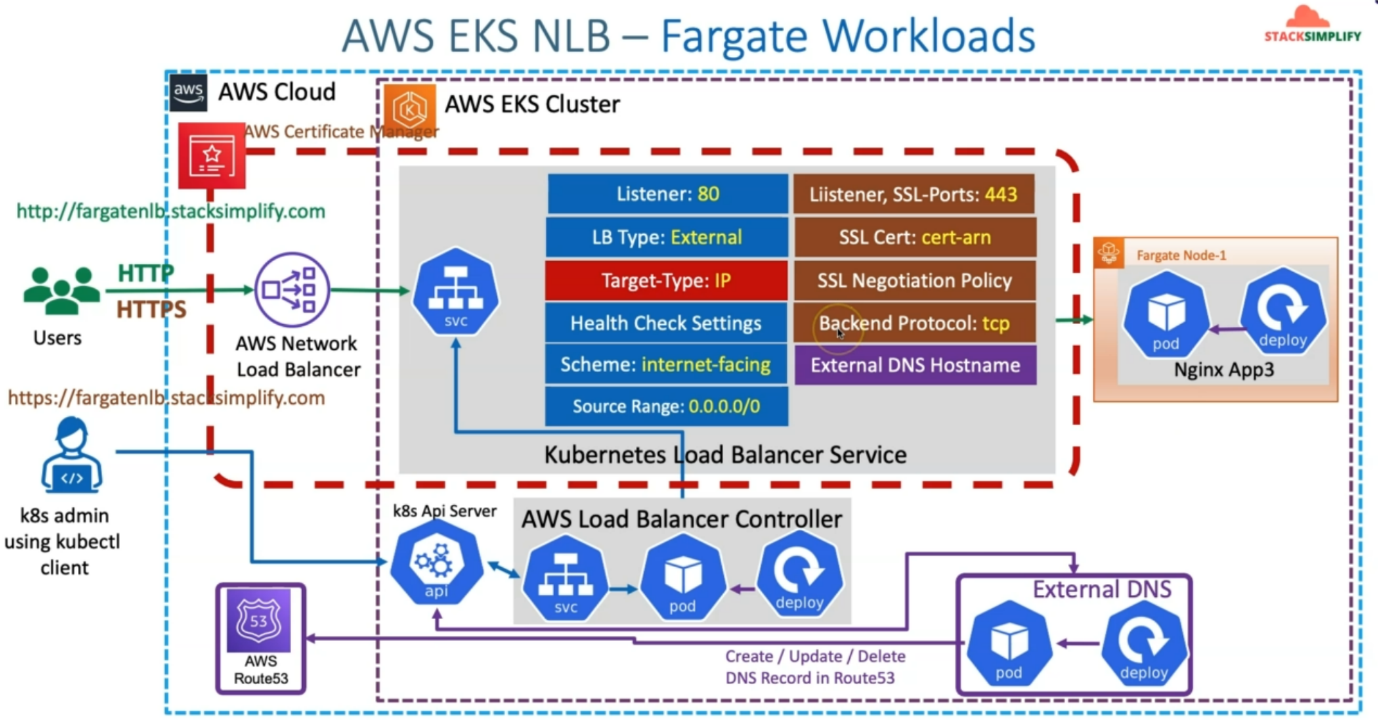
**9. NLB Fargate Demo with Target Type IP**

--- in this demo, we are going to send the traffic coming to the network load balancer to a work load that is running on fargate in cluster.



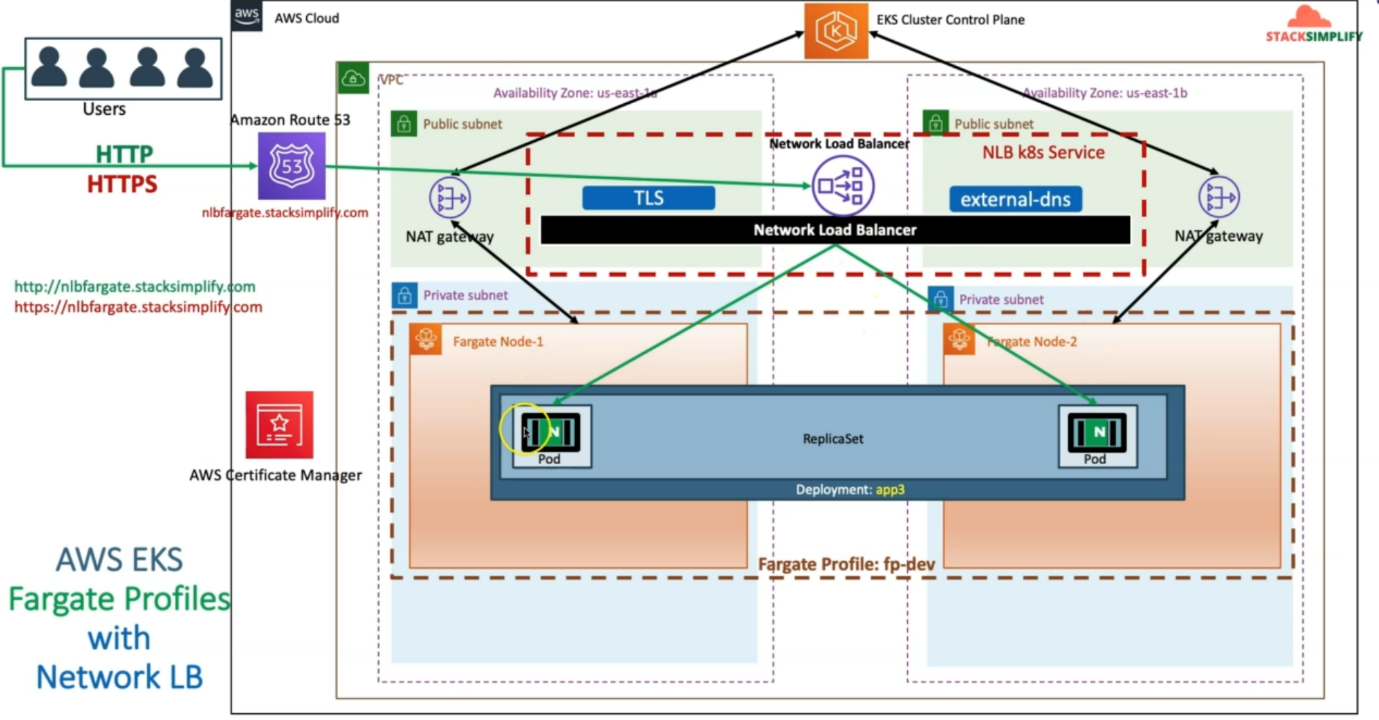
--- So, for that purpose, the changes included in the Kubernetes load balancer service off type network load balancer is **target type:IP**. we need to change the target type annotation to IP so that it can send the traffic to the pods running on the forget infrastructure.

--- here the Forget pods are going to be registered with pods IP as the targets in the target group

of this network load balancer.

--- let's also understand that this respective nginx app3 related pods will be scheduled on fargate nodes.

**Network design for same**



--- Let's understand the network architecture for the same thing.

--- in aws cloud, whenever we created the EKS cluster control plane using EKSCTL, equilent VPC public and private subnets are created and now we are also going to create forget profile and inside this forget profile. Whenever we deploy the workload, automatically it will create Forget nodes for us.

--- how many pods we create? So, those many no forget nodes will be created and another important thing we should be aware of is whenever we schedule pod on fargate infrastructure.

--- We need to ensure that pod initially will be in the pending state and it will take three to five to initialize all the forget nodes and then change the pod from pending to running.

--- Unlike regular worker nodes, the time taken to initialize the fargate scheduled pod is going

to be a little bit more so when you see it is in the pending state. Initially, we really don't need to worry. So, we need to wait for two to three minutes for the Forget node to get created and registered with the EKS cluster control plane and then ready to accept this pod to be running on those respective nodes.

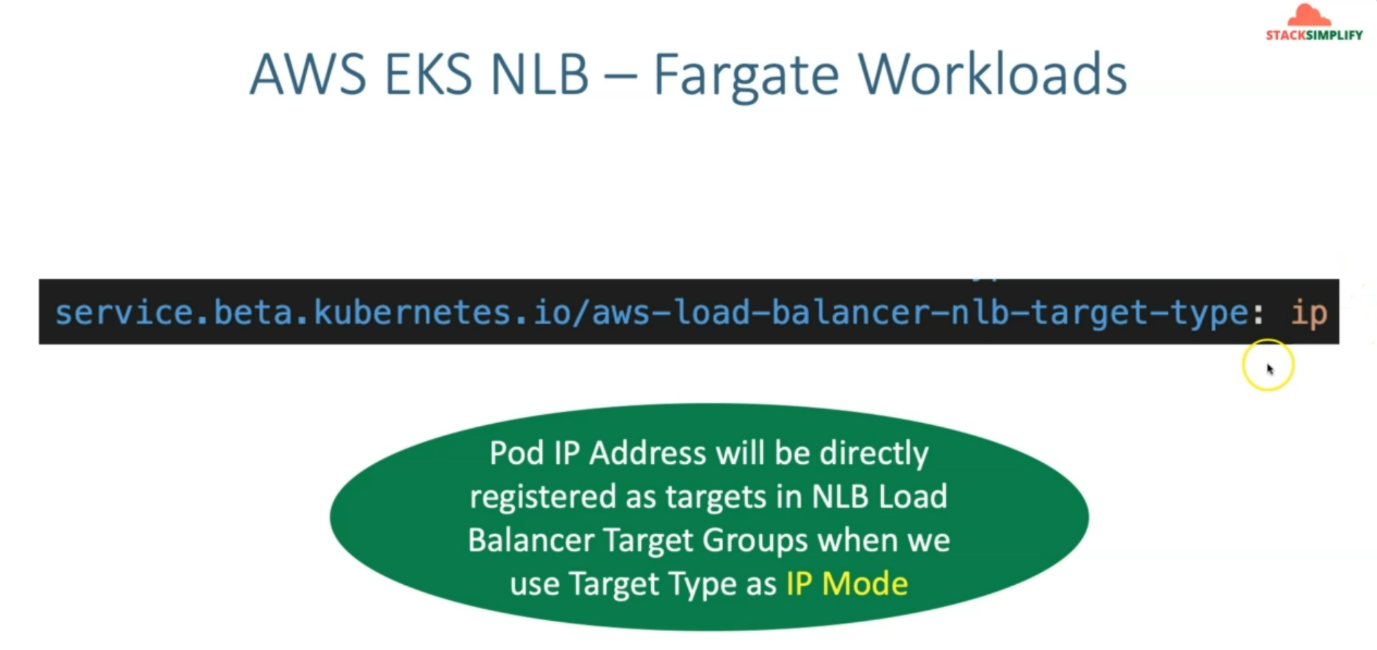
--- these fargate nodes also will be in communication with the EKS cluster control plane using that NAT gateways and the Kubernetes service off type network load balancer will be deployed,

--- as usual, it will have the standard features of the TLS, external DNS. All those things which we have already demonstrated in our previous sections.

--- only change here is target type annotation is going to be changed from instance to IP.

--- **NOTE** – the user request directly reaches to pods from network load balancer.

**Annotation change**



**Introduction**

--- Create advanced AWS Fargate Profile

--- Schedule App3 on Fargate Pod

--- Update NLB Annotation aws-load-balancer-nlb-target-type with ip from instance mode

**Review Fargate Profile**

--- **File Name: fargate-profile/01-fargate-profiles.yml**

apiVersion: eksctl.io/v1alpha5

kind: ClusterConfig

metadata:

  name: eksdemo1  # Name of the EKS Cluster

  region: us-east-1

fargateProfiles:

  - name: fp-app3

    selectors:

      # All workloads in the "ns-app3" Kubernetes namespace will be

      # scheduled onto Fargate:

      - namespace: ns-app3

--- **note** – any deployment uses ns-app3 then workloads will be deployed on fargate profile.

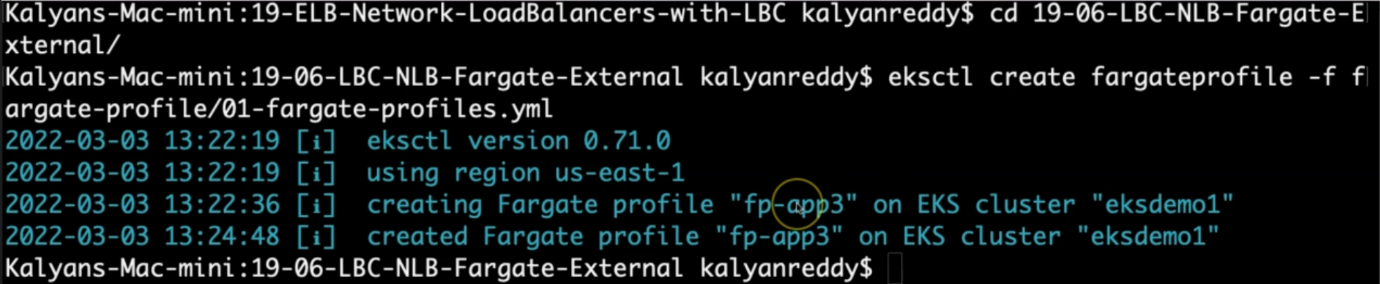
**Create Fargate Profile**

**# Change Directory**

--- **cd 19-06-LBC-NLB-Fargate-External**

**# Create Fargate Profile**

--- **eksctl create fargateprofile -f fargate-profile/01-fargate-profiles.yml**



--- **note** - eksctl takes 3 to 5 minute to create fargate profile.

**Update Annotation aws-load-balancer-nlb-target-type to IP**

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

**service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip # For Fargate Workloads we should use target-type as ip**

apiVersion: v1

kind: Service

metadata:

  name: fargate-lbc-network-lb

  namespace: ns-app3

  annotations:

    # Traffic Routing

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

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

    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip # For Fargate Workloads we should use target-type as ip

    #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/load-balancer-source-ranges: 0.0.0.0/0

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

    # AWS Resource Tags

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

    # TLS

    service.beta.kubernetes.io/aws-load-balancer-ssl-cert: arn:aws:acm:us-east-1:180789647333:certificate/d86de939-8ffd-410f-adce-0ce1f5be6e0d

    service.beta.kubernetes.io/aws-load-balancer-ssl-ports: 443, # Specify this annotation if you need both TLS and non-TLS listeners on the same load balancer

    service.beta.kubernetes.io/aws-load-balancer-ssl-negotiation-policy: ELBSecurityPolicy-TLS13-1-2-2021-06

    service.beta.kubernetes.io/aws-load-balancer-backend-protocol: tcp

    # External DNS - For creating a Record Set in Route53

    external-dns.alpha.kubernetes.io/hostname: nlbfargate901.stacksimplify.com

spec:

  type: LoadBalancer

  selector:

    app: app3-nginx

  ports:

    - name: http

      port: 80

      targetPort: 80

    - name: https

      port: 443

      targetPort: 80

--- **00-namespace.yml**

apiVersion: v1

kind: Namespace

metadata:

  name: ns-app3

# Apps deployed in this namespace will run on a Fargate fp-app3

**Review the k8s Deployment Metadata for namespace**

--- **File Name: kube-manifests/01-Nginx-App3-Deployment.yml**

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

apiVersion: apps/v1

kind: Deployment

metadata:

  name: app3-nginx-deployment

  labels:

    app: app3-nginx

  namespace: ns-app3    # Update Namespace given in Fargate Profile 01-fargate-profiles.yml

spec:

  replicas: 2

  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

          resources:

            requests:

              memory: "128Mi"

              cpu: "500m"

            limits:

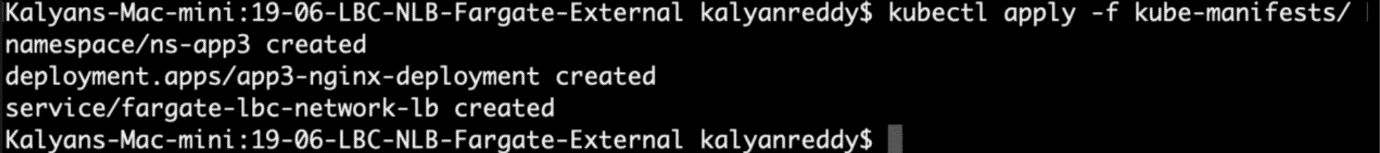
              memory: "500Mi"

              cpu: "1000m"

**Deploy all kube-manifests**

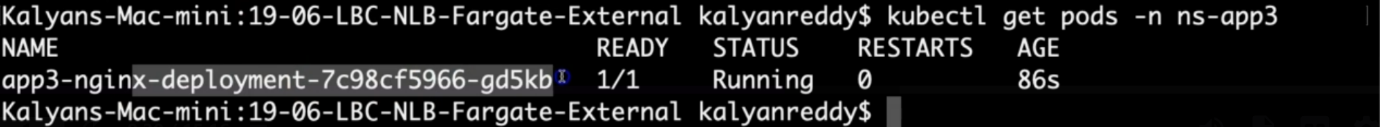
**# Deploy kube-manifests**

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



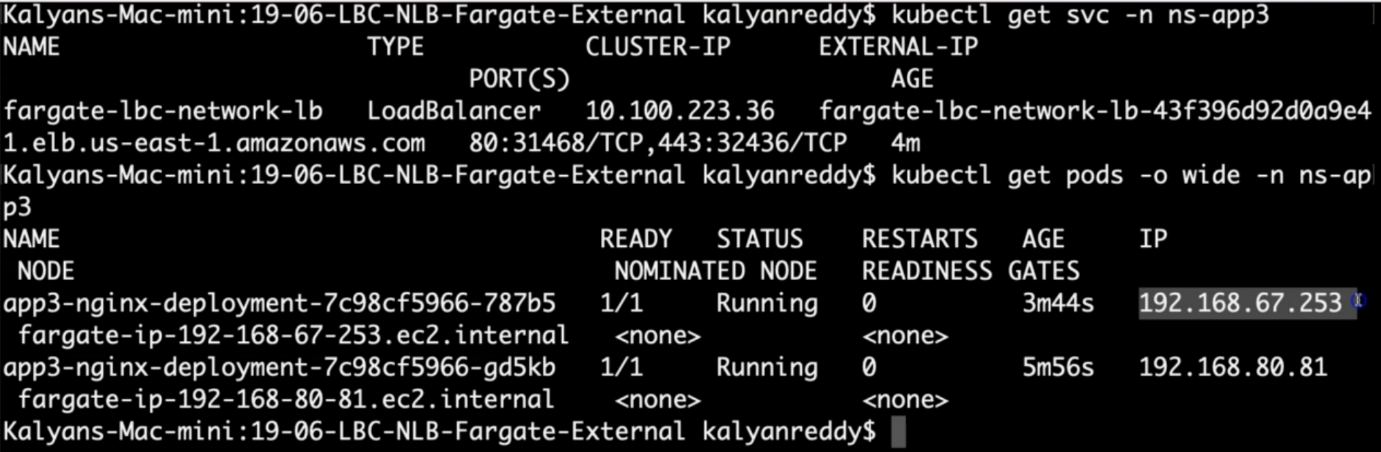
**# list the pods in ns-app3**

--- **kubectl get pods -n ns-app3**



**# Verify Pods**

--- **kubectl get pods -o wide**

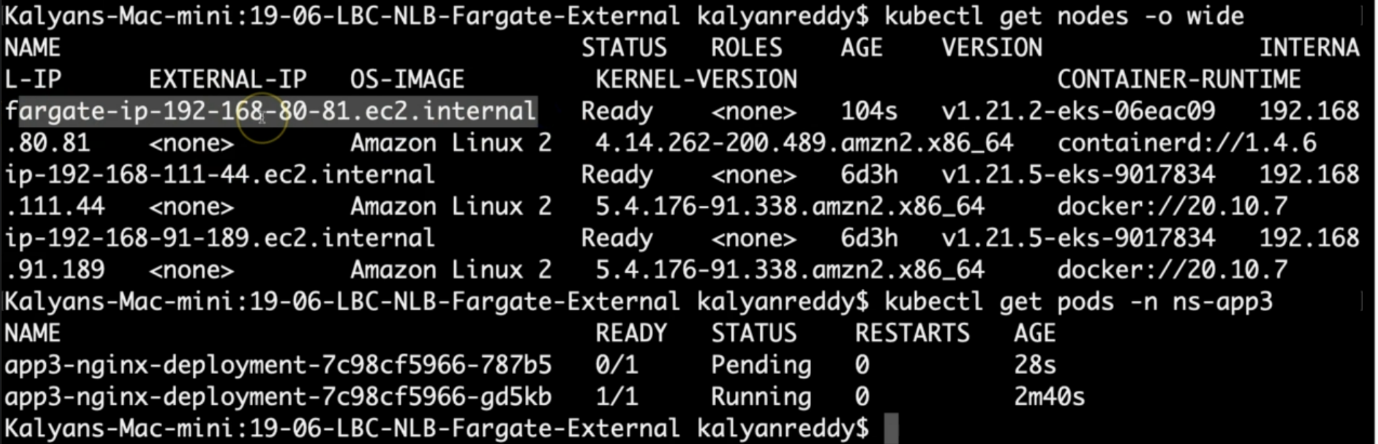


Observation:

1. It will take couple of minutes to get the pod from pending to running state due to Fargate Mode.

**# Verify Worker Nodes**

--- **kubectl get nodes -o wide**

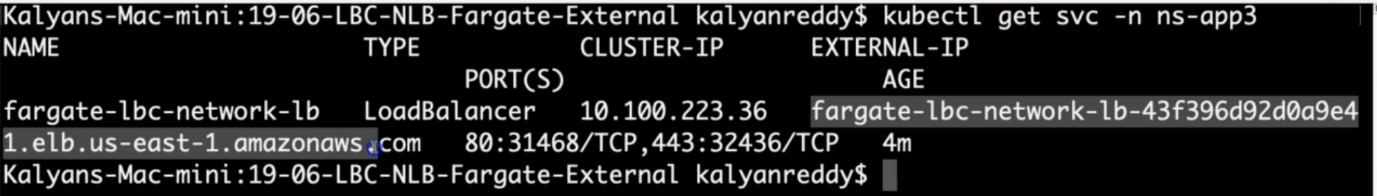


Obseravtion:

1. wait for Fargate worker node to create

**# Verify Services**

--- **kubectl get svc -n ns-app3**



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

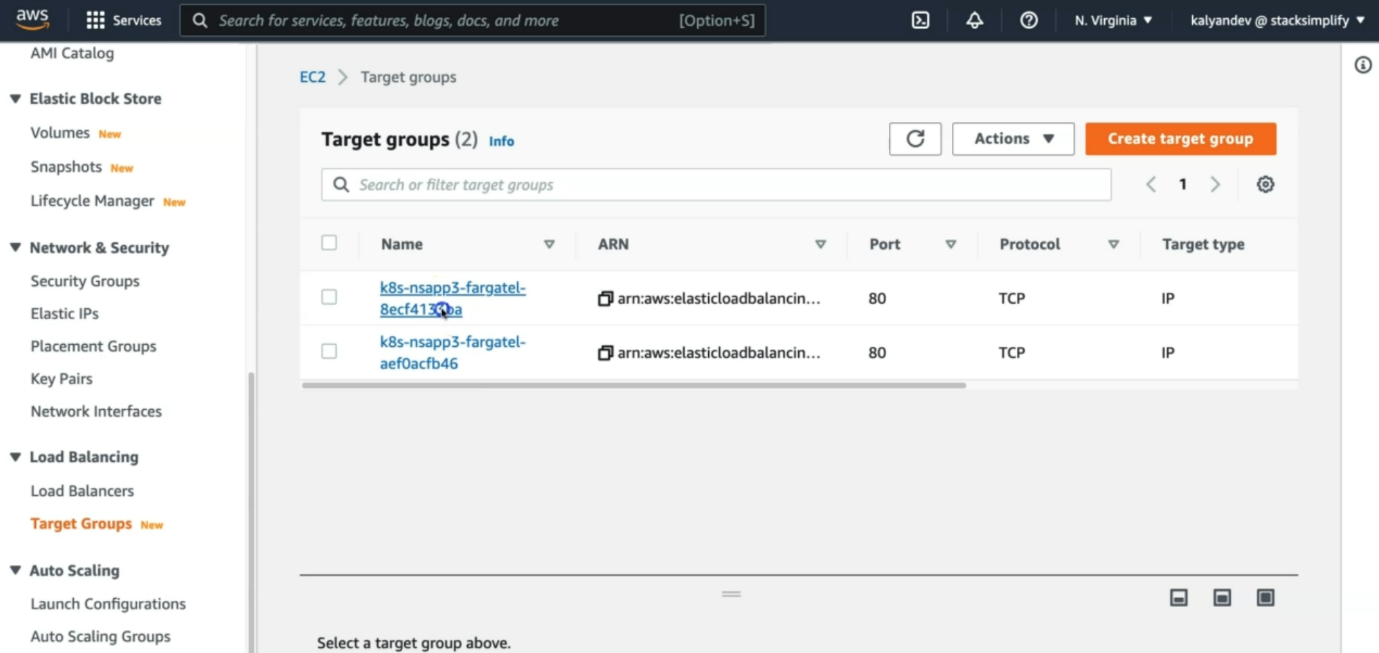
1. Verify Description Tab - DNS Name matching output of "kubectl get svc" External IP

2. Verify Listeners Tab

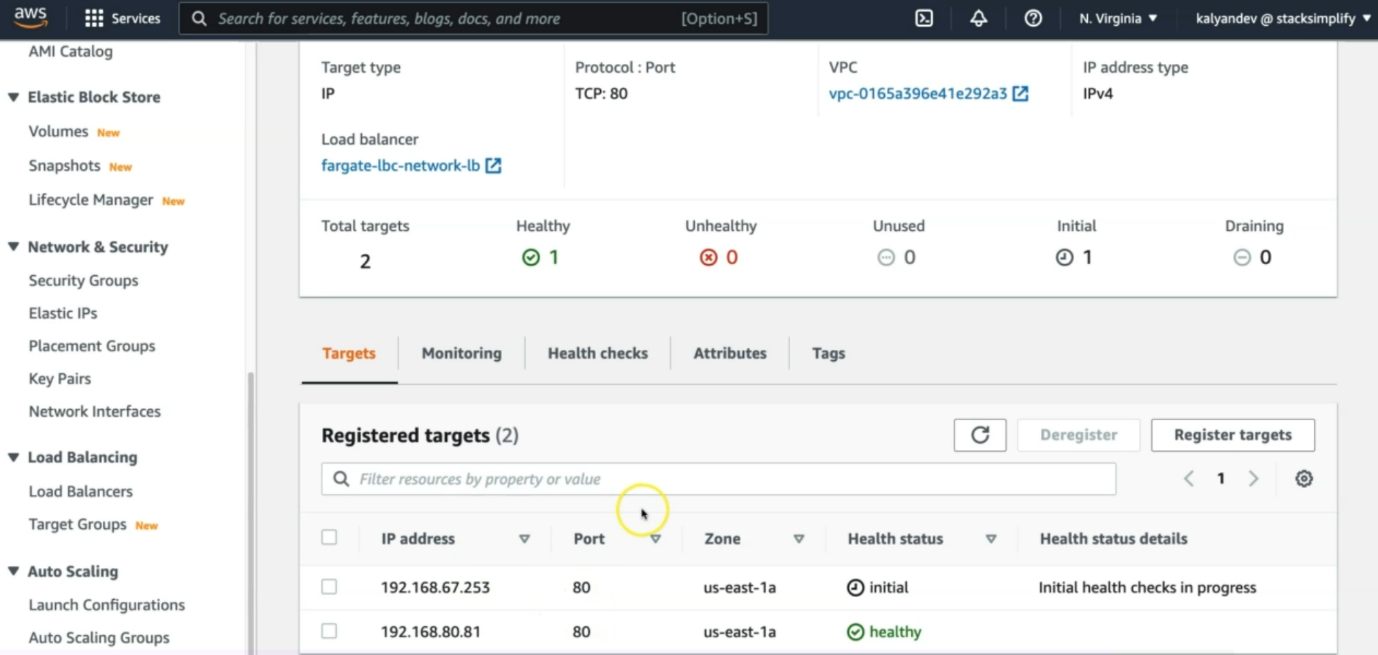
Go to Services -> EC2 -> Load Balancing -> Target Groups

1. Verify Registered targets

2. Verify Health Check path



--- we had created 2 target groups.



**# Perform nslookup Test**

--- nslookup nlbfargate901.stacksimplify.com

**# Access Application**

**# Test HTTP URL**

--- <http://nlbfargate901.stacksimplify.com>

**# Test HTTPS URL**

--- <https://nlbfargate901.stacksimplify.com>

**Clean-Up**

**# Delete or Undeploy kube-manifests**

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

**# 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/>

**Delete Fargate Profile**

**# List Fargate Profiles**

--- **eksctl get fargateprofile --cluster eksdemo1**

**# Delete Fargate Profile**

--- **eksctl delete fargateprofile --cluster eksdemo1 --name <Fargate-Profile-NAME> --wait**

--- **eksctl delete fargateprofile --cluster eksdemo1 --name fp-app3 --wait**