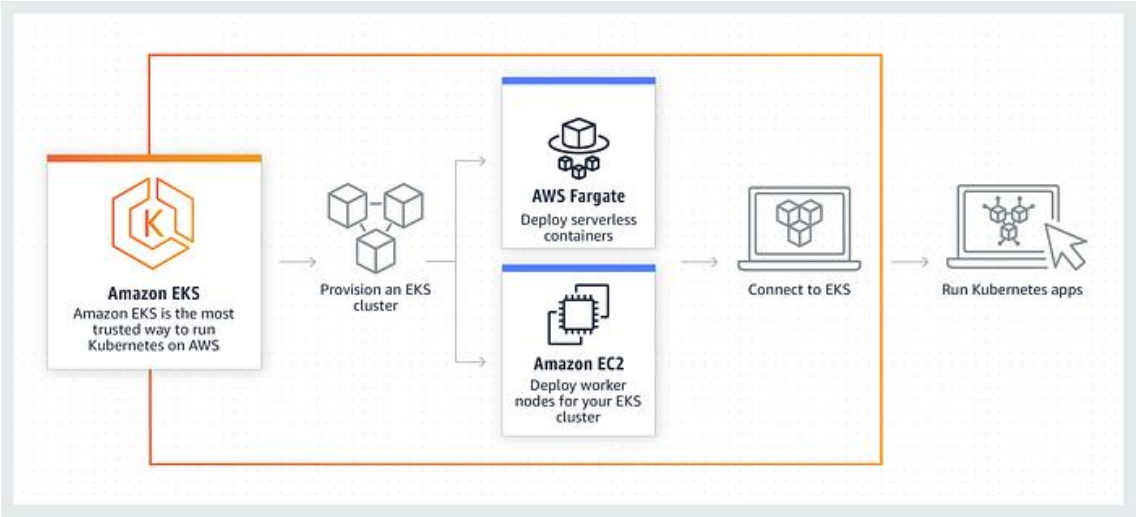
Mi primera Aplicación con EKS



Amazon Elastic Kubernetes Service (Amazon EKS) es un servicio Kubernetes completamente administrado.

EKS es el mejor lugar para ejecutar Kubernetes por varios motivos. Primero, puede elegir ejecutar sus clústeres de EKS con AWS Fargate, que es un cómputo sin servidor para contenedores.

Segundo, EKS está profundamente integrado con servicios tales como Amazon CloudWatch, grupos de Auto Scaling, AWS Identity and Access Management (IAM), y Amazon Virtual Private Cloud (VPC).

En este articulo veremos como podemos desplegar nuestra aplicación en Elastic Kubernetes Service . A continuación se muestra el resumen de actividades que se estarán realizando.

- *Crear y subir nuestra imagen a Elastic Container Registry.*
- *Instalar kubectl & eksctl.*
- *EKS Cluster precio.*
- *Crear nuestro Cluster y Node Groups.*
- *Ejecutar nuestro Deployments.*
- *Exponer nuestro Deployment.*
- *Eliminar nuestro cluster.*

1.-*Crear y subir nuestra imagen a ECR*.

Para esta actividad seguir el laboratorio anterior donde se detalla como subir una imagen a *Elastic Container Registry.*

2.-*Instalar AWS CLI & kubectl & eksctl.*

- Instalar AWS CLI
- Instalar kubectl CLI
- Instalar eksctl CLI

3.-*EKS Cluster costo*.

Mencionar que EKS no se encuentra dentro de la capa gratuita que ofrece AWS.

EKS Cluster :

Se paga $0.10 por hora por cada Amazon EKS cluster.

EKS Worker Nodes Precio - EC2 :

Pagas por los recursos que utilizas ejemplo (EC2 instances or EBS volumes)

T3 Medium Server en N.Virginia

- $0.0416 per Hour

Si trabajamos 1 dia completo para este articulo y utilizamos :

* 1 EKS Cluster

* 2 Worker nodes t3.medium

por 1 dia el costo total aproximadamente sera de $3.4.

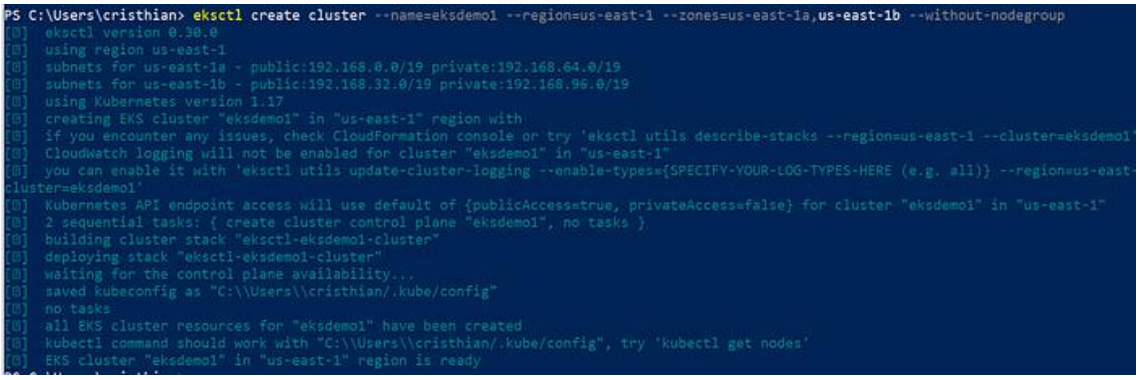
4.-*Crear nuestro Cluster y Node Groups*.

- En este caso utilizaremos Eksctl que es una herramienta simple de CLI para crear clústeres en EKS ,con tal solo unos comando podemos crear un cluster , esto demorara aproximadamente de 10 a 15 minutos.

Creación Cluster

****

```
eksctl create cluster --name=eksdemo1 \
                --region=us-east-1 \
                --zones=us-east-1a,us-east-1b \
                --without-nodegroup
```



- Asociar el clúster EKS al proveedor IAM OIDC.

Crear y asociar el proveedor OIDC de IAM para nuestro clúster EKS

****

eksctl utils associate-iam-oidc-provider \

--region us-east-1 \

--cluster eksdemo1 \

--approve

- Luego debemos de crear un grupo de nodos de Linux administrados por Amazon EKS que se registran en su clúster de Amazon EKS. Una vez que los nodos se unen al clúster, puede implementar aplicaciones de Kubernetes en ellos.
- Aunque --ssh-public-keyes opcional, le recomiendo que lo especifique cuando cree su grupo de nodos con un clúster. Esta opción habilita el acceso SSH a los nodos en su grupo de nodos administrados. Habilitar el acceso SSH le permite conectarse a sus instancias y recopilar información de diagnóstico si hay problemas, en este caso poner el nombre del key pair que tiene creado en esa región o en todo caso cree uno nuevo.

eksctl create nodegroup --cluster=eksdemo1 \

--region=us-east-1 \

--name=eksdemo1-ng-public1 \

--node-type=t3.medium \

--nodes=2 \

--nodes-min=2 \

--nodes-max=4 \

--node-volume-size=20 \

```
--ssh-access \
--ssh-public-key=kube-demo \
--managed \
--asg-access \
--external-dns-access \
--full-ecr-access \
--appmesh-access \
--alb-ingress-access
```

5.-*Ejecutar nuestro Deployments*.

Creamos nuestro archivo 02-deployment-definition.yml con la siguiente instrucción , donde básicamente indica crear 3 pods y menciona que imagen utilizara el contenedor. Luego ingresamos al CMD y nos dirigimos a la raíz donde se encuentra nuestro archivo y ejecutamos el siguiente código :

kubectl apply -f 02-deployment-definition.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: myapp-deployment
spec:
 replicas: 3
 selector:
  matchLabels:
   app: myapp
 template:
  metadata: # Dictionary
   name: myapp-pod
   labels: # Dictionary
    app: myapp
```

```
spec:
  containers: # List
   - name: myapp-container
     image: 818929933852.dkr.ecr.us-east-1.amazonaws.com/demo-ninja:latest
     ports:
      - containerPort: 80
```

6.-*Exponer nuestro Deployment*.

Luego creamos nuestro archivo 03-deployment-nodeport-service.yml para poder exponer nuestro deploy y así poder acceder a nuestra aplicación. De igual manera ingresamos al CMD y nos ubicamos en la raíz donde se encuentra el archivo y ejecutamos el siguiente comando.

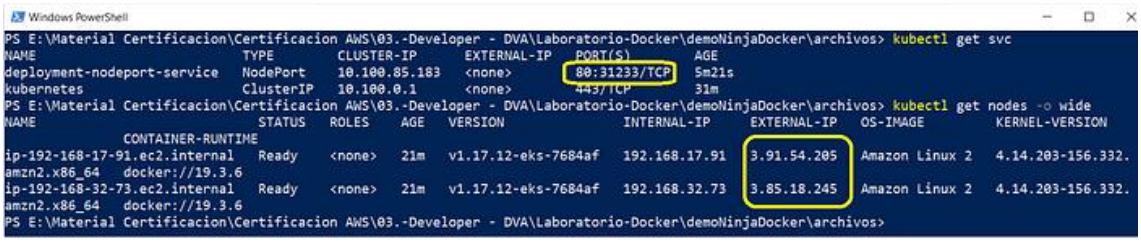
kubectl apply -f 03-deployment-nodeport-service.yml

```
apiVersion: v1
kind: Service
metadata:
  name: deployment-nodeport-service
spec:
  type: NodePort
  selector:
    app: myapp
  ports:
   - name: http
     port: 80
     targetPort: 80
     nodePort: 31233
```

Luego de ejecutar nuestros dos archivos necesitamos saber las ip de nuestros nodos para poder acceder al aplicativo.

```
# Listamos el servicio
kubectl get svc
```

mostramos la ip publica.
kubectl get nodes -o wide



Antes de acceder a nuestro aplicativo debemos dirigirnos al Security Group del Node Group y en Inbound rules añadir All TCP .

Teniendo las ip y el puerto podemos ingresar a nuestro aplicativo.

7.-Eliminar nuestro cluster.

Seguir los siguientes pasos para eliminar nuestro Cluster y Node Group.

Espero que te pueda ser útil este tutorial.