Type        VCPUs        Mem (GiB)
-------------        -----        ---------
m4.xlarge            4            16.000
m5.xlarge            4            16.000
m5a.xlarge           4            16.000
m5ad.xlarge          4            16.000
m5dn.xlarge          4            16.000
…
```

https://github.com/aws/amazon-ec2-instance-selector

Managed node group Spot Capacity Rebalancing activity



Controlling pod scheduling: nodeAffinity




```
apiVersion: v1
Kind: Deployment
…
spec:
  …
  affinity:
    nodeAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 1
          preference:
          matchExpressions:
            - key: eks.amazonaws.com/capacityType
              operator: In
              values:
                - SPOT
  …
```

```
eks.amazonaws.com/capacityType:
ON_DEMAND
```

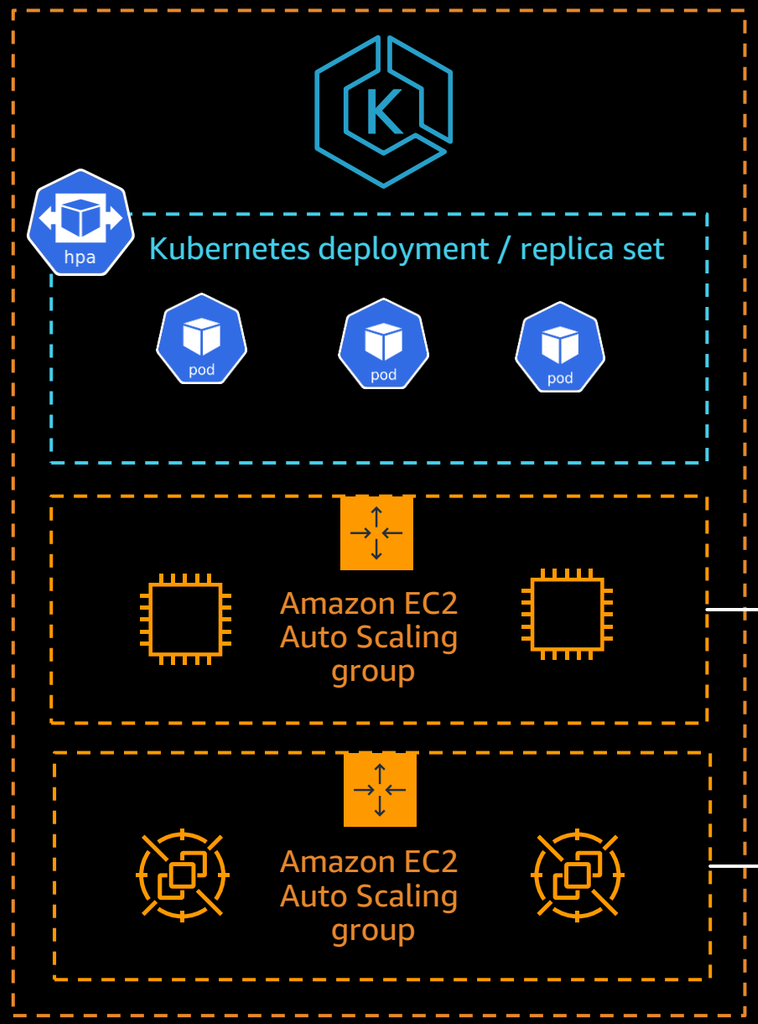
```
eks.amazonaws.com/capacityType:
SPOT
```

Use Kubernetes labels for simple node selection constraints with *nodeSelector*

Or use richer expressions to control pod scheduling using nodeAffinty:

- Soft / preference: *preferredDuringSchedulingIgnoredDuringExecution*
- Hard / requirement: *requiredDuringSchedulingIgnoredDuringExecution*

Scaling the application and the underlying infrastructure



Horizontal pod autoscaler

Scale number of pods in a deployment on the basis of observed CPU utilization or application-provided metrics

```
desiredReplicas = ceil[currentReplicas * ( currentMetricValue / desiredMetricValue )]
```

Kubernetes Cluster Autoscaler

- Scale-out Amazon EC2 Auto Scaling groups of nodes matching pod scheduling requirements when pods are in a *pending* state because of insufficient resources
- Scale in when nodes are underutilized and important pods can be rescheduled somewhere else

Auto Scaling group tags:

```
k8s.io/cluster-autoscaler/enabled: true
k8s.io/cluster-autoscaler/${cluster_name}: owned
```

Going large scale with multiple node groups

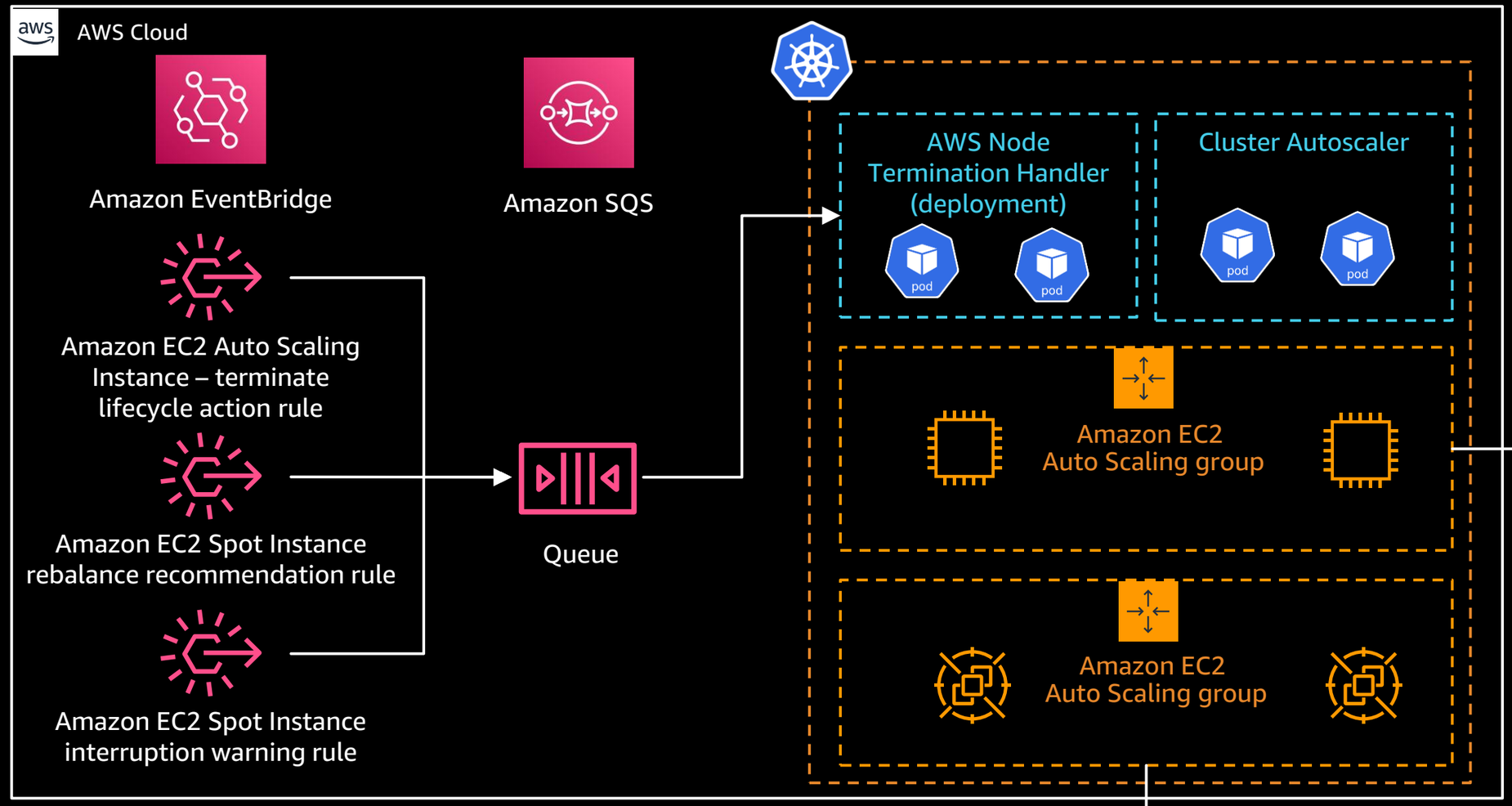


- Create multiple node groups with same-size instance types and the same labels / taints
- Let the Kubernetes Cluster Autoscaler *random* expander span your instances across similar node groups (i.e., the same labels / taints)

DIY: Handling interruptions on Kubernetes

New!

AWS Node Termination Handler queue processor for self-managed node groups / DIY Kubernetes



All ASGs configuration:

EC2_INSTANCE_TERMINATING lifecycle hook: enabled

- Scale-in events
- AZRebalance events
- Instance termination events

Spot Instances ASGs:

Capacity Rebalancing: enabled

- Attempts launch of replacement capacity
- Upon succesful replacement, triggers instance termination lifecycle hook

Auto Scaling group tags:
`aws-node-termination-handler/managed:`

https://github.com/aws/aws-node-termination-handler

Takeaways

- Save up to 90% on compute costs with Spot Instances for your interruption-tolerant and flexible workloads
- Leverage Amazon EKS managed node groups to offload Spot Instances lifecycle management and configuration of Kubernetes nodes
- Use multiple instance types (families and generations) and Availability Zones, and the capacity-optimized allocation strategy to provision and maintain the desired capacity
- Leverage ec2-instance-selector to select suitable instance types
- For self-managed options, use the aws-node-termination-handler queue processor with Auto Scaling Capacity Rebalancing to manage Spot Instances lifecycle

Thank you!

Isaac Vallhonrat

@ivallho

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.