Application Details

Instruction to run and maintain the Application

1. Firstly, we will create an Amazon Linux EC2 instance manually and set up the configuration as per our need.

2. We will Install terraform and kubernetes in the AWS by using the required installation instruction and use aws configure to connect it with AWS Account.

3. Installed terraform in the EC2 Instance using the below command.
    - "yum install -y yum-utils"
    - Then we added the repo using the command "yum-config-manager --add-repo https://rpm.releases.hashicorp.com/RHEL/hashicorp.repo"
    - Then install the terraform using "yum -y install terraform".

4. Checked if the EC2 is up-to-date or not using the command sudo yum update.

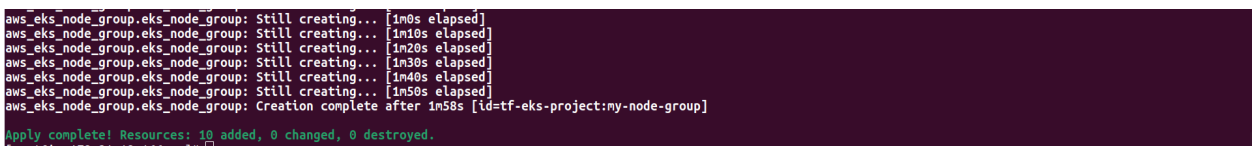
5. Now we will create a new directory in the AWS EC2 instance where we will be storing the terraform configuration file. In this folder we will create a filename "main.tf" and initialize it with the terraform code.

6. Now we have to create an AWS EKS cluster, so we will start with editing the main.tf file and add our terraform code in it.

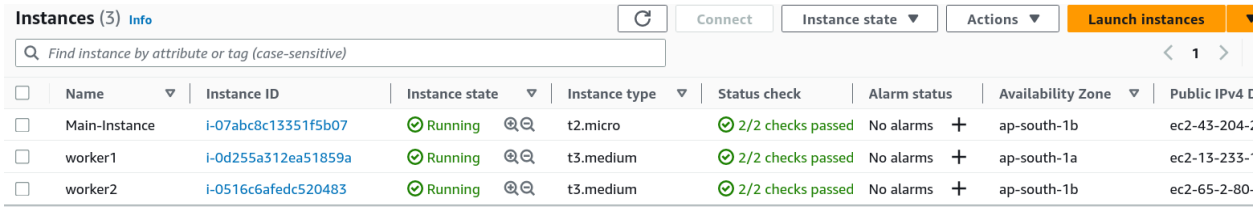
7. Now we will add the IAM roles and policies for the cluster and worker node by adding the code in the terraform file.

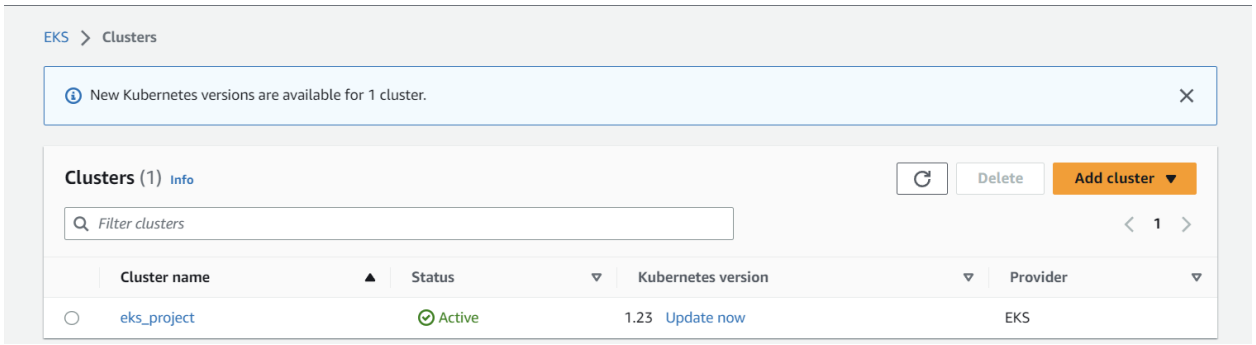
8. In the EKS node group we have desired size of 2 for our worker nodes and we have used t3.medium as our instance type.

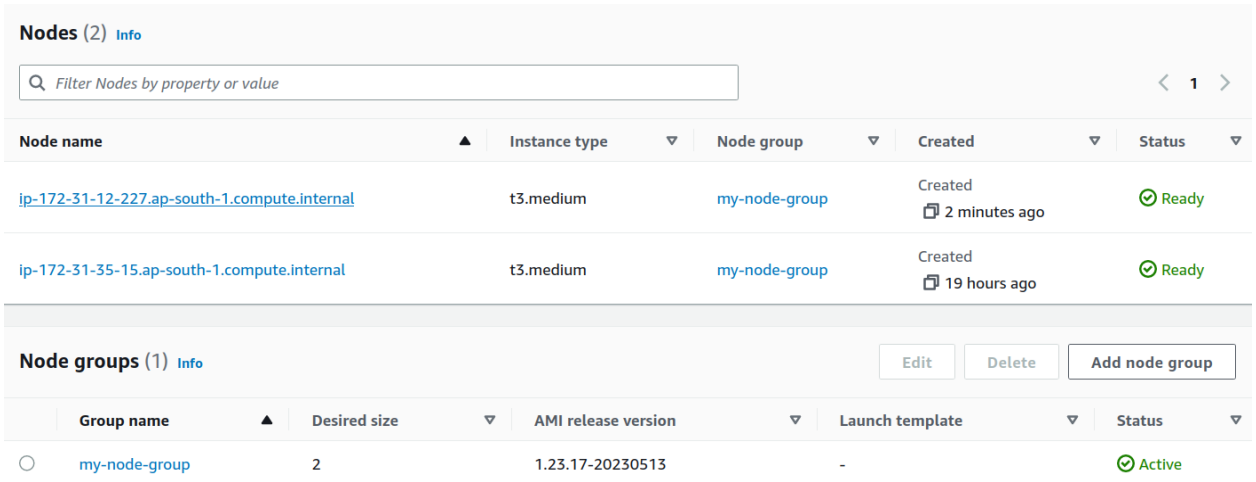
9. Once done, we will apply the new terraform code and wait for it to get completed. If the plan is correct, we will confirm it by typing "yes" when prompted. We will see the below output once the infra creation is completed



10. We can see the infrastructure created in the AWS as well.







11. Once the infrastructure is up, we will configure the newly created cluster by running the below command.

```
[root@ip-172-31-12-166 worker-node]# aws eks --region ap-south-1 update-kubeconfig --name eks_project
```

This will update the kubernetes config to access our cluster.

12. Now we will check if the cluster is accessible. We will see the output as per the below image.

```
[root@ip-172-31-12-166 worker-node]# kubectl get nodes
NAME                                          STATUS   ROLES    AGE   VERSION
ip-172-31-35-15.ap-south-1.compute.internal   Ready    <none>   15h   v1.23.17-eks-0a21954
```

13. We have downloaded EKSCTL for creating and managing clusters on EKS with the below commands.

```
curl --silent --location
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname
-s)_amd64.tar.gz" | tar xz -C /tmp
mv /tmp/eksctl /usr/local/bin
eksctl version
```

After running the above command we can see the version of downloaded EKSCTL.

```
[root@ip-172-31-12-166 ~]# eksctl version
0.143.0
```

14. Now we will create deployment configuration in the yaml format with the file name as deploy.yml for the Kubernetes with the name swordhealth-deployment using a docker image.

```
[root@ip-172-31-12-166 ~]# cat deploy.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: swordhealth-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: swordhealth
  template:
    metadata:
      labels:
        app: swordhealth
    spec:
      containers:
      - name: swordhealth
        image: swordhealth/node-example:0.0.1
        ports:
        - containerPort: 3000

[root@ip-172-31-12-166 ~]#
```

15. Now we will apply the above created deploy.yml to kubernetes using the below command

```
[root@ip-172-31-12-166 ~]# kubectl apply -f deploy.yml
deployment.apps/swordhealth-deployment created
[root@ip-172-31-12-166 ~]#
```

16. Once everything has been set up correctly we will access the pods and we will get to see the details of the pods as below.

```
[root@ip-172-31-12-166 ~]# kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
swordhealth-deployment-d98bdb5f4-7mhnm   1/1     Running   0          15s
[root@ip-172-31-12-166 ~]#
```

17. Once the pod is ready, we need to expose the application and for that we need the service resource. A Service provides a stable network endpoint (IP address and port) to access your application within the Kubernetes cluster.

```
[root@ip-172-31-12-166 ~]# cat service.yml
apiVersion: v1
kind: Service
metadata:
  name: swordhealth-service
spec:
  selector:
    app: swordhealth
  ports:
    - protocol: TCP
      port: 3000
      targetPort: 3000
  type: LoadBalancer
```

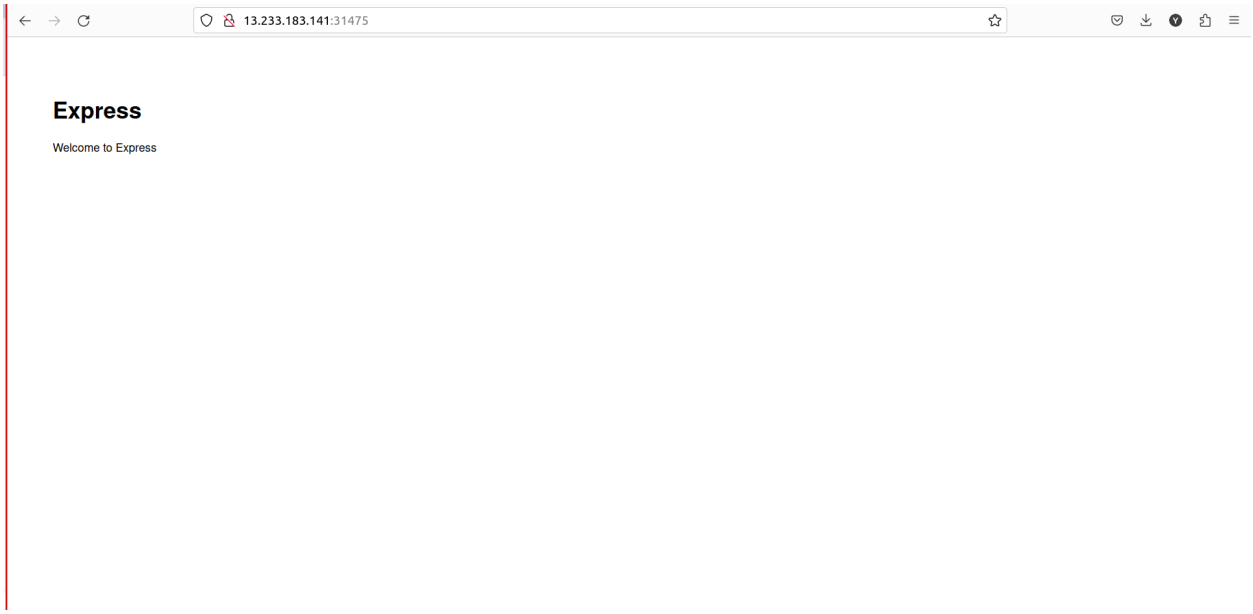
18. Now we will apply the above created service.yml to kubernetes using the below command

```
[root@ip-172-31-12-166 ~]# kubectl apply -f service.yml
service/swordhealth-service created
```

19. Once the service.yaml file has been applied, we will check if the service is up or not.

```
[root@ip-172-31-12-166 ~]# kubectl get service
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP                                                              PORT(S)          AGE
kubernetes          ClusterIP      10.100.0.1      <none>                                                                   443/TCP          32m
swordhealth-service LoadBalancer   10.100.132.40   a4071adf388414b5d834a94655405ceb-666562026.ap-south-1.elb.amazonaws.com   3000:31475/TCP   13s
[root@ip-172-31-12-166 ~]#
```

20. We are able to access the application.


13.233.183.141:31475
Express
Welcome to Express