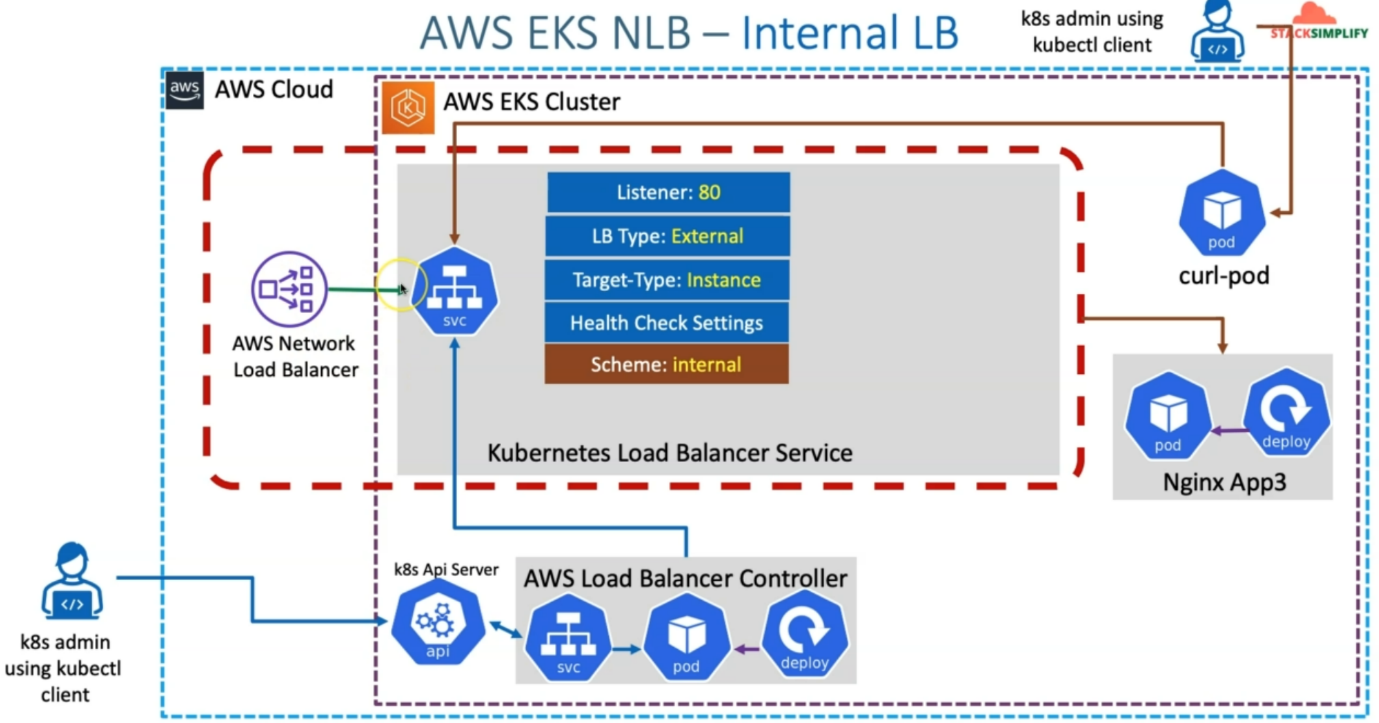
**8. NLB InternalLB Demo using k8s Service**

--- Reference - <https://github.com/stacksimplify/aws-eks-kubernetes-masterclass/tree/master/19-ELB-Network-LoadBalancers-with-LBC/19-05-LBC-NLB-Internal>



--- we are going to create the internal network load balancer using Kubernetes service and for this purpose, we will change the load balance scheme in our Kubernetes load balancer service manifest from internet facing to internal.

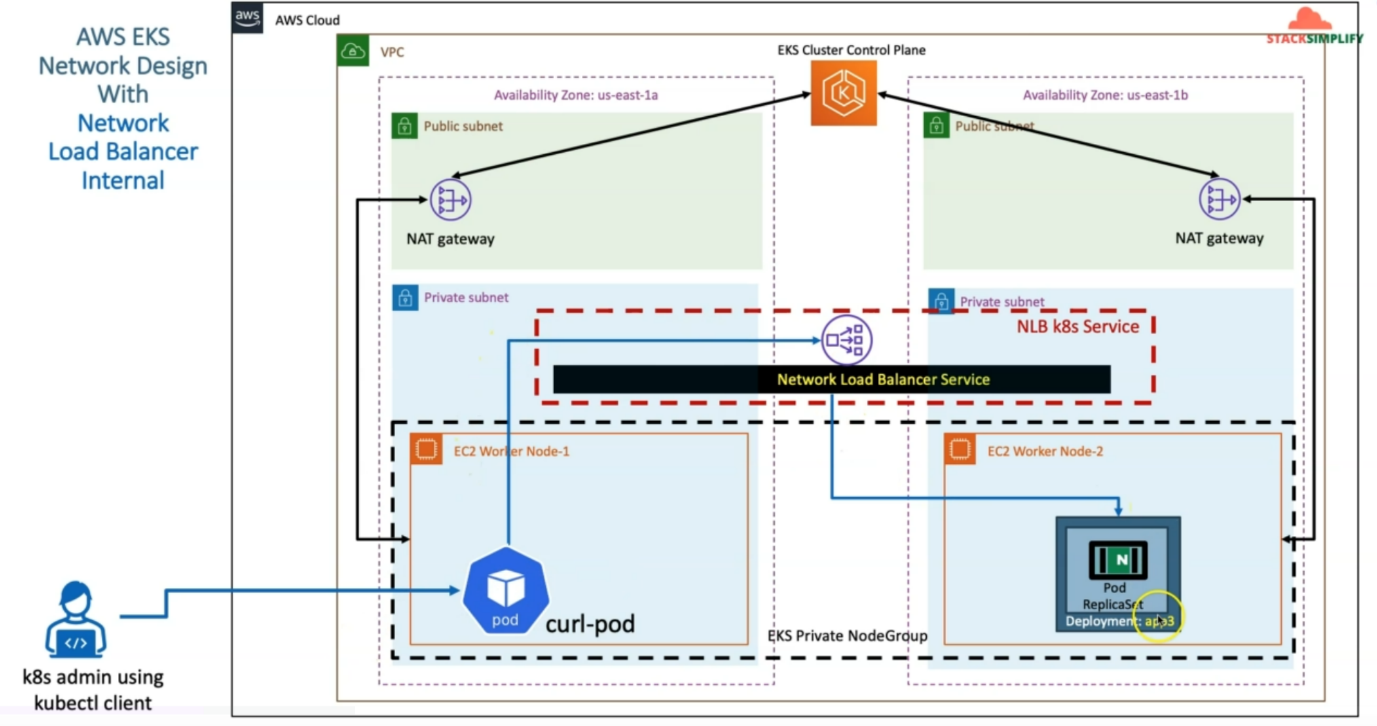
--- we will also ensure that we will remove all other annotations which are unrelated for this use case.

--- we will ensure that we will have only LB type external, which will help us in reconciling this respect to load balance of service with the latest load balancer controller. we'll use the target type instance and will use the standard health check settings.

--- once we deploy this Kubernetes Load Balancer service, it is going to create the internal network load balancer.

--- how are we going to test this? So, we are going to test this using a curl pod, so we will deploy a curl pod in the EKS cluster and as a Kubernetes admin using the kubectl client, I will connect to this curl pod and from Curl Pod. I will run the call command to access this network load balancer, which is internally created and test the sample application. Which is nginx app3 related page.

**See the network design for same thing**



--- in aws cloud, when we create the EKS cluster control plane. It creates the VPC public subnet, private subnet and we also created EKS Private Node Group with the worker nodes created in the private subnet and these communicate to the EKS cluster control plane outbound via NAT gateway.

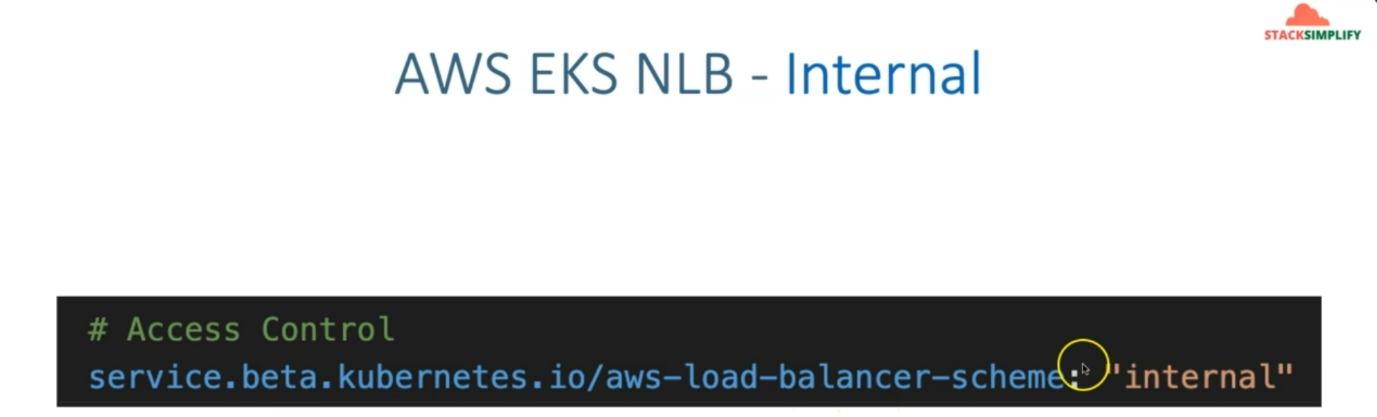
--- we'll also deploy our App3 deployment in our EKS private node group and we create the network

load balancer in the private subnet itself.

--- as we changed the load balancer scheme from internet facing to internal. So, this network load balancer is created in the private subnet for us and to access the sample application app3 inside this private subnet. We just deploy a curl pod in this same EkS cluster and we connect using k8s admin using kubectl client.

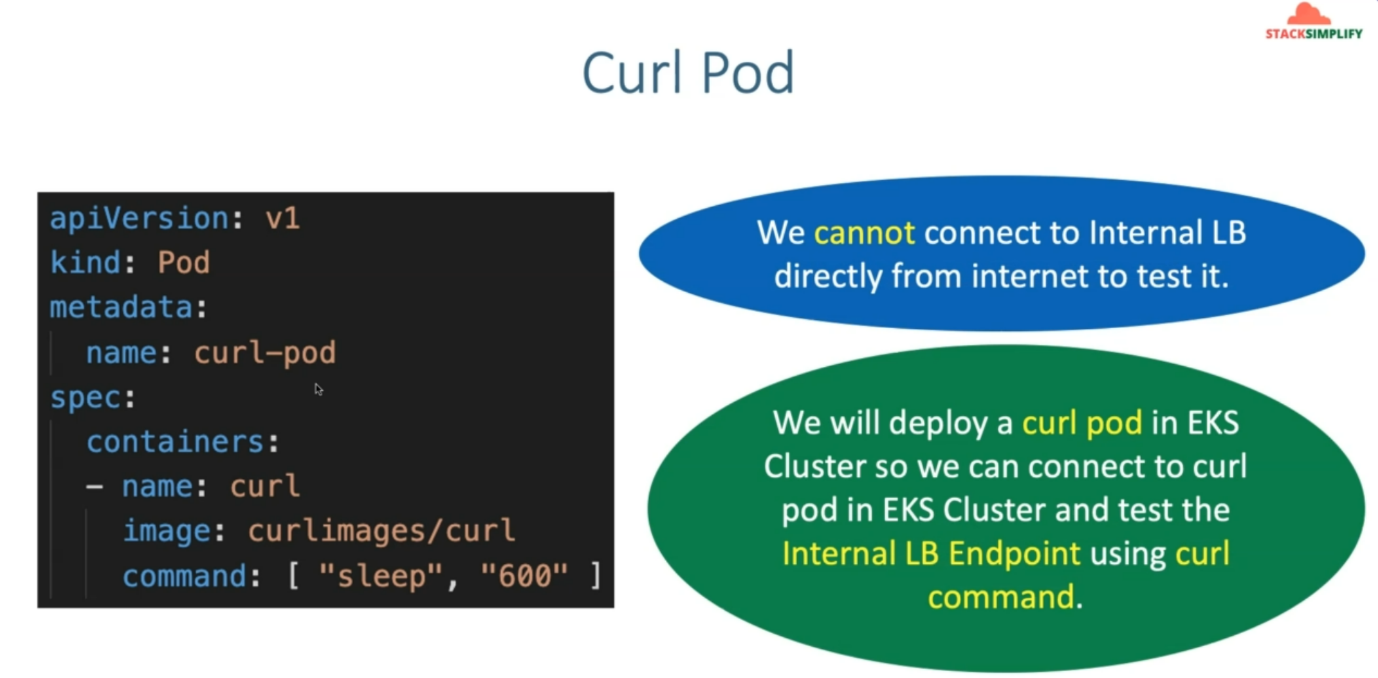
--- the request will go to the internal network load balancer and from there it reaches to the app3 application.

**Internal NLB scheme annotations**



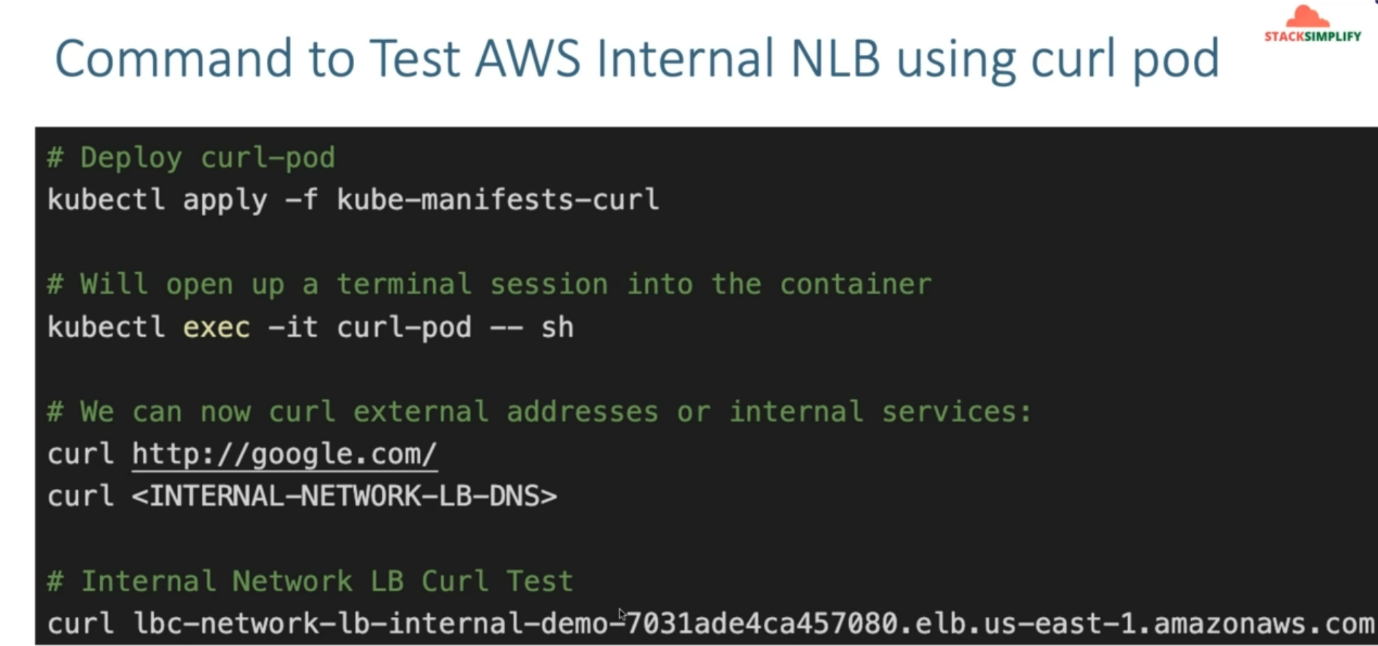
--- this annotation, we need to change from internet-facing to internal.

**Curl pod**



--- we cannot connect curl pod from internet so we will connect curl pod from EKS cluster plane.

**Commands to deploy curl pod**



**Introduction**

--- Create Internal NLB

--- Update NLB Service k8s manifest with aws-load-balancer-scheme Annotation as internal

--- Deploy curl pod

--- Connect to curl pod and access Internal NLB endpoint using curl command.

**Review LB Scheme Annotation**

--- **File Name: kube-manifests\02-LBC-NLB-LoadBalancer-Service.yml**

**# Access Control**

**service.beta.kubernetes.io/aws-load-balancer-scheme: "internal"**

apiVersion: v1

kind: Service

metadata:

  name: lbc-network-lb-internal

  annotations:

    # Traffic Routing

    service.beta.kubernetes.io/aws-load-balancer-name: lbc-network-lb-internal

    service.beta.kubernetes.io/aws-load-balancer-type: external

    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: instance

    #service.beta.kubernetes.io/aws-load-balancer-subnets: subnet-xxxx, mySubnet ## Subnets are auto-discovered if this annotation is not specified, see Subnet Discovery for further details.

    # Health Check Settings

    service.beta.kubernetes.io/aws-load-balancer-healthcheck-protocol: http

    service.beta.kubernetes.io/aws-load-balancer-healthcheck-port: traffic-port

    service.beta.kubernetes.io/aws-load-balancer-healthcheck-path: /index.html

    service.beta.kubernetes.io/aws-load-balancer-healthcheck-healthy-threshold: "3"

    service.beta.kubernetes.io/aws-load-balancer-healthcheck-unhealthy-threshold: "3"

    service.beta.kubernetes.io/aws-load-balancer-healthcheck-interval: "10" # The controller currently ignores the timeout configuration due to the limitations on the AWS NLB. The default timeout for TCP is 10s and HTTP is 6s.

    # Access Control

    service.beta.kubernetes.io/aws-load-balancer-scheme: "internal"

    # The VPC CIDR will be used if service.beta.kubernetes.io/aws-load-balancer-scheme is internal

    #service.beta.kubernetes.io/load-balancer-source-ranges: 0.0.0.0/0

    # AWS Resource Tags

    service.beta.kubernetes.io/aws-load-balancer-additional-resource-tags: Environment=dev,Team=test

spec:

  type: LoadBalancer

  selector:

    app: app3-nginx

  ports:

    - port: 80

      targetPort: 80

--- **01-Nginx-App3-Deployment.yml**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: app3-nginx-deployment

  labels:

    app: app3-nginx

spec:

  replicas: 1

  selector:

    matchLabels:

      app: app3-nginx

  template:

    metadata:

      labels:

        app: app3-nginx

    spec:

      containers:

        - name: app2-nginx

          image: stacksimplify/kubenginx:1.0.0

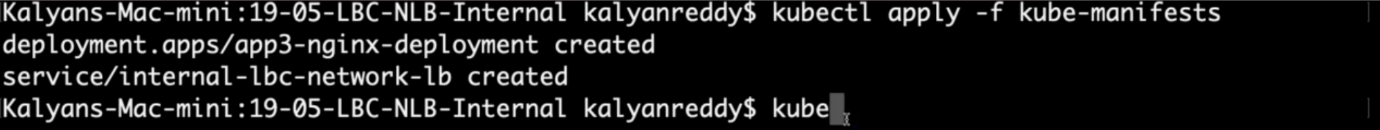
          ports:

            - containerPort: 80

**Deploy all kube-manifests**

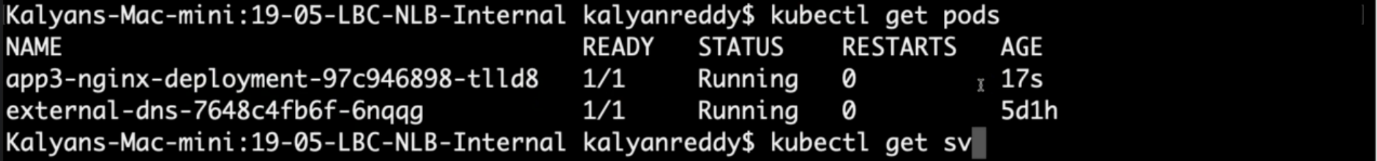
**# Deploy kube-manifests**

--- **kubectl apply -f kube-manifests/**



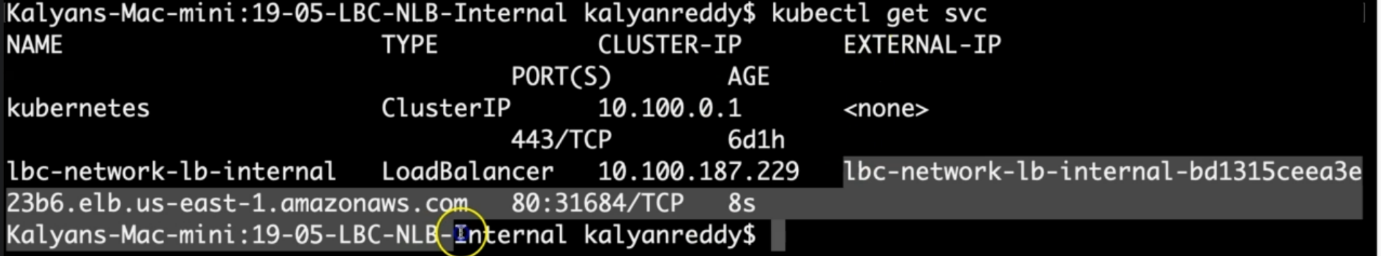
**# Verify Pods**

--- **kubectl get pods**



**# Verify Services**

--- **kubectl get svc**



Observation:

1. Verify the network lb DNS name

**# Verify AWS Load Balancer Controller pod logs**

--- **kubectl -n kube-system get pods**

--- **kubectl -n kube-system logs -f <aws-load-balancer-controller-POD-NAME>**

**# Verify using AWS Mgmt Console**

Go to Services -> EC2 -> Load Balancing -> Load Balancers

1. Verify Description Tab - DNS Name matching output of "kubectl get svc" External IP

2. Verify Listeners Tab

Observation: Should see two listeners Port 80

Go to Services -> EC2 -> Load Balancing -> Target Groups

1. Verify Registered targets

2. Verify Health Check path

**Deploy curl pod and test Internal NLB**

--- **01-curl-pod.yml**

apiVersion: v1

kind: Pod

metadata:

  name: curl-pod

spec:

  containers:

  - name: curl

    image: curlimages/curl

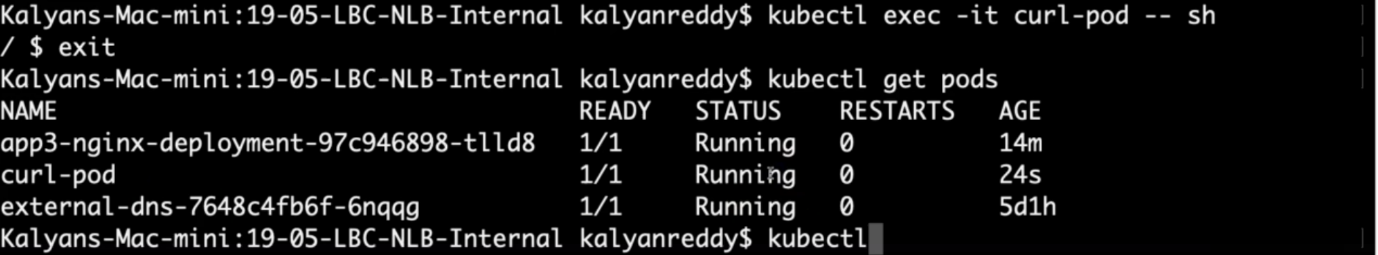
    command: [ "sleep", "600" ]

**# Deploy curl-pod**

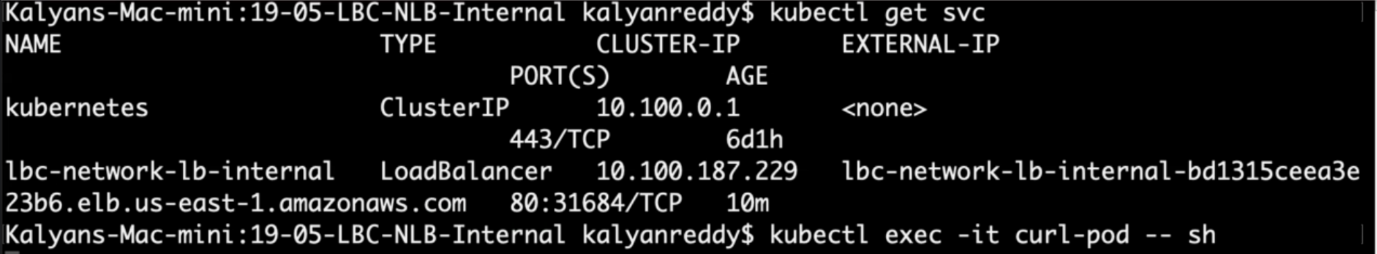
--- **kubectl apply -f kube-manifests-curl**

**# Will open up a terminal session into the container**

--- **kubectl exec -it curl-pod – sh**



--- curl pod is running.



--- this is the internal load balancer endpoint.

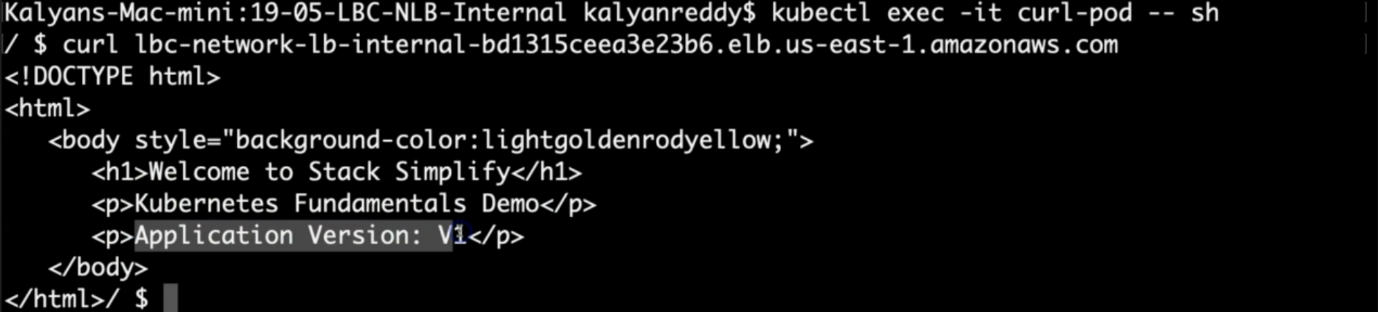
**# We can now curl external addresses or internal services:**

--- **curl http://google.com/**

--- **curl <INTERNAL-NETWORK-LB-DNS>**

**# Internal Network LB Curl Test**

--- curl lbc-network-lb-internal-demo-7031ade4ca457080.elb.us-east-1.amazonaws.com



--- **note** - we were able access the app3 via internal network load balancer.

**Clean-Up**

**# Delete or Undeploy kube-manifests**

--- **kubectl delete -f kube-manifests/**

--- **kubectl delete -f kube-manifests-curl/**

**# Verify if NLB deleted**

In AWS Mgmt Console,

Go to Services -> EC2 -> Load Balancing -> Load Balancers

**References**

--- Network Load Balancer - <https://docs.aws.amazon.com/eks/latest/userguide/network-load-balancing.html>

--- NLB Service - <https://kubernetes-sigs.github.io/aws-load-balancer-controller/v2.4/guide/service/nlb/>

--- NLB Service Annotations - <https://kubernetes-sigs.github.io/aws-load-balancer-controller/v2.4/guide/service/annotations/>