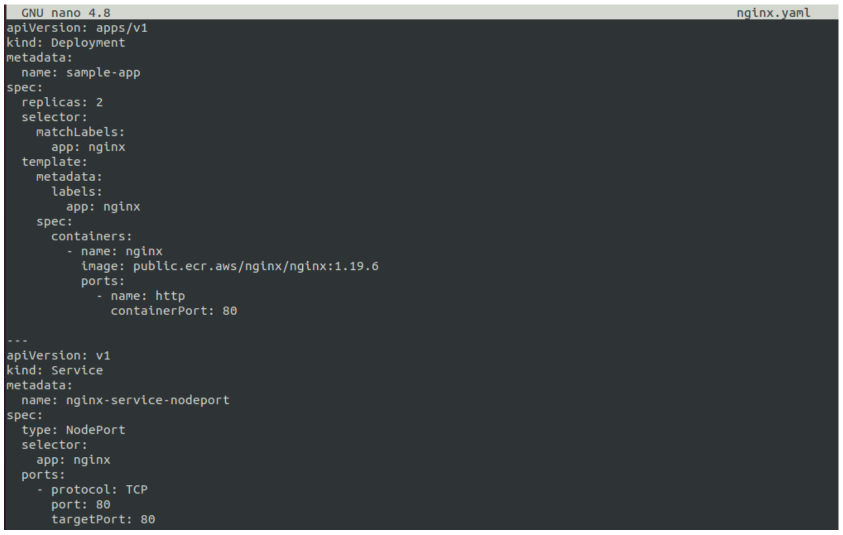
Task 1

Steps:

1. Create a cluster and give it a name using the following command ($ eksctl create cluster –name mycluster03).
2. Then create a Nginx deployment yaml file using the command ($ nano nginx.yaml) and paste the following:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: public.ecr.aws/nginx/nginx:1.19.6
          ports:
            - name: http
              containerPort: 80

---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service-nodeport
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

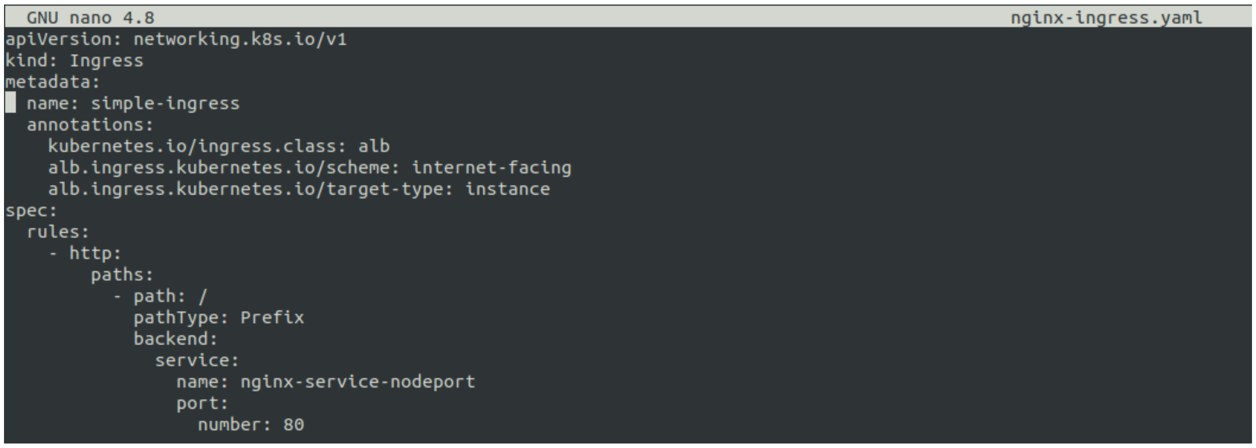


```
  GNU nano 4.8                                                                                              nginx.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: public.ecr.aws/nginx/nginx:1.19.6
          ports:
            - name: http
              containerPort: 80

---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service-nodeport
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

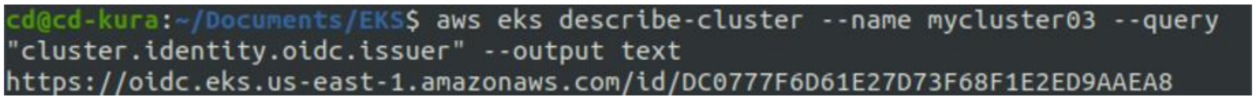
3. Then create an Ingress controller yaml file using the command ($ nano nginx-ingress.yaml) and paste the following:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: simple-ingress
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: instance
spec:
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: nginx-service-nodeport
                port:
                  number: 80
```


```
  GNU nano 4.8                                                                        nginx-ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: simple-ingress
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: instance
spec:
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: nginx-service-nodeport
                port:
                  number: 80
```

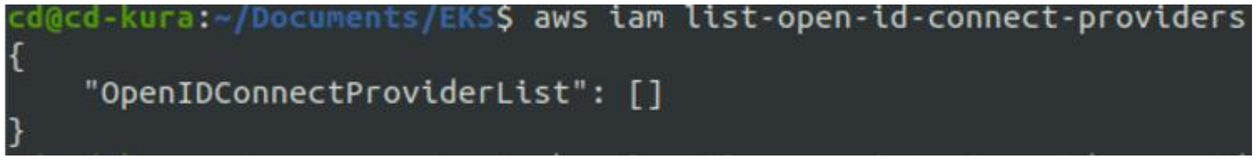
Task 2

4. Describe the cluster using the following command ($ aws eks describe-cluster --name mycluster03 --query "cluster.identity.oidc.issuer" --output text).



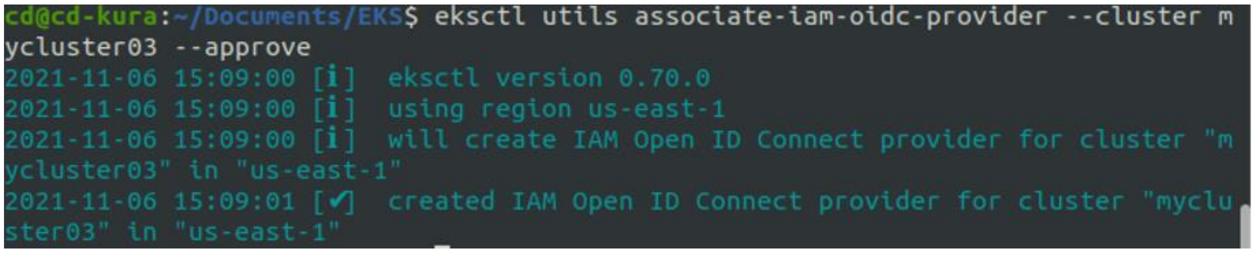
```
cd@cd-kura:~/Documents/EKS$ aws eks describe-cluster --name mycluster03 --query
"cluster.identity.oidc.issuer" --output text
https://oidc.eks.us-east-1.amazonaws.com/id/DC0777F6D61E27D73F68F1E2ED9AAEA8
```

5. Then check the open-id provider list using the following command ($ aws iam list-open-id-connect-providers).



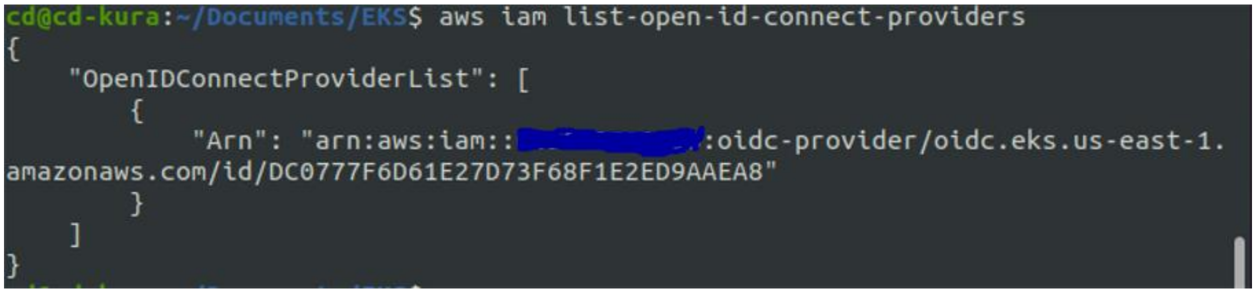
```
cd@cd-kura:~/Documents/EKS$ aws iam list-open-id-connect-providers
{
    "OpenIDConnectProviderList": []
}
```

6. Once done sign up for an openID provider using the command ($ eksctl utils associate-iam-oidc-provider --cluster mycluster03 –approve).



```
cd@cd-kura:~/Documents/EKS$ eksctl utils associate-iam-oidc-provider --cluster m
ycluster03 --approve
2021-11-06 15:09:00 [ℹ]  eksctl version 0.70.0
2021-11-06 15:09:00 [ℹ]  using region us-east-1
2021-11-06 15:09:00 [ℹ]  will create IAM Open ID Connect provider for cluster "m
ycluster03" in "us-east-1"
2021-11-06 15:09:01 [✔]  created IAM Open ID Connect provider for cluster "myclu
ster03" in "us-east-1"
```

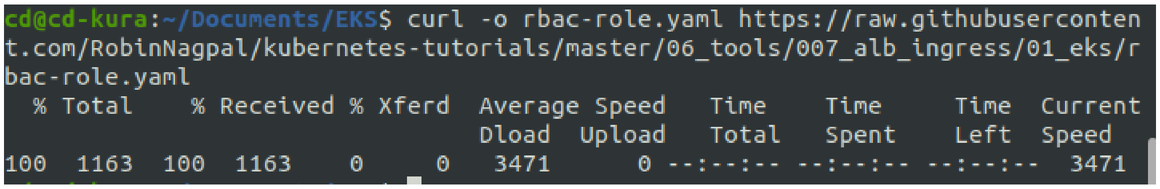
7. Then check the openID provider list to see the new change using the same command as earlier ($ aws iam list-open-id-connect-providers).



```
cd@cd-kura:~/Documents/EKS$ aws iam list-open-id-connect-providers
{
    "OpenIDConnectProviderList": [
        {
            "Arn": "arn:aws:iam::[redacted]:oidc-provider/oidc.eks.us-east-1.
amazonaws.com/id/DC0777F6D61E27D73F68F1E2ED9AAEA8"
        }
    ]
}
```

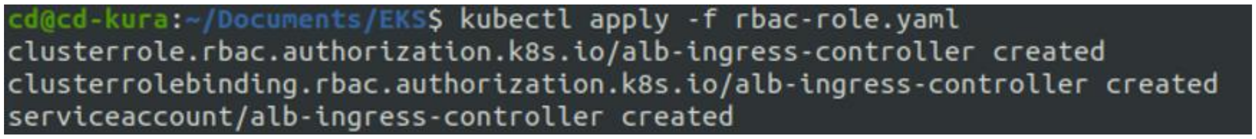
Task 3

8. Now download and save a file to rbac-role.yaml using the command ($ curl -o rbac-role.yaml \ https://raw.githubusercontent.com/RobinNagpal/kubernetes-tutorials/master/06_tools/007_alb_ingress/01_eks/rbac-role.yaml).



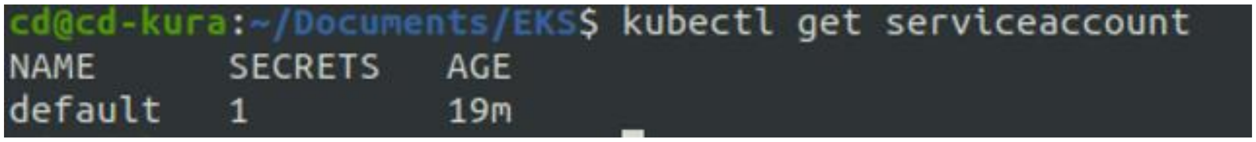
```
cd@cd-kura:~/Documents/EKS$ curl -o rbac-role.yaml https://raw.githubuserconten
t.com/RobinNagpal/kubernetes-tutorials/master/06_tools/007_alb_ingress/01_eks/r
bac-role.yaml
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1163  100  1163    0     0   3471      0 --:--:-- --:--:-- --:--:--  3471
```

9. Then create the rbac-role.yaml file using the following command ($ kubectl apply -f rbac-role.yaml). This will create a role which binds it to the ingress controller.



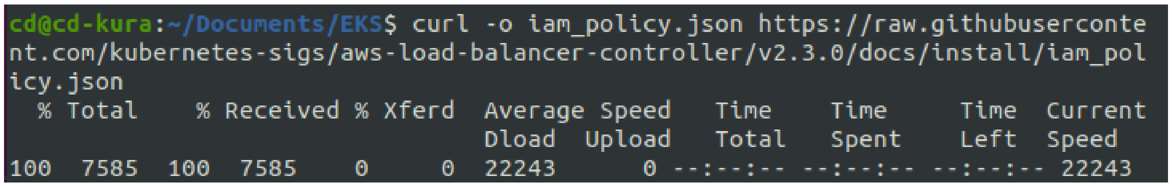
```
cd@cd-kura:~/Documents/EKS$ kubectl apply -f rbac-role.yaml
clusterrole.rbac.authorization.k8s.io/alb-ingress-controller created
clusterrolebinding.rbac.authorization.k8s.io/alb-ingress-controller created
serviceaccount/alb-ingress-controller created
```

10. Then view the service account using the command ($ kubectl get serviceaccount).



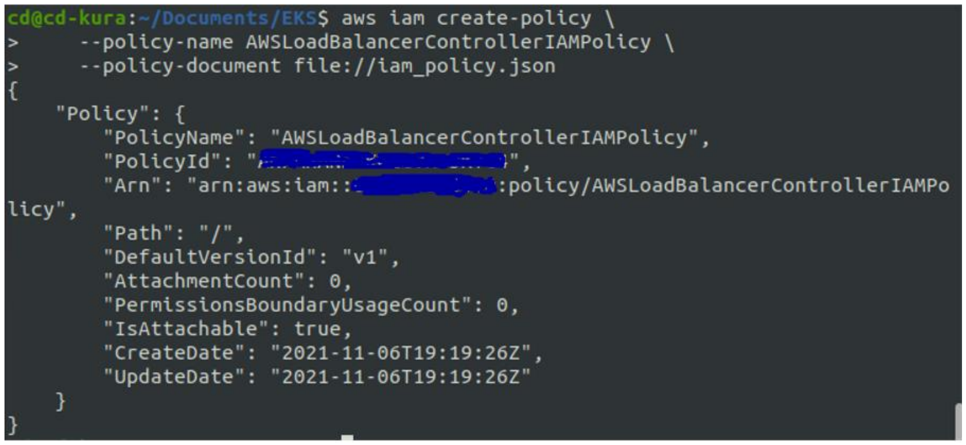
```
cd@cd-kura:~/Documents/EKS$ kubectl get serviceaccount
NAME      SECRETS   AGE
default   1         19m
```

11. Then create the IAM policy which will allow the ingress controller to create everything it needs on aws. Use the command ($ curl -o iam_policy.json https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.3.0/docs/install/iam_policy.json).



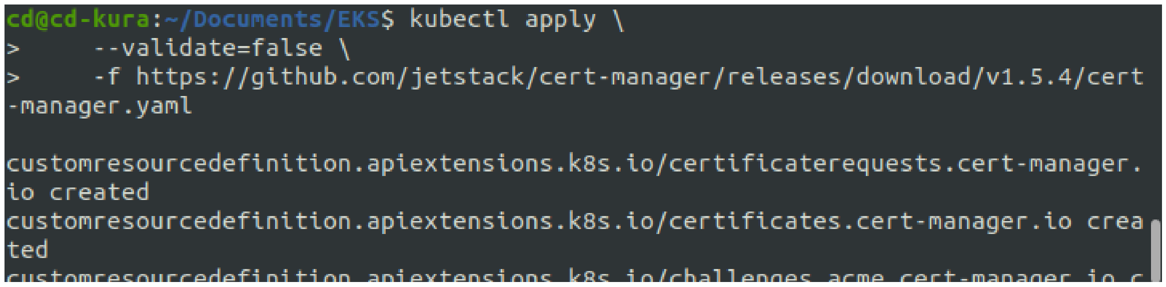
```
cd@cd-kura:~/Documents/EKS$ curl -o iam_policy.json https://raw.githubuserconte
nt.com/kubernetes-sigs/aws-load-balancer-controller/v2.3.0/docs/install/iam_pol
icy.json
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  7585  100  7585    0     0  22243      0 --:--:-- --:--:-- --:--:-- 22243
```

12. Now create the AWS policy using the following command ($ aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json).



```
cd@cd-kura:~/Documents/EKS$ aws iam create-policy \
>     --policy-name AWSLoadBalancerControllerIAMPolicy \
>     --policy-document file://iam_policy.json
{
    "Policy": {
        "PolicyName": "AWSLoadBalancerControllerIAMPolicy",
        "PolicyId": "[redacted]",
        "Arn": "arn:aws:iam::[redacted]:policy/AWSLoadBalancerControllerIAMPo
licy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2021-11-06T19:19:26Z",
        "UpdateDate": "2021-11-06T19:19:26Z"
    }
}
```

13. Then using the following command ($ eksctl create iamserviceaccount   --cluster=mycluster03   --namespace=kube-system   --name=aws-load-balancer-controller   --attach-policy-arn=arn:aws:iam::069598533000:policy/AWSLoadBalancerControllerIAMPolicy   --override-existing-serviceaccounts   --approve). Ensure that you use the correct **cluster name** and **ARN number**.
14. Now create a certificate manager for the ingress using the command ($ kubectl apply \ --validate=false \ -f https://github.com/jetstack/cert-manager/releases/download/v1.5.4/cert-manager.yaml).

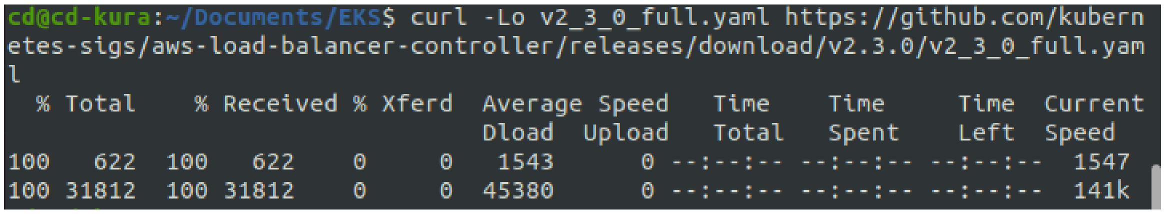


```
cd@cd-kura:~/Documents/EKS$ kubectl apply \
>     --validate=false \
>     -f https://github.com/jetstack/cert-manager/releases/download/v1.5.4/cert
-manager.yaml

customresourcedefinition.apiextensions.k8s.io/certificaterequests.cert-manager.
io created
customresourcedefinition.apiextensions.k8s.io/certificates.cert-manager.io crea
ted
customresourcedefinition.apiextensions.k8s.io/challenges.acme.cert-manager.io c
```

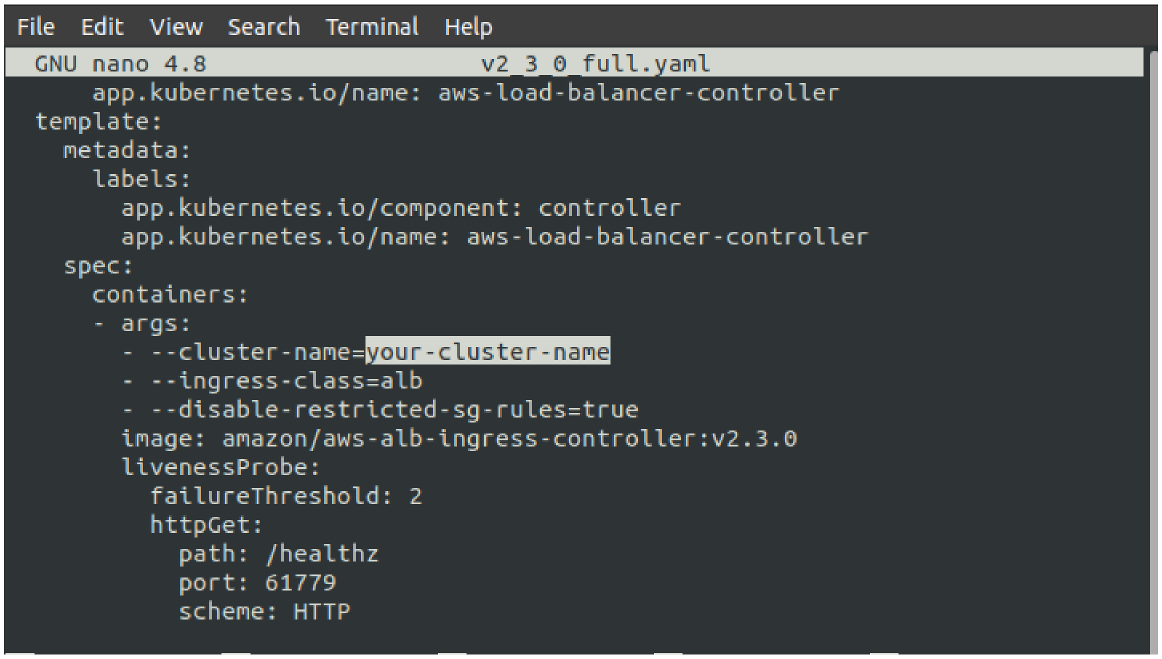
Task 4

15. Now create the load balancer controller by running the following command which will also download it ($ curl -Lo v2_3_0_full.yaml https://github.com/kubernetes-sigs/aws-load-balancer-controller/releases/download/v2.3.0/v2_3_0_full.yaml).



```
cd@cd-kura:~/Documents/EKS$ curl -Lo v2_3_0_full.yaml https://github.com/kubern
etes-sigs/aws-load-balancer-controller/releases/download/v2.3.0/v2_3_0_full.yam
l
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   622  100   622    0     0   1543      0 --:--:-- --:--:-- --:--:--  1547
100 31812  100 31812    0     0  45380      0 --:--:-- --:--:-- --:--:--  141k
```

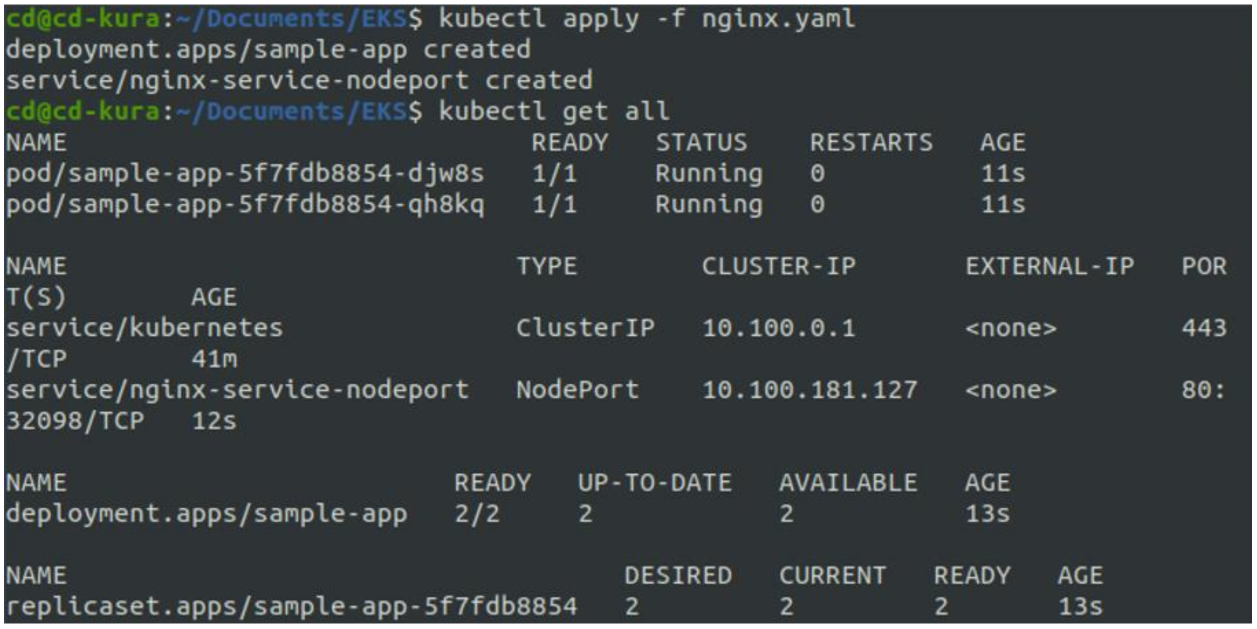
16. Now edit the yaml file to replace “your-cluster-name” with the name of your cluster. Use the command ($ nano v2_3_0_full.yaml). N.B if cant find it press “ctrl W” to search for it in nano.



```
  GNU nano 4.8                        v2_3_0_full.yaml
      app.kubernetes.io/name: aws-load-balancer-controller
  template:
    metadata:
      labels:
        app.kubernetes.io/component: controller
        app.kubernetes.io/name: aws-load-balancer-controller
    spec:
      containers:
      - args:
        - --cluster-name=your-cluster-name
        - --ingress-class=alb
        - --disable-restricted-sg-rules=true
        image: amazon/aws-alb-ingress-controller:v2.3.0
        livenessProbe:
          failureThreshold: 2
          httpGet:
            path: /healthz
            port: 61779
            scheme: HTTP
```

17. Save the changes then use the following command ($ kubectl apply -f v2_3_0_full.yaml).
18. To view the controller use the command ($ kubectl get deployment -n kube-system aws-load-balancer-controller).

19. Next create the yaml file using the command ($ kubectl apply -f nginx.yaml). To view what was created use the command ($ kubectl get all).



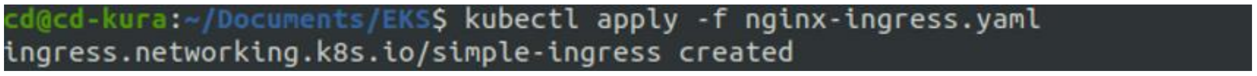
```
cd@cd-kura:~/Documents/EKS$ kubectl apply -f nginx.yaml
deployment.apps/sample-app created
service/nginx-service-nodeport created
cd@cd-kura:~/Documents/EKS$ kubectl get all
NAME                                 READY   STATUS    RESTARTS   AGE
pod/sample-app-5f7fdb8854-djw8s      1/1     Running   0          11s
pod/sample-app-5f7fdb8854-qh8kq      1/1     Running   0          11s

NAME                             TYPE        CLUSTER-IP       EXTERNAL-IP   POR
T(S)        AGE
service/kubernetes               ClusterIP   10.100.0.1       <none>        443
/TCP        41m
service/nginx-service-nodeport   NodePort    10.100.181.127   <none>        80:
32098/TCP   12s

NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/sample-app   2/2     2            2           13s

NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/sample-app-5f7fdb8854   2         2         2       13s
```

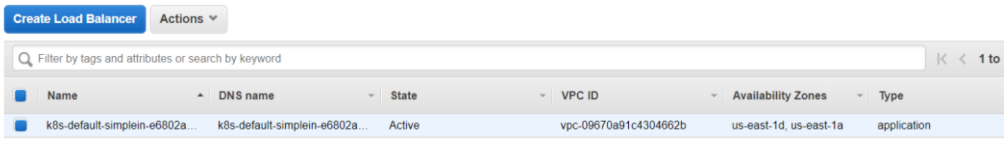
20. Then create the ingress controller yaml file using the command ($ kubectl apply -f nginx-ingress.yaml).



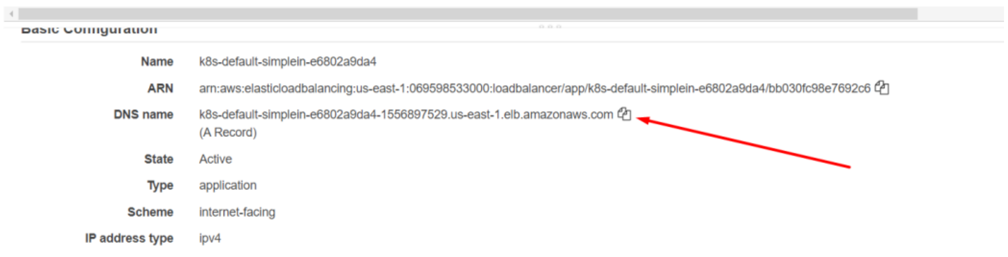
```
cd@cd-kura:~/Documents/EKS$ kubectl apply -f nginx-ingress.yaml
ingress.networking.k8s.io/simple-ingress created
```

Task 5

21. Then go to AWS to copy the DNS name. That is located in EC2 under load balancer.

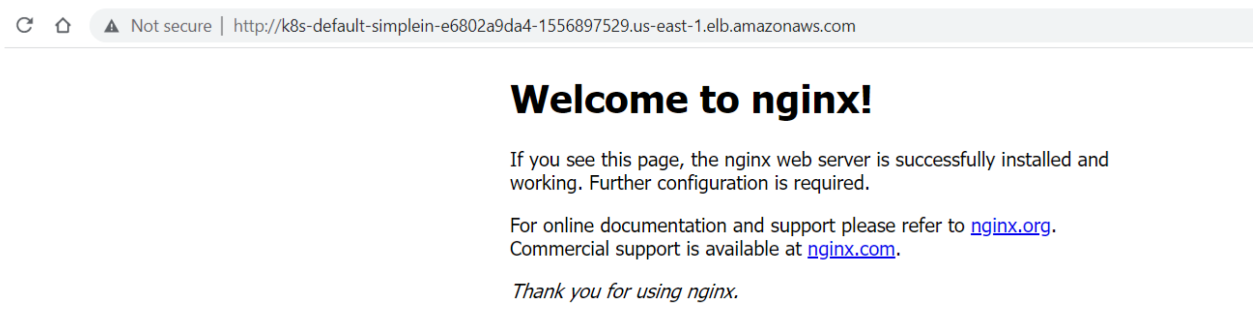


Name	DNS name	State	VPC ID	Availability Zones	Type
k8s-default-simplein-e6802a...	k8s-default-simplein-e6802a...	Active	vpc-09670a91c4304662b	us-east-1d, us-east-1a	application



Basic Configuration

Name	k8s-default-simplein-e6802a9da4
ARN	arn:aws:elasticloadbalancing:us-east-1:069598533000:loadbalancer/app/k8s-default-simplein-e6802a9da4/bb030fc98e7692c6
DNS name	k8s-default-simplein-e6802a9da4-1556897529.us-east-1.elb.amazonaws.com (A Record)
State	Active
Type	application
Scheme	internet-facing
IP address type	ipv4



Not secure | http://k8s-default-simplein-e6802a9da4-1556897529.us-east-1.elb.amazonaws.com

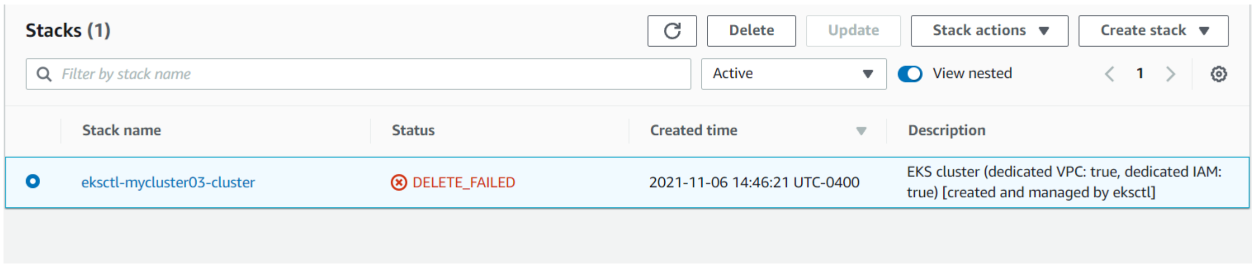
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

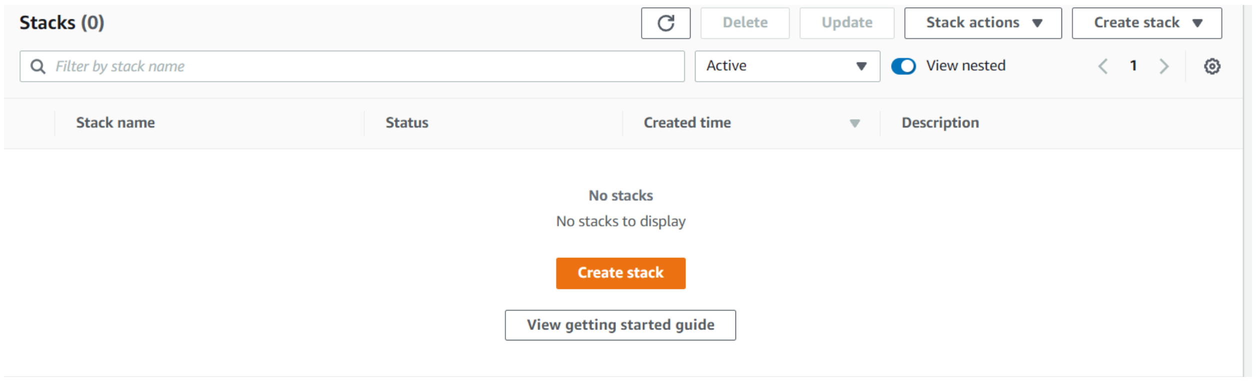
Thank you for using nginx.

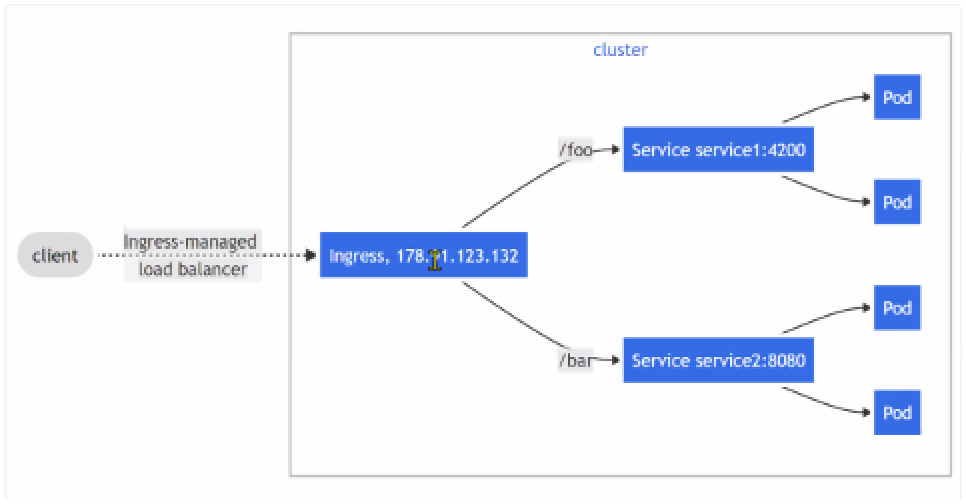
22. Delete the cluster created using the command ($ eksctl delete cluster --name mycluster03).



Stack name	Status	Created time	Description
eksctl-mycluster03-cluster	DELETE_FAILED	2021-11-06 14:46:21 UTC-0400	EKS cluster (dedicated VPC: true, dedicated IAM: true) [created and managed by eksctl]

NOTE: if cluster fails to delete go to security groups and delete the one associated with the cluster


Stacks (0)
Delete
Update
Stack actions
Create stack
Filter by stack name
Active
View nested
1
Stack name
Status
Created time
Description
No stacks
No stacks to display
Create stack
View getting started guide



Client accessing the NGINx web

After the client access the NGINX web, it goes to the load balancer then hits the ingress controller. The Ingress controller sends the traffic to the service port, it's also the NodePort. The NodePort then uses the port that's open on the node. The traffic is sent over to the Pod from the NodePort services.