Auto-deploy your container to EKS

Pre-requisites:

- Should have completed the steps mentioned in **Setup CI/CD for a GitHub repository with Codebuild + Codepipeline** tutorial:
  - Have a ECR repository
  - Completed the AWS Code build setup
  - Have the code pipeline setup

Install Required Packages:

As part of this setup, we need to install AWS CLI, Kubectl, aws-iam-authenticator, EKSCTL otherwise the setup will not work as expected.

- **Installing aws cli (Install CLI V2)**
- **Installing kubectl**
- **Installing aws-iam-authenticator**
- **Installing eksctl**

Creating EKS Cluster:

Use the following command to create an EKS cluster in your AWS environment.

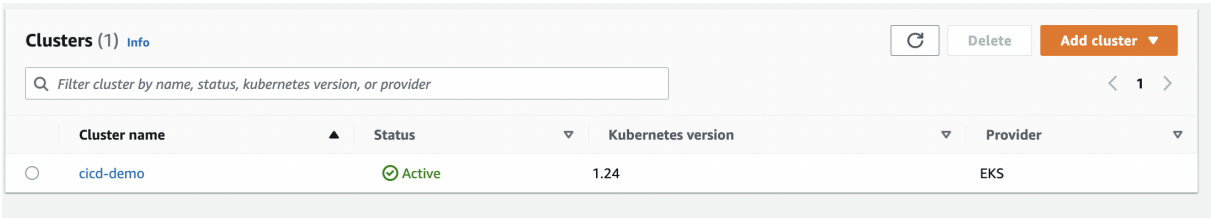
```
eksctl create cluster -f cluster.yaml
```

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: cicd-demo #cluster name
  region: us-east-1 #desired region

nodeGroups:
  - name: ng-1 #cluster node group name
    instanceType: t3.small #desired instance type
    desiredCapacity: 1 #desired nodes count / capacity
    ssh:
      allow: false # if true - will use ~/.ssh/id_rsa.pub as the default ssh key
      #publicKeyPath: ~/.ssh/ec2_id_rsa.pub #you can specify the public key path likr this as well
```

- Go to EKS Console on https://console.aws.amazon.com/eks
- Cluster will be created and the 1 node.



Create/Update the buildspec file:

- Create a file named "buildspec.yml" in the root of your GitHub repository if not already present
- Paste the following code into the buildspec file, which builds the Docker image and pushes it to the ECR repository:

```
version: 0.2
phases:
  install:
    commands:
      - echo Installing app dependencies...
      - curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.18.9/2020-11-02/bin/linux/amd64/kubectl
      - chmod +x ./kubectl
      - mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin
      - echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc
      - source ~/.bashrc
      - echo 'Check kubectl version'
      - kubectl version --short --client
  pre_build:
    commands:
      - echo Logging into Amazon EKS...
      - aws eks --region us-east-1 update-kubeconfig --name cicd-demo
      - echo check config
      - kubectl config view --minify
      - echo check kubectl access
      - kubectl get svc
      - echo logging in to AWS ECR...
      - $(aws ecr get-login --no-include-email --region us-east-1)
  build:
    commands:
      - docker build -t <image name> .
  post_build:
    commands:
      - docker tag <image name> <ECR repository URI>:latest
      - docker push <ECR repository URI>:latest
      - kubectl apply -f deployment.yaml
      - kubectl apply -f service.yaml
      - kubectl rollout restart -f deployment.yaml
      - kubectl get svc --all-namespaces
```

- Replace <image name> and <ECR repository URI> with the actual values.
- Replace <cluster name> with the name you specified in `cluster.yaml`
- Run `git add buildspec.yml` , `git commit` and `git push` to commit the file

Create Deployment and Service Files:

- Create a file named "deployment.yaml" in the root of your GitHub repository

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app.kubernetes.io/name: cicd-demo
    app.kubernetes.io/instance: cicd-demo-instance
    app.kubernetes.io/version: '1.0.0'
    app.kubernetes.io/managed-by: kubectl
  name: cicd-demo-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: cicd-demo
  template:
    metadata:
      labels:
        app: cicd-demo
    spec:
      containers:
        - image: <ECR repository URI>:latest
          imagePullPolicy: Always
          name: cicd-demo
          ports:
            - containerPort: <Port>
```

- Create a file named "service.yaml" in the root of your GitHub repository

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/name: cicd-demo
    app.kubernetes.io/instance: cicd-demo-instance
    app.kubernetes.io/version: "1.0.0"
    app.kubernetes.io/component: backend
    app.kubernetes.io/managed-by: kubectl
  name: cicd-demo
spec:
  selector:
    app: cicd-demo
  type: NodePort
  ports:
   -  protocol: TCP
      port: 80
      targetPort: <Port>
```

Configuring AWS EKS Cluster for CI/CD:

Even though the CodeBuild role has permission to authenticate to the cluster, it doesn't have the requisite RBAC access to do any other action on the cluster.

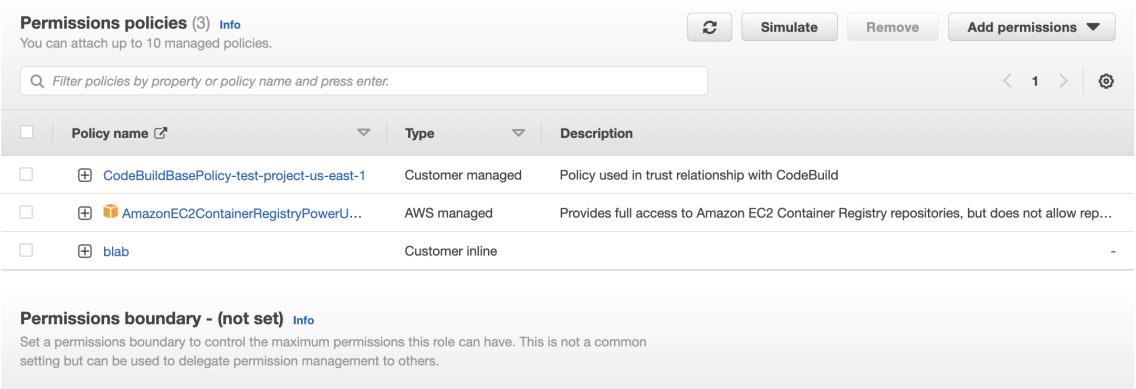
- Open the AWS CodeBuild console
- Open your build project you created in previous exercise. In **Build details** tab, scroll down to Environment where you can see the Service Role ARN, copy that.
- Edit the **aws-auth configmap** by running the following command

```
eksctl create iamidentitymapping --cluster cicd-demo --arn <Role Arn> --group system:masters --username <Role Name> --region us-east-1
# Eg: eksctl create iamidentitymapping --cluster cicd-demo --arn arn:aws:iam::<ac id>:role/TestCodeBuild --group system:masters --username
```

- Additionally to the CodeBuild Service Role (that you created in last exercise), attach a policy with **eks:DescribeCluster** action allowed. This will allow codebuild to download the kubeconfig file onto it's server.
  - Open the IAM console at https://console.aws.amazon.com/iam/
  - Click on the "Roles" menu from the sidebar
  - Select the **CodeBuild** Role you created
  - CLick Add **Permission → Create Inline Policy,** Switch to JSON Tab and paste the following

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "eks:DescribeCluster",
            "Resource": "*"
        }
    ]
}
```

  - Choose **Review, Provide a name and Create Policy**



Test your Pipeline:

- Make a code change to your configured source repository, commit, and push the change.
- Open the CodePipeline console at https://console.aws.amazon.com/codepipeline/.
- Choose your pipeline from the list.
- Watch the pipeline progress through its stages. Your pipeline should complete and your Amazon ECS service runs the Docker image that was created from your code change.