Spot Instance Interruption Handling on EKS

Spot Instance Interruption can be handled in two ways,
1)Node termination Handler with ASG.
2)Node termination Handler with Karpenter.

Since we are using the Karpenter as the cluster autoscaler, we are using the NTH with the Karpenter setup.

Step-1:
Installing Karpenter:
For the Installation of Karpenter, you can refer to the following documentation,
https://docs.google.com/document/d/1nRwpUmlomZZ9WjT-EpWNfRCrGJdtH_0iJvBiGD1UG3I/edit?usp=sharing

Step-2:
Setup the Node Termination Handler:

NTH runs on 2 different modes,
i.e IMDS Processor Mode, and Queue Processor Mode

For using the node termination handler with Karpenter, it is preferable to use Queue Processor mode.

For Queue mode, we need some more infrastructure setup.

You'll need the following AWS infrastructure components:

1. Amazon Simple Queue Service (SQS) Queue
2. AutoScaling Group Termination Lifecycle Hook
3. Amazon EventBridge Rule
4. IAM Role for the aws-node-termination-handler
5. Queue Processing Pods

Here, NTH works on ASG, for balancing the nodes but Karpenter doesn't follow ASG so we need to skip some steps.

->Creating the SQS queue,

```
$ QUEUE_POLICY=$(cat <<EOF
{
    "Version": "2012-10-17",
    "Id": "MyQueuePolicy",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": ["events.amazonaws.com", "sqs.amazonaws.com"]
        },
        "Action": "sqs:SendMessage",
        "Resource": [
            "arn:aws:sqs:${AWS_REGION}:${ACCOUNT_ID}:${SQS_QUEUE_NAME}"
        ]
    }]
}
EOF
)

## make sure the queue policy is valid JSON
$ echo "$QUEUE_POLICY" | jq .

## Save queue attributes to a temp file
$ cat << EOF > /tmp/queue-attributes.json
{
  "MessageRetentionPeriod": "300",
  "Policy": "$(echo $QUEUE_POLICY | sed 's/\"/\\"/g' | tr -d -s '\n' " ")"
}
EOF

$ aws sqs create-queue --queue-name "${SQS_QUEUE_NAME}" --attributes
file:///tmp/queue-attributes.json
```

The above commands create the SQS queue with policy.

Since we are not using ASG, we can skip step 2.

->Creating Amazon EventBridge Rules

We can skip this step if we are using ASG, but in our case, we are not using ASG, so for lifecycle notification, we need to configure the EventBridge

```
$ aws events put-rule \
  --name MyK8sSpotTermRule \
  --event-pattern "{\"source\": [\"aws.ec2\"],\"detail-type\": [\"EC2
Spot Instance Interruption Warning\"]}"

$ aws events put-targets --rule MyK8sSpotTermRule \
  --targets
"Id"="1","Arn"="arn:aws:sqs:us-east-1:123456789012:MyK8sTermQueue"


$ aws events put-rule \
  --name MyK8sRebalanceRule \
  --event-pattern "{\"source\": [\"aws.ec2\"],\"detail-type\": [\"EC2
Instance Rebalance Recommendation\"]}"

$ aws events put-targets --rule MyK8sRebalanceRule \
  --targets
"Id"="1","Arn"="arn:aws:sqs:us-east-1:123456789012:MyK8sTermQueue"


$ aws events put-rule \
  --name MyK8sInstanceStateChangeRule \
  --event-pattern "{\"source\": [\"aws.ec2\"],\"detail-type\": [\"EC2
Instance State-change Notification\"]}"

$ aws events put-targets --rule MyK8sInstanceStateChangeRule \
  --targets
"Id"="1","Arn"="arn:aws:sqs:us-east-1:123456789012:MyK8sTermQueue"
```

```
$ aws events put-rule \
  --name MyK8sScheduledChangeRule \
  --event-pattern "{\"source\": [\"aws.health\"],\"detail-type\":
[\"AWS Health Event\"],\"detail\": {\"service\":
[\"EC2\"],\"eventTypeCategory\": [\"scheduledChange\"]}}"

$ aws events put-targets --rule MyK8sScheduledChangeRule \
  --targets
"Id"="1","Arn"="arn:aws:sqs:us-east-1:123456789012:MyK8sTermQueue"
```

Update the ARN with your SQS arn.

->Attach the IAM Policy for the node termination handler:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:CompleteLifecycleAction",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeTags",
                "ec2:DescribeInstances",
                "sqs:DeleteMessage",
                "sqs:ReceiveMessage"
            ],
            "Resource": "*"
        }
    ]
}
```

Create the policy and attach the policy to the eks-cluster-node role.

->Add the below policy to the access policy in Amazon SQS.

```
{
  "Version": "2012-10-17",
  "Id": "Policy1670568194281",
  "Statement": [
    {
      "Sid": "Stmt1670568193321",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "sqs:*",
      "Resource":
"arn:aws:sqs:ap-south-1:467564224058:NTH-SQS-Template-Queue-9SEw
4rPocHVO"
    }
  ]
}
```

Update the ARN of the SQS

->Installing the node termination handler:

Add the helm repo

helm repo add eks https://aws.github.io/eks-charts

Installing helm chart:

```
helm upgrade --install aws-node-termination-handler \
  --namespace kube-system \
  --set
queueURL=https://sqs.ap-south-1.amazonaws.com/467564224058/NTH-SQS-Template-Queue-9
SEw4rPocHVO \
  --set awsRegion=ap-south-1 \
  --set checkASGTagBeforeDraining=false \
  --set enableSpotInterruptionDraining=true \
  eks/aws-node-termination-handler
```

Or configure the changes and apply the values.yaml file

-> Configuring the karpenter to provision the spot instances.
On the capacity type mention [“spot”,”on-demand”] , so the first preference would be the spot instance if spot instance is not available then on-demand would be provisioned.

Behaviour of NTH:

Before spot instance goes down, our event bridge sends the spot interruption message through SQS to NTH. We will receive the interruption notification 2 minutes before the instance goes down.

When the spot interruption notice is received if will cordon the node and drain the node, but if no capacity is available to schedule the pods, the pods remain unscheduled and downtime may occur.

To maintain the downtime we need to add the PodDisruptionBudget.

PodDisruptionBudget:

It will help in maintaining the minimum availability of the deployment. If Our nodes goes down and pod remains unscheduled it will make sure that each deployment maintain the minimum availability.

This will help us in holding the NTH while draining the nodes. Till the time our new node becomes available when the node drains.

Testing the Spot Interruption Handling

We can use the AWS FIS to test the spot interruption handling,
FIS(Fault Injection Simulator) used to simulate the spot interruption to check what happens when the spot instance goes down.