How can I set up Cluster Autoscaler on Amazon EKS?

Last updated: 2020-08-13

I want to set up Cluster Autoscaler on Amazon Elastic Kubernetes Service (Amazon EKS).

Short description

The Kubernetes Cluster Autoscaler automatically adjusts the size of a Kubernetes cluster when one of the following conditions is true:

- There are pods that fail to run in the cluster due to insufficient resources.
- There are nodes in the cluster that are underutilized for an extended period of time and their pods can be placed on other existing nodes.

The Cluster Autoscaler scales worker nodes within any specified Auto Scaling group and runs as a deployment in your cluster.

Note: The following resolution assumes that you have an active Amazon EKS cluster with associated worker nodes created by an AWS CloudFormation template. The resolution uses the auto-discovery setup, but you can also configure Cluster Autoscaler by specifying one or multiple Auto Scaling groups.

Resolution

Set up Auto-Discovery

1. Open the AWS CloudFormation console, select your stack, and then choose the **Resources** tab.

2. To find the Auto Scaling group resource created by your stack, find the **NodeGroup** in the **Logical ID** column. For more information, see Launching self-managed Amazon Linux 2 nodes.

3. Open the Amazon Elastic Compute Cloud (Amazon EC2) console, and then choose **Auto Scaling Groups** from the navigation pane.

4. Choose the **Tags** tab, and then choose **Add/Edit tags**.

```
Key: k8s.io/cluster-autoscaler/enabled
Key: k8s.io/cluster-autoscaler/awsExampleClusterName
```

Note: The keys for the tags that you entered don't have values. Cluster Autoscaler ignores any value set for the keys.

Create an IAM policy

1. Create an AWS Identity and Access Management (IAM) policy called **ClusterAutoScaler** based on the following example. This gives the worker node running the Cluster Autoscaler access to required resources and actions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeLaunchConfigurations",
                "autoscaling:DescribeTags",
                "autoscaling:SetDesiredCapacity",
                "autoscaling:TerminateInstanceInAutoScalingGroup",
                "ec2:DescribeLaunchTemplateVersions"
            ],
            "Resource": "*"
        }
    ]
}
```

Note: By adding the preceding policy to the worker nodes role, you enable all pods or applications running on the respective EC2 instances to use the additional IAM permissions.

2. Attach the new policy to the instance role that's attached to your Amazon EKS worker nodes.

Deploy the Cluster Autoscaler

1. To download a deployment example file provided by the Cluster Autoscaler project on GitHub, run the following command:

2. Open the downloaded YAML file, and set the cluster name (**awsExampleClusterName**) based on the following example. Then, save your changes.

```
...
          command:
            - ./cluster-autoscaler
            - --v=4
            - --stderrthreshold=info
            - --cloud-provider=aws
            - --skip-nodes-with-local-storage=false
            - --expander=least-waste
            - --node-group-auto-discovery=asg:tag=k8s.io/cluster-autoscaler/enabled,k8s.i
...
```

3. To create a Cluster Autoscaler deployment, run the following command:

```
kubectl apply -f cluster-autoscaler-autodiscover.yaml
```

4. To check the Cluster Autoscaler deployment logs for deployment errors, run the following command:

```
kubectl logs -f deployment/cluster-autoscaler -n kube-system
```

Test the scale out of the worker nodes

1. To see the current number of worker nodes, run the following command:

```
kubectl get nodes
```

2. To increase the number of worker nodes, run the following commands:

```
kubectl create deployment autoscaler-demo --image=nginx
kubectl scale deployment autoscaler-demo --replicas=50
```

3. To check the status of your deployment and see the number of pods increasing, run the following command:

```
kubectl get deployment autoscaler-demo --watch
```

4. When the number of available pods equals 50, check the number of worker nodes by running the following command:

```
kubectl get nodes
```

Clean up the test deployment

1. To scale down the worker nodes by deleting the deployment **autoscaler-demo** that was created before, run the following command:

```
kubectl delete deployment autoscaler-demo
```

2. To see the number of worker nodes, wait about 10 minutes, and then run the following command:

```
kubectl get nodes
```

What Is Cloud Computing?
What Is DevOps?
What Is a Container?
What Is a Data Lake?
AWS Cloud Security
What's New
Blogs
Press Releases

AWS Solutions Portfolio
Architecture Center
Product and Technical FAQs
Analyst Reports
AWS Partner Network

.NET on AWS
Python on AWS
Java on AWS
PHP on AWS
Javascript on AWS

Help

Contact Us
AWS Careers
File a Support Ticket
Knowledge Center
AWS Support Overview
Legal

Sign In to the Console

Language

عربي |
Bahasa Indonesia |
Deutsch |
English |
Español |
Français |
Italiano |
Português |
Tiếng Việt |
Türkçe |