Accelerating Containerized Workloads With EC2 Spot

Madhuri Peri, Sr. Specialist Solution Architect, AWS

June 2018

Agenda

Recap of EC2 Spot

- What is EC2 Spot? & Why use Spot?
- Best Practices

Provision EC2 Spot

Handling Interruptions

Container workloads with EC2 Spot

- ECS
- EKS
- Batch

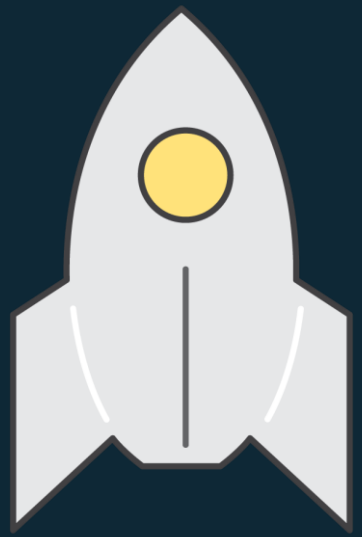
aws

Recap of EC2 Spot Concepts

What is EC2 Spot & Why Spot

Spare EC2 Capacity that AWS can reclaim with 2-minutes notice

Savings up to 90% off the On Demand Price

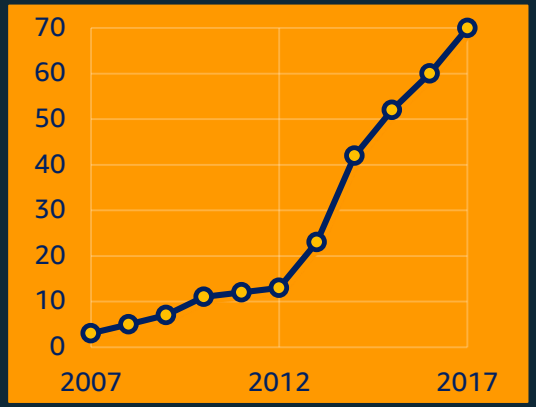


Turbo Boost your Results

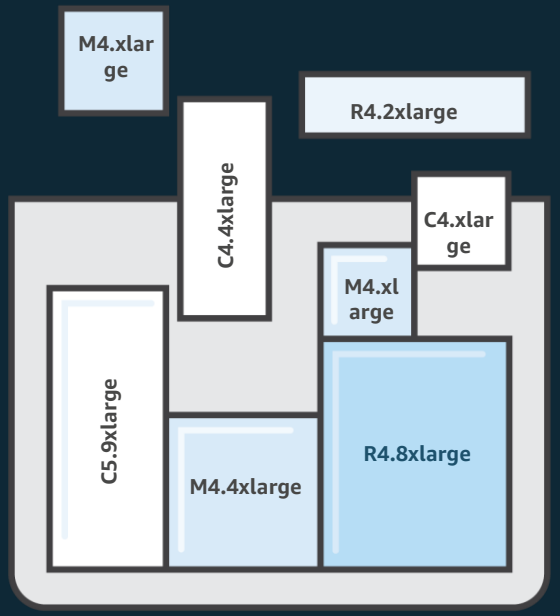
aws

Understanding EC2 Capacity & Spot Pools

Each instance family, each instance size, in each Availability Zone, in every Region is a ***separate*** Spot pool



100s of Instance Options



Instance Flexibility and Diversification Made Easy with Spot Fleet

aws

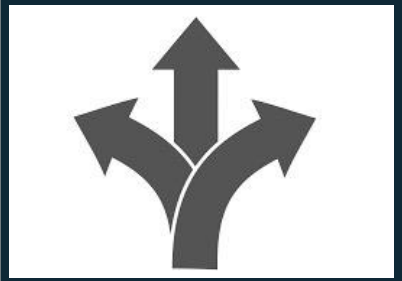
Best Practices for Running Apps on Spot Instances

Save more on distributed and flexible applications

Stateless

Fault-Tolerant

Can you Restart ?



Flexible

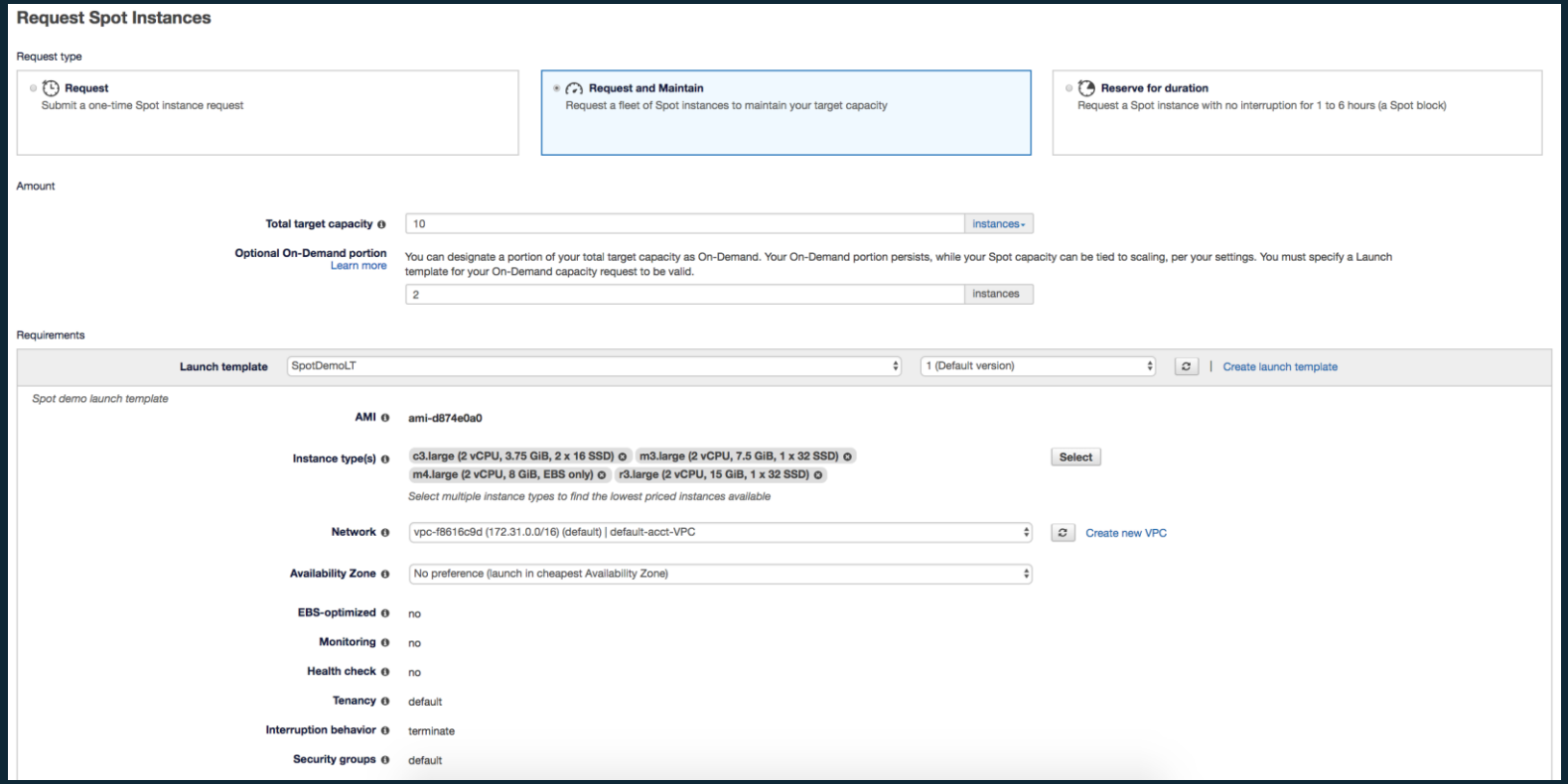
Multi-AZ and Instance Flexibility

Loosely Coupled

Distributed? Do you Checkpoint?

aws

Spot Provisioning - EC2/Spot Fleet



Spot Provisioning - RunInstances

```
acbc32918437:~ mperi$ aws ec2 run-instances --image-id ami-f2d3638a --count 1 --instance-type c3.large --key-name spotinst -
--security-groups default --instance-market-options file:///Users/mperi/Desktop/spotconfig.json --region us-west-2
```

```
{
  "MarketType": "spot",
  "SpotOptions": {
    "MaxPrice": "1.0",
    "SpotInstanceType": "persistent",
    "InstanceInterruptionBehavior": "stop"
  }
}
```


aws

Spot Interruptions

EC2 Spot is interrupted when

- On-Demand requires capacity back
- EC2 Spot price exceeds specified *Max Price* in the request

Notified of interruptions in 2 ways

- CloudWatch events
- Instance metadata

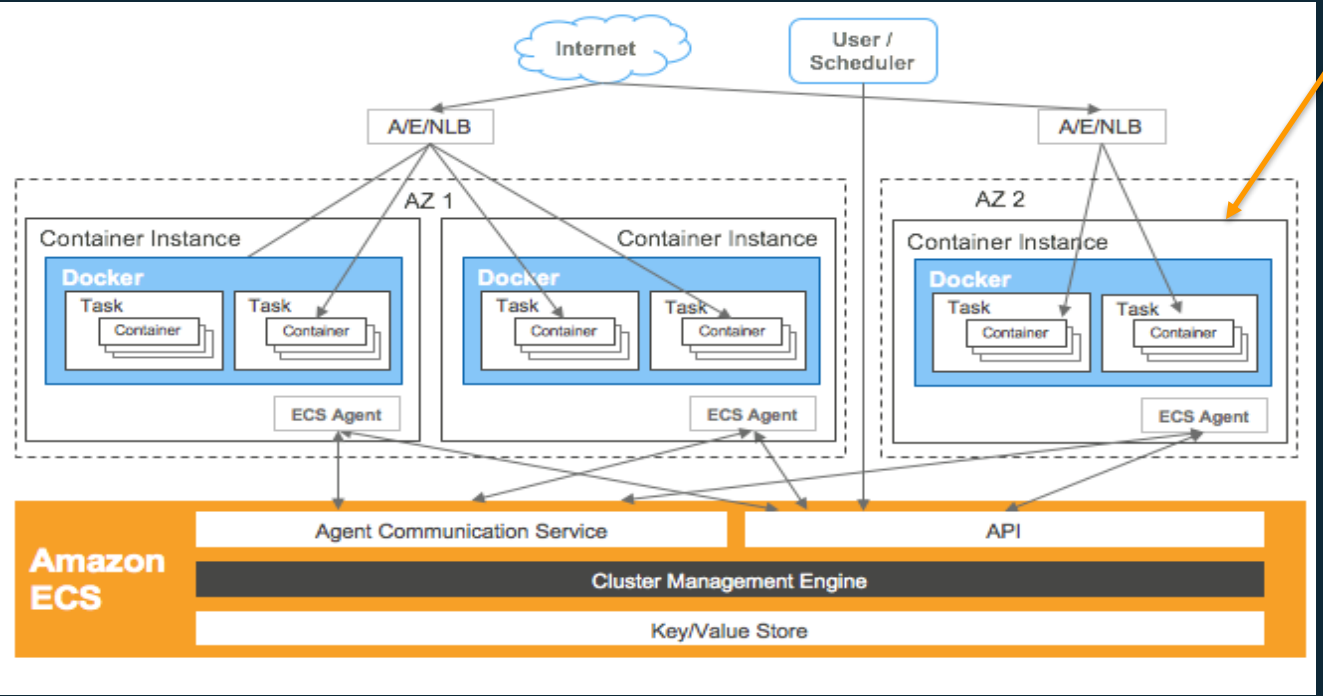
```
[ec2-user@ip-192-168-133-1 ~]$ curl http://169.254.169.254/latest/meta-data/instance-action/
none[ec2-user@ip-192-168-133-1 ~]$
```


aws

ECS with EC2 Spot

ECS concept (recap)

Worker nodes on EC2 Spot or On-Demand



ECS with EC2 Spot - Provisioning

- ECS is integrated via Console with Spot Fleet
- Spot Fleet launches instances
  - Each EC2 Instance registers with ECS Cluster

```
#!/bin/bash
# Set any ECS agent configuration options
echo "ECS_CLUSTER=default" >> /etc/ecs/ecs.config
```


aws

ECS with EC2 Spot - Scaling

- 2 dimensions to scaling
  - EC2 Instance and Container
- EC2 Instance via Automatic Scaling
  - 2 scaling policies
    - Target tracking scaling
    - Step scaling

aws

ECS with EC2 Spot – Scaling (cont…)

- Target tracking scaling
  - Select a metric, set target value
  - Spot Fleet creates & manages CW alarms that trigger scaling policy
  - Adds/removes capacity to keep metric at/close to target value specified.

aws

ECS with EC2 Spot - Scaling (cont…)

- Step scaling
  - You specify the CloudWatch alarms or trigger the scaling policies

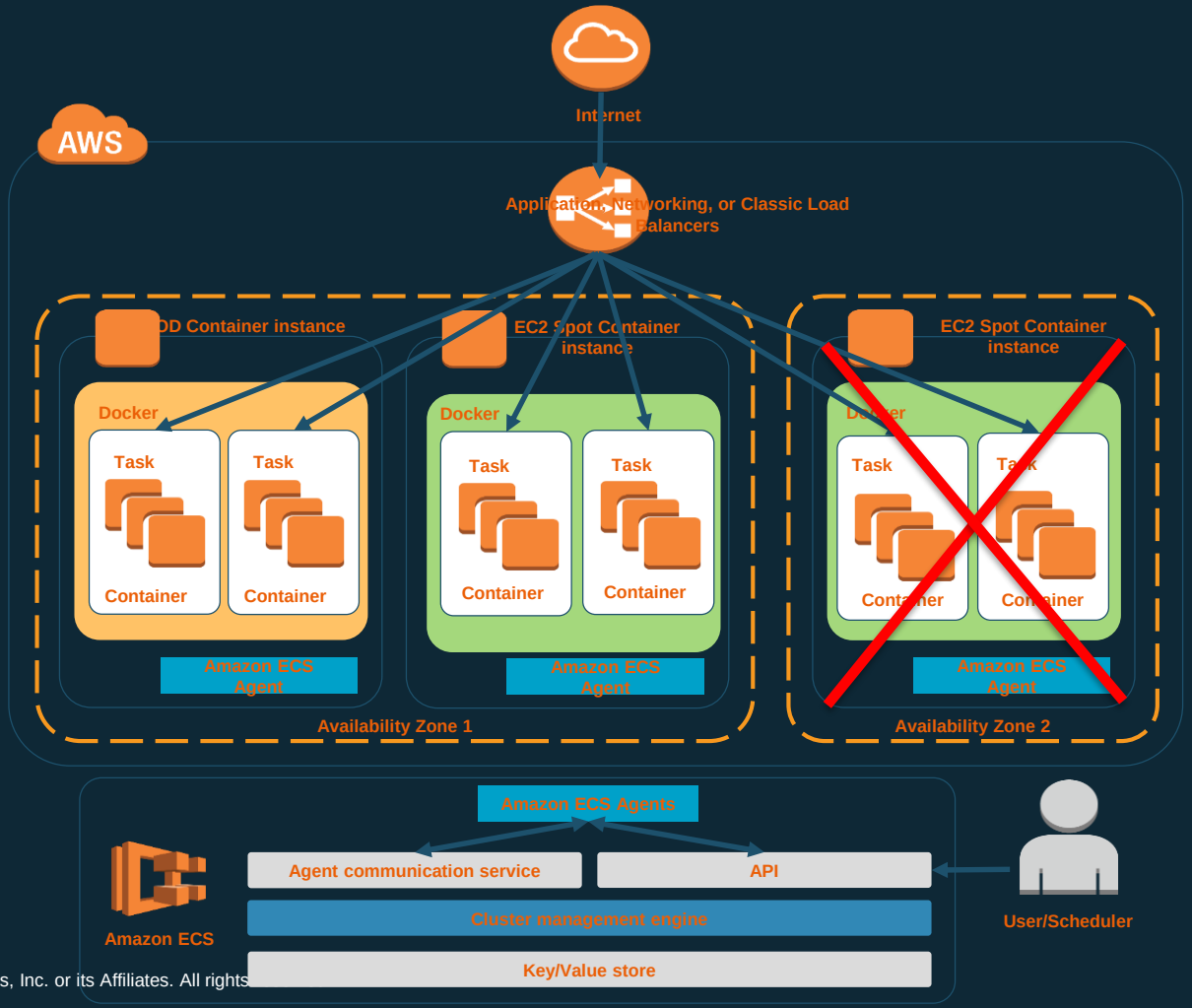
ECS with EC2 Spot – Termination/Interruption

- ECS Service availability is maintained
- Need interruption to be managed
  - Stop sending new traffic to terminating instance
  - Active connections are drained
  - Place a new copy on another running EC2 instance

Termination/Interruption (cont…)

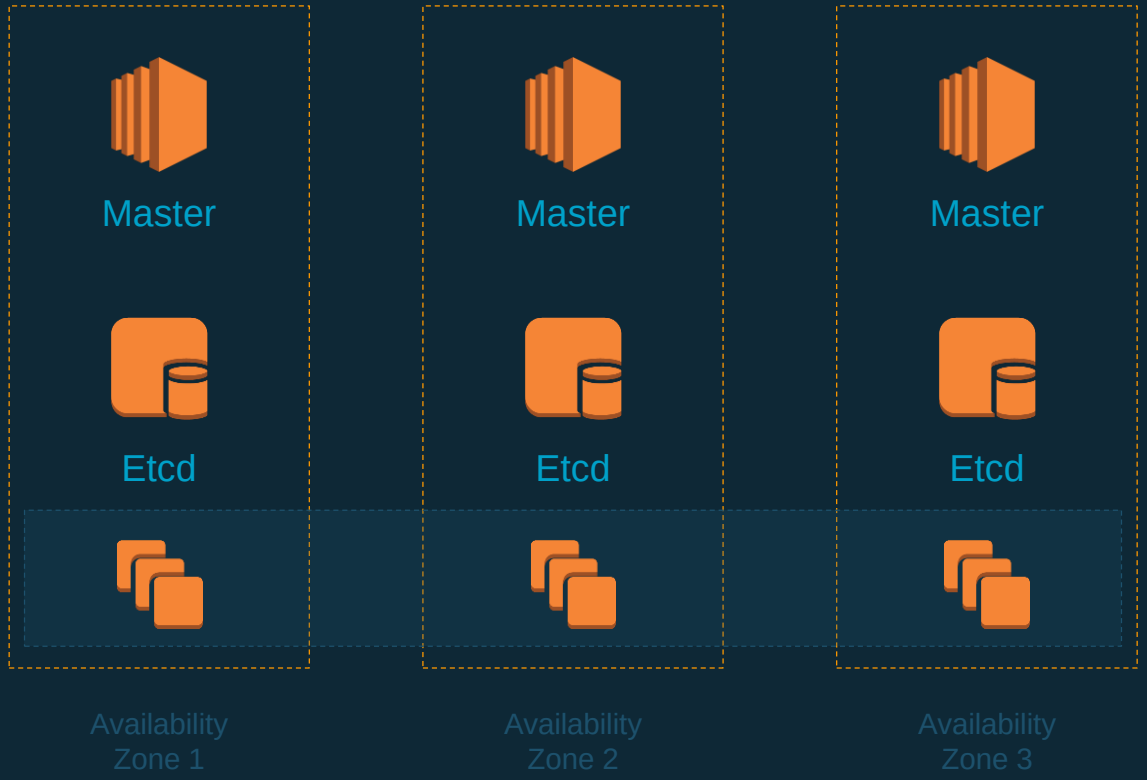
- Spot Fleet will make new requests to fulfill target capacity
- ECS
  - Move EC2 instance to DRAINING state
  - This prevents any new tasks to be scheduled on the host.
- Service scheduler will place tasks on other hosts in cluster

aws



EKS with EC2 Spot

EKS – Master Nodes (Recap)



EKS – Logical layout of Master Endpoint and Worker Nodes

EKS with EC2 Spot - Provisioning

- Worker node user data needs to connect to the Master node endpoint
  - If you use Cloudformation EC2::Instance, then EC2 Spot instance is requested via RunInstances API
  - If you use LaunchConfig, then EC2 Spot instance is requested via ASG.
    - ASG makes the Spot Instance Request (*sir-**) on your behalf
- K8's optionally uses *label* to manage pod placement
  - --node-labels *lifecycle=Ec2Spot*

aws

EKS Spot provisioning (cont…)

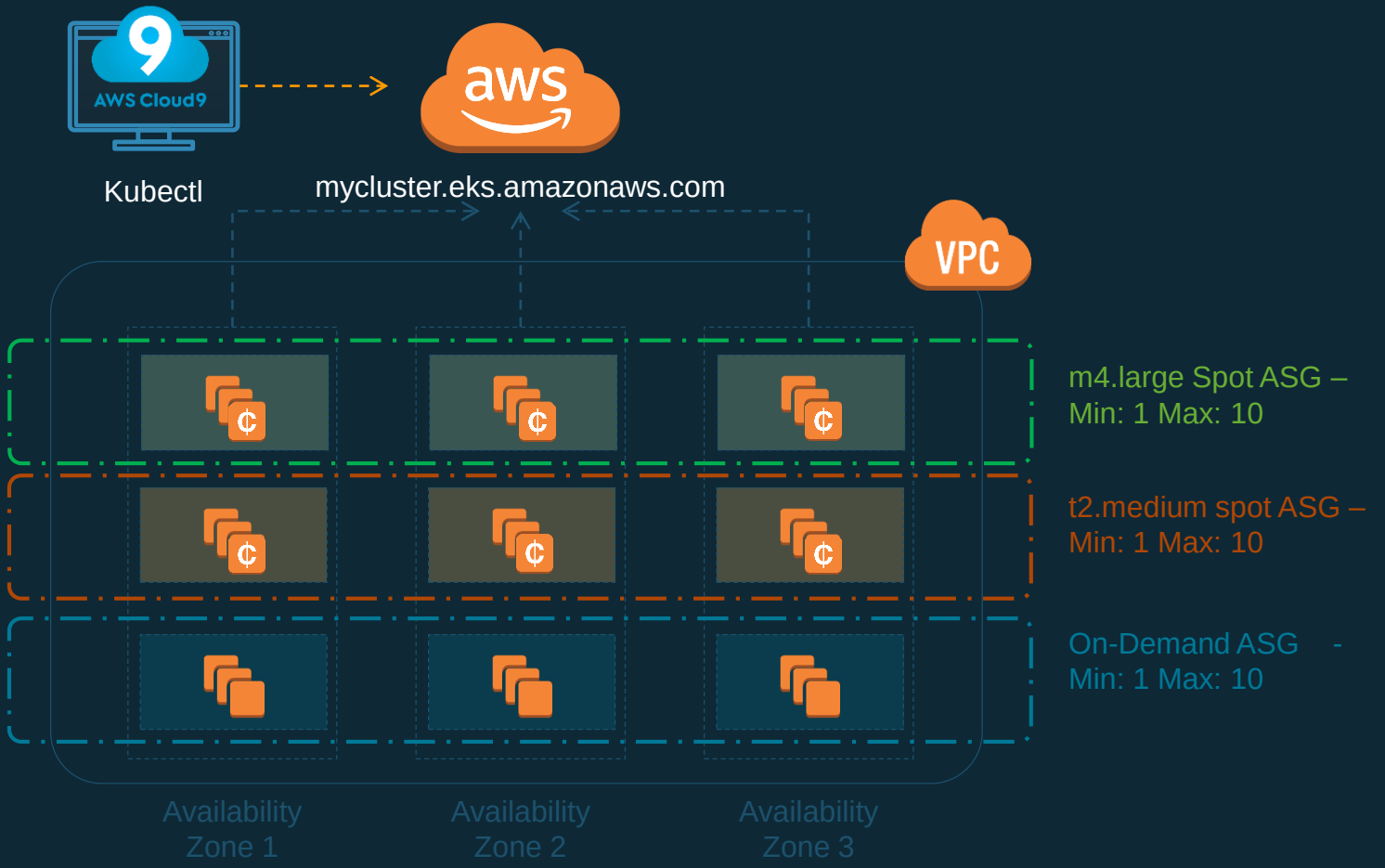
- BUT Wait……..
- Remember EC2 Spot best practice?...
    - Use Instance flexibility!!

Spot provisioning (cont…)

- Put each of the instance type in a separate ASG with EC2 Spot.
  - Gives you multiple instance types – best practice.
  - ASG makes additional requests to replace interrupted instances

EKS – Scaling

- Horizontal *pod* scaling
  - Automatically scales number of pods in replication controller
- Cluster scaling (Scaling EC2 Instances)
  - Adjusts the number of nodes in the cluster when
    - Pods failed to run in cluster due to insufficient resources
    - Nodes are underutilized, for an extended period of time
    - And pods could be placed elsewhere on other nodes



EKS – EC2 Spot scaling

```
spec:
  serviceAccountName: cluster-autoscaler
  containers:
    - image: gcr.io/google_containers/cluster-autoscaler:v1.2.2
      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
        - --expander=least-waste
        - --nodes=1:10:eks-DemoSpotClusterScale-workers-NodeGroup-1T05UYPCM91ZI
        - --nodes=1:10:eks-DemoSpotClusterScale-workers-OnDemandNodeGroup-1N7U1LX3MAXDK
        - --skip-nodes-with-system-pods=false
      env:
        - name: AWS_REGION
          value: us-west-2
```


aws

EKS with EC2 Spot – Termination/Interruptions

- Catch interruption on EC2 Instance
- Have a pod running in daemon mode on each Spot Node
    - Pod will intercept the interruption notice
    - Set K8 Node to *drain* state
        - Marks Node unschedulable
        - Graceful termination
        - Vacates pods via eviction API

aws

AWS Batch with EC2 Spot

Batch with EC2 Spot

- With Batch you have 3 main aspects:
  - Jobs/Job definitions
  - Job queues
  - Compute environments

Batch with EC2 Spot

- Managed compute
- Unmanaged compute

Batch with EC2 Spot - Managed Compute

- Managed Compute
  - AWS Batch manages configuring & scaling of instances
  - Under the hood, uses ECS cluster with EC2 instances
  - You define the compute provisioning model
    - On-Demand or EC2 Spot

aws

Batch Managed Compute with EC2 Spot

- Uses Spot Fleet
- You define the instance types or leave it as optimal
- If Optimal,
    - Batch chooses best fit based on vCPU, mem requirements
    - Match job with most-appropriate instance
    - For example –
        - 2 vCPU, and 16GB for each job, and 300 jobs
        - Picks biggest instance type that can fit most # of jobs

Batch Managed Compute with EC2 Spot

- If single instance type, then doesn’t align with Spot Fleet best practices
- Override optimal setting*
    - Assign multiple Managed CE to same job queue with order
    - Up to 5 can be assigned

aws

Batch Un-Managed Compute with EC2 Spot

- You configure & manage scaling of the ECS Cluster
- ECS integrated with Spot Fleet directly

aws

Batch with EC2 Spot – Manage interruptions

- Managed compute – optimal setting
  - If interruption was because OD needed capacity, no progress
- Managed compute – multiple CE
  - Diversified instance types likely to have capacity, jobs progress
- Unmanaged compute – Spot Fleet
  - Diversified instance types likely to have capacity, jobs progress

aws

Thank you!