AWS re:Invent

CMP416-R

Scale Kubernetes for less using Spot Instances

Isaiah Campbell
Sr. Business Development Manager, EC2 Spot
Amazon Web Services

AWS re:Invent

Agenda

- Amazon containers landscape & Amazon Elastic Kubernetes Service (Amazon EKS) recap
- Amazon Elastic Compute Cloud (Amazon EC2) Spot Instances: Overview and best practices
- Applying Spot best practices to Kubernetes/Amazon EKS
    - Adding Spot Instances to your EKS clusters
    - Handling Spot interruptions to avoid application impact
    - Scaling mechanisms for application and cluster elasticity
    - Taints, tolerations and affinity & tools
- Main takeaways

AWS container services landscape

Management
Deployment, scheduling, scaling & management of containerized applications

Amazon Elastic Container Service (Amazon ECS)

Amazon EKS

Hosting
Where the containers run

Amazon EC2

AWS Fargate

Image registry
Container image repository

Amazon Elastic Container Registry (Amazon ECR)

Amazon EKS

63%

of Kubernetes workloads run on AWS today

—CNCF survey

How are customers using Amazon EKS?

Microservices

PaaS

Platform-as-a-Service

Enterprise App Migration

Machine Learning

Amazon EKS is Kubernetes-certified

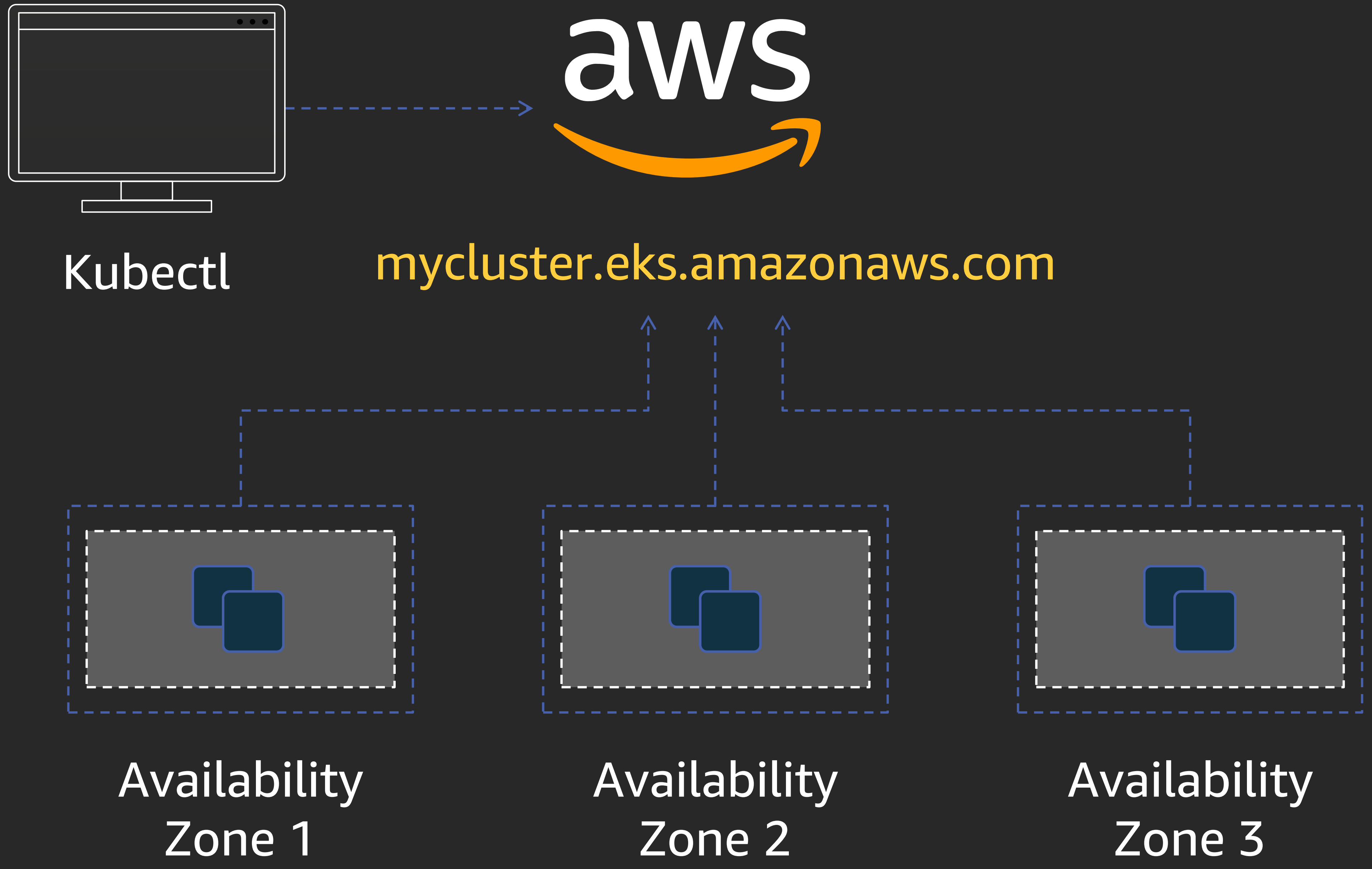
Tenet 1: EKS is a platform for enterprises to run production-grade workloads

Tenet 2: EKS provides a native and upstream Kubernetes experience

Tenet 3: If EKS customers want to use additional AWS services, the integrations are seamless and eliminate undifferentiated heavy lifting

Tenet 4: The EKS team actively contributes to the Kubernetes project

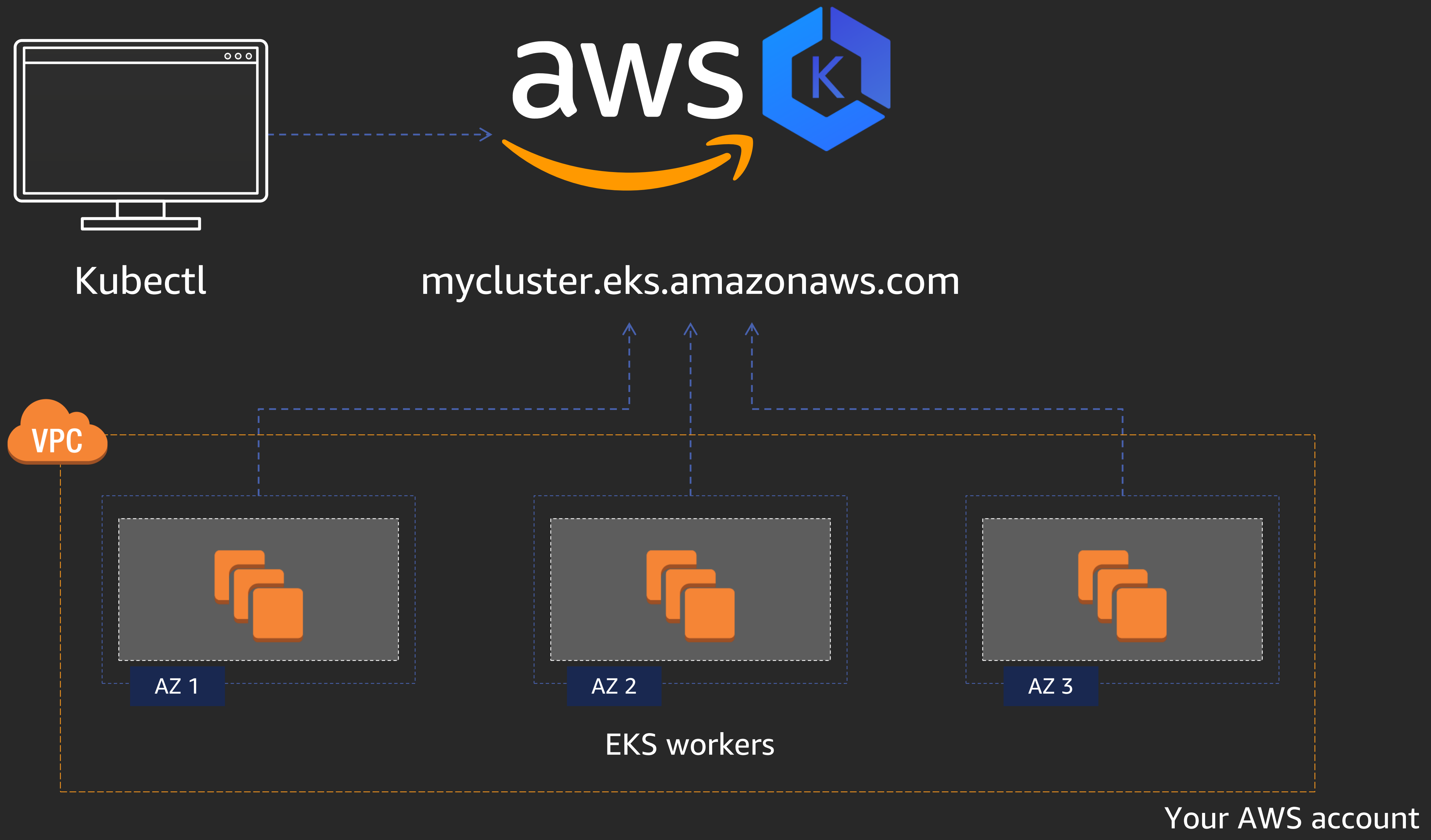
certified kubernetes

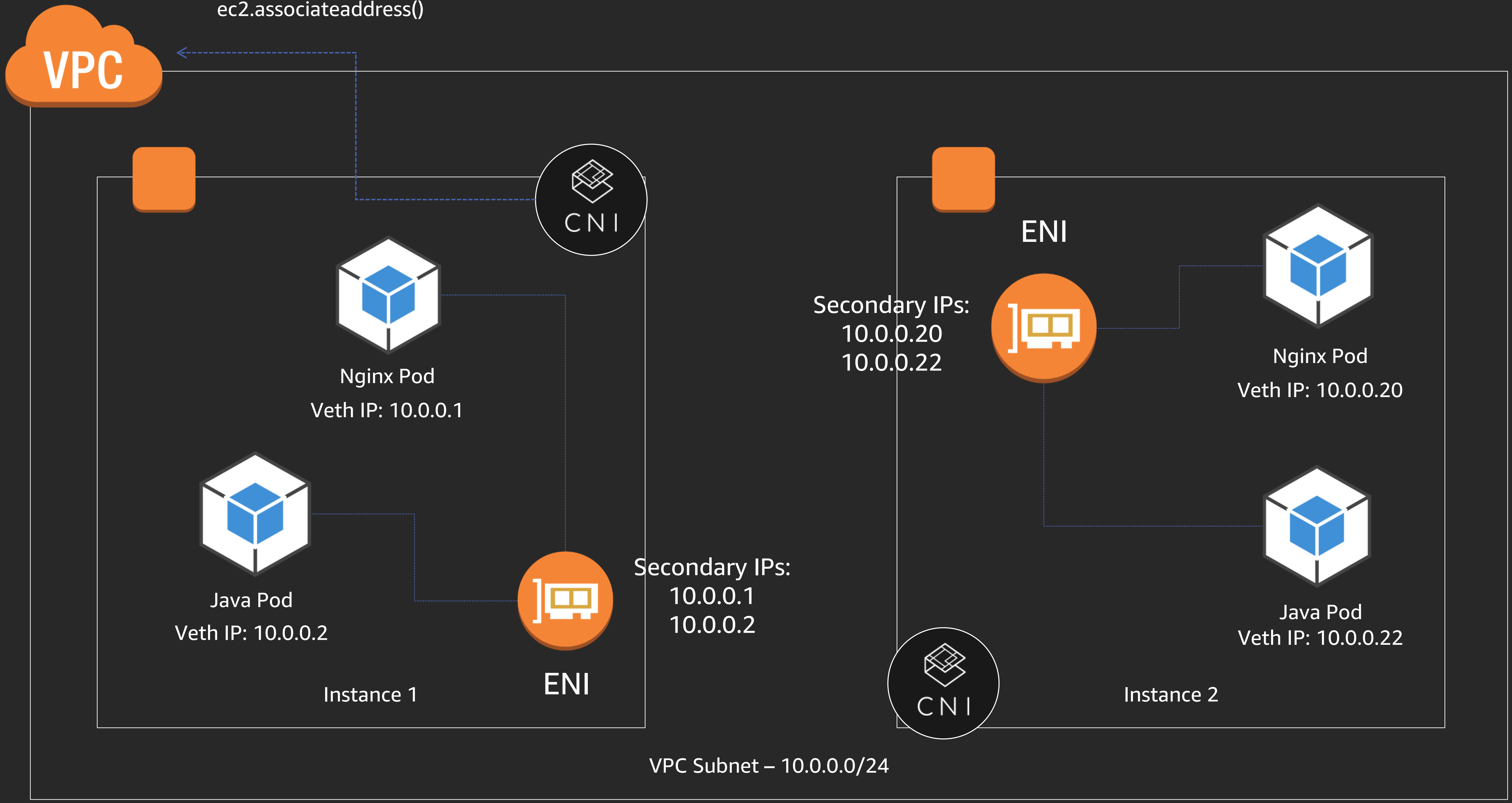

Kubectl
mycluster.eks.amazonaws.com
Availability Zone 1
Availability Zone 2
Availability Zone 3

Kubernetes master

API server	Controller manager	Cloud controller
Scheduler	KubeDNS	Add-ons

Amazon EKS


Kubectl
mycluster.eks.amazonaws.com
VPC
AZ 1
AZ 2
AZ 3
EKS workers
Your AWS account

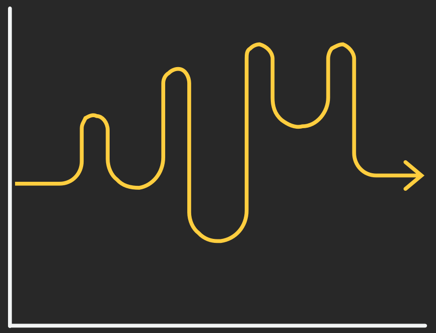

ec2.associateaddress()
VPC
CNI
Nginx Pod
Veth IP: 10.0.0.1
Java Pod
Veth IP: 10.0.0.2
ENI
Secondary IPs:
10.0.0.1
10.0.0.2
Instance 1
ENI
Secondary IPs:
10.0.0.20
10.0.0.22
Nginx Pod
Veth IP: 10.0.0.20
Java Pod
Veth IP: 10.0.0.22
CNI
Instance 2
VPC Subnet – 10.0.0.0/24

How do I provision Amazon EKS worker nodes?

Amazon EC2 purchase options

On-Demand

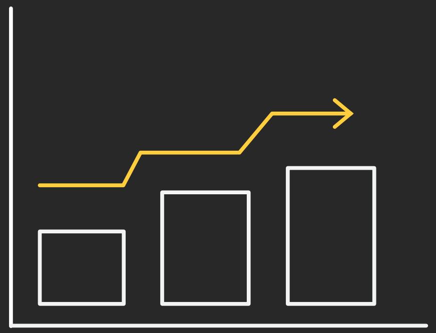
Pay for compute capacity by the second with no long-term commitments



Spiky workloads, to define needs

Reserved Instances

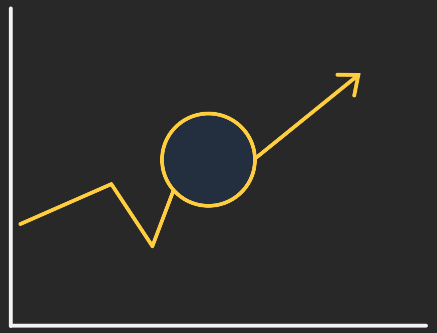
Make a 1- or 3-year commitment and receive a significant discount off On-Demand prices



Committed & steady-state usage

Spot Instances

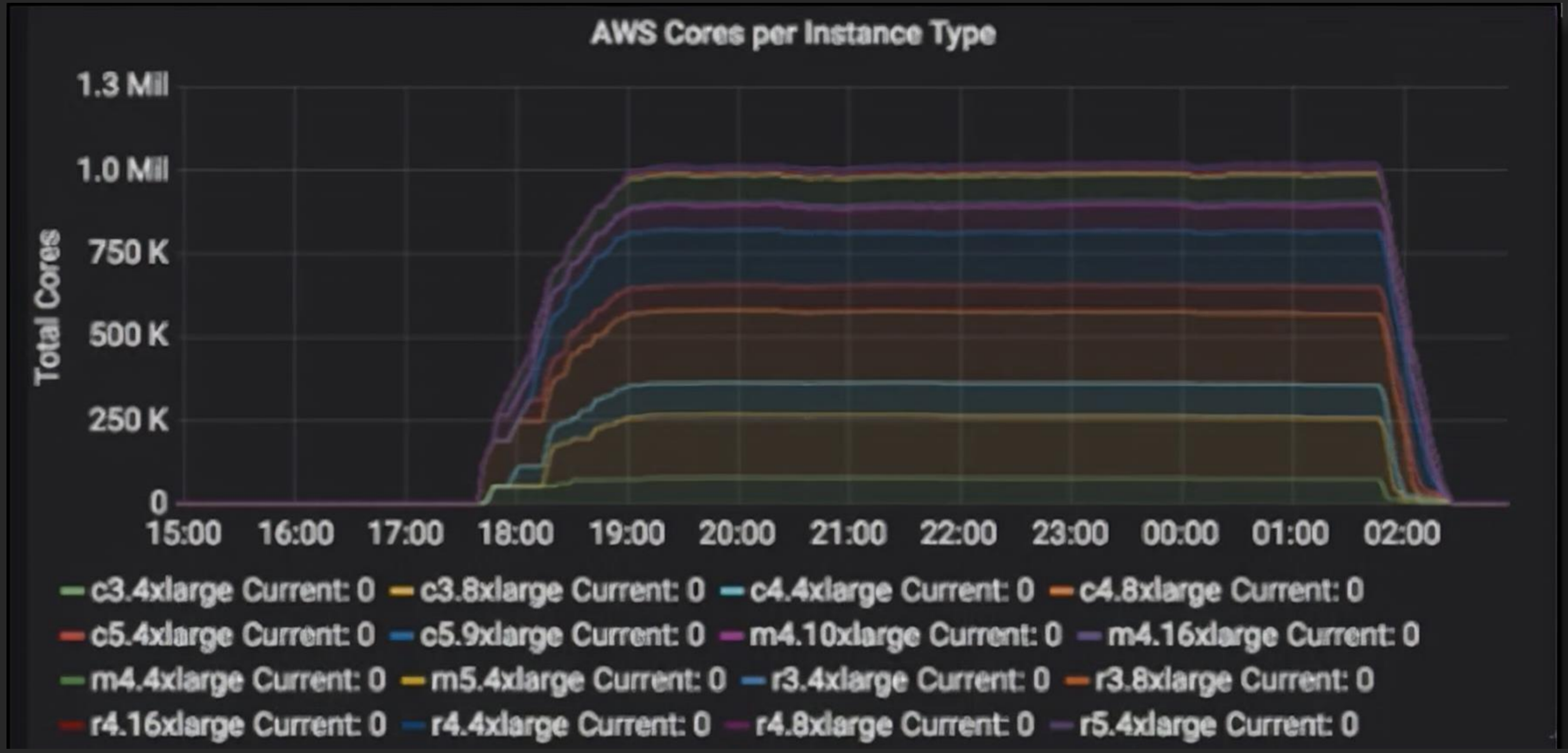
Spare EC2 capacity at savings of up to 90% off On-Demand prices



Fault-tolerant, flexible, stateless workloads

Amazon EC2 Spot Instances

Spare capacity at scale



Western Digital®

More than 2.3 million simulation jobs on a single HPC cluster of 1 million vCPUs built using Amazon EC2 Spot Instances.
Time to results: 20 Days → 8 hours

EC2 Spot pools—instance flexibility

C4	1a	1b	1c	On-Demand
8XL	$0.50	$0.27	$0.29	$1.76
4XL	$0.21	$0.30	$0.16	$0.88
2XL	$0.08	$0.07	$0.08	$0.44
XL	$0.04	$0.05	$0.04	$0.22
L	$0.01	$0.01	$0.04	$0.11

Each instance family

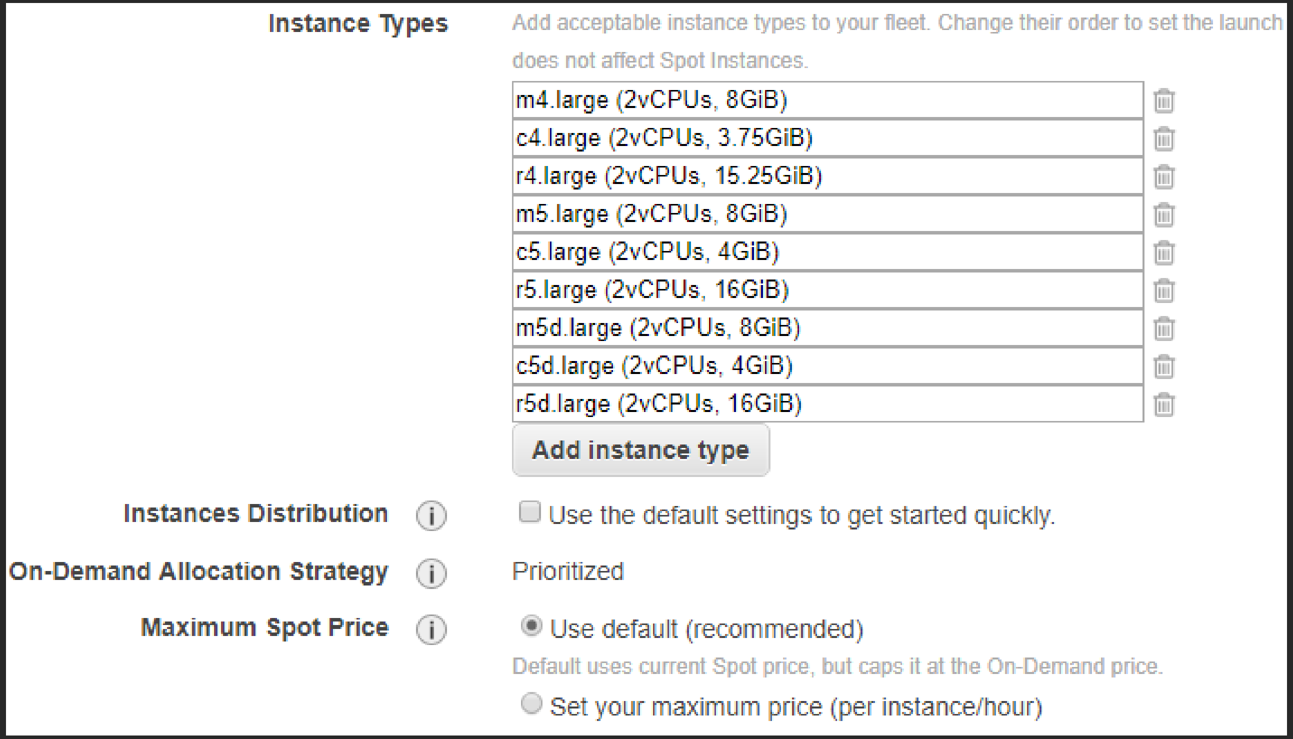
Each instance size

Each Availability Zone (66)

In every Region (21)

Is a separate Spot pool

R5 M4 I3 C5n M5d C5 R4 D2



Instance Types
Add acceptable instance types to your fleet. Change their order to set the launch
does not affect Spot Instances.

- m4.large (2vCPUs, 8GiB)
- c4.large (2vCPUs, 3.75GiB)
- r4.large (2vCPUs, 15.25GiB)
- m5.large (2vCPUs, 8GiB)
- c5.large (2vCPUs, 4GiB)
- r5.large (2vCPUs, 16GiB)
- m5d.large (2vCPUs, 8GiB)
- c5d.large (2vCPUs, 4GiB)
- r5d.large (2vCPUs, 16GiB)

Add instance type

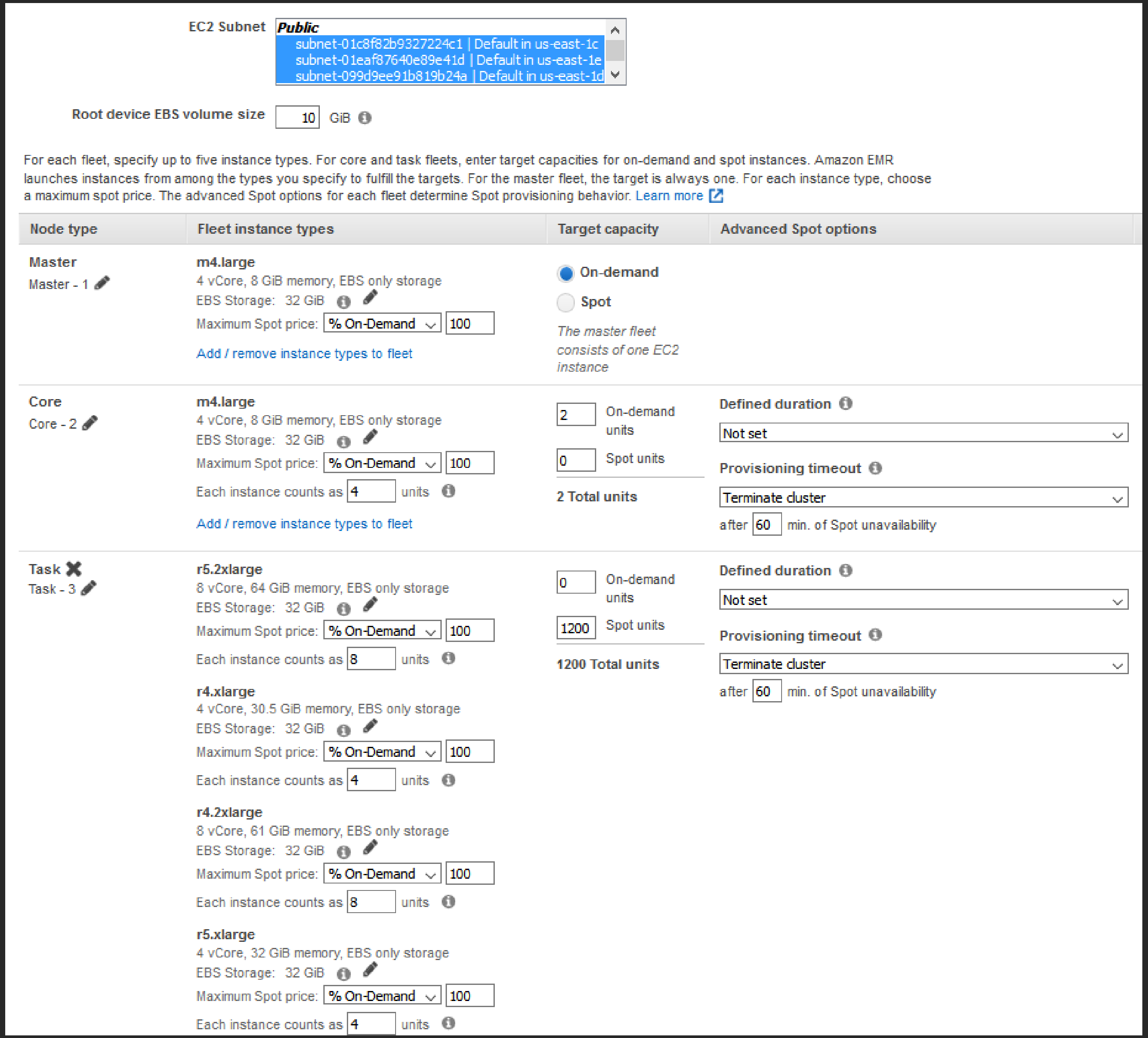
Instances Distribution: ☐ Use the default settings to get started quickly.

On-Demand Allocation Strategy: Prioritized

Maximum Spot Price: [✓ Use default (recommended)]
Default uses current Spot price, but caps it at the On-Demand price.
○ Set your maximum price (per instance/hour)

Amazon EC2 Auto Scaling groups

Amazon EMR instance fleets



EC2 Subnet: *Public*
subnet-01c8f82b9327224c1 | Default in us-east-1c
subnet-01eaf87640e89e41d | Default in us-east-1e
subnet-099d9ee91b819b24a | Default in us-east-1d

Root device EBS volume size: 10 GiB

For each fleet, specify up to five instance types. For core and task fleets, enter target capacities for on-demand and spot instances. Amazon EMR launches instances from among the types you specify to fulfill the targets. For the master fleet, the target is always one. For each instance type, choose a maximum spot price. The advanced Spot options for each fleet determine Spot provisioning behavior. Learn more

Node type | **Fleet instance types** | **Target capacity** | **Advanced Spot options**

Master
Master - 1

m4.large
4 vCore, 8 GiB memory, EBS only storage
EBS Storage: 32 GiB
Maximum Spot price: % On-Demand 100
Add / remove instance types to fleet

[✓ On-demand] ○ Spot
The master fleet consists of one EC2 instance

Core
Core - 2

m4.large
4 vCore, 8 GiB memory, EBS only storage
EBS Storage: 32 GiB
Maximum Spot price: % On-Demand 100
Each instance counts as 4 units
Add / remove instance types to fleet

2 On-demand units
0 Spot units
2 Total units

Defined duration: Not set
Provisioning timeout: Terminate cluster
after 60 min. of Spot unavailability

Task
Task - 3

r5.2xlarge
8 vCore, 64 GiB memory, EBS only storage
EBS Storage: 32 GiB
Maximum Spot price: % On-Demand 100
Each instance counts as 8 units

r4.xlarge
4 vCore, 30.5 GiB memory, EBS only storage
EBS Storage: 32 GiB
Maximum Spot price: % On-Demand 100
Each instance counts as 4 units

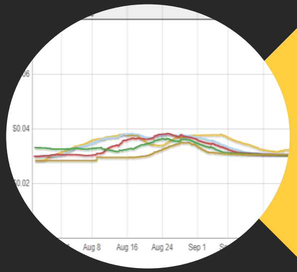
r4.2xlarge
8 vCore, 61 GiB memory, EBS only storage
EBS Storage: 32 GiB
Maximum Spot price: % On-Demand 100
Each instance counts as 8 units

r5.xlarge
4 vCore, 32 GiB memory, EBS only storage
EBS Storage: 32 GiB
Maximum Spot price: % On-Demand 100
Each instance counts as 4 units

0 On-demand units
1200 Spot units
1200 Total units

Defined duration: Not set
Provisioning timeout: Terminate cluster
after 60 min. of Spot unavailability

Spot is easy



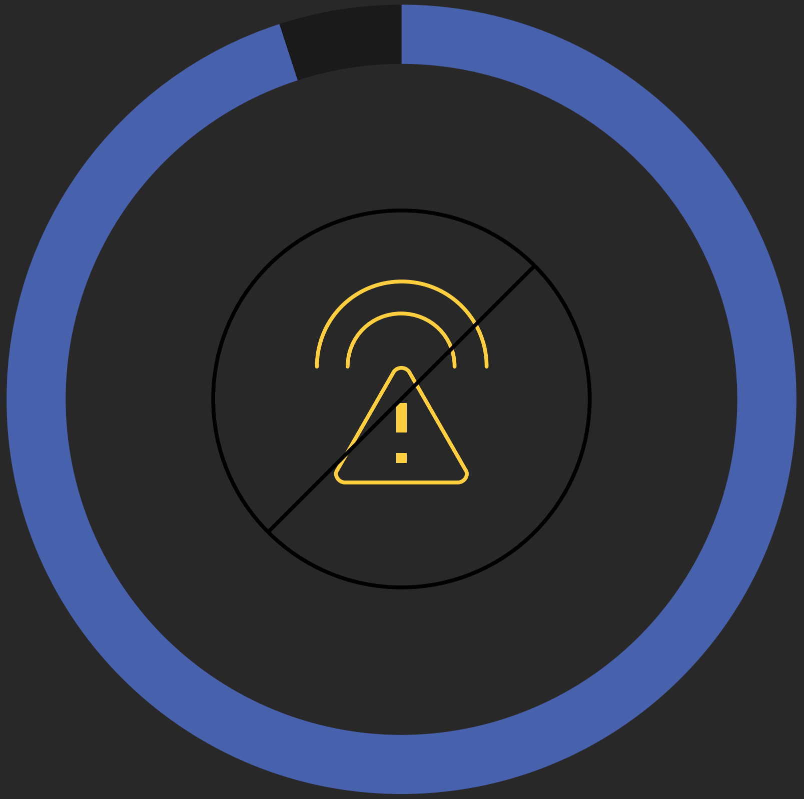
Price changes infrequently based on *long-term* supply and demand of spare capacity in each pool independently

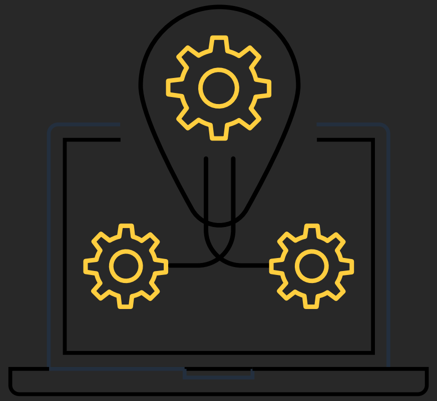
Just request capacity and pay the current rate. No bidding.

Interruptions only happen when OD needs capacity. No outbidding.

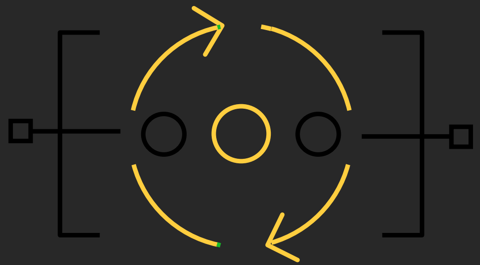
What about interruptions?

Minimal interruptions





The work you are doing to make your applications fault-tolerant also benefits Spot



Spot is optimized for stateless, fault-tolerant, or flexible workloads

Any application that can have part or all of the work paused and resumed or restarted can use Spot

Check for two-minute instance termination notice via instance metadata or Amazon CloudWatch Events and automate by:

- ✓ Checkpointing
- ✓ Draining from ELB
- ✓ Using stop-start and hibernate to restart faster

Containers + Spot = match made in heaven

- ✓ Containers are often stateless, fault-tolerant, and a great fit for Spot Instances
- ✓ Deploy containerized workloads and easily manage clusters at any scale at a fraction of the cost with Spot Instances
- ✓ Spot Instances can be used with ECS or Kubernetes to run any containerized workload

skyscanner

Skyscanner is a travel fare aggregator website and travel metasearch engine based in Edinburgh, Scotland

"We are currently tracking **74% savings** over all regions."

—Paul Gillespie,
Senior Principal Engineer

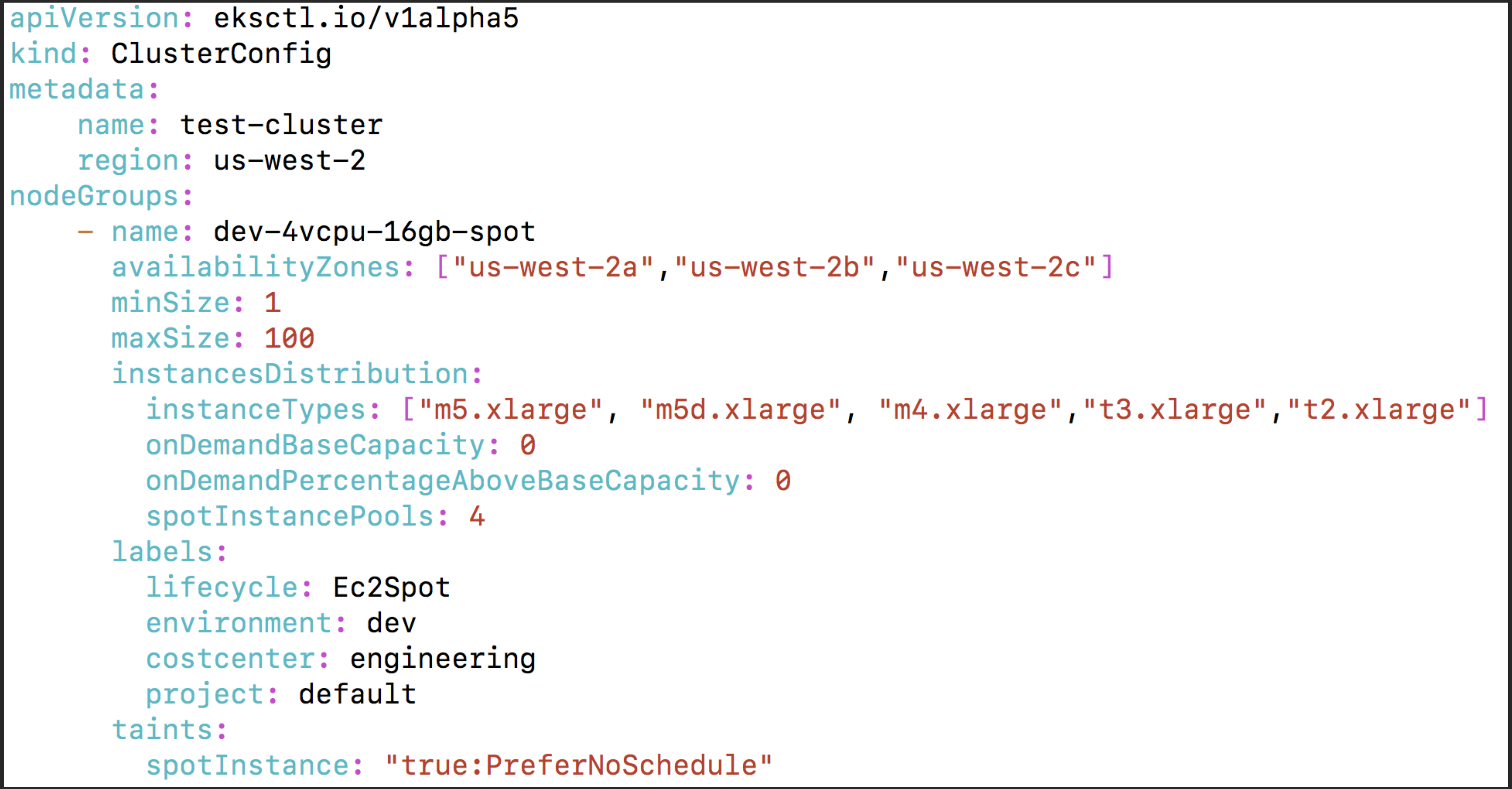
lyft yelp MapBox Expedia Caltech

EKS & EC2 Spot Instances

1. Acquiring capacity
2. Handling interruptions: DaemonSets
3. Scaling mechanisms
4. Taints, tolerations, and affinity
5. Tools

Demo

eksctl—adding a diversified Spot nodegroup



```yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
    name: test-cluster
    region: us-west-2
nodeGroups:
    - name: dev-4vcpu-16gb-spot
      availabilityZones: ["us-west-2a","us-west-2b","us-west-2c"]
      minSize: 1
      maxSize: 100
      instancesDistribution:
        instanceTypes: ["m5.xlarge", "m5d.xlarge", "m4.xlarge","t3.xlarge","t2.xlarge"]
        onDemandBaseCapacity: 0
        onDemandPercentageAboveBaseCapacity: 0
        spotInstancePools: 4
      labels:
        lifecycle: Ec2Spot
        environment: dev
        costcenter: engineering
        project: default
      taints:
        spotInstance: "true:PreferNoSchedule"
```

EKS & EC2 Spot instances

1. Acquiring capacity
2. Handling interruptions, DaemonSets
3. Scaling mechanisms
4. Taints, tolerations, and affinity
5. Tools

Handling interruptions & DaemonSets

```
NOTICE_URL=${NOTICE_URL:-http://169.254.169.254/latest/meta-data/spot/termination-time}

echo "Polling ${NOTICE_URL} every ${POLL_INTERVAL} second(s)"

# To whom it may concern: http://superuser.com/questions/590099/can-i-make-curl-fail-with-an-exitcode-different-
while http_status=$(curl -o /dev/null -w '%{http_code}' -sL "${NOTICE_URL}"); [ "${http_status}" -ne 200 ]; do
  verbose && echo "$(date): ${http_status}"
  sleep "${POLL_INTERVAL}"
done
```

```
GRACE_PERIOD=${GRACE_PERIOD:-120}
kubectl drain "${NODE_NAME}" --force --ignore-daemonsets --delete-local-data --grace-period="${GRACE_PERIOD}"
```

```
      nodeSelector:
        lifecycle: Ec2Spot
```

https://github.com/kube-aws/kube-spot-termination-notice-handler

EKS & EC2 Spot Instances

1. Acquiring capacity
2. Handling interruptions, DaemonSets
3. Scaling mechanisms
4. Taints, tolerations, and affinity
5. Tools

Auto Scaling the app & cluster

- HPA (horizontal pod autoscaler)
  - ➢ Auto scales the number of pods in a Deployment/ReplicaSet

```
kubectl autoscale deployment hello-k8s --cpu-percent=$HPA_MIN_CPU --min=$HPA_MIN_PODS --max=$HPA_MAX_PODS
```

- CA (cluster-autoscaler)
  - ➢ Auto scales the number of worker nodes in the cluster when:
    - Pods cannot be scheduled due to lack of resources (pending state)
    - Nodes are underutilized and important pods can be rescheduled elsewhere

Auto Scaling the app & cluster

```yaml
- image: k8s.gcr.io/cluster-autoscaler:v1.13.6
  name: cluster-autoscaler
  resources:
    limits:
      cpu: 100m
      memory: 300Mi
    requests:
      cpu: 100m
      memory: 300Mi
  command:
    - ./cluster-autoscaler
    - --v=4
    - --stderrthreshold=info
    - --cloud-provider=aws
    - --skip-nodes-with-local-storage=false
    - --nodes=1:100:eksctl-test-cluster-nodegroup-dev-8vcpu-32gb-spot-NodeGroup-1BIF74YAF5BMQ
    - --nodes=1:100:eksctl-test-cluster-nodegroup-dev-4vcpu-16gb-spot-NodeGroup-C0Y9VUB6VDK1
    - --expander=random
    - --balance-similar-node-groups
  env:
    - name: AWS_REGION
      value: us-west-2
```

- CA Nodegroups are still expected to be homogeneous. Implement diversification!

EKS & EC2 Spot Instances

1. Acquiring capacity
2. Handling interruptions, DaemonSets
3. Scaling mechanisms
4. Taints, tolerations, and affinity
5. Tools

Taints, toleration, and affinity

```
requiredDuringSchedulingIgnoredDuringExecution:
  nodeSelectorTerms:
  - matchExpressions:
    - key: environment
      operator: In
      values:
      - dev
```

Multi-tenant cluster
Group affinity

```
affinity:
  nodeAffinity:
    preferredDuringSchedulingIgnoredDuringExecution:
    - weight: 1
      preference:
        matchExpressions:
        - key: lifecycle
          operator: In
          values:
          - OnDemand
```

Lifecycle affinity & toleration

```
tolerations:
- key: "spotInstance"
  operator: "Equal"
  value: "true"
  effect: "PreferNoSchedule"
```

EKS & EC2 Spot Instances

1. Acquiring capacity
2. Handling interruptions, DaemonSets
3. Scaling mechanisms
4. Taints, tolerations, and affinity
5. Tools

Tools

- eksctl: https://github.com/weaveworks/eksctl
- Amazon CloudWatch Container Insights
- Descheduler: https://github.com/kubernetes-incubator/descheduler
- K8s-node-drainer: https://github.com/aws-samples/amazon-k8s-node-drainer
- Overprovisioner: https://github.com/helm/charts/tree/master/stable/cluster-overprovisioner

Main takeaways

- Understand Spot best practices: Pricing model, termination, Instance diversification, Spot Instance advisor, launch template, ASGs
- Apply Spot best practices to Kubernetes: Instance diversification, HPA, CA
- Apply instance termination DaemonSet, taints, tolerations, affinities
- Know your tools: cluster auto scaler, eksctl
- Amazon EKS/Amazon ECS roadmap: https://github.com/aws/containers-roadmap/projects/1

The definitive guide to running EC2 Spot Instances as Kubernetes worker nodes: http://bit.ly/DefintiveSpotK8sGuide

Amazon EKS workshop: https://eksworkshop.com/

Amazon EC2 Spot Instances Workshops: https://ec2spotworkshops.com

Thank you!

Please complete the session survey in the mobile app.