**EKS Cluster**

For deploy our cluster using “managed” node group,

AWS will create an auto-scaling group and manage it as a part of an EKS cluster deployment.

Also need to define the*maximum, minimum, and desired capacity*for the auto-scaling group (ASG).

**Create EC2 Instance:**

1. Login into AWS account
2. Go to Services “EC2” and Click on Launch Instance.
3. Choose AMI and Instance Type.
4. Provide IAM role to give Permission to EC2 instance
5. Create New KeyPair and Download PPK file.
6. Click on Launch Instance

**Install kubectl:**

Command line tool that lets you communicate and control Kubernetes clusters for creating, managing, or deleting resources on Kubernetes platform

$ curl -LO <https://dl.k8s.io/release/v1.24.0/bin/linux/amd64/kubectl>

$ sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

**Install eksctl**

CLI tool for creating and managing clusters on EKS - Amazon's managed Kubernetes service for EC2. It is written in Go, uses CloudFormation.

$ curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp

$ sudo mv /tmp/eksctl /usr/local/bin

$ eksctl version

**Install AWS CLI**

To take control on AWS services from the command line its required to install awscli.To install the AWS CLI by following these commands:

$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

$ unzip awscliv2.zip

$ sudo ./aws/zip

$ aws –version

$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

$ unzip awscliv2.zip

$ sudo ./aws/install --update

**Create Cluster**

[Amazon Elastic Kubernetes Service](http://aws.amazon.com/eks/) (Amazon **EKS**) is a fully managed by AWS that simplifies the process of creating and maintaining Kubernetes clusters on AWS.

Amazon EKS integrates with core AWS services for monitoring, scaling and load balancing your containerized applications.

* CloudWatch,
* Auto Scaling Groups, and
* IAM to provide a seamless experience

$ git clone https://github.com/swaroopmuni/eksclustertask/tree/main/.github/workflows /eks-clustercreation

$ cd /eks-clustercreation

$ eksctl create cluster -f managed-nodegroup.yaml

managed-nodegroup.yaml

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  | apiVersion: eksctl.io/v1alpha5 |
|  |  | kind: ClusterConfig |
|  |  | metadata: |
|  |  | name: kubehub-cluster-01 |
|  |  | region: ap-northeast-1 |
|  |  | version: "1.24" |
|  |  | managedNodeGroups: |
|  |  | - name: managed-ng-1 |
|  |  | instanceType: t3.small |
|  |  | desiredCapacity: 2 |
|  |  | minSize: 2 |
|  |  | maxSize: 10 |
|  |  | volumeSize: 10 |
|  |  | volumeType: gp2 |

Deploy workloads on EKS Cluster

EKS cluster requires additional resources for running workloads in EKS, more instances will be deployed by ASG for the existing node group.

 kubectl create -f [database.yaml](https://github.com/swaroopmuni/eksclustertask/blob/main/.github/workflows/database.yaml)

database.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: db

spec:

replicas: 4

selector:

matchLabels:

app: mongodb

template:

metadata:

labels:

app: mongodb

spec:

containers:

- name: mongodb-container

image: mongo:latest

resources:

requests:

memory: "1Gi"

cpu: 1

**Check the pods and deployment status:**

# **List the deployment**

$ kubectl get deployment db

  NAME        READY   UP-TO-DATE   AVAILABLE   AGE

  db    2/4     4            2           7m47s

# **List the pods of the deployment**

 $ kubectl get pods

  NAME                         READY   STATUS    RESTARTS   AGE

  db-58d5794487-6lbhw   1/1     Running   0          11s

  db-58d5794487-9wn4r   1/1     Running   0          11s

  db-58d5794487-kmzcr   0/1     Pending   0          10s

  db-58d5794487-tt552   0/1     Pending   0          10s

**# notice the pending state**

#  **Inspect the events of one of the "pending" pod**

$  kubectl describe pod db-58d5794487-6lbhw | grep -A20 Events

   Events:

   Type     Reason            Age                  From               Message

   Warning  FailedScheduling  41s (x6 over 5m49s)  default-scheduler  0/2 nodes are available: 2 Insufficient cpu, 2 Insufficient memory.

Cluster Autoscaler

SG will scale when **CPU**and **Memory**usage goes high. But in our current situation, CPU or Memory usage isn’t really high; what’s stopping the pod to be created is the resource requests exceeding the available resource. ASG does not have information about the allocated resources for the pods. This is where**Cluster Autoscaler**comes into the image.

The **Custer autoscaler** will be deployed as a set of Kubernetes Pods. These Pods must have permission to perform AWS API operations, such as the ability to examine and modify EC2 Auto Scaling Groups. Create an IAM policy that grants the permissions that the**Cluster Autoscaler**requires and attach it to the IAM role.

OIDC :

OIDC allows individuals to use single sign-on (SSO) to access relying party sites using OpenID Providers (OPs), such as an email provider or social network, to authenticate their identities.

$ CLUSTER\_NAME=kubehub-cluster-01

**# Create an OIDC provider**

$ eksctl utils associate-iam-oidc-provider --cluster $CLUSTER\_NAME --approve

aws iam list-open-id-connect-providers | grep $OIDC\_ID

Configure IAM Role and Service Account

Kubernetes service account **provides an identity for processes that run in a pod**. If **your pod needs access to AWS services**, you can map the service account to an AWS IAM to grant that access.

#**Download IAM policy from Link**

$ curl https://github.com/swaroopmuni/eksclustertask/blob/main/.github/workflows/AmazonEKSClusterAutoscalerPolicy.json > AmazonEKSClusterAutoscalerPolicy.json

#**Create IAM policy**

$ aws iam create-policy  \

--policy-name AmazonEKSClusterAutoscalerPolicy \

--policy-document file://AmazonEKSClusterAutoscalerPolicy.json

**#Save the POLICY\_ARN as an environment variable**

$ POLICY\_ARN=$(aws iam list-policies --query 'Policies

 [?PolicyName==`AmazonEKSClusterAutoscalerPolicy`].Arn' --output text)

**#Set Env Variables**

|  |  |  |
| --- | --- | --- |
|  |  | $ CLUSTER\_NAME=kubehub-cluster-01 |
|  |  | $ ROLE\_NAME=AmazonEKSClusterAutoscalerRole |
|  |  | $ SA\_NAME=cluster-autoscaler |
|  |  |  |
|  |  | **#Create a Service Account and IAM Role with POLICY** |
|  |  | $ eksctl create iamserviceaccount \ |
|  |  | --name $SA\_NAME \ |
|  |  | --cluster $CLUSTER\_NAME \ |
|  |  | --attach-policy-arn=$POLICY\_ARN \ |
|  |  | --role-name $ROLE\_NAME \ |
|  |  | --namespace kube-system \ |
|  |  | --override-existing-serviceaccounts \ |
|  |  | --approve |
|  |  |  |
|  |  | **#Save the ROLE\_ARN as an environment variable** |
|  |  | $ export ROLE\_ARN=$(aws iam list-roles --query 'Roles[?RoleName==`AmazonEKSClusterAutoscalerRole`].Arn' --output text) |
|  |  | export CLUSTER\_NAME=kubehub-cluster-01 |
|  |  |  |
|  |  | **# Download the manifest file** |
|  |  | curl https://github.com/swaroopmuni/eksclustertask/blob/main/.github/workflows/cluster-creation.yaml > cluster-creation.yaml |
|  |  |  |
|  |  | **# Replace <ROLE ARN> place holder with "ROLE\_ARN" environment variable** |
|  |  | sed -i "s#<ROLE ARN>#$ROLE\_ARN#" cluster-creation.yaml |
|  |  |  |
|  |  | **# Replace the <YOUR CLUSTER NAME> placeholder with the CLUSTER\_NAME and** |
|  |  | **# Two commands under the the cluster-autoscaler deployment** |
|  |  | printf -v spc %12s |
|  |  | sed -i "s#<YOUR CLUSTER NAME>#$CLUSTER\_NAME\n${spc}- --balance-similar-node-groups\n${spc}- --skip-nodes-with-system-pods=false#g" cluster-creation.yaml |
|  |  |  |
|  |  | **# Deploy the manifest file** |
|  |  | kubectl create -f cluster-creation.yaml |
|  |  |  |

Test Autoscaler :

**#List the deployment**

$ kubectl get deployment webserver

NAME        READY   UP-TO-DATE   AVAILABLE   AGE

webserver   1/4     4            1           7m47s

$ kubectl get pods

NAMESPACE     NAME                           READY   STATUS    RESTARTS   AGE

default    db-58d5794487-8fmh8     1/1     Running   0          16s

default   db-58d5794487-m4kn     0/1     Pending   0          16s

default    db-58d5794487-d2lkh     0/1     Pending   0          16s

default    db-58d5794487-dnkr8     0/1     Pending   0          16s

**After 30mins all the pods are in Running state:**

> kubectl get pods

NAME                         READY   STATUS    RESTARTS   AGE

db-58d5794487-8fmh8   1/1     Running   0          47m

db-58d5794487-9m4kn   1/1     Running   0          47m

db-58d5794487-d2lkh   1/1     Running   0          47m

db-58d5794487-dnkr8   1/1     Running   0          47m