# Spot ←→ Ondemand Failover Strategies V0.3

Oct 2020

Jayaprakash Alawala (JP)
EC2 Spot Specialist SA

# AGENDA

1. Important Note
2. Workload based on ONLY ASG
3. Workload based on EKS

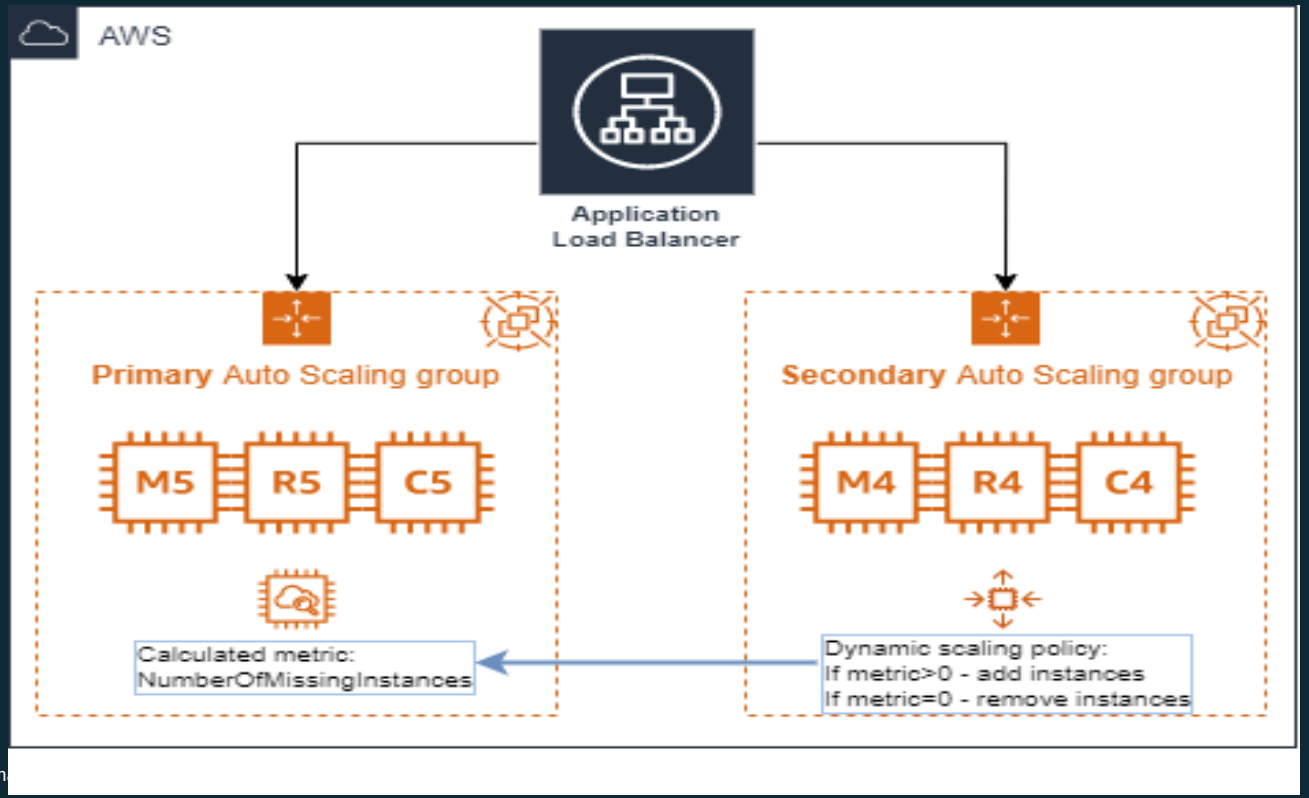
# Important Note

- Please note it is always recommended to
  - use all of  EC2 spot best practices
  - Run Stateless and Fault tolerant workloads on EC2 Spot
  - Ideal way is to design  architectures considering potential delays for Replacement spot to be available if there are capacity constraints so you don’t need failover strategies.
  - But if  workload is very cost conscious or needs failover strategy for some reason,  pls keep below considerations while adopting these approaches.
- Failover strategies are created ONLY based on many customer requests
- These are just few guidelines to implement your own strategy for as per your requirement
- These are NOT tested / proved for production. So request to please test it enough and own the solution before deploying on  production

# Workloadd based on ONLY ASGs

# ASG CWT Metrics Strategy

https://medium.com/@ranshn/using-preferred-instance-types-availability-zones-or-purchase-options-with-ec2-auto-scaling-groups-1a5997f3cb0d

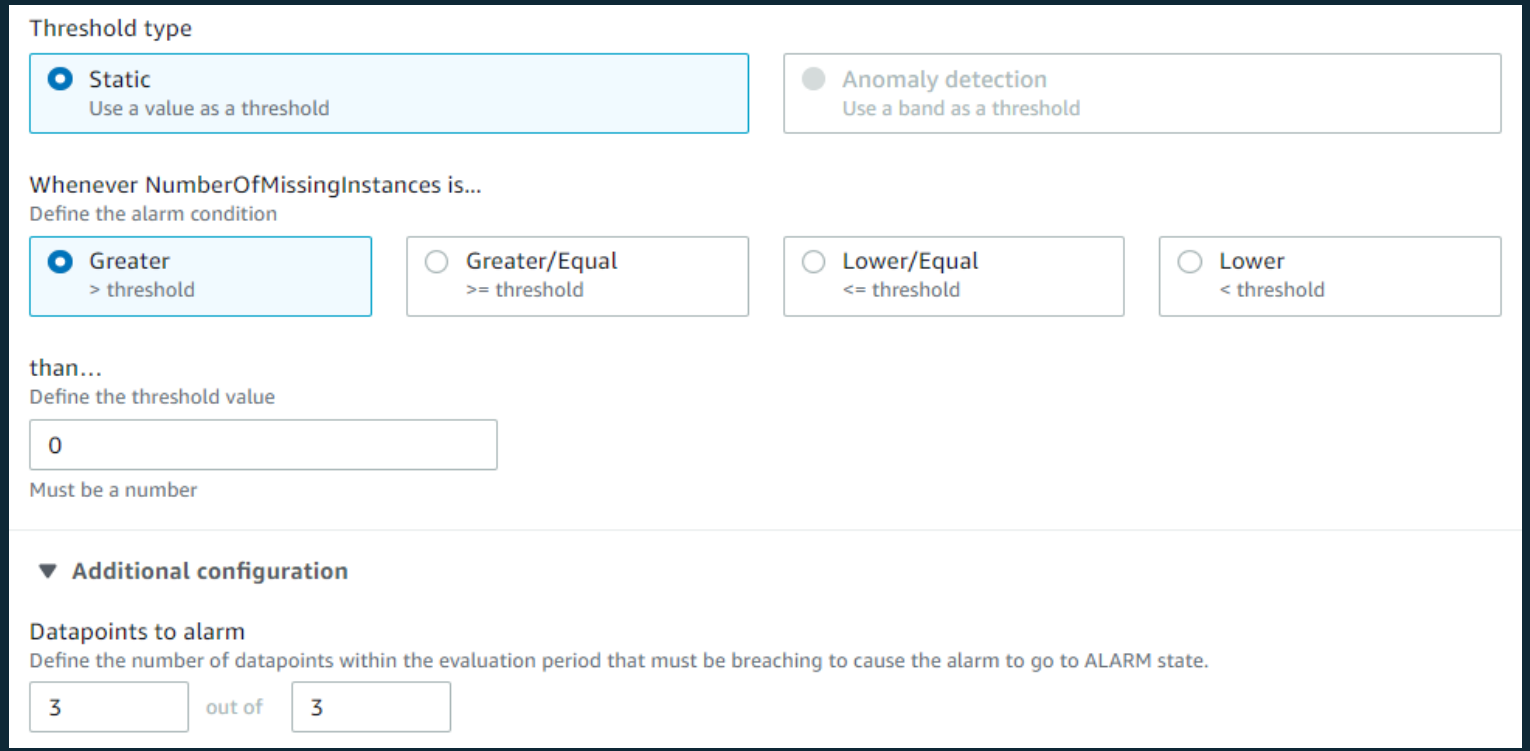


# Cloud Watch Metrics Math

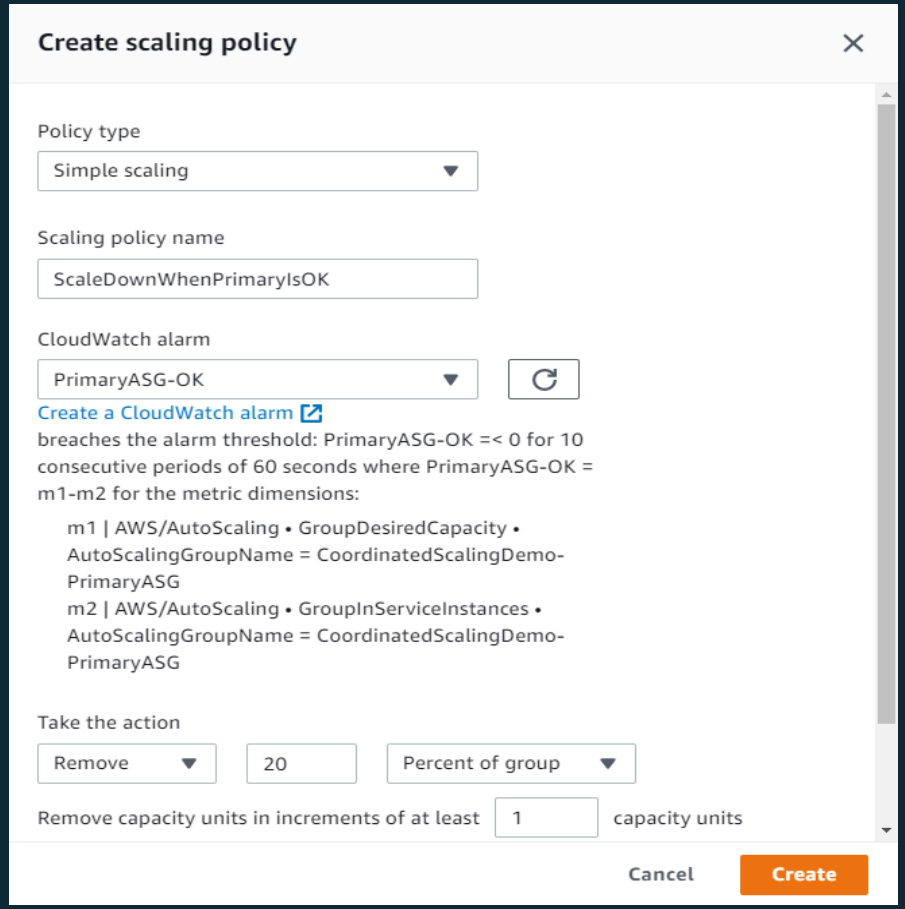
Math expression | Dynamic labels | Statistic: Maximum | Period: 1 Minute | Remove all

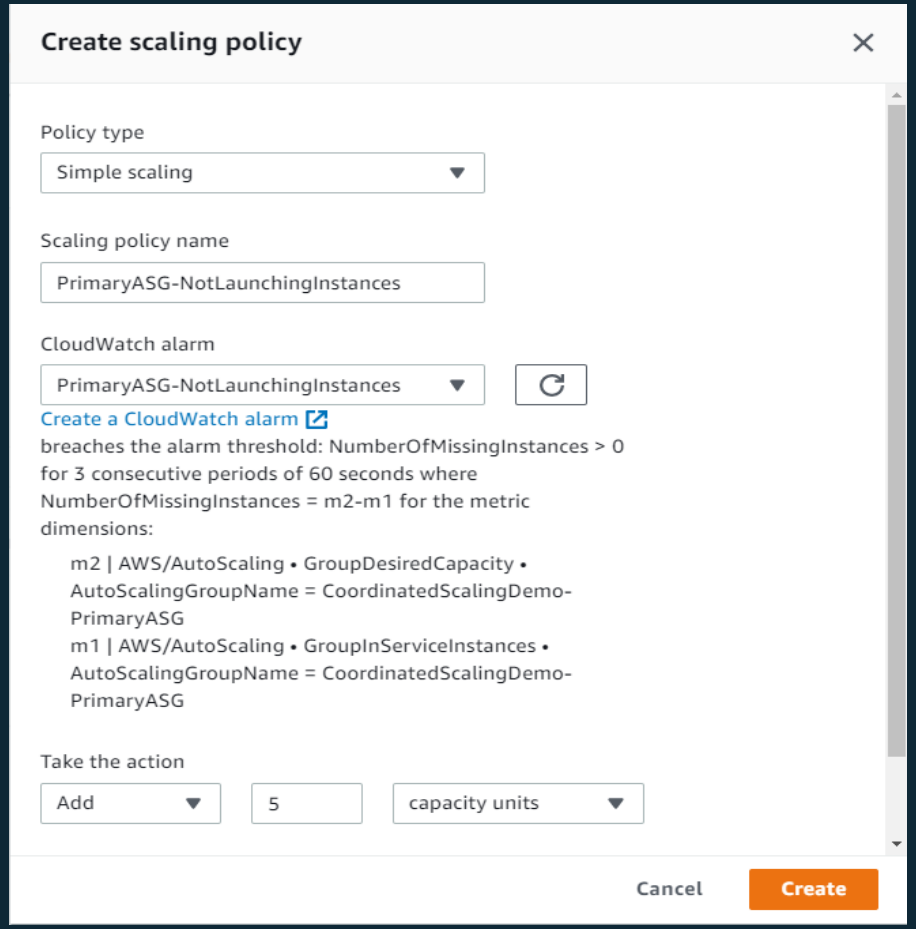
| | Id | Label | Details | Statistic | Period | Y Axis | Actions |
|---|---|---|---|---|---|---|---|
| ☑ | e1 | NumberOfMissingInstances | m2-m1 | | | | |
| ☐ | m1 | GroupInServiceInstances | Auto Scaling • GroupInServiceInstances • AutoScalingGroupName: Coordinated... | Maximum | 1 Minute | | |
| ☐ | m2 | GroupDesiredCapacity | Auto Scaling • GroupDesiredCapacity • AutoScalingGroupName: CoordinatedSc... | Maximum | 1 Minute | | |

# Cloud Watch Alarms

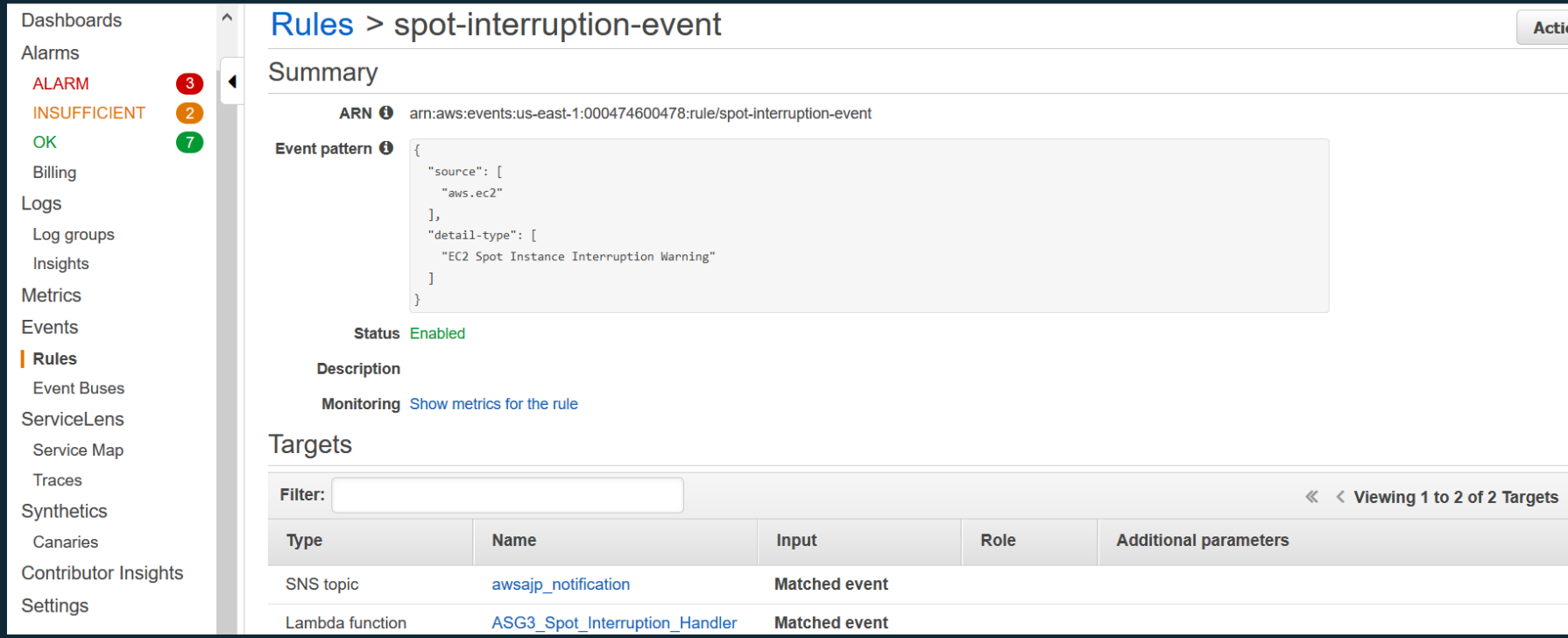


# OD ASG Scale Out/In Policies

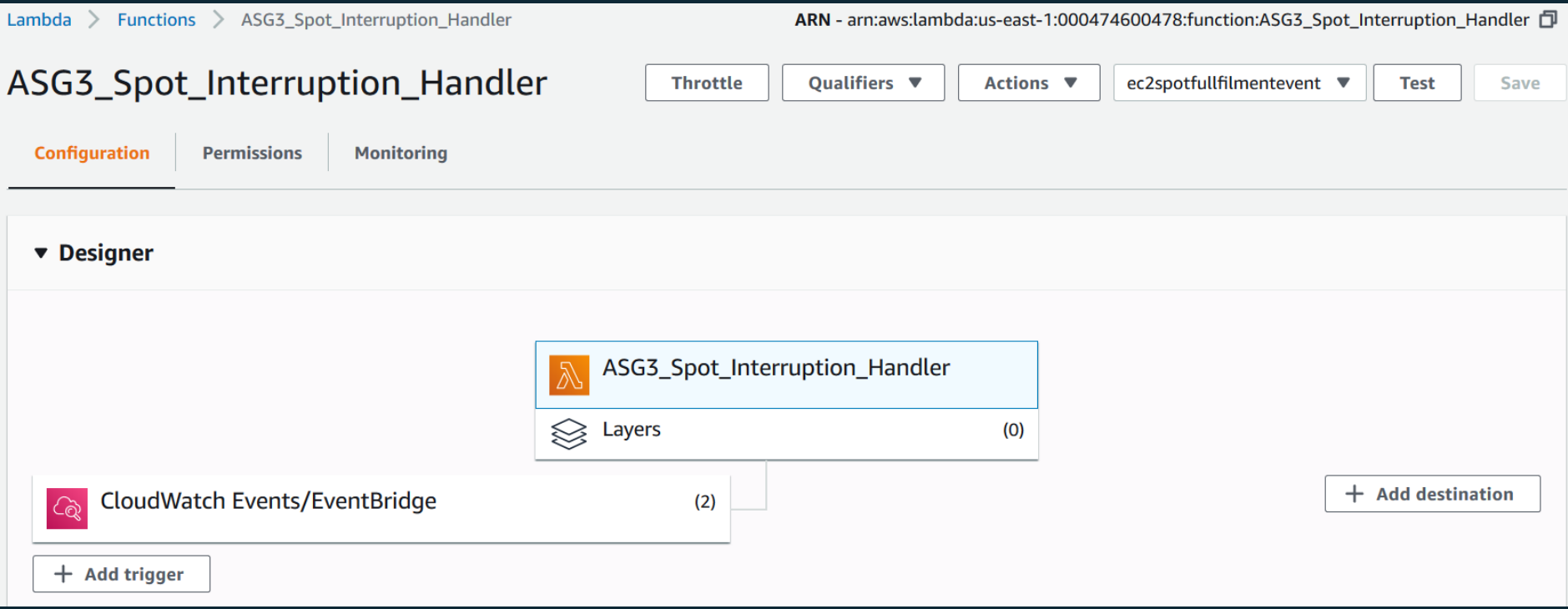


# Spot Interruption Handler



# Lambda Function

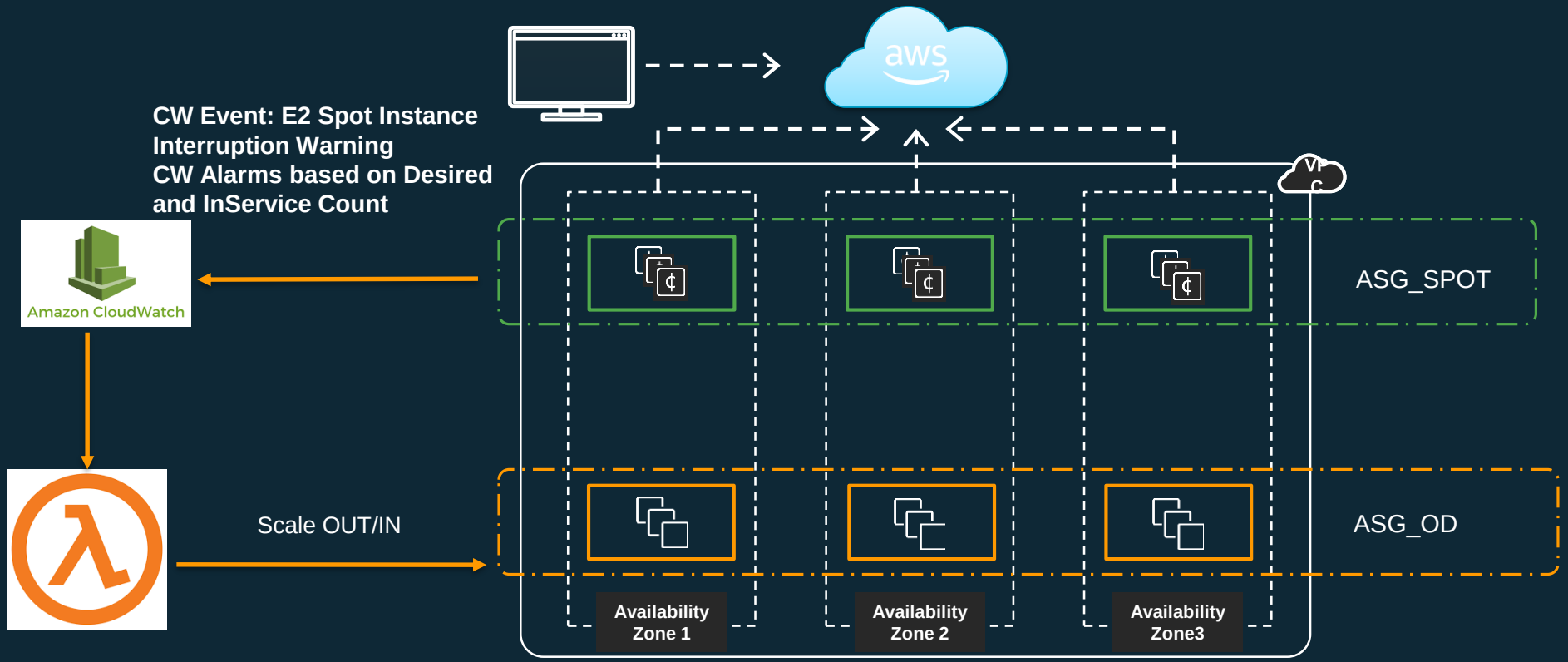


# Detach Instance from ASG

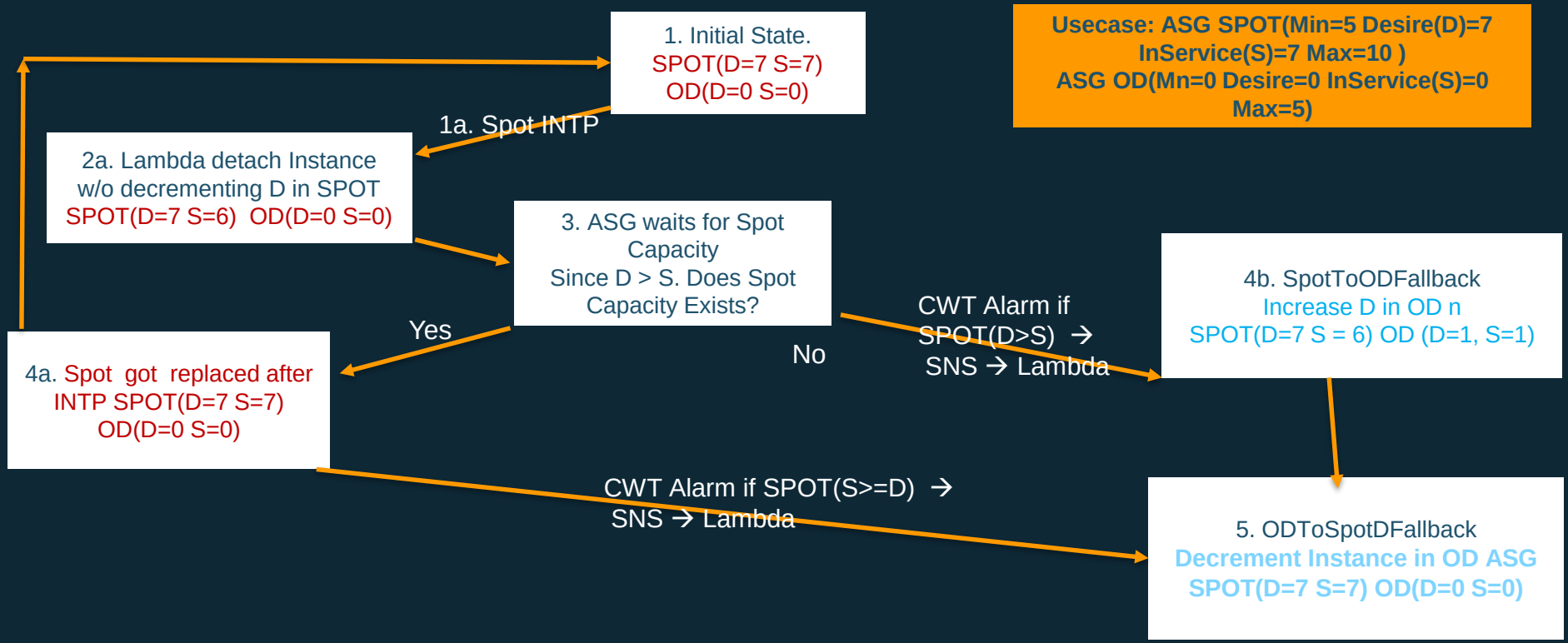
https://aws.amazon.com/blogs/compute/running-web-applications-on-amazon-ec2-spot-instances/

```
def detach_instance_from_asg(instance_id,as_group_name):
    try:
        # detach instance from ASG and launch replacement instance
        response = asgclient.detach_instances(
            InstanceIds=[instance_id],
            AutoScalingGroupName=as_group_name,
            ShouldDecrementDesiredCapacity=False)
        logger.info(response['Activities'][0]['Cause'])
    except ClientError as e:
        error_message = "Unable to detach instance {id} from AutoS
            id=instance_id,asg_name=as_group_name)
        logger.error( error_message + e.response['Error']['Message
        raise e
```

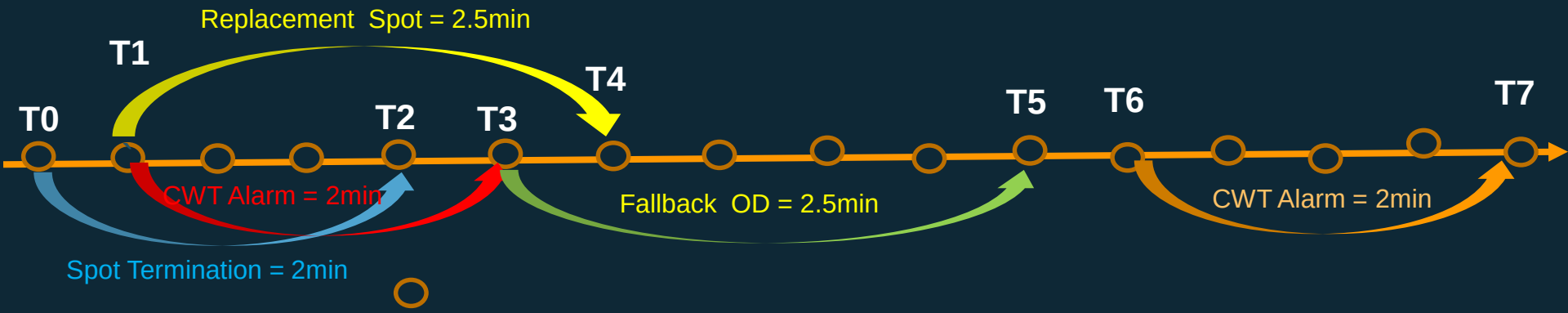
# Spot ←→ OD Failover Strategy



# ASG Spot ←→ OD Fallback Strategy



# ASG Spot ←→ OD Failover Timeline




T0 - Spot INTP occurs
T0 - CWT Event → lambda
T0 - ASG detaches the Instance
T1 = T0+30se (appx) = ASG tries to bring replacement instance
T2 = T0+2min = Spot is Terminated
T3 = T1+2min = CWT Alarm triggers for SpotToOD fallback → Launches OD Instance
T4 = T1+2.5min(approx.) = Replacement Spot if ASG find capacity at T1
T5 = T3+2.5min(approx.) = Fallback OD launched at T3
T5 = T3+2.5min(approx.) = Fallback OD launched at T3
T6 = T1 + 5min = Replacement came back
T7 = T6 + 1 min = CWT alarm for ODtoSpot Fallback → Terminating OD Instance



aws

# Workloads based on EKS

# K8S CA ASGs/NGs Strategy

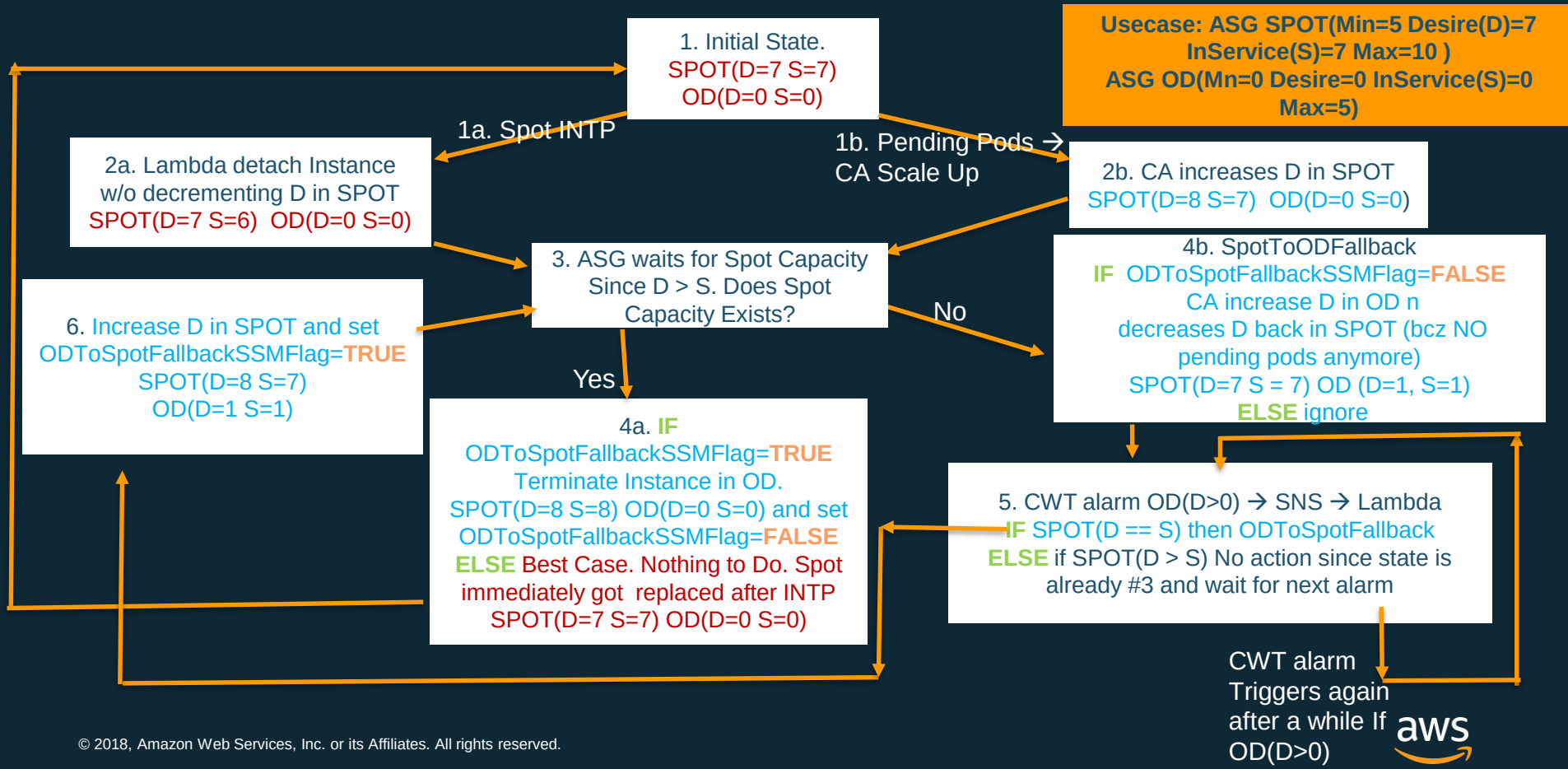
| | Option #1: Separate ASG for Spot and OD | Option #2: Single ASG for Spot and OD | Comments |
|---|---|---|---|
| Labels/Taints to Spot/OD Instances | Different since each NG can be different settings | Same since they inherit from same ASG | Use Bootstrapping.sh to make it diff in Option 2 |
| Scaling Speed | Low | High | Num of NGs affect CA scan simulations |
| Cost Optimization | Higher with high priority for spot and Spot to OD fallback leveraged | High | |
| Spot to OD fallback | Easy with priority expander. Ensure OD also scales. Fine tune *--max-node-provision-time* | N/A. Very difficult to automate. Not recommended | |
| Split Instances across Spot / OD | Not Simple. Tweak to Desire/min/max in ASGs | Easy. Use OD base and Percent options | |
| Pods Scheduling using labels/taints | Works well if labels/taints applied accordingly | Works well if labels/taints applied accordingly | No difference |
| Recommended Expander Strategy | Priority if both spot to OD fallback and scaling on OD is needed else Random bcz there is only spot | Random | |
| Usecase: Base on OD and Scale on Spot.<br>Desire 10, Min 2, Max 20 | OD_NG1: Desire=2, Min=2, Max=2 OR<br>OD_NG2: Desire=1, Min=1, Max=1<br>SPOT_NG1: Desire=8, Min=0, Max=18<br>SPOT_NG2: Desire=4, Min=0, Max=9<br><br>CA scales only SPOT_NG1 and SPOT_NG2 | SPOT_OD_NG1: Desire=10, Min=2, Max=20<br>OD_base=2, Percent=0%<br>SPOT_OD_NG2: Desire=0, Min=0, Max=9<br>OD_base=0, Percent=0%<br><br>CA scales on SPOT_OD_NG1 and SPOT_OD_NG2 | NG1 with m5.large<br>NG2 with m5.2xlarge<br><br>OD_NG* can be managed NG |

# ASG/NG Config

- on-demand NG with lower priority
- Spot NG with higher priority
- s*--max-node-provision-time* (default 15 min)
- Stops considering NG in simulations → attempt to scale up a different NG if pods still pending
- Attempt to remove any nodes left unregistered after this time

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-autoscaler-priority-expander
data:
  priorities: |-
    10:
      - .*high-cost-guaranteed-instances.*
      - .*medium-cost-instances.*
    50:
      - .*low-cost-spot-instances.*
```

# EKS CA Spot ←→ OD Fallback Strategy



# Usecase and ASG / CA Config

- Usecase Description:
  - 100% scale on Spot. Use OD only for fallback until spot is available to maximize cost savings
  - Min number of Instance in Cluster running all the time is 5
  - Max number of Instance in Cluster is 10.  Current / Desired Instances  say 7
- ASG / NG Configuration
  - ASG SPOT Configuration
    - Min=5 Desire(D)=7 InService(S)=7 Max=10
  - ASG OD
    - Min=0 → bcz OD is meant ONLY for fallback and no need of any minimum OD instances.
    - Desire=0 → Assume  we have required number of Spot instances. No need of any OD for initially.
    - InService(S)=0  is same as Desired
    - Max=5 → i.e. to implement fallback ONLY for min of SPOT ASG i.e. 5. Ideally OD Max  should be same as SPOT Max to implement fallback for ALL Spot Instances not just min spot instances
- CA Config
  - Priority expander strategy.  SPOT ASG is higher priority than OD ASG
  - Set *--max-node-provision-time* to lesser value say 3min or 5min (default is 15 min)

# Sequence of Events Description - 1

1. This is initial state of the ASG
2. There can be two options
   - 2a. Spot Interruption occurs
     - Lambda is triggered upon on spot interruption notification
     - Lambda calls detach API of ASG for this spot instance
       - Instance is taken out of ASG and kept in the draining mode
       - ASG will try to get the replacement spot immediately
   - 2b. CA scaling up occurs bcz there are some pending pods in the cluster
     - CA will increment the desired capacity of Spot ASG by 1
   - In both 2a and 2b, the desired capacity is more than number of instances in Service.
3. ASG waits for spot capacity to launch a new instance

# Sequence of Events Description - 2

4. There can be two options
   - 4a. spot capacity is available.
     - There can be two options
       - ODToSpotFallbackSSMFlag is TRUE
         - In this case, decrement Instance in OD ASG and set the flag to FALSE
       - ODToSpotFallbackSSMFlag is FALSE
         - That means it is a normal spot interruption
   - 4b . Spot is not available
     - CA will fallback to OD ASG
5. CWT alarm triggers if there are instances running in the Ondemand ASG
   - There are two options
   - If desired and InService capacity is same in Spot ASG, then implement
   - Else if desiredCount > InService instances, then SPOT ASG is already waiting for spot, so no need to do anything
6. ODToSpotFallback
   - Increase the desired capacity in Spot ASG

Thank you

https://aws.amazon.com/ec2/spot