Oded Bachenheimer, 22-10-2019

Intro

EKS is a fully managed service where you can deploy, scale and manage different type of containers using Kebernetes on AWS.

Pods: group of containers. Includes specification on how to run the containers, network and storage.

Containers run on workers which are nothing but EC2 instances.

\*\* make sure Hong Kong is chosen as the region at the top of AWS page. Otherwise higher charges would occur for the long distance data transmission.

Create an EKS cluster

Create an IAM Role: (to assign permissions for K8S to manage resources on your behalf)

* Search IAM in AWS search box
* On dashboard, click on *ROLES*, *create role*.
* Select *EKS*, then click on *Next: Permissions*, the *Next: Review*.
* Role Name: eksrole, then *Create Role.*

Create VPC and security groups for your cluster to use

* Search CloudFormation in AWS search box. (this is a tool that allows you to use a single text file to manage all the resources that you are using on AWS)
* Click on *Create Stack*
* Paste this under *Amazon S3 URL:* (will create a VPC, security group and 3 subnets)

<https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2019-10-08/amazon-eks-vpc-sample.yaml>

* *Next*
* *Next*
* *Next*
* *Create Stack*
* When the stack is ready, click on the stack name, go to *outputs* and copy and copy SecurityGroups, SubnetIds, VpcId to a text editor.

Create an EKS cluster

* Search EKS in AWS search box.
* Click on *Next Step*
* Cluster Name: nginxcluster
* Role Name: eksrole
* Choose the VPC and security group that we copied to the text editor
* Click on *Create*

Connect to EKS (point kubectl to K8S cluster)

* Using AWS CLI (from PowerShell) : aws eks update-kubeconfig --name nginxcluster (maybe should add --region ap-east-1).

Now if you go to C:\Users\USER\.kube and open config file you should see that the cluster you created was added.

Deploy worker nodes (CloudFormation)

Workers are just EC2 instances so we can use CloudFormation to create them.

* Search CloudFormation in AWS search box.
* *Create Stack*
* Paste this under *Amazon S3 URL:*

<https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2019-10-08/amazon-eks-nodegroup.yaml>

* Stack Name: nginxcluster-worker-nodes
* EKS Cluster: nginxcluster (same as we created before)
* ClusterControlPanelSecurityGroup: Same we created before
* NodeGroupName: nodegp1
* Choose min&max number of instances
* NodeImageId: ami-0921568c05f370e97 (Hong Kong Region’s AMI)
* KeyName: mykeypair
* If don’t have one, create it by going to EC2, choosing *Key Pairs* under *Network & security* at the dashboard, and clicking on *Create Key Pair.*
* VpcId: same as we created before.
* Subnets: same 3 we created before.
* Click *Next*
* Click *Next*
* Click on *I acknowledge…* and then click on *Create Stack*

If you type kubectl get nodes from PowerShell you can see there are no worker nodes yet. That’s because we haven’t connected them to the cluster yet. To that we use a K8S object of type ConfigMap.

Create a file aws-auth-cm.yaml and save it in C:\bin (where kubectl.exe is located):

apiVersion: v1

kind: ConfigMap

metadata:

name: aws-auth

namespace: kube-system

data:

mapRoles:

- rolearn: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

username: system:node:{{EC2PrivateDNSName}}

groups:

- system:bootsrappers

- system:nodes

On AWS web page, under *Stacks*, click on the stack you have just created (nginxcluster-worker-nodes), then go to *Outputs*, copy the NodeInstanceRole, and paste it under ‘rolearn’ in aws-auth-cm.yaml.

On PowerShell: kubectl apply –f aws-auth-cm.yaml

If you type kubectl get nodes now you can see that there are 3 worker nodes attached to cluster.

Run Kubernetes apps on EKS cluster (the same like you do with any other K8S environment)

Let’s launch a simple application on the cluster we have just created.

nginx.yaml:

apiVersion: app/v1 # for versions before 1.9.0 use apps/v1beta2

kind: Deployment

metadata:

name: nginx

spec:

selector:

matchLabels:

run: nginx

replicase: 2 # tells deployment to run 2 pods matching the template

template:

metadata:

labels:

run: nginx

spec:

containers:

- name: nginx

impage: nginx:1.7.9

ports:

- containerPort: 80 # the port to be used when accessing the application from a web browser

Service to manage the pods, in case one fails or for load balancing.

An abstraction to define the logical set of pods running somewhere on your cluster.

All of these pods have the same configuration as specified on the Deployment Object.

nginx-svc.yaml:

apiVersion: v1

kind: Service

metadata:

name: nginx

labels:

run: nginx

spec:

ports:

# the port that this service should serve on

- port: 80

protocol: TCP

selector:

run: nginx

type: LoadBalancer

Save the files in C:\bin (where kubectl.exe is located) as well.

From C:\bin:

kubectl create –f nginx.yaml => deployment.apps “nginx” created

kubectl create –f nginx-svc.yaml => service “nginx“ created

kubectl get services –o wide should show the newly created service, and give you the external IP address.

kubectl describe svc nginx to get all the information about the application. Copy the external IP address and using the port specified here too, use a web browser to access your application (might take a few minutes to be ready).