**Alif Rahi**

***Project 1***

**CSCI 381 - Cloud Computing**

To start the project, I created an EKS cluster using the following command below.

**eksctl create cluster -f cluster.yaml**

```
byalif@Alifs-MacBook-Pro eks_cicd % eksctl create cluster -f cluster.yaml
2023-03-19 16:07:27 [ℹ]  eksctl version 0.133.0
2023-03-19 16:07:27 [ℹ]  using region us-east-1
2023-03-19 16:07:27 [ℹ]  setting availability zones to [us-east-1f us-east-1d]
2023-03-19 16:07:27 [ℹ]  subnets for us-east-1f - public:192.168.0.0/19 private:192.168.64.0/19
2023-03-19 16:07:27 [ℹ]  subnets for us-east-1d - public:192.168.32.0/19 private:192.168.96.0/19
2023-03-19 16:07:27 [ℹ]  nodegroup "ng-1" will use "ami-0b4795e99297c2650" [AmazonLinux2/1.24]
2023-03-19 16:07:28 [ℹ]  using Kubernetes version 1.24
2023-03-19 16:07:28 [ℹ]  creating EKS cluster "cicd-demo" in "us-east-1" region with un-managed nodes
2023-03-19 16:07:28 [ℹ]  1 nodegroup (ng-1) was included (based on the include/exclude rules)
2023-03-19 16:07:28 [ℹ]  will create a CloudFormation stack for cluster itself and 1 nodegroup stack(s)
2023-03-19 16:07:28 [ℹ]  will create a CloudFormation stack for cluster itself and 0 managed nodegroup stack(s)
2023-03-19 16:07:28 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-east-1
2023-03-19 16:07:28 [ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "cicd-dem
2023-03-19 16:07:28 [ℹ]  CloudWatch logging will not be enabled for cluster "cicd-demo" in "us-east-1"
2023-03-19 16:07:28 [ℹ]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all
2023-03-19 16:07:28 [ℹ]
2 sequential tasks: { create cluster control plane "cicd-demo",
    2 sequential sub-tasks: {
        wait for control plane to become ready,
        create nodegroup "ng-1",
    }
}
2023-03-19 16:07:28 [ℹ]  building cluster stack "eksctl-cicd-demo-cluster"
2023-03-19 16:07:28 [ℹ]  deploying stack "eksctl-cicd-demo-cluster"
2023-03-19 16:07:58 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-cluster"
2023-03-19 16:08:28 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-cluster"
2023-03-19 16:09:29 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-cluster"
2023-03-19 16:10:29 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-cluster"
2023-03-19 16:11:29 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-cluster"
2023-03-19 16:12:29 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-cluster"
2023-03-19 16:13:30 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-cluster"
2023-03-19 16:14:30 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-cluster"
2023-03-19 16:15:30 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-cluster"
2023-03-19 16:16:31 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-cluster"
2023-03-19 16:17:31 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-cluster"
2023-03-19 16:18:31 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-cluster"
2023-03-19 16:19:32 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-cluster"
2023-03-19 16:21:34 [ℹ]  building nodegroup stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:21:34 [ℹ]  --nodes-min=3 was set automatically for nodegroup ng-1
2023-03-19 16:21:34 [ℹ]  --nodes-max=3 was set automatically for nodegroup ng-1
2023-03-19 16:21:34 [ℹ]  deploying stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:21:34 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:22:05 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:22:53 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:23:59 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:24:40 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:25:17 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:26:56 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:28:06 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:30:05 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:31:10 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:32:22 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:33:45 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:34:36 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:36:21 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:38:08 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:39:13 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:40:27 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:41:30 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:42:43 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:44:23 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:45:07 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:46:05 [ℹ]  waiting for CloudFormation stack "eksctl-cicd-demo-nodegroup-ng-1"
2023-03-19 16:46:05 [!]  1 error(s) occurred and cluster hasn't been created properly, you may wish to check CloudFormation console
2023-03-19 16:46:05 [ℹ]  to cleanup resources, run 'eksctl delete cluster --region=us-east-1 --name=cicd-demo'
2023-03-19 16:46:05 [✖]  exceeded max wait time for StackCreateComplete waiter
Error: failed to create cluster "cicd-demo"
```

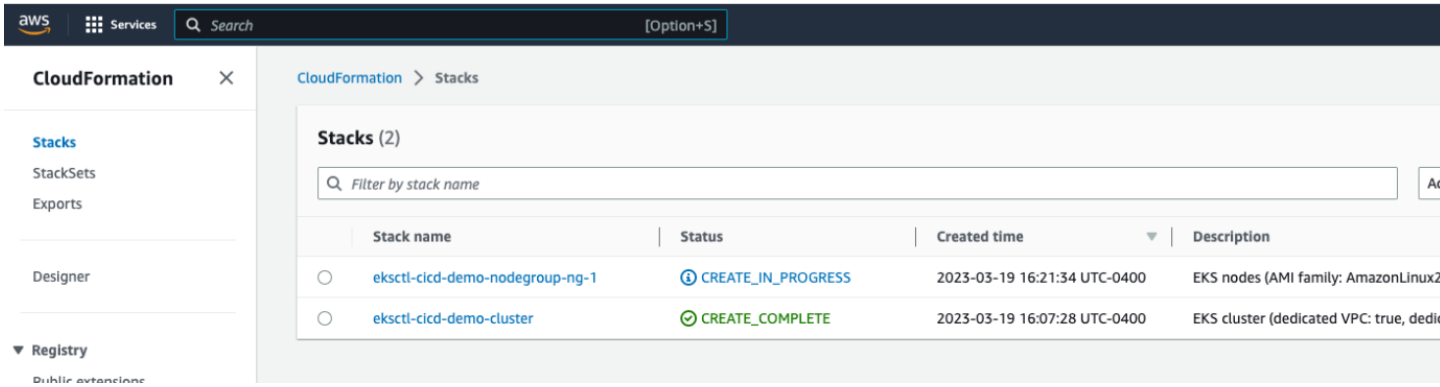
I used a cluster.yaml file to specify 3 worker nodes and some other metadata like region.

eks_cicd > cluster.yaml

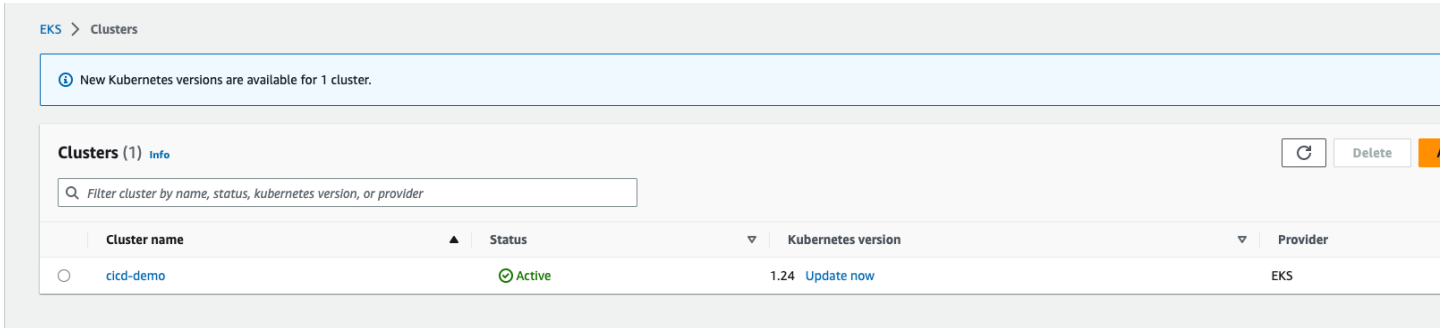
You, 50 minutes ago | 2 authors (Sandip Das and others)

```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: t2.medium #desired instance type
    desiredCapacity: 3 #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
```

This cluster creation kept timing out for some reason and I spent 4-5 days retracing my steps. Everytime i made a cluster, it took about 15-20 minutes. It ended up making 2 cloudformation stacks which creates the cluster in aws EKS.




## Cluster



This cluster was supposed to have 2 worker nodes/pods running as EC2 instances. I was planning on deploying a node.js and express server onto the EC2 instances.

**Instances (2)** Info

Connect | Instan

Find instance by attribute or tag (case-sensitive)

Instance state = running | Clear filters

| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | Public IPv4 DNS | Public |
|---|---|---|---|---|---|---|---|---|
| cicd-demo-ng-... | i-0c5cc511d50ad4e62 | Running | t2.medium | 2/2 checks passed | No alarms | us-east-1d | ec2-44-202-234-200.co... | 44.20 |
| cicd-demo-ng-... | i-0969e621912646428 | Running | t2.medium | 2/2 checks passed | No alarms | us-east-1d | ec2-44-214-119-119.co... | 44.21 |

Here is my application's yaml file that includes metadata about my server that I was going to push to one of the instances.

eks_cicd > deployment.yaml

Sandip Das, 23 months ago | 1 author (Sandip Das)

```yaml
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      Sandip Das, 23 months ago • code changes …
  selector:
    matchLabels:
      app: cicd-demo
  template:
    metadata:
      labels:
        app: cicd-demo
    spec:
      containers:
        - image: 120717539064.dkr.ecr.us-west-2.amazonaws.com/cicd-demo:latest
          imagePullPolicy: Always
          name: cicd-demo
          ports:
            - containerPort: 3000
```

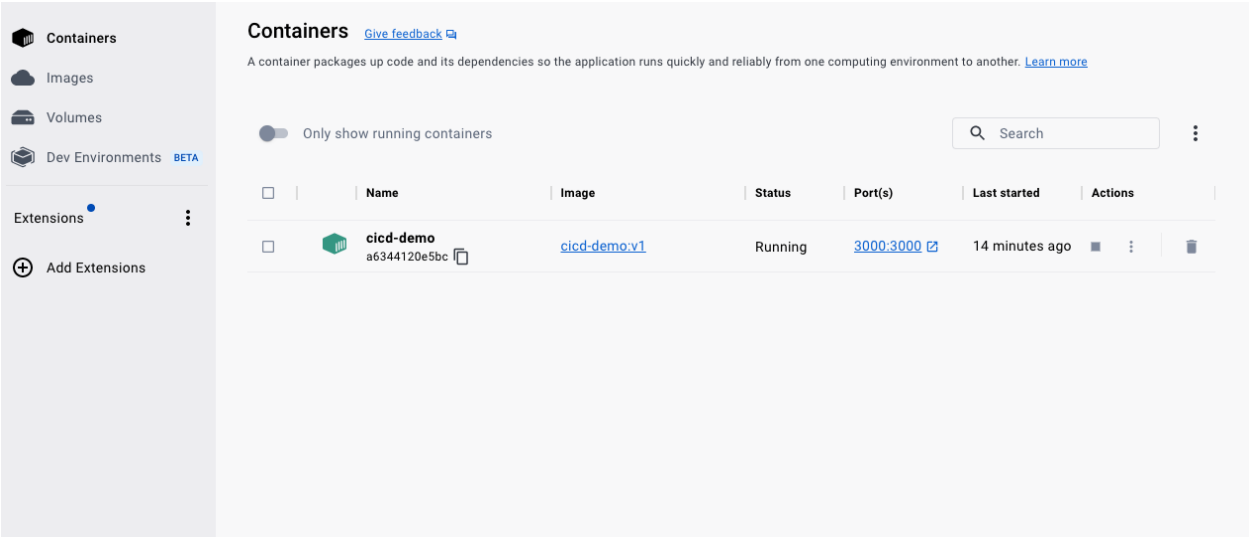
I used Docker desktop to containerize my application and then created an Image of it to push to aws EKS. These are the commands I used to containerized my server;

**To Make Docker Build**

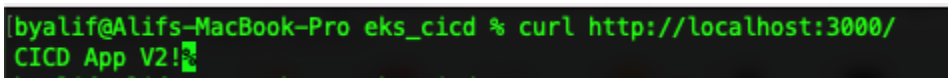
docker image build -t cicd-demo:v1 .

**Test image running fine or not:**

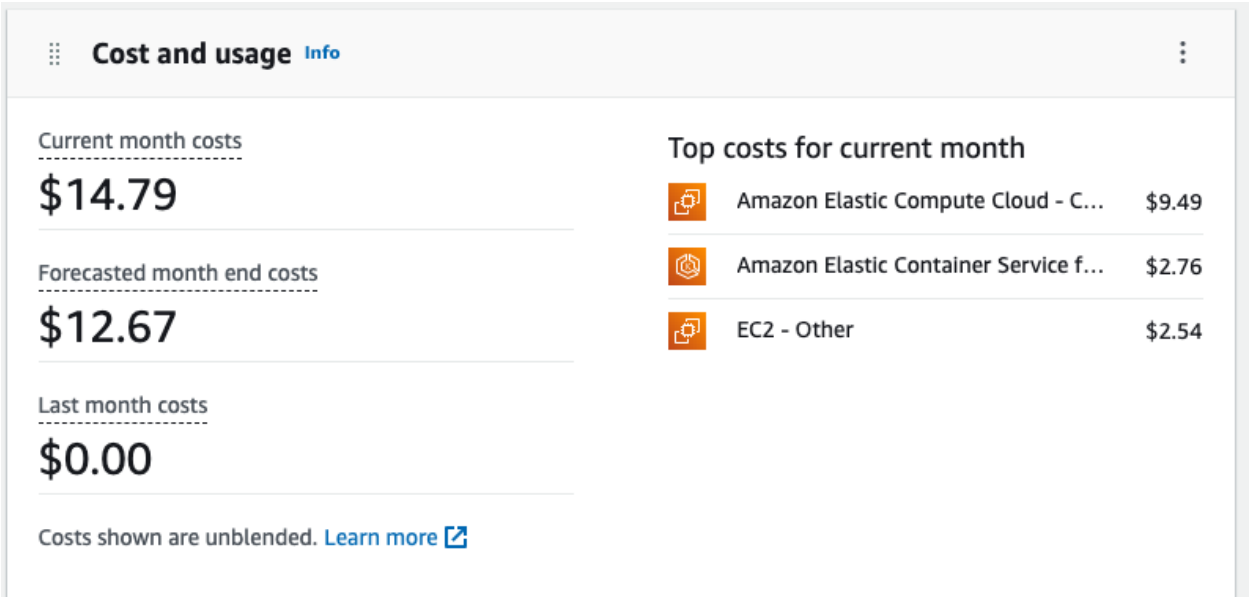
docker run -d --name cicd-demo -p 3000:3000 cicd-demo:v1




This is my containerized application which runs on [http://localhost:3000](http://localhost:3000)




I have been testing the EKS cluster for a few days now and didn't realize how much money it accumulated. I did not have any free student credits so I will have to pay a fee. From my calculations running an instance all month would cost about $75. Unfortunately I will be paying a fee because I forgot to delete my clusters after testing them. I was not able to scale the pods in my EKS cluster because it kept giving me timeout errors. But I really tried to get this to work.



In order to update the application, I would have run these commands,

```
kubectl set image deployment/<Deployment-Name>
<Container-Name>=<Container-Image> --record=true
kubectl set image deployment/my-first-deployment
kubenginx=stacksimplify/kubenginx:2.0.0 --record=true
```